Cluster Analysis

- Overview
- Partitioning methods
- Hierarchical methods
  - Classical methods
  - Recent methods
- Density-based methods
- Other Methods
- Outlier analysis
- Summary
Recent Hierarchical Clustering Methods

- **BIRCH (1996)**: uses CF-tree and incrementally adjusts the quality of sub-clusters
- **CURE (1998)**: uses representative points for inter-cluster distance
- **ROCK (1999)**: clustering categorical data by neighbor and link analysis
- **CHAMELEON (1999)**: hierarchical clustering using dynamic modeling
Birch

Birch: Balanced Iterative Reducing and Clustering using Hierarchies (Zhang, Ramakrishnan & Livny, SIGMOD’96)

- SIGMOD 10 year test of time award

Main ideas:
- Incremental (does not need the whole dataset)
- Use in-memory clustering feature to summarize existing cluster
- Use hierarchical clustering for microclustering and other clustering methods (e.g. partitioning) for macroclustering

Features:
- Scales linearly: single scan and improves the quality with a few additional scans
- handles only numeric data, and sensitive to the order of the data record.
BIRCH Overview

Data

Phase 1: Load into memory by building a CF tree

Initial CF tree

Phase 2 (optional): Condense into desirable range by building a smaller CF tree

smaller CF tree

Phase 3: Global Clustering

Good Clusters

Phase 4: (optional and offline): Cluster Refining

Better Clusters
Building Clustering Feature (CF) Tree

- **Clustering Feature (CF)** to represent a cluster
- **CF tree** to represent a hierarchy of clusters
- **CF tree insertion** - insert a new point/cluster
Cluster Statistics

Given a cluster of instances \( \{ \vec{X}_i \} \)

**Centroid**: 
\[
\vec{X}_0 = \frac{\sum_{i=1}^{N} \vec{X}_i}{N}
\]

**Radius**: average distance from member points to centroid
\[
R = \left( \frac{\sum_{i=1}^{N} (\vec{X}_i - \vec{X}_0)^2}{N} \right)^{\frac{1}{2}}
\]

**Diameter**: average pair-wise distance within a cluster
\[
D = \left( \frac{\sum_{i=1}^{N} \sum_{j=1}^{N} (\vec{X}_i - \vec{X}_j)^2}{N(N - 1)} \right)^{\frac{1}{2}}
\]
Intra-Cluster Distance

Given two clusters

Centroid Euclidean distance:

\[ D_0 = \left( (\overrightarrow{X_{01}} - \overrightarrow{X_{02}})^2 \right)^{\frac{1}{2}} \]

Centroid Manhattan distance:

\[ D_1 = |\overrightarrow{X_{01}} - \overrightarrow{X_{02}}| = \sum_{i=1}^{d} |\overrightarrow{X_{01}^{(i)}} - \overrightarrow{X_{02}^{(i)}}| \]

Average distance:

\[ D_2 = \left( \frac{\sum_{i=1}^{N_1} \sum_{j=N_{1}+1}^{N_1+N_2} (\overrightarrow{X_i} - \overrightarrow{X_j})^2}{N_1 N_2} \right)^{\frac{1}{2}} \]
Clustering Feature (CF)

Given a cluster \( \{ \vec{X}_1, \vec{X}_2, \ldots, \vec{X}_N \} \)

\[
\text{CF} = (N, \bar{LS}, SS)
\]

\[N \text{ is the number of data points}\]
\[\bar{LS} = \sum_{i=1}^{N} \vec{X}_i\]
\[SS = \sum_{i=1}^{N} \vec{X}_i^2\]

\[
\text{CF}_1 + \text{CF}_2 = (N_1 + N_2, \bar{LS}_1 + \bar{LS}_2, SS_1 + SS_2)
\]

\[
\text{CF} = (5, (16,30),(54,190))
\]

\[
\begin{align*}
(3,4) \\
(2,6) \\
(4,5) \\
(4,7) \\
(3,8)
\end{align*}
\]
Properties of Clustering Feature

- CF entry is more compact
  - Stores significantly less than all of the data points in the sub-cluster
- A CF entry has sufficient information to calculate statistics about the cluster and intra-cluster distances
- Additivity theorem allows us to merge sub-clusters incrementally & consistently
Hierarchical CF-Tree

- A CF tree is a height-balanced tree that stores the clustering features for a hierarchical clustering
  - A nonleaf node in a tree has descendants or “children”
  - The nonleaf nodes store sums of the CFs of their children
- A CF tree has two parameters
  - Branching factor: specify the maximum number of children.
  - threshold: max diameter of sub-clusters stored at the leaf nodes
CF-Tree Insertion

- Traverse down from root, find the appropriate leaf
  - Follow the "closest"-CF path, w.r.t. intra-cluster distance measures
- Modify the leaf
  - If the closest-CF leaf cannot absorb (diameter exceeds the threshold), make a new CF entry.
  - If there is no room for new leaf (# of children exceeds branching factor), split the parent node
- Traverse back & up
  - Updating CFs on the path or splitting nodes

February 25, 2011
The Algorithm: BIRCH

- **Phase 1**: Scan database to build an initial in-memory CF-tree
  - Subsequent phases become fast, accurate, less order sensitive
- **Phase 2**: Condense data (optional)
  - Rebuild the CF-tree with leaf nodes
- **Phase 3**: Global clustering
  - Use existing clustering algorithm on leaf nodes
  - Helps fix problem where natural clusters span nodes
- **Phase 4**: Cluster refining (optional)
  - Do additional passes over the dataset & reassign data points to the closest centroid from phase 3
Birch

Main ideas:
- Incremental (does not need the whole dataset)
- Use in-memory **clustering feature** to summarize existing cluster
- Use hierarchical clustering for microclustering and other clustering methods (e.g. partitioning) for macroclustering

Features:
- *Scales linearly*: single scan and improves the quality with a few additional scans
- Flexibility of combining partitioning methods with hierarchical methods
- handles only numeric data, and sensitive to the order of the data record.
CURE

- CURE: An Efficient Clustering Algorithm for Large Databases (1998) Sudipto Guha, Rajeev Rastogi, Kyusckok Shim

- Potential problems with min, max, centroid distance based hierarchic clustering

- Main ideas:
  - Use representative points for inter-cluster distance
  - Random sampling and partitioning

- Features:
  - More robust to outliers
  - Handles non-spherical shapes and arbitrary sizes better
CURE: Cluster Points

- Uses a number of points to represent a cluster (tradeoff between using centroid and using all points)

  ![Diagram showing cluster points and centroid](image)

- Representative points are found by selecting a constant number of points from a cluster and then “shrinking” them toward the center of the cluster

- Cluster similarity is the similarity of the closest pair of representative points from different clusters
Experimental Results: CURE

Actual performance depending on the shrinking factor
CURE Cannot Handle Differing Densities

Original Points

CURE
Clustering Categorical Data: The ROCK Algorithm

- ROCK: RObust Clustering using linKs
  - S. Guha, R. Rastogi & K. Shim, ICDE’99
- Major ideas
  - Use links to measure similarity/proximity
  - Sampling-based clustering
- Features:
  - More meaningful clusters
  - Emphasizes interconnectivity but ignores proximity
Similarity Measure in ROCK

- Market basket data clustering
- Jaccard co-efficient-based similarity function: 
  \[ Sim(T_1, T_2) = \frac{|T_1 \cap T_2|}{|T_1 \cup T_2|} \]

- Example: Two groups (clusters) of transactions
  - \( C_1. <a, b, c, d, e> \)
    - \{a, b, c\}, \{a, b, d\}, \{a, b, e\}, \{a, c, d\}, \{a, c, e\}, \{a, d, e\}, \{b, c, d\}, \{b, c, e\}, \{b, d, e\}, \{c, d, e\}
  - \( C_2. <a, b, f, g> \)
    - \{a, b, f\}, \{a, b, g\}, \{a, f, g\}, \{b, f, g\}
  - Let \( T_1 = \{a, b, c\} \), \( T_2 = \{c, d, e\} \), \( T_3 = \{a, b, f\} \)
  - Jaccard coefficient may lead to wrong clustering result
    \[ Sim(T_1, T_2) = \frac{|\{c\}|}{|\{a, b, c, d, e\}|} = \frac{1}{5} = 0.2 \]
    \[ Sim(T_1, T_3) = \frac{|\{c, f\}|}{|\{a, b, c, f\}|} = \frac{2}{4} = 0.5 \]
Link Measure in ROCK

- Neighbor: \( \text{Sim} (T_i, T_j) \geq \theta \)
- Links: # of common neighbors
- Example:
  - \( \text{link}(T_1, T_2) = 4, \text{since they have 4 common neighbors} \)
    - \{a, c, d\}, \{a, c, e\}, \{b, c, d\}, \{b, c, e\}
  - \( \text{link}(T_1, T_3) = 3, \text{since they have 3 common neighbors} \)
    - \{a, b, d\}, \{a, b, e\}, \{a, b, g\}
Rock Algorithm

1. Obtain a sample of points from the data set
2. Compute the link value for each set of points (computed by Jaccard coefficient)
3. Perform an agglomerative hierarchical clustering on the data using the “number of shared neighbors” as similarity measure
4. Assign the remaining points to the clusters that have been found

- CHAMELEON: by G. Karypis, E.H. Han, and V. Kumar’99
- Basic ideas:
  - A graph-based clustering approach
  - A two-phase algorithm:
    - Partitioning: cluster objects into a large number of relatively small sub-clusters
    - Agglomerative hierarchical clustering: repeatedly combine these sub-clusters
  - Measures the similarity based on a dynamic model
    - interconnectivity and closeness (proximity)
- Features:
  - Handles clusters of arbitrary shapes, sizes, and density
  - Scales well
Graph-Based Clustering

- Uses the proximity graph
  - Start with the proximity matrix
  - Consider each point as a node in a graph
  - Each edge between two nodes has a weight which is the proximity between the two points
- Fully connected proximity graph
- Sparsification of the fully connected graph
  - kNN
    - Threshold based
- Clusters are connected components in the graph
Overall Framework of CHAMELEON

Data Set → Construct Sparse Graph → Partition the Graph → Merge Partition → Final Clusters
Chameleon: Steps

- **Preprocessing Step:** Represent the Data by a Graph
  - Given a set of points, construct the k-nearest-neighbor (k-NN) graph to capture the relationship between a point and its k nearest neighbors
  - Concept of neighborhood is captured dynamically (even if region is sparse)

- **Phase 1:** Use a multilevel graph partitioning algorithm on the graph to find a large number of clusters of well-connected vertices
  - Each cluster should contain mostly points from one “true” cluster, i.e., is a sub-cluster of a “real” cluster
Chameleon: Steps ...

- Phase 2: Use Hierarchical Agglomerative Clustering to merge sub-clusters
  - Two clusters are combined if the resulting cluster shares certain properties with the constituent clusters
  - How to merge?
Limitations of Current Merging Schemes

Closeness schemes will merge (a) and (b)

Average connectivity schemes will merge (c) and (d)
Chameleon: Clustering Using Dynamic Modeling

- Adapt to the characteristics of the data set to find the natural clusters
- Use a dynamic model to measure the similarity between clusters
  - Main property is the relative closeness and relative interconnectivity of the cluster
  - Two clusters are combined if the resulting cluster shares certain properties with the constituent clusters
  - The merging scheme preserves self-similarity
CHAMELEON (Clustering Complex Objects)
Cluster Analysis

- Overview
- Partitioning methods
- Hierarchical methods
- **Density-based methods**
- Other methods
- Cluster evaluation
- Outlier analysis
- Summary