Data Mining
Cluster Analysis: Basic Concepts and Algorithms

Lecture Notes for Chapter 5

Data Mining

by

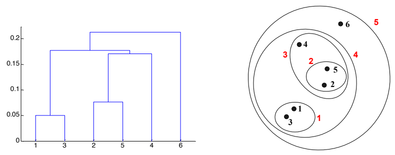
Zhaonian Zou

5.3 Hierarchical Clustering

5.3.1 What is Hierarchical Clustering?

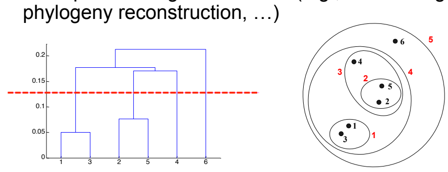
Hierarchical Clustering

- Hierarchical clustering produces a set of nested clusters organized as a hierarchical tree
- Hierarchical clustering can be visualized as a dendrogram
  – A tree like diagram that records the sequences of merges or splits



Strengths of Hierarchical Clustering

- Do not have to assume any particular number of clusters
  – Any desired number of clusters can be obtained by 'cutting' the dendogram at the proper level
- They may correspond to meaningful taxonomies
  – Example in biological sciences (e.g., animal kingdom, phylogeny reconstruction, …)



Types of Hierarchical Clustering

- Two main types of hierarchical clustering
  – Agglomerative:
    - Start with the points as individual clusters
    - At each step, merge the closest pair of clusters until only one cluster (or k clusters) left
  – Divisive:
    - Start with one, all-inclusive cluster
    - At each step, split a cluster until each cluster contains a point (or there are k clusters)
- Traditional hierarchical algorithms use a similarity or distance matrix (also known as proximity matrix)
  – Merge or split one cluster at a time

Proximity Matrix

- The raw data is D = $\{p_1, p_2, \ldots, p_n\}$.
- The raw data is transformed into the form of a proximity matrix.
- The element M_{ij} of the proximity matrix M is the similarity or distance between points p_i and p_j.
  – Euclidean distance
  – Cosine similarity
  – Jaccard similarity

5.3 Hierarchical Clustering

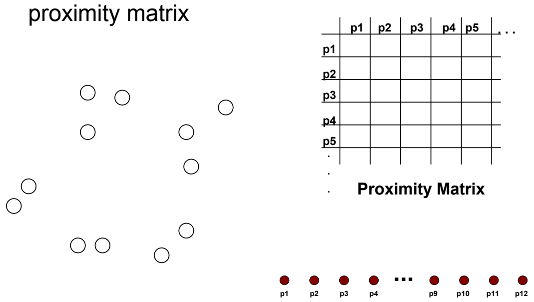
5.3.2 Agglomerative Clustering

Agglomerative Clustering Algorithm

- More popular hierarchical clustering technique
- Basic algorithm is straightforward
  1. Compute the proximity matrix
  2. Let each data point be a cluster
  3. **Repeat**
  4. Merge the two closest clusters
  5. Update the proximity matrix
  6. **Until** only a single cluster remains
- Key operation is the computation of the proximity of two clusters
  - Different approaches to defining the distance between clusters distinguish the different algorithms

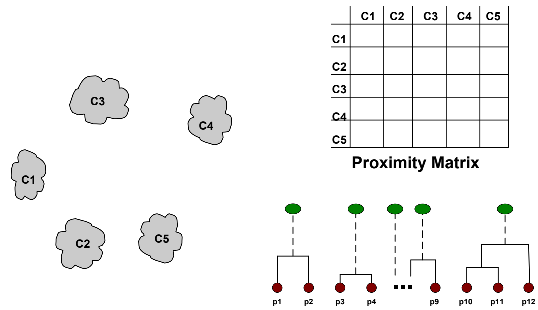
Starting Situation

- Start with clusters of individual points and a proximity matrix



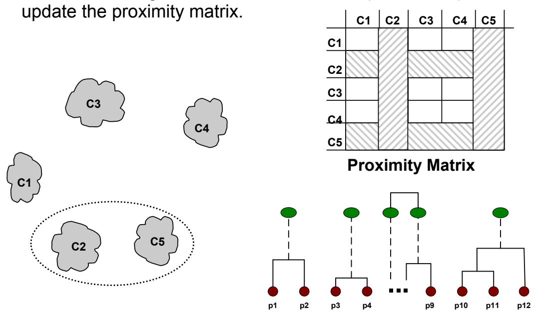
Intermediate Situation

- After some merging steps, we have some clusters



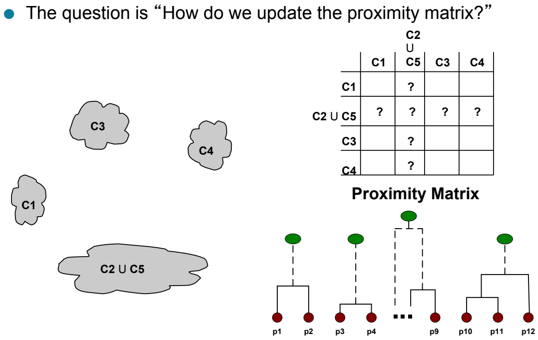
Intermediate Situation

- We want to merge the two closest clusters (C2 and C5) and update the proximity matrix.



After Merging

- The question is "How do we update the proximity matrix?"



Hierarchical Clustering: Time and Space requirements

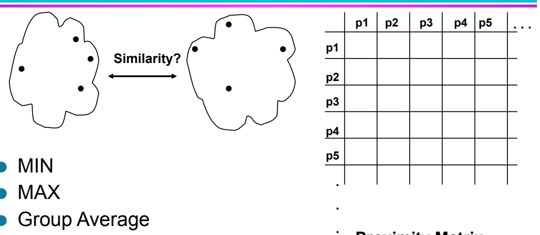
- $O(N^2)$ space since it uses the proximity matrix.
  - N is the number of points.
- $O(N^3)$ time in many cases
  - There are N steps and at each step the size, N^2, proximity matrix must be updated and searched
  - Complexity can be reduced to $O(N^2 \log(N))$ time for some approaches by using priority queues

5.3 Hierarchical Clustering

5.3.2 Agglomerative Clustering

Inter-Cluster Similarity

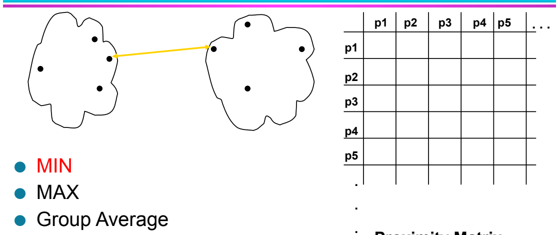
How to Define Inter-Cluster Similarity



Proximity Matrix

- MIN
- MAX
- Group Average
- Distance Between Centroids
- Other methods driven by an objective function
  - Ward's Method uses squared error

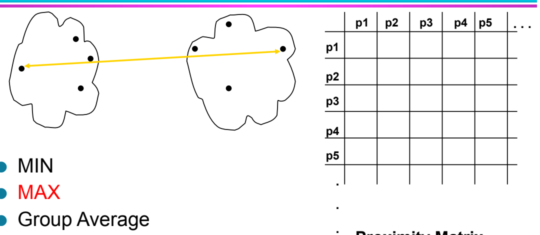
How to Define Inter-Cluster Similarity



Proximity Matrix

- MIN
- MAX
- Group Average
- Distance Between Centroids
- Other methods driven by an objective function
  - Ward's Method uses squared error

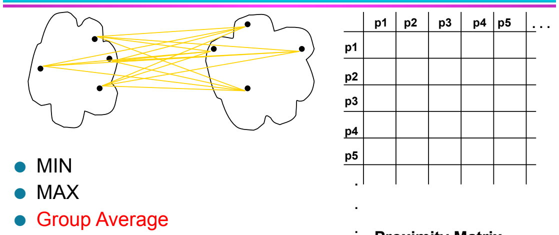
How to Define Inter-Cluster Similarity



Proximity Matrix

- MIN
- MAX
- Group Average
- Distance Between Centroids
- Other methods driven by an objective function
  - Ward's Method uses squared error

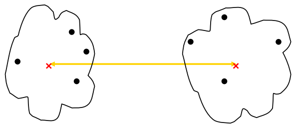
How to Define Inter-Cluster Similarity

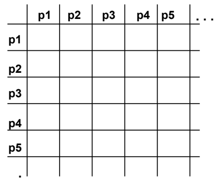


Proximity Matrix

- MIN
- MAX
- Group Average
- Distance Between Centroids
- Other methods driven by an objective function
  - Ward's Method uses squared error

How to Define Inter-Cluster Similarity





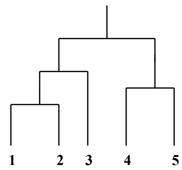
Proximity Matrix

- MIN
- MAX
- Group Average
- Distance Between Centroids
- Other methods driven by an objective function
  - Ward's Method uses squared error

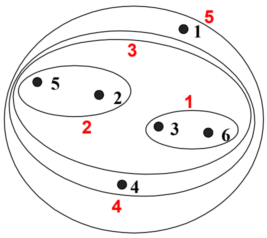
Cluster Similarity: MIN or Single Link

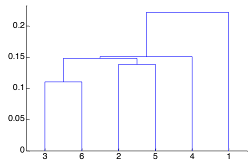
- Similarity of two clusters is based on the two most similar (closest) points in the different clusters
  - Determined by one pair of points, i.e., by one link in the proximity graph.

	I1	I2	I3	I4	I5
I1	1.00	0.90	0.10	0.65	0.20
I2	0.90	1.00	0.70	0.60	0.50
I3	0.10	0.70	1.00	0.40	0.30
I4	0.65	0.60	0.40	1.00	0.80
I5	0.20	0.50	0.30	0.80	1.00



Hierarchical Clustering: MIN

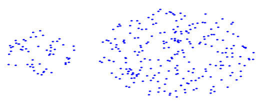




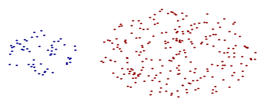
Nested Clusters

Dendrogram

Strength of MIN



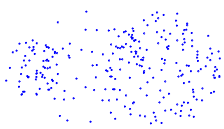
Original Points



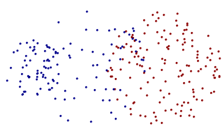
Two Clusters

- Can handle non-elliptical shapes

Limitations of MIN



Original Points



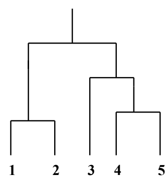
Two Clusters

- Sensitive to noise and outliers

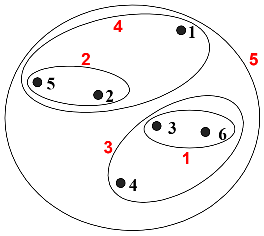
Cluster Similarity: MAX or Complete Linkage

- Similarity of two clusters is based on the two least similar (most distant) points in the different clusters
  - Determined by all pairs of points in the two clusters

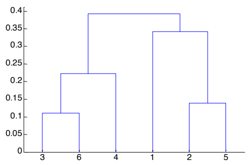
	I1	I2	I3	I4	I5
I1	1.00	0.90	0.10	0.65	0.20
I2	0.90	1.00	0.70	0.60	0.50
I3	0.10	0.70	1.00	0.40	0.30
I4	0.65	0.60	0.40	1.00	0.80
I5	0.20	0.50	0.30	0.80	1.00



Hierarchical Clustering: MAX

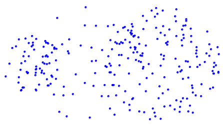


Nested Clusters

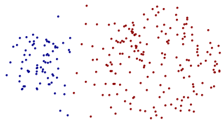


Dendrogram

Strength of MAX



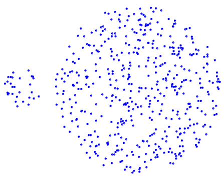
Original Points



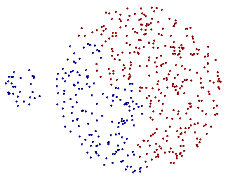
Two Clusters

• Less susceptible to noise and outliers

Limitations of MAX



Original Points



Two Clusters

•Tends to break large clusters

•Biased towards globular clusters

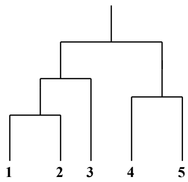
Cluster Similarity: Group Average

- Proximity of two clusters is the average of pairwise proximity between points in the two clusters.

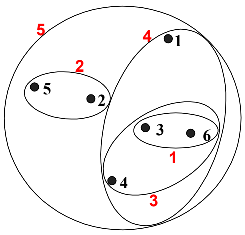
$$\text{proximity}(\text{Cluster}_i, \text{Cluster}_j) = \frac{\sum\limits_{\substack{p_i \in \text{Cluster}_i \\ p_j \in \text{Cluster}_j}} \text{proximity}(p_i, p_j)}{|\text{Cluster}_i| * |\text{Cluster}_j|}$$

- Need to use average connectivity for scalability since total proximity favors large clusters

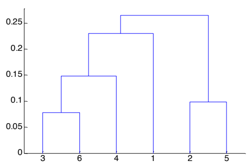
	I1	I2	I3	I4	I5
I1	1.00	0.90	0.10	0.65	0.20
I2	0.90	1.00	0.70	0.60	0.50
I3	0.10	0.70	1.00	0.40	0.30
I4	0.65	0.60	0.40	1.00	0.80
I5	0.20	0.50	0.30	0.80	1.00



Hierarchical Clustering: Group Average



Nested Clusters



Dendrogram

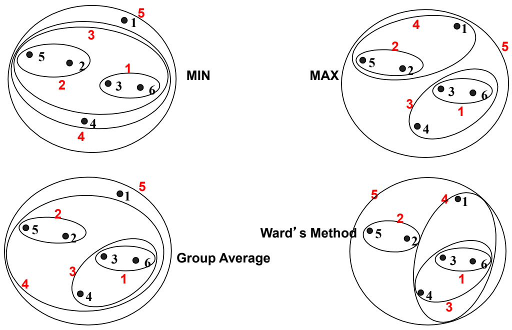
Hierarchical Clustering: Group Average

- Compromise between Single and Complete Link
- Strengths
  - Less susceptible to noise and outliers
- Limitations
  - Biased towards globular clusters

Cluster Similarity: Ward's Method

- Similarity of two clusters is based on the increase in squared error when two clusters are merged
  - Similar to group average if distance between points is distance squared
- Less susceptible to noise and outliers
- Biased towards globular clusters
- Hierarchical analogue of K-means
  - Can be used to initialize K-means

Hierarchical Clustering: Comparison



Hierarchical Clustering: Problems and Limitations

- Once a decision is made to combine two clusters, it cannot be undone
- No objective function is directly minimized
- Different schemes have problems with one or more of the following:
  - Sensitivity to noise and outliers
  - Difficulty handling different sized clusters and convex shapes
  - Breaking large clusters

5.3 Hierarchical Clustering

5.3.2 Agglomerative Clustering

Handling Non-Euclidean Spaces

Hierarchical Clustering in Non-Euclidean Spaces

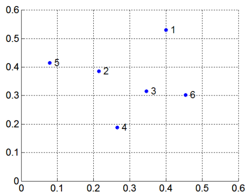
- The points in a cluster cannot be averaged.
- Select as the clustroid the point that minimizes
  - the sum of the distances to the other points in the cluster
  - the maximum distance to another point in the cluster
  - the sum of the squares of the distances to the other points in the cluster

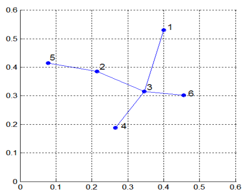
5.3 Hierarchical Clustering

5.3.3 Divisive Clustering

MST: Divisive Hierarchical Clustering

- Build MST (Minimum Spanning Tree)
  - Start with a tree that consists of any point
  - In successive steps, look for the closest pair of points (p, q) such that one point (p) is in the current tree but the other (q) is not
  - Add q to the tree and put an edge between p and q





MST: Divisive Hierarchical Clustering

- Use MST for constructing hierarchy of clusters

Algorithm 7.5 MST Divisive Hierarchical Clustering Algorithm

1: Compute a minimum spanning tree for the proximity graph.
2: **repeat**
3: Create a new cluster by breaking the link corresponding to the largest distance (smallest similarity).
4: **until** Only singleton clusters remain

5.3 Hierarchical Clustering

5.3.4 BIRCH Algorithm

BIRCH

- BIRCH = Balanced Iterative Reducing and Clustering using Hierarchies

- BIRCH introduces two concepts
  - Clustering Feature (CF)
  - Clustering Feature Tree (CF Tree)

- CF and CF trees summarize the inherent clustering structures of the data

Clustering Features

- The clustering feature of a cluster is a triple <N, LS, SS>
  - N is the number of points in the cluster
  - LS is the sum of the points in the cluster

$$LS = \mathbf{x}_1 + \mathbf{x}_2 + \cdots + \mathbf{x}_N$$

  - SS is the square sum of the points in the cluster

$$SS = \mathbf{x}_1^2 + \mathbf{x}_2^2 + \cdots + \mathbf{x}_N^2$$

Additivity of Clustering Features

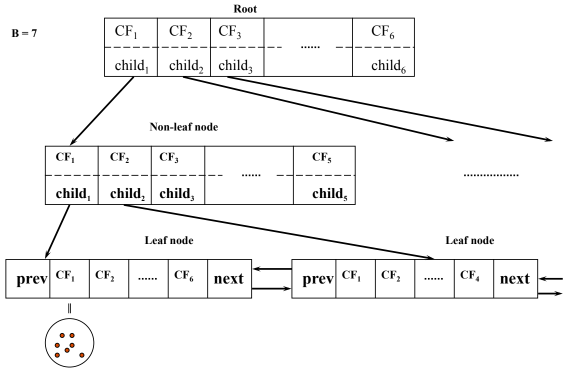
- Clustering features are additive
  - The clustering feature of C_1 is CF_1
  - The clustering feature of C_2 is CF_2
  - C_1 and C_2 are disjoint.

  - The clustering feature of $C_1 \cup C_2$ is $CF_1 + CF_2$

Clustering Feature Trees

- A clustering feature (CF) tree is a balanced tree that stores the clustering features for a hierarchical clustering
  - A non-leaf node has children
  - A non-leaf node stores the sum of the CFs of its children
- A CF tree has two parameters
  - Branching factor (B): the maximum number of children that a non-leaf node can have
  - Threshold (T): the maximum diameter of the sub-clusters stored at the leaf nodes

A CF Tree



BIRCH Algorithm

- **Step 1:** Scan the data to build an initial in-memory CF tree
- **Step 2:** Use an arbitrary clustering algorithm to cluster the leaf nodes of the CF-tree
- **Step 3:** Scan the data again and assign the data points using the cluster centers found in the previous step as seeds
- Time complexity: O(n), where n is the number of points

Limitations of BIRCH

- BIRCH does not work in non-Euclidean spaces
  - There is no average of a set of points
- BIRCH is inaccurate in case of non-globular clusters

5.3 Hierarchical Clustering

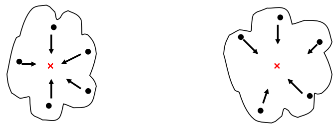
5.3.4 CURE Algorithm

CURE: Clustering Using REpresentatives

- The CURE algorithm does not assume anything about the shape of clusters.
  - Clusters need not be normally distributed.
  - They can even have strange bends, S-shapes, or even rings
- Instead of representing clusters by their centroid, it uses a collection of representative points.

Representative Points of a Cluster

- Uses a number of points to represent a cluster



- 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

Representative Points of a Cluster

- Shrinking representative points toward the center helps avoid problems with noise and outliers
- CURE is better able to handle clusters of arbitrary shapes and sizes

CURE Step 1: Initialization

- Take a small sample of the data and cluster it in main memory using a hierarchical method in which clusters are merged when they have a close pair of points (MIN).
- Select a small set of points from each cluster to be representative points. These points should be chosen to be as far from one another as possible.
- Move each of the representative points a fixed fraction (say 20%) of the distance between its location and the centroid of its cluster.

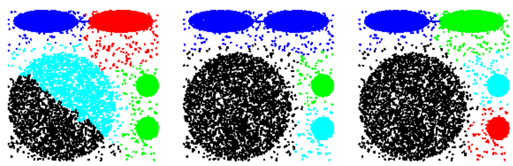
CURE Step 2: Merging Clusters

- Merge two clusters if they have a pair of representative points, one from each cluster, that are sufficiently close.

CURE Step 3: Point Assignment

- Each point p is brought from secondary storage and compared with the representative points.
- We assign p to the cluster of the representative point that is closest to p.

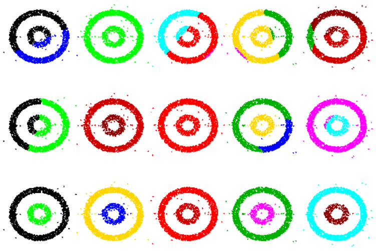
Experimental Results: CURE



a) BIRCH b) MST METHOD c) CURE

Picture from *CURE*, Guha, Rastogi, Shim.

Experimental Results: CURE



a) BIRCH

(centroid)

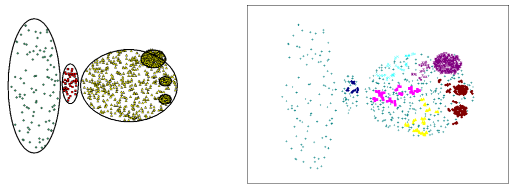
b) MST METHOD

(single link)

c) CURE

Picture from *CURE*, Guha, Rastogi, Shim.

CURE Cannot Handle Differing Densities



Original Points

CURE

5.3 Hierarchical Clustering

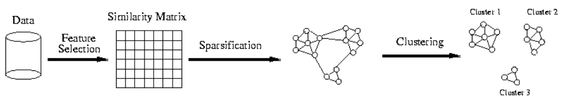
5.3.5 Graph-based Perspective

Proximity Graphs

- View the clustering process from the perspective of 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
  - Initially the proximity graph is fully connected
  - MIN (single-link) and MAX (complete-link) can be viewed as starting with this graph

Sparsifying Proximity Graphs

- Eliminate edges with low similarity weights
- In the simplest case, clusters are connected components in the sparcified graph



Why Sparsifying Proximity Graphs?

- The amount of data that needs to be processed is drastically reduced
  - Sparsification can eliminate more than 99% of the entries in a proximity matrix
  - The amount of time required to cluster the data is drastically reduced
  - The size of the problems that can be handled is increased

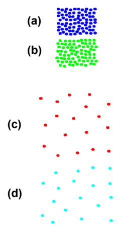
Why Sparsifying Proximity Graphs?

- Clustering may work better
  - Sparsification techniques keep the connections to the most similar (nearest) neighbors of a point while breaking the connections to less similar points.
  - The nearest neighbors of a point tend to belong to the same class as the point itself.
  - This reduces the impact of noise and outliers and sharpens the distinction between clusters.
- Sparsification facilitates the use of graph partitioning algorithms or algorithms based on graph partitioning algorithms.

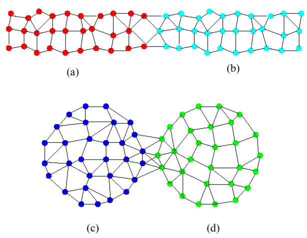
Limitations of Current Merging Schemes

- Existing merging schemes in hierarchical clustering algorithms are static in nature
  - MIN or CURE:
    - merge two clusters based on their *closeness* (or minimum distance)
  - GROUP-AVERAGE:
    - merge two clusters based on their average *connectivity*

Limitations of Current Merging Schemes



Closeness schemes will merge (a) and (b)



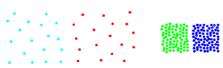
Average connectivity schemes will merge (c) and (d)

5.3 Hierarchical Clustering

5.3.6 Chameleon Algorithm

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 inter-connectivity of the cluster
  - Two clusters are combined if the resulting cluster shares certain *properties* with the constituent clusters
  - The merging scheme preserves *self-similarity*

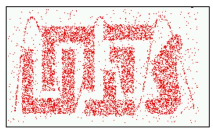


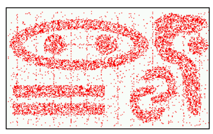
- One of the areas of application is spatial data

Characteristics of Spatial Data Sets

- **Clusters are defined as densely populated regions of the space**
- **Clusters have arbitrary shapes, orientation, and non-uniform sizes**
- **Difference in densities across clusters and variation in density within clusters**
- **Existence of special artifacts (*streaks*) and noise**

The clustering algorithm must address the above characteristics and also require minimal supervision.





Chameleon Algorithm

- Sparsification: 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)
- Graph Partitioning: 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 Algorithm

- Hierarchical Clustering: Use Hierarchical Agglomerative Clustering to merge sub-clusters
  - Two clusters are combined if the *resulting cluster shares certain properties with the constituent clusters*
  - Two key properties used to model cluster similarity:
    - Relative Interconnectivity: Absolute interconnectivity of two clusters normalized by the internal connectivity of the clusters
    - Relative Closeness: Absolute closeness of two clusters normalized by the internal closeness of the clusters

Merging Clusters

- Relative Interconnectivity (RI)

$$\boxed{RI(C_i, C_j) = \frac{2EC(C_i, C_j)}{EC(C_i) + EC(C_i)}}$$

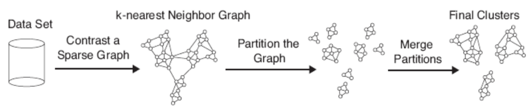
- Relative Closeness (RC)

$$\boxed{RC(C_i, C_j) = \frac{S_{EC}(C_i, C_j)}{\frac{m_i}{m_i + m_j} S_{EC}(C_i) + \frac{m_j}{m_i + m_j} S_{EC}(C_j)}}$$

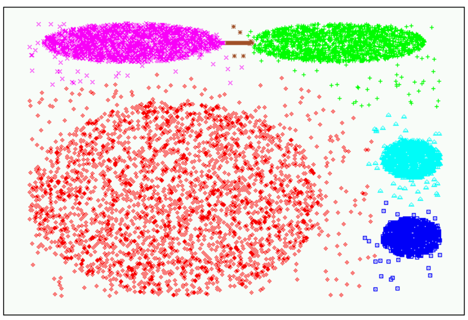
- Two clusters C_i and C_j are merged if

$$RI(C_i, C_j) RC(C_i, C_j) > t$$

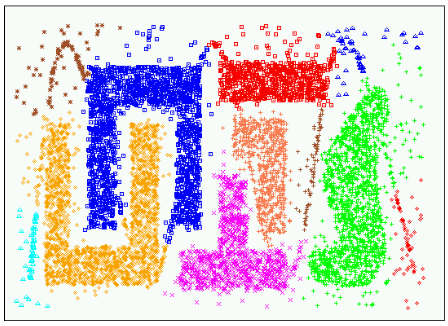
Chameleon Algorithm: Illustration



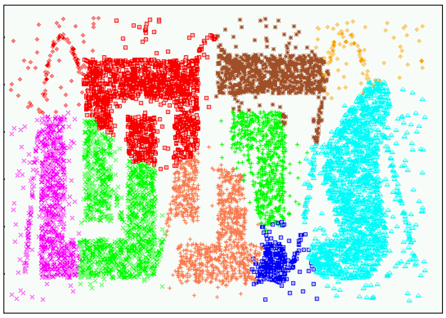
Experimental Results: CHAMELEON



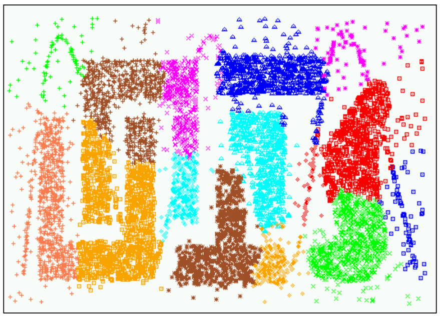
Experimental Results: CHAMELEON



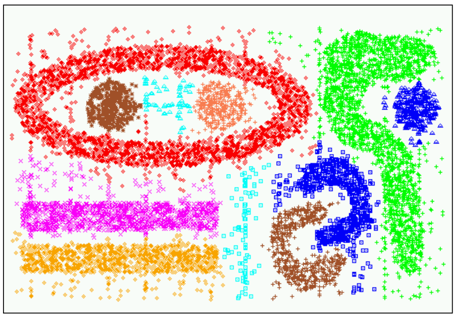
Experimental Results: CURE (*10 clusters*)



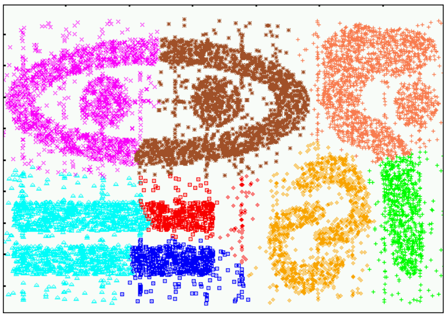
Experimental Results: CURE (*15 clusters*)



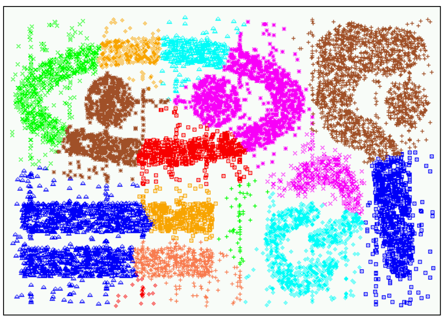
Experimental Results: CHAMELEON



Experimental Results: CURE (*9 clusters*)



Experimental Results: CURE (*15 clusters*)

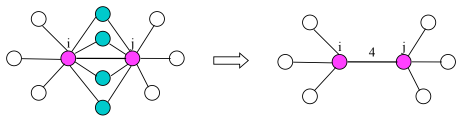


5.3 Hierarchical Clustering

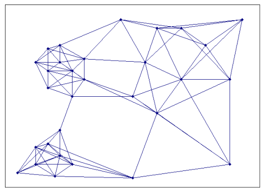
5.3.7 SNN Clustering

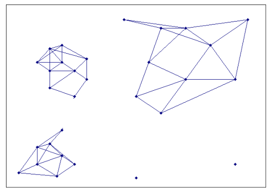
Shared Near Neighbor Approach

SNN graph: the weight of an edge is the number of shared neighbors between vertices given that the vertices are connected



Creating the SNN Graph





Sparse Graph

Link weights are similarities between neighboring points

Shared Near Neighbor Graph

Link weights are number of Shared Nearest Neighbors

ROCK (RObust Clustering using linKs)

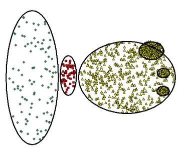
- Clustering algorithm for data with categorical and Boolean attributes
  - A pair of points is defined to be neighbors if their similarity is greater than some threshold
  - Use a hierarchical clustering scheme to cluster the data.

1. Obtain a sample of points from the data set
2. Compute the link value for each set of points, i.e., transform the original similarities (computed by Jaccard coefficient) into similarities that reflect the number of shared neighbors between points
3. Perform an agglomerative hierarchical clustering on the data using the "number of shared neighbors" as similarity measure and maximizing "the shared neighbors" objective function
4. Assign the remaining points to the clusters that have been found

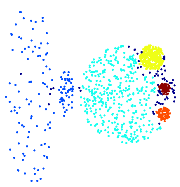
Jarvis-Patrick Clustering

- First, the k-nearest neighbors of all points are found
  - In graph terms this can be regarded as breaking all but the k strongest links from a point to other points in the proximity graph
- A pair of points is put in the same cluster if
  - any two points share more than T neighbors and
  - the two points are in each others k nearest neighbor list
- For instance, we might choose a nearest neighbor list of size 20 and put points in the same cluster if they share more than 10 near neighbors
- Jarvis-Patrick clustering is too brittle

When Jarvis-Patrick Works Reasonably Well



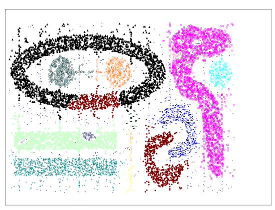
Original Points



Jarvis Patrick Clustering

6 shared neighbors out of 20

When Jarvis-Patrick Does NOT Work Well



Smallest threshold, T, that does not merge clusters.

Threshold of T - 1

SNN Clustering Algorithm

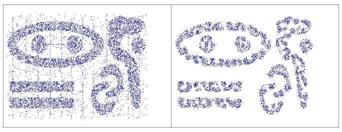
1. **Compute the similarity matrix**
   This corresponds to a similarity graph with data points for nodes and edges whose weights are the similarities between data points
2. **Sparsify the similarity matrix by keeping only the *k* most similar neighbors**
   This corresponds to only keeping the *k* strongest links of the similarity graph
3. **Construct the shared nearest neighbor graph from the sparsified similarity matrix.**
   At this point, we could apply a similarity threshold and find the connected components to obtain the clusters (Jarvis-Patrick algorithm)
4. **Find the SNN density of each Point.**
   Using a user specified parameters, *Eps*, find the number points that have an SNN similarity of *Eps* or greater to each point. This is the SNN density of the point

SNN Clustering Algorithm …

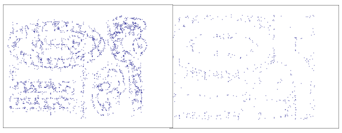
5. **Find the core points**
   Using a user specified parameter, *MinPts*, find the core points, i.e., all points that have an SNN density greater than *MinPts*
6. **Form clusters from the core points**
   If two core points are within a radius, *Eps*, of each other they are place in the same cluster
7. **Discard all noise points**
   All non-core points that are not within a radius of *Eps* of a core point are discarded
8. **Assign all non-noise, non-core points to clusters**
   This can be done by assigning such points to the nearest core point

(Note that steps 4-8 are DBSCAN)

SNN Density

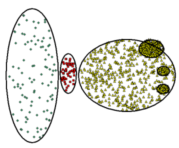


a) All Points　　b) High SNN Density

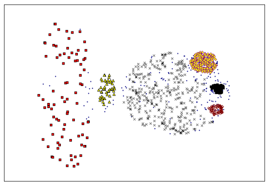


c) Medium SNN Density　　d) Low SNN Density

SNN Clustering Can Handle Differing Densities

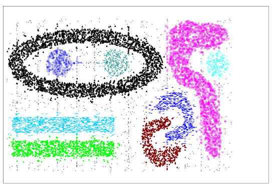


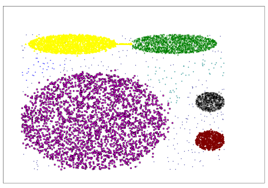
Original Points



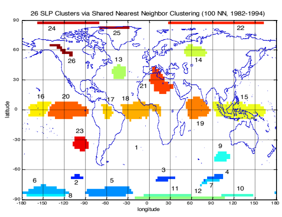
SNN Clustering

SNN Clustering Can Handle Other Difficult Situations

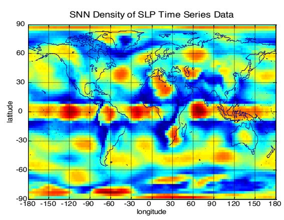




Finding Clusters of Time Series In Spatio-Temporal Data



SNN Clusters of SLP.



SNN Density of Points on the Globe.

Features and Limitations of SNN Clustering

- Does not cluster all the points

- Complexity of SNN Clustering is high
  - O(n * time to find numbers of neighbor within *Eps*)
  - In worst case, this is $O(n^2)$
  - For lower dimensions, there are more efficient ways to find the nearest neighbors
    - R* Tree
    - k-d Trees