

# **Data Mining Cluster Analysis: Advanced Concepts and Algorithms**

---

Lecture Notes for Chapter 9

Introduction to Data Mining

by

Tan, Steinbach, Kumar

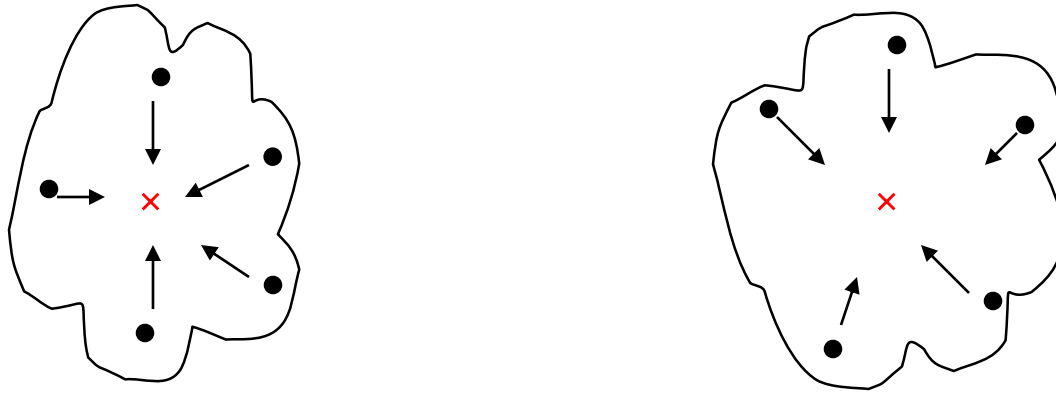
# Hierarchical Clustering: Revisited

---

- Creates nested clusters
- Agglomerative clustering algorithms vary in terms of how the proximity of two clusters are computed
  - ◆ MIN (single link): susceptible to noise/outliers
  - ◆ MAX/GROUP AVERAGE:
    - may not work well with non-globular clusters
  - CURE algorithm tries to handle both problems
- Often starts with a proximity matrix
  - A type of graph-based algorithm

# CURE: Another Hierarchical Approach

- 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

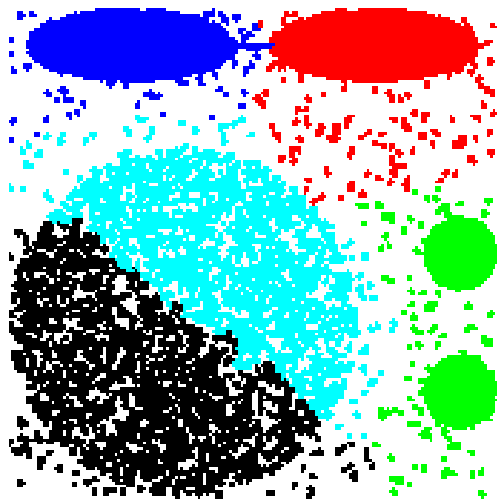
# CURE

---

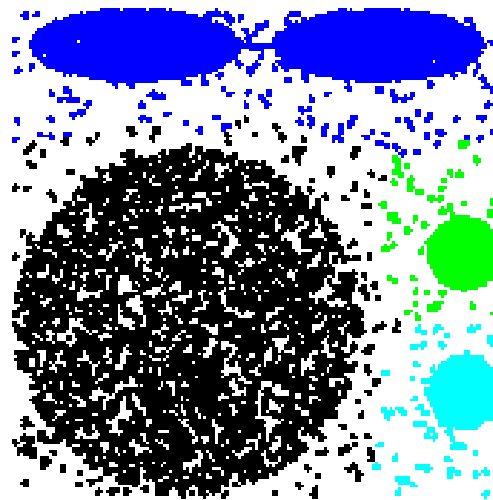
- Shrinking representative points toward the center helps avoid problems with noise and outliers
- CURE is better able to handle clusters of arbitrary shapes and sizes

# Experimental Results: CURE

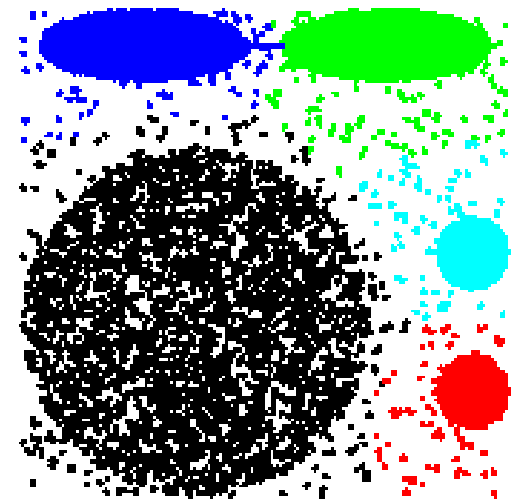
---



a) BIRCH



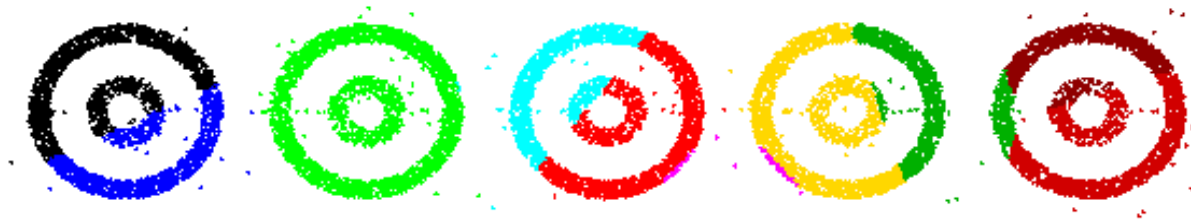
b) MST METHOD



c) CURE

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

# Experimental Results: CURE



a) BIRCH

(centroid)



b) MST METHOD

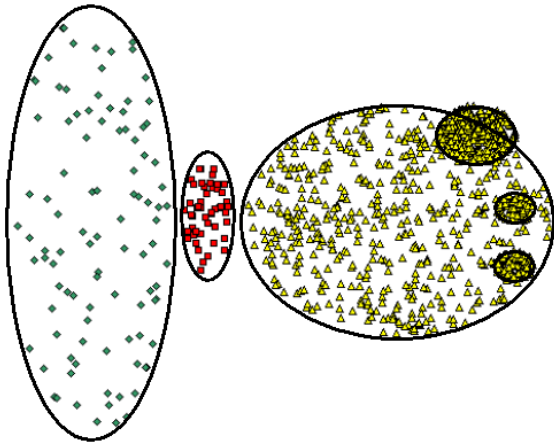
(single link)



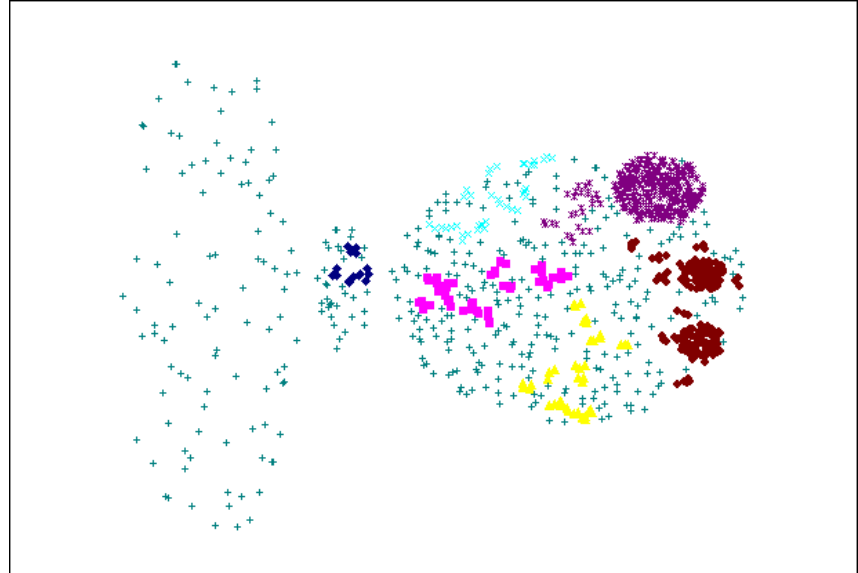
c) CURE

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

# CURE Cannot Handle Differing Densities



**Original Points**



**CURE**

# Graph-Based Clustering

---

- 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
  - Initially the proximity graph is fully connected
  - MIN (single-link) and MAX (complete-link) can be viewed as starting with this graph
  
- In the simplest case, clusters are connected components in the graph.



# Graph-Based Clustering: Sparsification

---

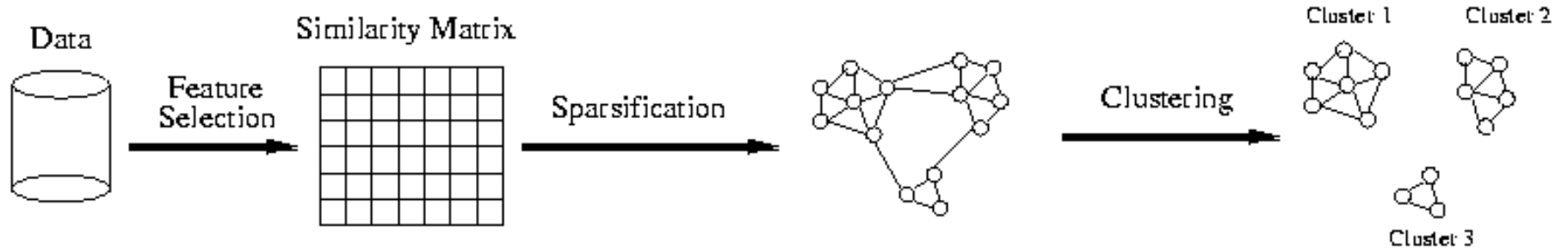
- The amount of data that needs to be processed is drastically reduced
  - Sparsification can eliminate more than 99% of the entries in a proximity matrix
  - The amount of time required to cluster the data is drastically reduced
  - The size of the problems that can be handled is increased

# Graph-Based Clustering: Sparsification ...

---

- 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).
  - Chameleon and Hypergraph-based Clustering

# Sparsification in the Clustering Process

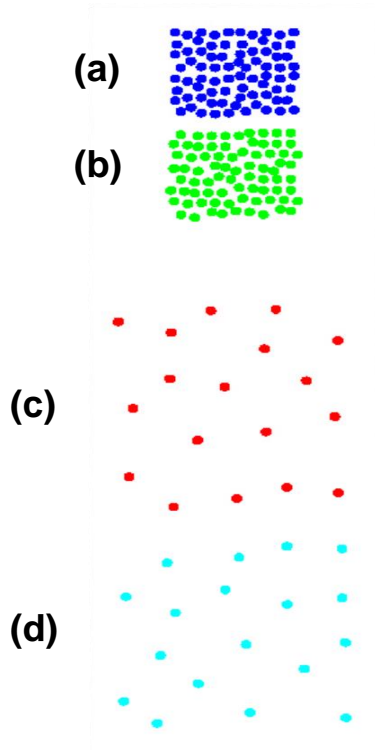


# Limitations of Current Merging Schemes

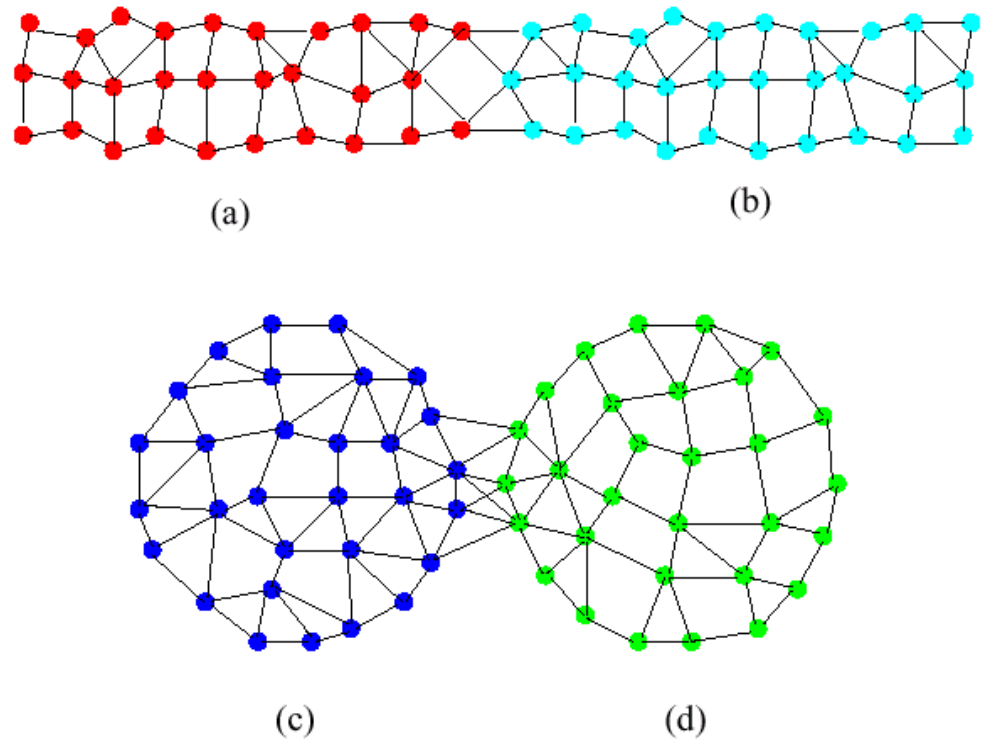
---

- 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)**

# 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*

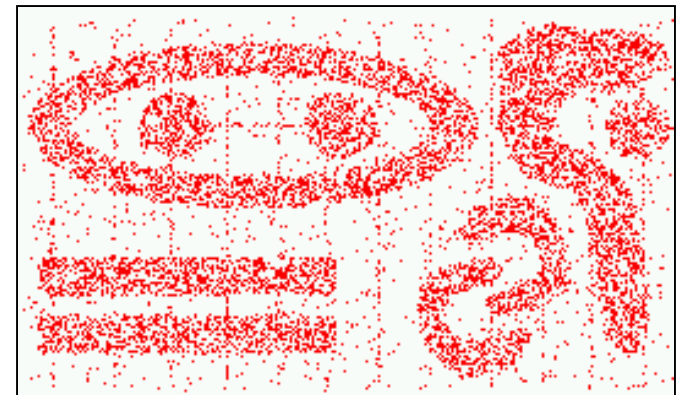
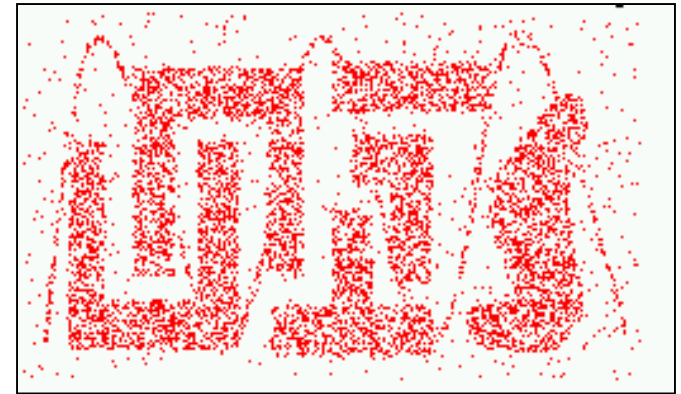


- One of the areas of application is *spatial data*

# 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: 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

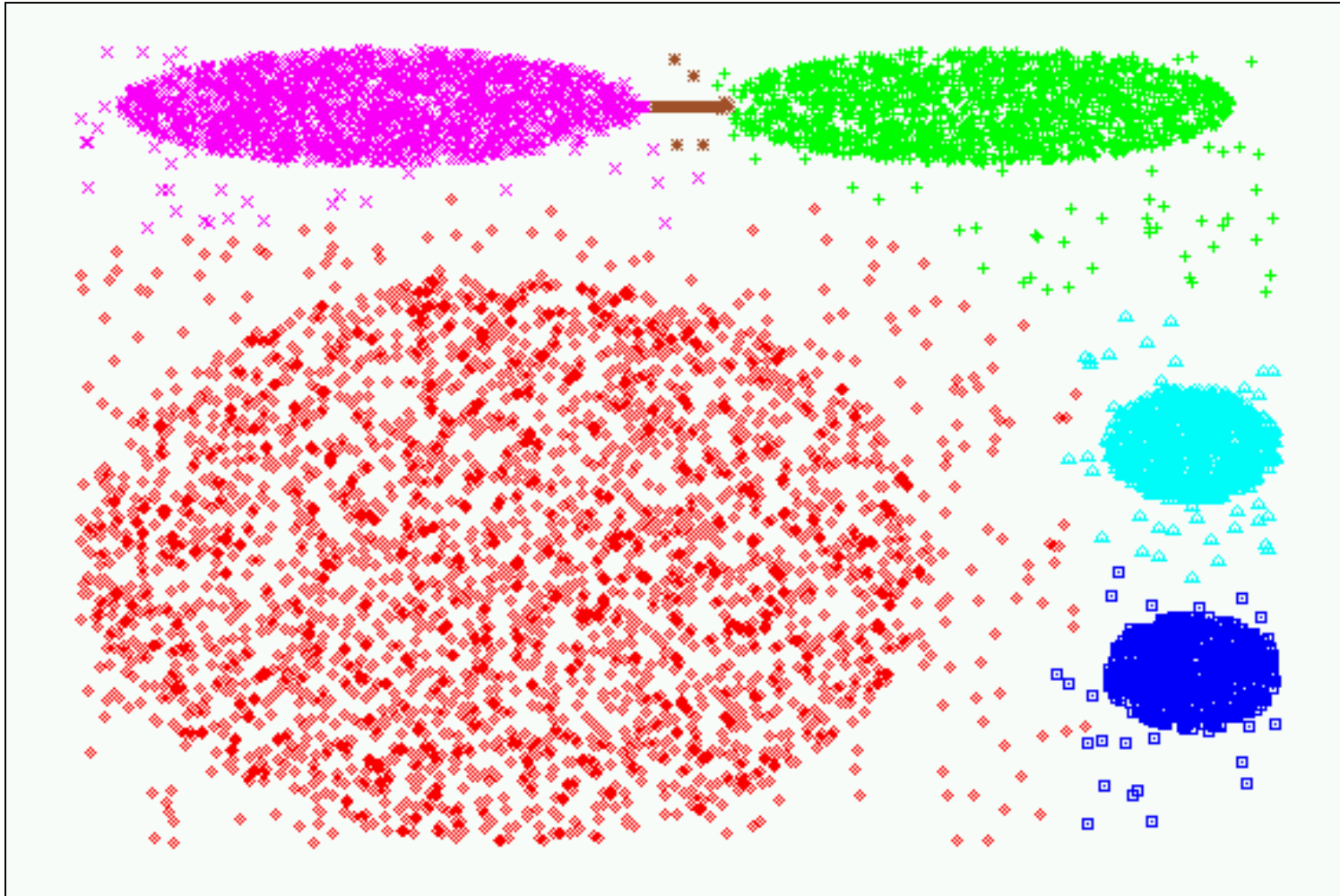


# Chameleon: Steps ...

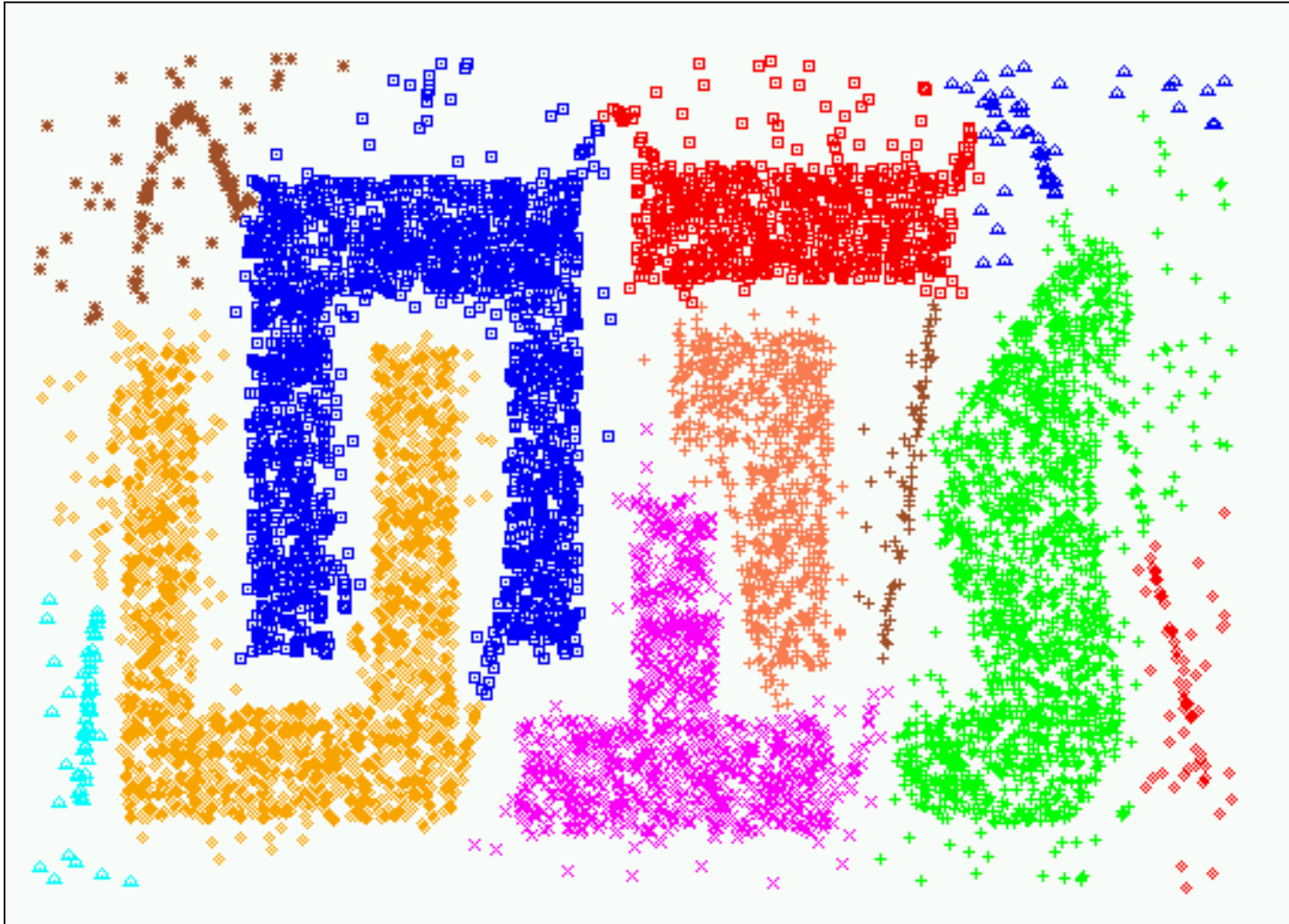
---

- **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*
  - 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

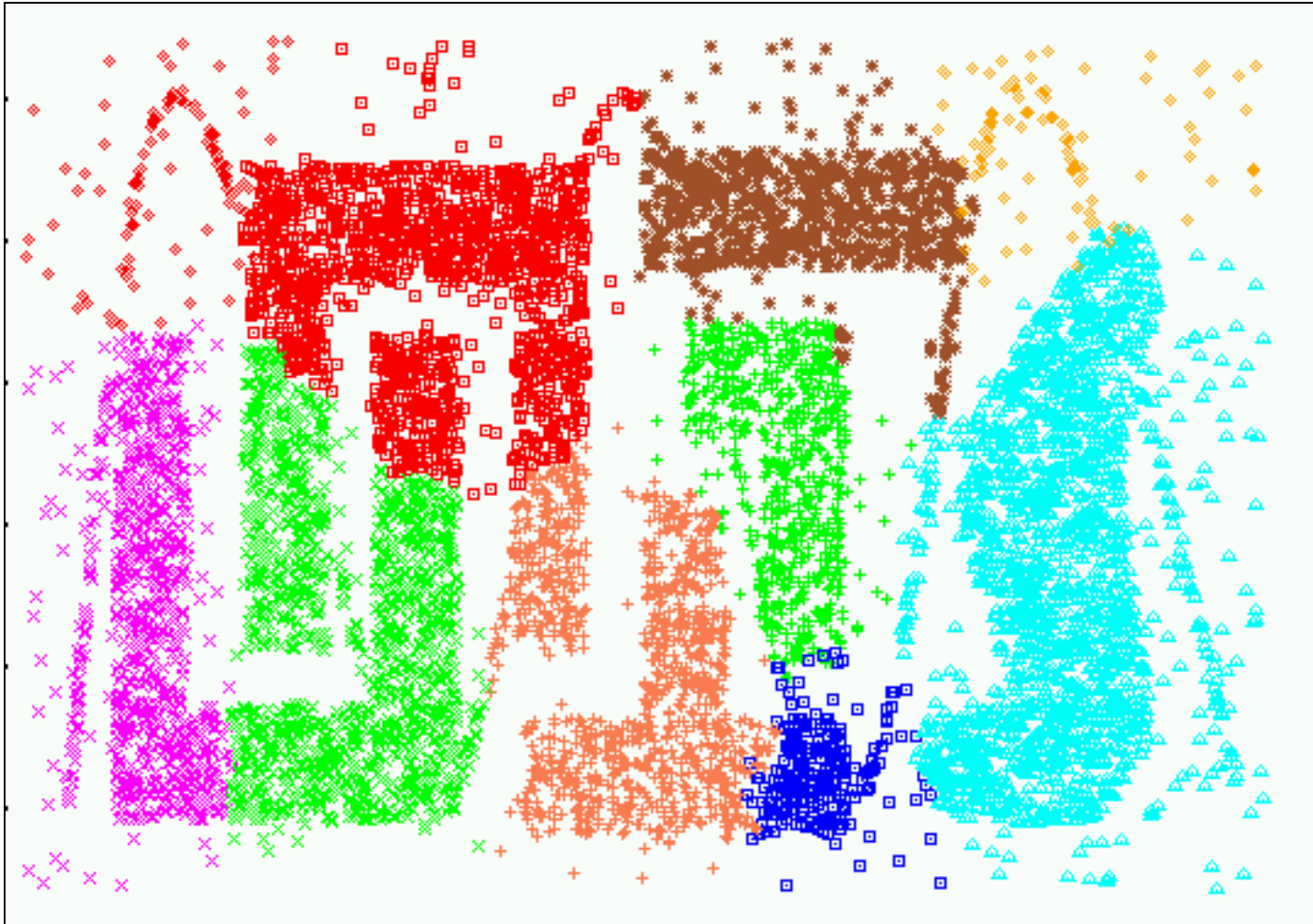
# Experimental Results: CHAMELEON



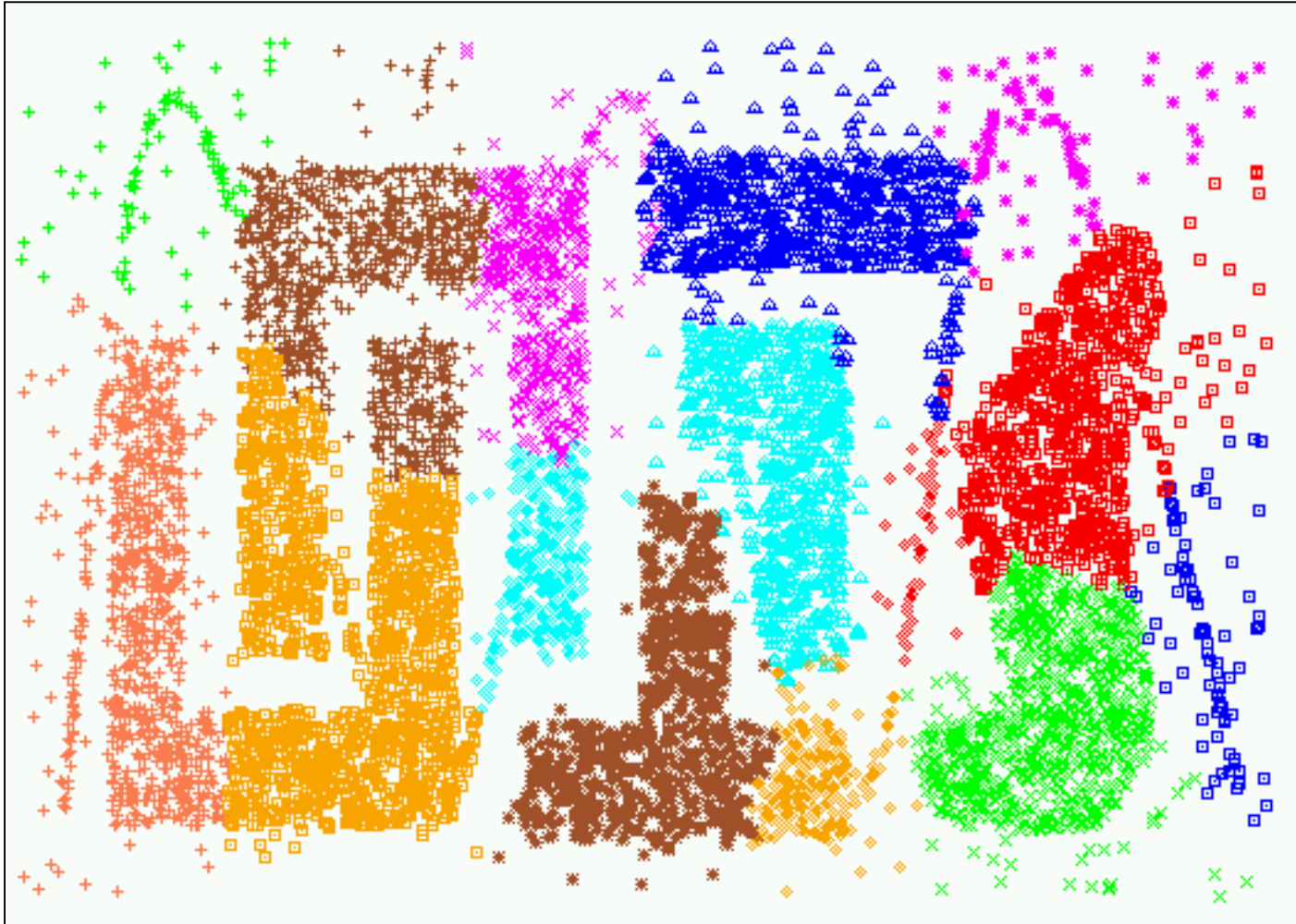
# Experimental Results: CHAMELEON



# Experimental Results: CURE (*10 clusters*)

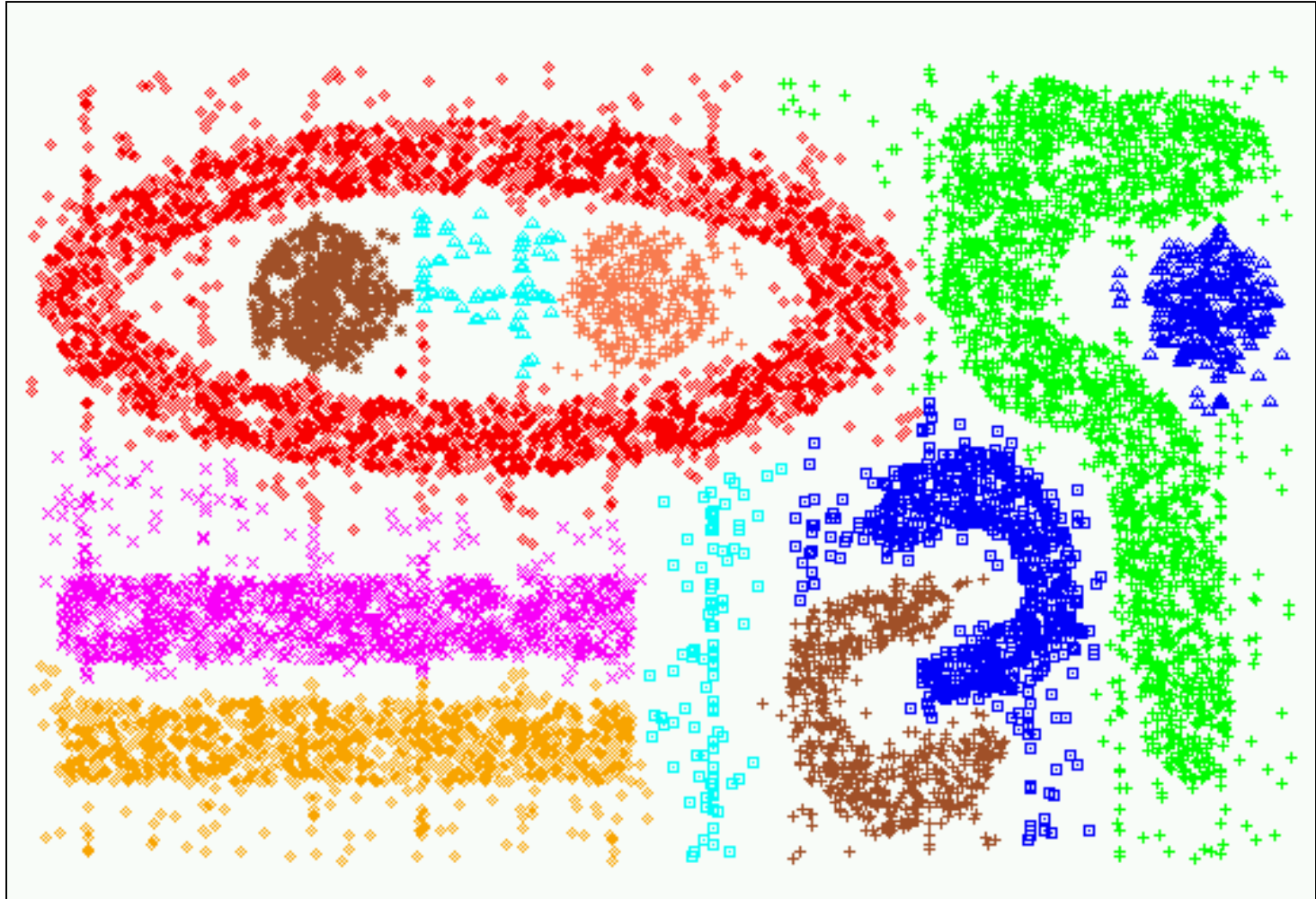


# Experimental Results: CURE (*15 clusters*)

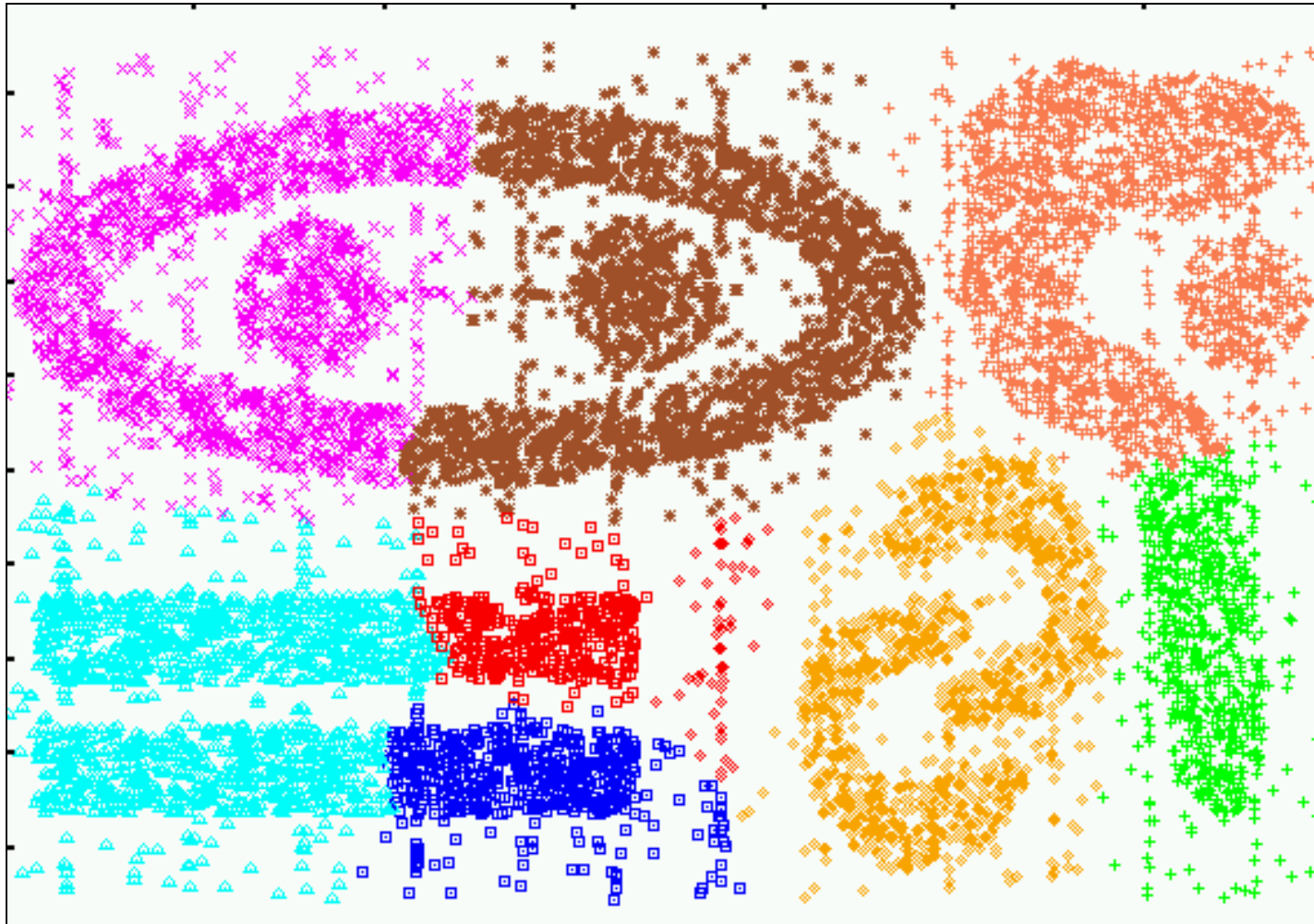




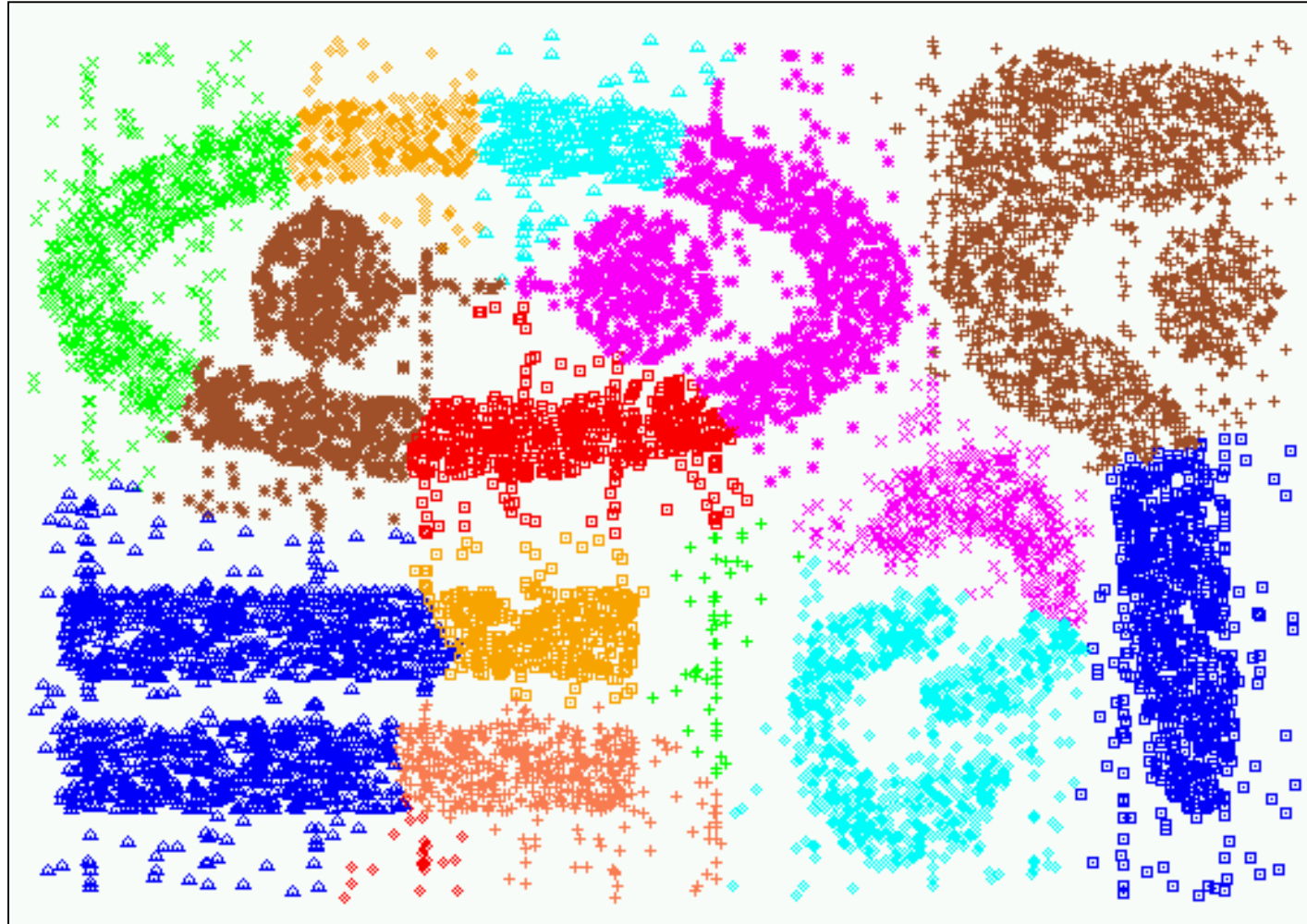
# Experimental Results: CHAMELEON



# Experimental Results: COKE (9 clusters)



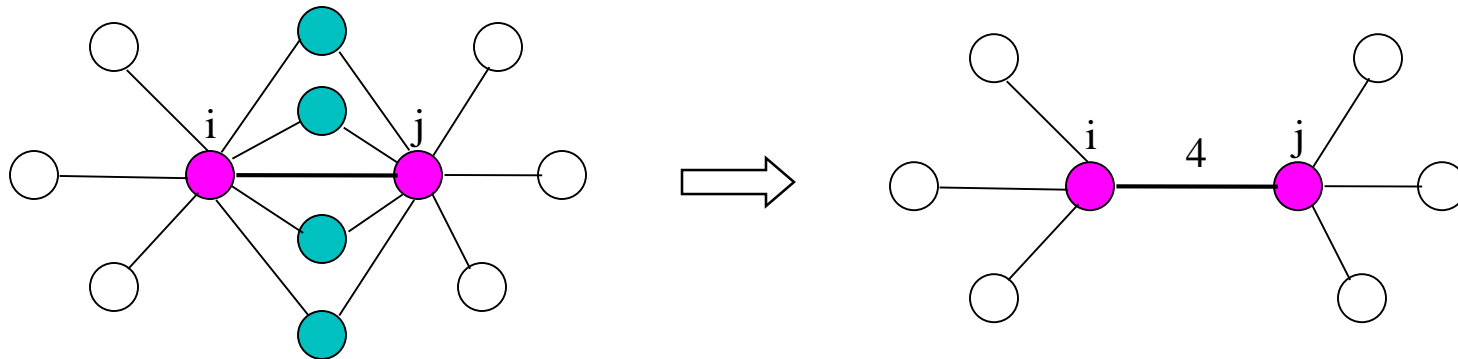
# Experimental Results: CURE ( *15 clusters* )



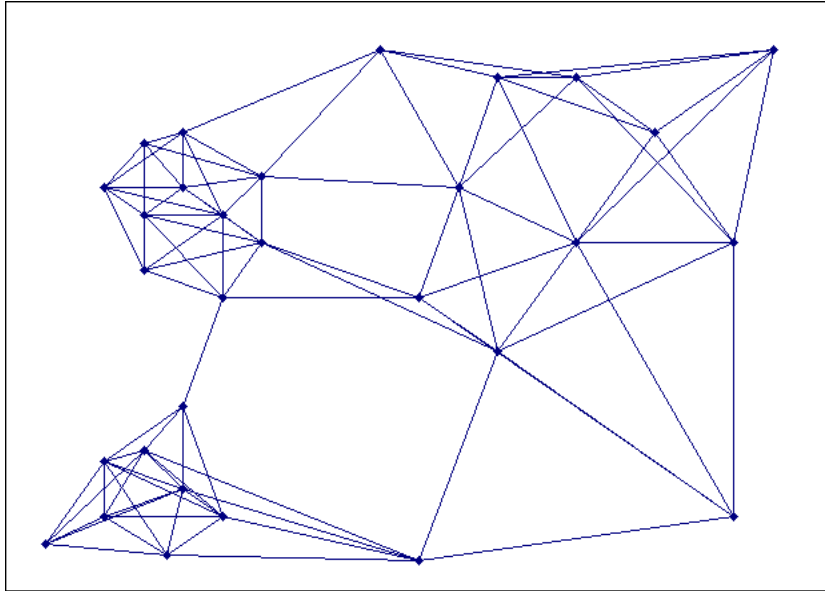


# Shared Near Neighbor Approach

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

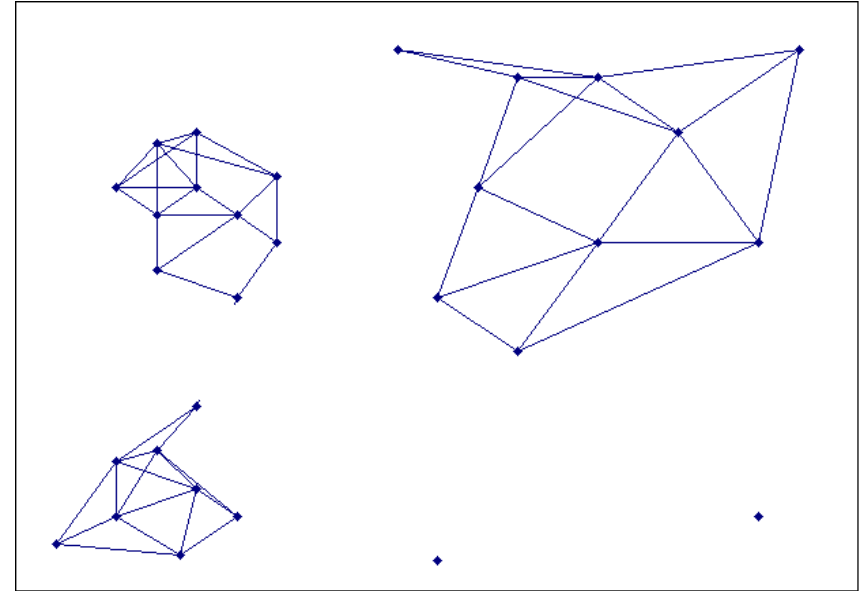


# Creating the SNN Graph



**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)

---

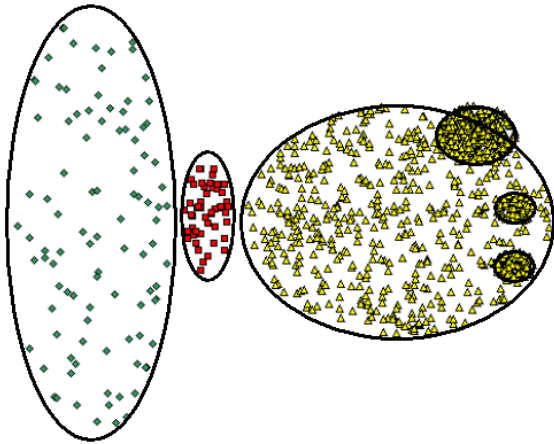
- Clustering algorithm for data with categorical and Boolean attributes
  - A pair of points is defined to be neighbors if their similarity is greater than some threshold
  - Use a hierarchical clustering scheme to cluster the data.
- 1. Obtain a sample of points from the data set
- 2. Compute the link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

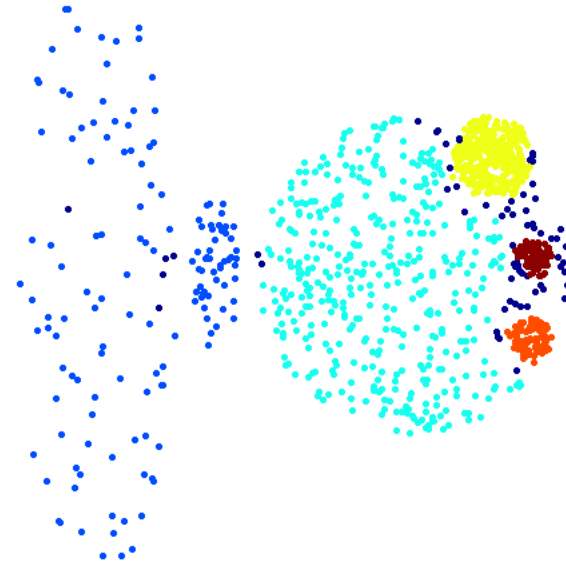
---

- 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



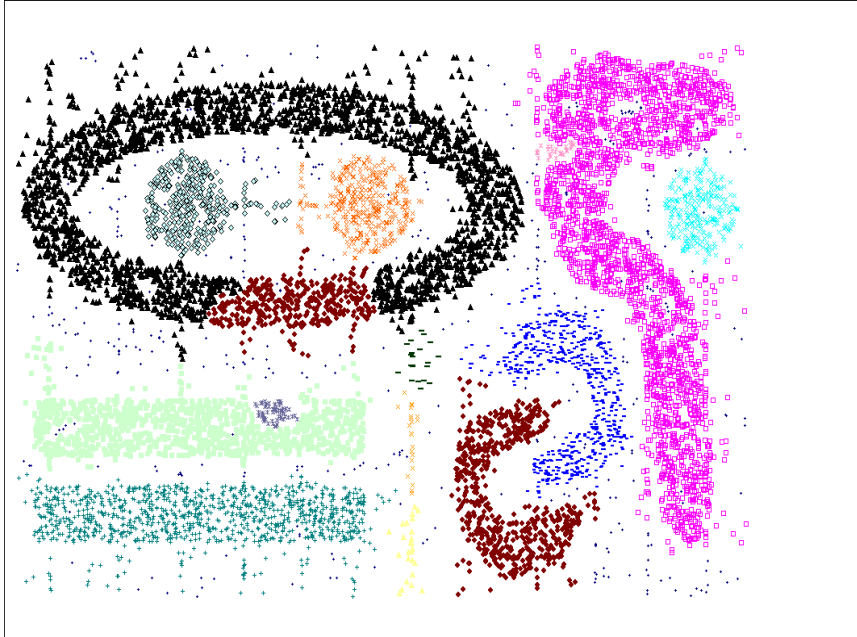
**Original Points**



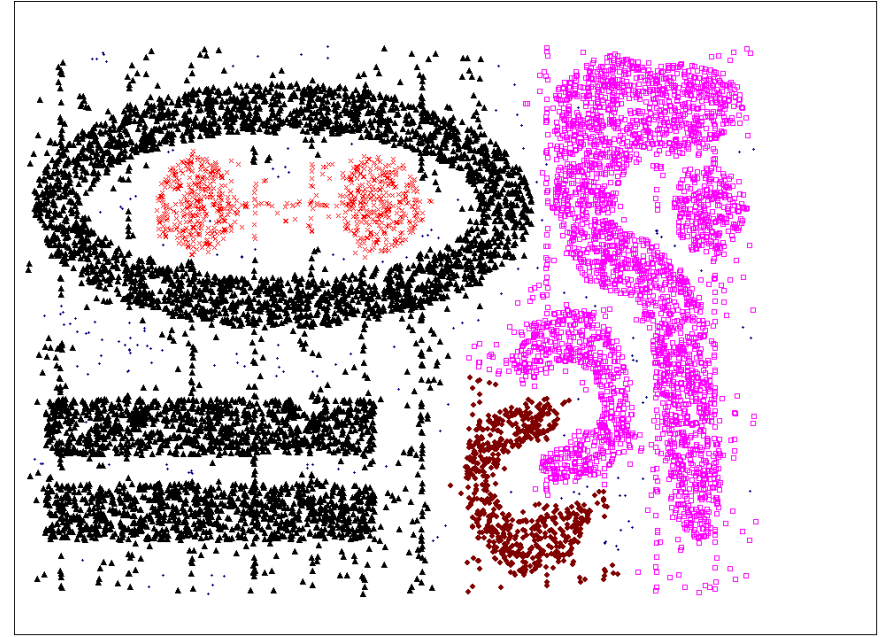
**Jarvis Patrick Clustering**

**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 placed 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

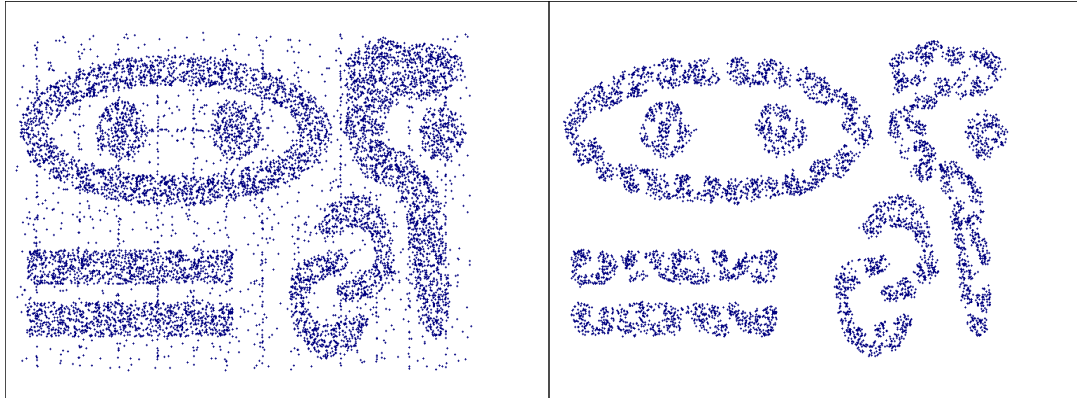
## 8. Assign all non-noise, non-core points to clusters

This can be done by assigning such points to the nearest core point

(Note that steps 4-8 are DBSCAN)

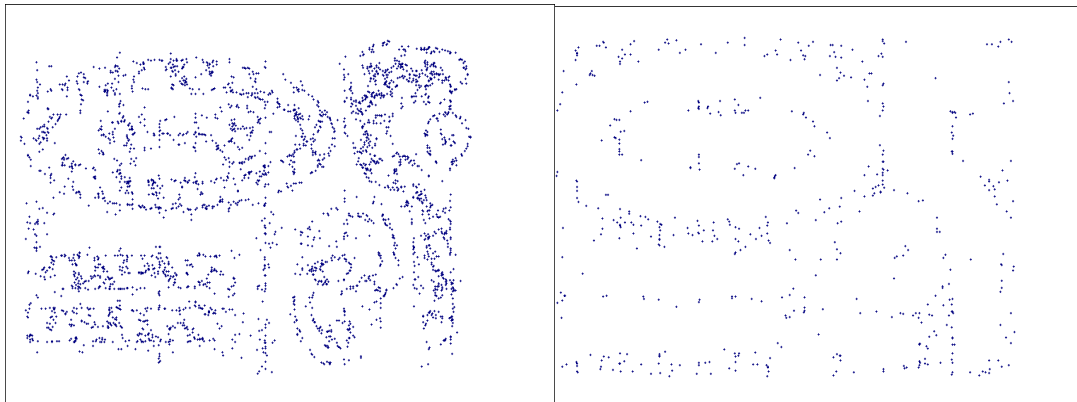


# SNN Density



**a) All Points**

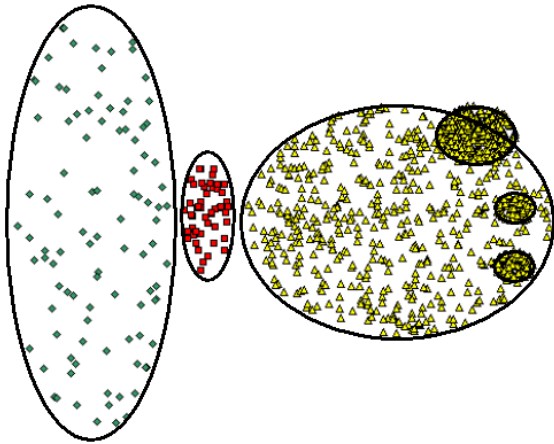
**b) High SNN Density**



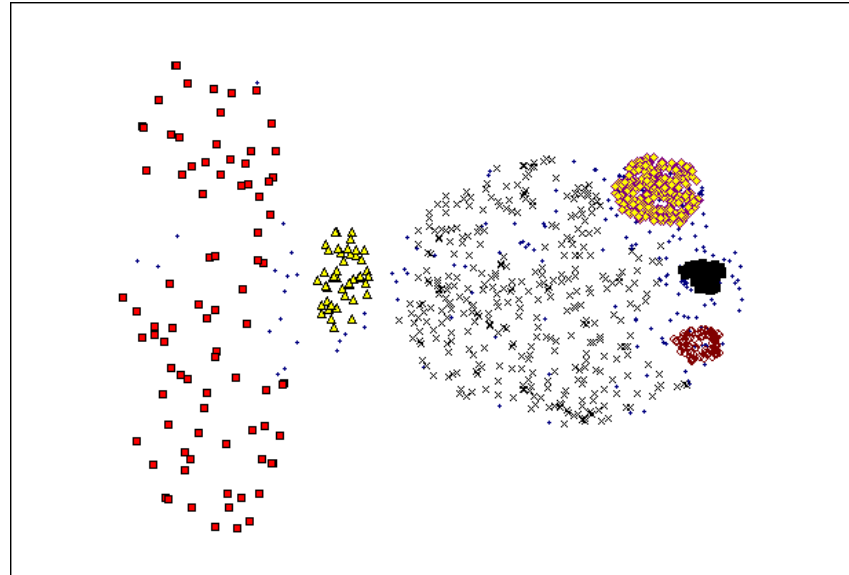
**c) Medium SNN Density**

**d) Low SNN Density**

# SNN Clustering Can Handle Differing Densities

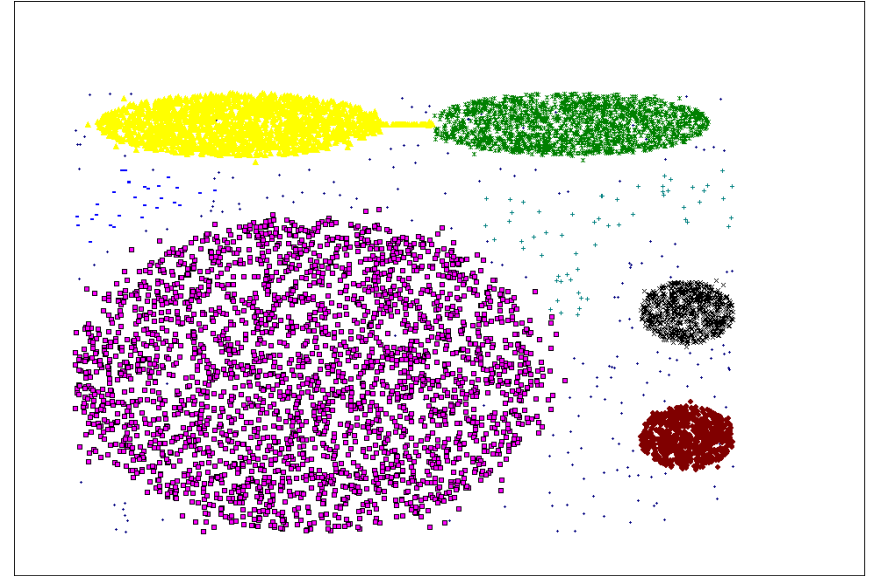
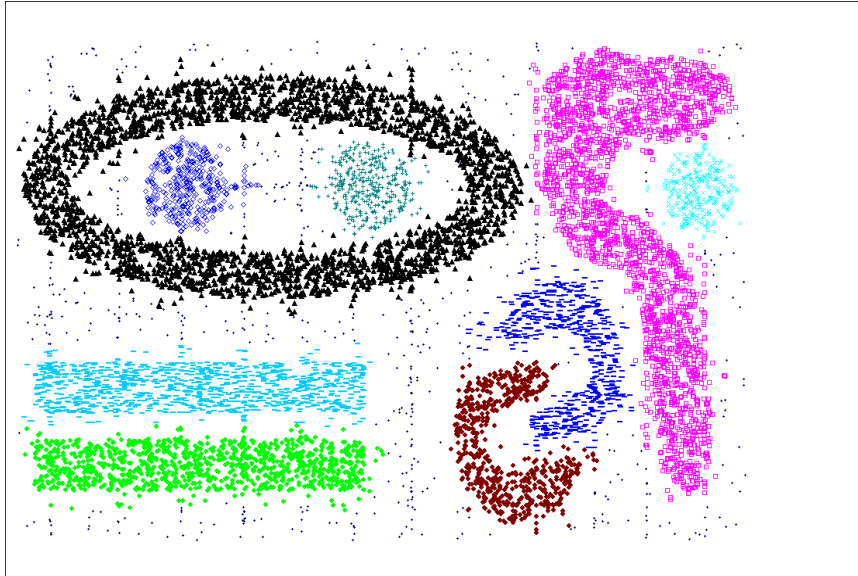


**Original Points**

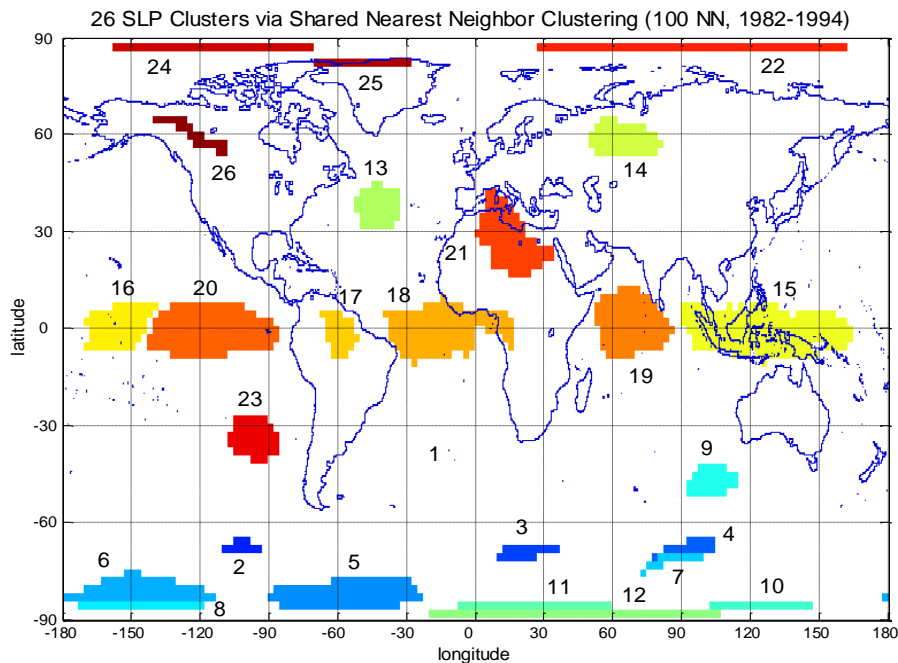


**SNN Clustering**

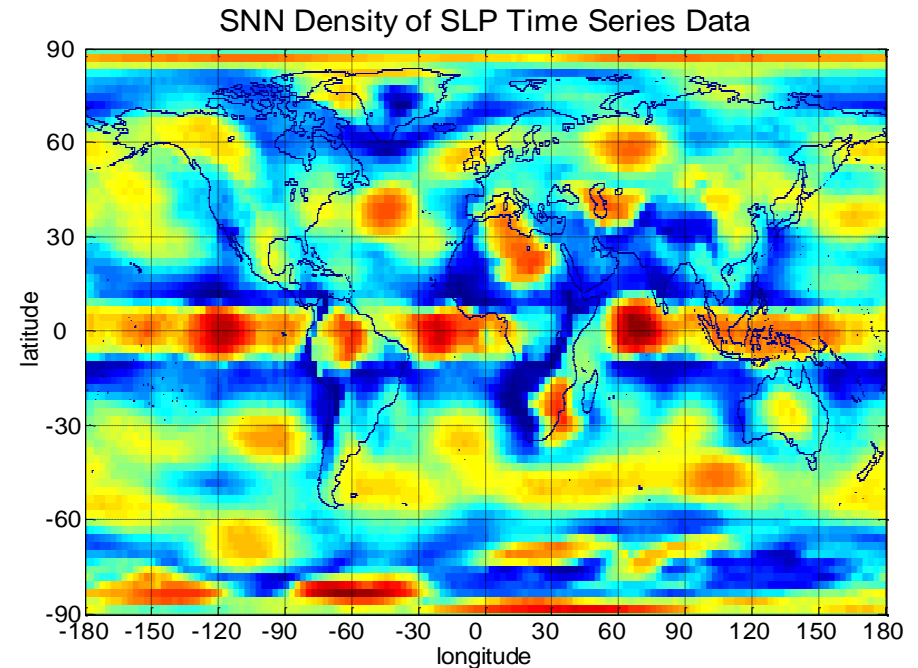
# SNN Clustering Can Handle Other Difficult Situations



# 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
  - $O(n * \text{time to find numbers of neighbor within } Eps)$
  - 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