Clustering Techniques

Clustering Overview

Today's lecture
- What is clustering
- Hierarchical algorithms
- Partitional algorithms

Next lecture
- Techniques for clustering large databases
  - BIRCH
  - DBSCAN
  - CURE

Clustering Examples

- **Segment** customer database based on similar buying patterns
- Group houses in a town into neighborhoods based on similar features
- Group plants into categories
- Identify similar Web usage patterns

Clustering Example

Attribute Value Based Clustering
Geographic Distance Based Clustering

Clustering Example: Direct Marketing

<table>
<thead>
<tr>
<th>Income</th>
<th>Age</th>
<th>Children</th>
<th>Marital Status</th>
<th>Education</th>
</tr>
</thead>
<tbody>
<tr>
<td>$25,000</td>
<td>35</td>
<td>3</td>
<td>Single</td>
<td>High School</td>
</tr>
<tr>
<td>$15,000</td>
<td>25</td>
<td>1</td>
<td>Married</td>
<td>High School</td>
</tr>
<tr>
<td>$20,000</td>
<td>40</td>
<td>0</td>
<td>Single</td>
<td>High School</td>
</tr>
<tr>
<td>$30,000</td>
<td>20</td>
<td>0</td>
<td>Divorced</td>
<td>High School</td>
</tr>
<tr>
<td>$20,000</td>
<td>25</td>
<td>3</td>
<td>Divorced</td>
<td>College</td>
</tr>
<tr>
<td>$70,000</td>
<td>60</td>
<td>0</td>
<td>Married</td>
<td>College</td>
</tr>
<tr>
<td>$90,000</td>
<td>30</td>
<td>0</td>
<td>Married</td>
<td>Graduate School</td>
</tr>
<tr>
<td>$200,000</td>
<td>45</td>
<td>5</td>
<td>Married</td>
<td>Graduate School</td>
</tr>
<tr>
<td>$100,000</td>
<td>50</td>
<td>2</td>
<td>Divorced</td>
<td>College</td>
</tr>
</tbody>
</table>

Clustering vs. Classification

- In classification the set of groups to which objects belong is given, and the task is to discriminate between groups on the basis of values for the attributes of the objects
- In clustering there is no prior knowledge: the meaning (and often the number) of clusters is unknown and the objective is to discover them
- Clustering is sometimes called unsupervised learning
**Issues which make Clustering Difficult**

- Outlier handling
- Dynamic data
- Interpreting results
- Evaluating results
- Number of clusters
- Data to be used
- Dimensionality curse
- Scalability

**Impact of Outliers on Clustering**

![Impact of Outliers](image)

**Clustering Problem**

- Given a database $D = \{t_1, t_2, ..., t_n\}$ of tuples and an integer value $k$, the **Clustering Problem** is to define a mapping $f: D \to \{1, ..., k\}$ where each $t_i$ is assigned to one cluster $K_j$, $1 \leq j \leq k$.

- A **Cluster**, $K_j$, contains precisely those tuples mapped to it.

- Unlike classification problem, clusters are not known a priori.

**Types of Clustering**

- **Hierarchical** - Nested set of clusters created
- **Partitional** - One set of clusters created
- **Incremental** - Each element handled one at a time
- **Simultaneous** - All elements handled together
- **Overlapping/Non-overlapping**

**Clustering Approaches**

- Cluster Care
- Hierarchical
- Partitional
- Categorical
- Large DB

- Agglomerative
- Divisive
- Sampling
- Compression

**Cluster Parameters**

- **Centroid** $C_m = \frac{\sum_{i=1}^{N} t_{mi}}{N}$
- **Radius** $R_m = \left( \frac{\sum_{i=1}^{N} (t_{mi} - C_m)^2}{N} \right)^{1/2}$
- **Diameter** $D_m = \left( \frac{\sum_{i=1}^{N} \sum_{j=1}^{N} (t_{mi} - t_{mj})^2}{(N)(N-1)} \right)$
Distance Between Clusters

- **Single Link**: smallest distance between points
- **Complete Link**: largest distance between points
- **Average Link**: average distance between points
- **Centroid**: distance between centroids

Hierarchical Clustering

Clusters are created in levels actually creating sets of clusters at each level.

- **Agglomerative**
  - Initially each item in its own cluster
  - Iteratively clusters are merged together
  - Bottom Up
- **Divisive**
  - Initially all items in one cluster
  - Large clusters are successively divided
  - Top Down

Hierarchical Clustering Algorithms

- Single Link
- MST (Minimum Spanning Tree) Single Link
- Complete Link
- Average Link

Dendrogram

*Dendrogram*: a tree data structure which illustrates hierarchical clustering techniques.

- Each level shows clusters for that level
  - Leaf - individual clusters
  - Root - one cluster
- A cluster at level i is the union of its children clusters at level i+1

Levels of Clustering

Agglomerative Clustering Example

Threshold of 1 2 3 4 5
**MST Example**

\[A \quad B \quad C \quad D \quad E\]

\[\begin{array}{c|ccccc}
A & 0 & 2 & 2 & 3 & \\
B & 1 & 0 & 2 & 4 & 3 \\
C & 2 & 2 & 0 & 1 & 5 \\
D & 2 & 4 & 1 & 0 & 3 \\
E & 3 & 3 & 5 & 3 & 0 \\
\end{array}\]

**Agglomerative Algorithm**

**Input:**
- \(E = \{e_1, e_2, \ldots, e_n\}\) // Set of elements
- \(K\) // Adjacency matrix showing distance between elements.

**Output:**
- \(D\) // Deedragoem represented as a set of ordered triplets.

**Agglomerative Algorithm:**

1. \(k = n\)
2. \(A = \{\{i\} : 1 \leq i \leq n\}\)
3. \(D = \{\{i,j\} : e_{ij} > 0\}\) // Initially deedragoem contains each element in its own cluster.
4. **repeat**
   \(k = k - 1\)
   \(d = \infty\)
   \(\{i,j\} = \arg\min_{\{i,j\} \in A} e_{ij}\)
   \(A = A \setminus \{i,j\}\)
   \(D = D \cup \{\{i,j\}\}\)
5. **until** \(k = 1\)

**Single Link Clustering**

- Views all items with links (distances) between them
- Finds maximal connected components in this graph
- Two clusters are merged if there is at least one edge which connects them
- Uses threshold distances at each level
- Could be agglomerative or divisive

**MST Single Link Algorithm**

**Input:**
- \(E = \{e_1, e_2, \ldots, e_n\}\) // Set of elements
- \(K\) // Adjacency matrix showing distance between elements.

**Output:**
- \(C\) // MST (minimum spanning tree) of \(E\) and \(K\).

**MST Single Link Algorithm:**

1. **repeat**
   \(\{i,j\} = \arg\min_{\{i,j\} \in A} e_{ij}\)
   \(A = A \setminus \{i,j\}\)
   \(C = C \cup \{\{i,j\}\}\)
   \(A = A \cup \{i,j\}\)

**Partitional Clustering**

- Nonhierarchical
- Usually deals with static sets
- Creates clusters in one step as opposed to several steps
- Since only one set of clusters is output, the user normally has to input the desired number of clusters, \(k\)
- Need some metric/criterion that determines the goodness of each proposed solution
Partitional Clustering Algorithms

- MST: Minimum Spanning Tree
- Squared Error
- K-Means
- Nearest Neighbor
- PAM: Partitioning Around Medoids
  - CLARA: Clustering LARge Applications
  - CLARANS: Clustering Large Applications based on RANdomized Search

MST Algorithm

Input:
\[ D = \{ d_{ij}, \ldots, d_{n} \} \] // Set of elements
\[ A \] // Adjacency matrix showing distance between elements,
\[ k \] // Number of desired clusters.

Output:
\[ f \] // Mapping represented as a set of ordered pairs.

Partitional MST Algorithm:
\[ M = MST(A); \]
identify inconsistent edges in M;
remove \( k-1 \) inconsistent edges;
create output representation;

Squared Error

Minimized squared error

\[ s\epsilon K_i = \sum_{j=1}^{m} ||x_j - C_k||^2 \]

\[ s\epsilon K = \sum_{j=1}^{k} s\epsilon K_j \]

Squared Error Algorithm

Input:
\[ D = \{ x_1, x_2, \ldots, x_n \} \] // Set of elements
\[ k \] // Number of desired clusters.

Output:
\[ \{ \} \] // Set of clusters.

Squared Error Algorithm:
assign each item \( i \) to a cluster;
calculate center for each cluster;
repeat
assign each item \( i \) to the cluster which has the closest center;
calculate new center for each cluster;
calculate squared error;
until the difference between successive squared errors is below a threshold;

K-Means Clustering

- Initial set of clusters randomly chosen
- Iteratively, items are moved among sets of clusters until the desired set is reached
- High degree of similarity among elements in a cluster is obtained
- Given a cluster \( K_i(\{x_1, x_2, \ldots, x_n\}) \), the cluster mean is \( m_i = \frac{1}{m} \sum_{j=1}^{m} x_j \)

K-Means Clustering Example

Given: \( \{2, 4, 10, 12, 3, 20, 30, 11, 25\} \), \( k=2 \)
- Randomly assign means: \( m_1=3, m_2=4 \)
- \( K_1 = (2, 3), K_2 = (4, 10, 12, 20, 30, 11, 25), m_1=2.5, m_2=16 \)
- \( K_1 = (2, 3, 4), K_2 = (10, 12, 20, 30, 11, 25), m_1=3, m_2=18 \)
- \( K_1 = (2, 3, 4, 10), K_2 = (12, 20, 30, 11, 25), m_1=4.75, m_2=19.6 \)
- \( K_1 = (2, 3, 4, 10, 11, 12), K_2 = (20, 30, 25), m_1=7, m_2=25 \)
Stop as the clusters with these means are the same.
K-Means Clustering Algorithm

Input:
- $D = \{d_1, d_2, ..., d_n\}$ // Set of elements
- $A$ // Adjacency matrix showing distance between elements.
- $k$ // Number of desired clusters.

Output:
- $K$ // Set of clusters.

K-Means Algorithm:
- Assign initial values for means $w_1, w_2, ..., w_k$;
- repeat
  - assign each item $t$ to the cluster which has the closest mean $w$;
  - calculate new mean for each cluster;
- until convergence criterion is met.

Nearest Neighbor Clustering

- Items are iteratively merged into the existing clusters that are closest.
- Incremental.
- Threshold, $t$, used to determine if items are added to existing clusters or whether a new cluster should be created.

Nearest Neighbor Clustering Algorithm

Input:
- $D = \{d_1, d_2, ..., d_n\}$ // Set of elements
- $A$ // Adjacency matrix showing distance between elements.
- $K$ // Set of clusters.

Nearest Neighbor Algorithm:
- $K_0 = \{i\}$;
- $K = [K_0];$
- for $i = 1$ to $k$ do
  - find the $j_i$ in some cluster $K_i$ in $K$ such that $d_j(t_i, j_i)$ is the smallest;
  - if $d_j(t_i, j_i) < t$ then
    - $K_0 = K_0 \cup \{i\}$;
  - else
    - $k = k + 1$
    - $K_0 = [K];$

PAM: Partitioning Around Medoids (K-Medoids)

- Handles outliers well.
- Ordering of input does not impact results.
- Computationally complex - does not scale well.
- Each cluster represented by one item, called the medoid.
- Initial set of $k$ medoids is randomly chosen.

PAM Example

- Start: Medoids $\{A, B\}$
- Change 1: $A \rightarrow C$
- Change 2: $A \rightarrow D$
- $K_C$: Change 0
- $K_D$: Change 1

PAM Cost Calculation

- At each step in algorithm, medoids are changed if the overall cost is improved.
- $C_{jk}$ - cost change for an item $t_j$ associated with swapping medoid $t_k$ with non-medoid $t_i$.

  1. $t_j \in K_i$ but $\exists$ another medoid $t_i_1$ where $\text{dist}(t_j, t_i_1) \leq \text{dist}(t_j, t_i)$
  2. $t_j \in K_i$ but $\text{dist}(t_j, t_i) \leq \text{dist}(t_j, t_i_1)$ for other medoids $t_i_1$;
  3. $t_j \in K_m \neq K_i$ and $\text{dist}(t_j, t_m) \leq \text{dist}(t_j, t_i)$; and
  4. $t_j \in K_m \neq K_i$ but $\text{dist}(t_j, t_m) \leq \text{dist}(t_j, t_i)$. 
PAM Algorithm

Input:
D := \{d_1, d_2, ..., d_n\} // Set of elements
k // Number of desired clusters.

Output:
E := \{C_1, C_2, ..., C_k\} // Set of clusters.
PAM Algorithm:
1. Arbitrarily select k medoids from D.
2. Repeat for each \( c_i \) not a medoid:
   - For each medoid \( c_j \) do:
     - Calculate \( D(c_i, c_j) \).
     - Find \( i, j \) where \( D(c_i, c_j) \) is the smallest.
   - If \( D(c_i, c_j) \leq \theta \) then:
     - Replace medoid \( c_j \) with \( c_i \).
3. Repeat until \( D(c_i, c_j) \geq \theta \) for each \( c_i \) not a medoid:
   - For each \( c_i \) not a medoid:
     - Assign \( c_i \) to \( C_j \) where \( \text{dist}(c_i, c_j) \) is the smallest over all medoids.

---

Clustering Large Databases

- Most clustering algorithms assume a large data structure which is memory resident.
- Clustering may be performed first on a sample of the database, then applied to the entire database.
- Algorithms:
  - BIRCH
  - DBSCAN
  - CURE

---

BIRCH: Balanced Iterative Reducing and Clustering using Hierarchies

- Incremental, hierarchical, one scan.
- Saves clustering information in a balanced tree.
- Each entry in the tree contains summary information about one cluster.
- New nodes inserted in closest entry in tree.
- Adapts to main memory size by changing the threshold value.
  - Larger threshold \( \Rightarrow \) Smaller tree.

---

CLARA and CLARANS

CLARA: Clustering LARge Applications
1. Determine set of medoids \( M \) by using a sample \( D' \) of the database \( D \), where \( |D'| \ll |D| \).
2. Do clustering of \( D \) based on the set of medoids \( M \).

CLARANS: Clustering Large Applications based on RANdomized Search
- Improved CLARA clustering algorithm which uses several randomly picked samples \( D' \) instead of only one.

---

Clustering Feature

- CF Triple: (N, SS)
  - N: Number of points in cluster
  - SS: Sum of squares of points in the cluster
- CF Tree
  - Balanced search tree
  - Leaf node represents cluster and has CF value for each subcluster in it
  - Subcluster has maximum diameter.
**BIRCH Algorithm**

**Input:**
- \( D = \{f_1, f_2, ..., f_n\} \) // Set of elements
- \( \tau \) // Threshold for CF tree construction.

**Output:**
- \( k \) // Set of clusters.

**BIRCH Clustering Algorithm:**

for each \( f_i \in D \) do
  - determine correct leaf node for \( f_i \), insertion;
  - if threshold condition is not violated then
    - add \( f_i \) to cluster and update CF triples;
  - else
    - if room to insert \( f_i \), then
      - insert \( f_i \) as single cluster and update CF triples;
    - else
      - split leaf node and redistribute CF features;

---

**DBSCAN: Density Based Spatial Clustering of Applications with Noise**

- Based on the notion of density
- Outliers will not effect creation of clusters
- Two parameters which are input:
  - \( \text{MinPts} \) - minimum number of points in cluster
  - \( \varepsilon \) - for each point in cluster there must be another point in it less than this distance away
- The number of clusters, \( k \), is not input but is determined by the algorithm itself.

---

**DBSCAN Example**

- database 1
- database 2
- database 3

**figure 1: Sample databases**

---

**DBSCAN: Improving Clusters**

1. Create initial CF tree using Algorithm. If there is insufficient memory to construct the CF tree with a given threshold, the threshold value is increased and a new smaller CF tree is constructed.

2. Apply another global clustering approach applied to the leaf nodes in the CF tree. Here each leaf node is treated as a single point for clustering.

3. The last phase (which is optional) re-clusters all points by placing them in the cluster which has the closest centroid.

---

**DBSCAN Example**

- database 1
- database 2
- database 3

**figure 5: Clusterings discovered by CLARANS**

---

**DBSCAN Example**

- database 1
- database 2
- database 3

**figure 6: Clusterings discovered by DBSCAN**
**DBSCAN Density Concepts**

- **Eps-neighborhood**: Points within Eps distance of a point.
- **Core point**: Point whose Eps-neighborhood is dense enough (MinPts) and forms the main portion of some cluster.
- **Directly density-reachable**: A point p is directly density-reachable from a point q if the distance between them is small (Eps) and q is a core point.
- **Density-reachable**: A point is density-reachable from another point if there is a path from one to the other consisting of only core points.

**DBSCAN Algorithm**

```
Input:
D = \{t_1, t_2, \ldots, t_n\} // Set of elements.
MinPts // Number of points in cluster.
Eps // Maximum distance for density measure.
Output:
K = \{K_1, K_2, \ldots, K_k\} // Set of clusters.
DBSCAN Algorithm:

k = 0 // Initially there are no clusters.
for i = 1 to n do
  if t_i is not in a cluster then
    X = \{t_i\} is density-reachable from t_i; // X = \{t_i\}
    if X is a valid cluster then
      k = k + 1;
      K_k = X;
```

**CURE: Clustering Using REpresentatives**

- Use many points to represent a cluster instead of only one
- Representative points need to be well scattered
- Handles outliers well - removes clusters which
  - Either grow very slowly
  - Or contain very few points

**CURE Algorithm**

```
Input:
D = \{d_1, d_2, \ldots, d_n\} // Set of elements.
k // Desired number of clusters.
Output:
Q // Heap containing one entry for each cluster.
CURE Algorithm:

T = buildTree(); // Put each point in Tree.
Q = heapify(T); // Initially build heap with one entry per item;
repeat
  s = min(Q); // s closest
  x = select(s); // x closest
  del(x, T, s); // delete(x, T, s);
  n(x, T, s); // insert(x, T, s);
  for each i \in Q do
    if i is closest to x then
      w.close = x;
      if w.close \in Q then
        del(w.close, T, Q);
      until number of nodes in Q is k;
```
**CURE for Large Databases**

1. Obtain a sample of the database
2. Partition the sample into $p$ partitions
3. Partially cluster the points in each partition
4. Remove outliers based on size of cluster
5. Completely cluster all data in the samples (representatives)
6. Cluster entire database on disk using $c$ points to represent each cluster. An item in the database is placed in the cluster which has the closest representative point to it.

**Comparison of Clustering Techniques**

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Type</th>
<th>Span</th>
<th>Time</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single Link</td>
<td>Hierarchical</td>
<td>$O(n)$</td>
<td>$O(n^2)$</td>
<td>Not incremental</td>
</tr>
<tr>
<td>Average Link</td>
<td>Hierarchical</td>
<td>$O(n)$</td>
<td>$O(n^2)$</td>
<td>Not incremental</td>
</tr>
<tr>
<td>Complete Link</td>
<td>Hierarchical</td>
<td>$O(n)$</td>
<td>$O(n^2)$</td>
<td>Not incremental</td>
</tr>
<tr>
<td>SNN</td>
<td>Hierarchical/Partitional</td>
<td>$O(n)$</td>
<td>$O(n)$</td>
<td>Not incremental</td>
</tr>
<tr>
<td>Squared Error</td>
<td>Partitional</td>
<td>$O(n^2)$</td>
<td>$O(n^2)$</td>
<td>Incremental</td>
</tr>
<tr>
<td>k-Means</td>
<td>Partitional</td>
<td>$O(nk)$</td>
<td>$O(nk)$</td>
<td>Incremental, No categorized</td>
</tr>
<tr>
<td>Nearest Neighbor</td>
<td>Partitional</td>
<td>$O(n^2)$</td>
<td>$O(n)$</td>
<td>Incremental</td>
</tr>
<tr>
<td>FAST</td>
<td>Partitional</td>
<td>$O(n)$</td>
<td>$O(n)$</td>
<td>Incremental</td>
</tr>
<tr>
<td>DBSCAN</td>
<td>Partitional</td>
<td>$O(n)$</td>
<td>$O(n)$</td>
<td>Incremental, Outliers</td>
</tr>
<tr>
<td>CUBE</td>
<td>Mixed</td>
<td>$O(n^2)\text{ or } O(n^2 \log n)$</td>
<td>$O(n^2 \log n)$</td>
<td>Incremental, Outliers</td>
</tr>
<tr>
<td>ROCK</td>
<td>Agglomerative</td>
<td>$O(n^2) \text{ or } O(n^2)$</td>
<td>$O(n^2)$</td>
<td>Sampling, Categorical Data</td>
</tr>
<tr>
<td>BRCLUS</td>
<td>Mixed</td>
<td>$O(n^2)$</td>
<td>$O(n^2)$</td>
<td>Sampling, Outliers</td>
</tr>
</tbody>
</table>