Data Mining
Cluster Analysis: Advanced Concepts and Algorithms

Lecture Notes for Chapter 8

Introduction to Data Mining

by

Tan, Steinbach, Kumar

Outline

- Prototype-based
  - Fuzzy c-means
  - Mixture Model Clustering
  - Self-Organizing Maps
- Density-based
  - Grid-based clustering
  - Subspace clustering: CLIQUE
  - Kernel-based: DENCLUE
- Graph-based
  - Chameleon
  - Jarvis-Patrick
  - Shared Nearest Neighbor (SNN)
- Characteristics of Clustering Algorithms

Hard (Crisp) vs Soft (Fuzzy) Clustering

Hard (Crisp) vs. Soft (Fuzzy) clustering

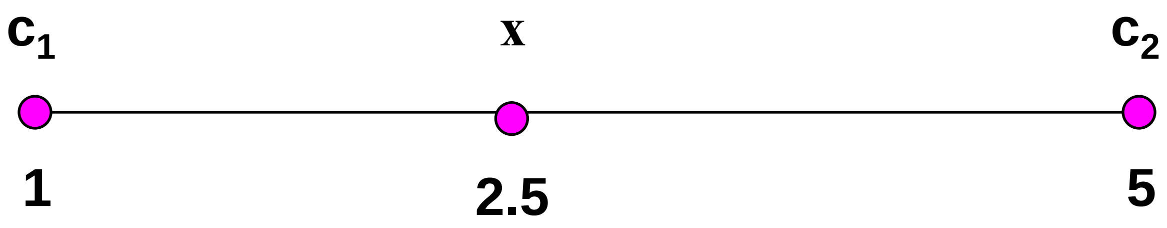
- For soft clustering allow point to belong to more than one cluster
- For K-means, generalize objective function

$$SSE = \sum_{j=1}^{k} \sum_{i=1}^{m} w_{ij}\, dist(\boldsymbol{x}_i, \boldsymbol{c}_j)^2 \qquad \sum_{j=1}^{k} w_{ij} = 1$$

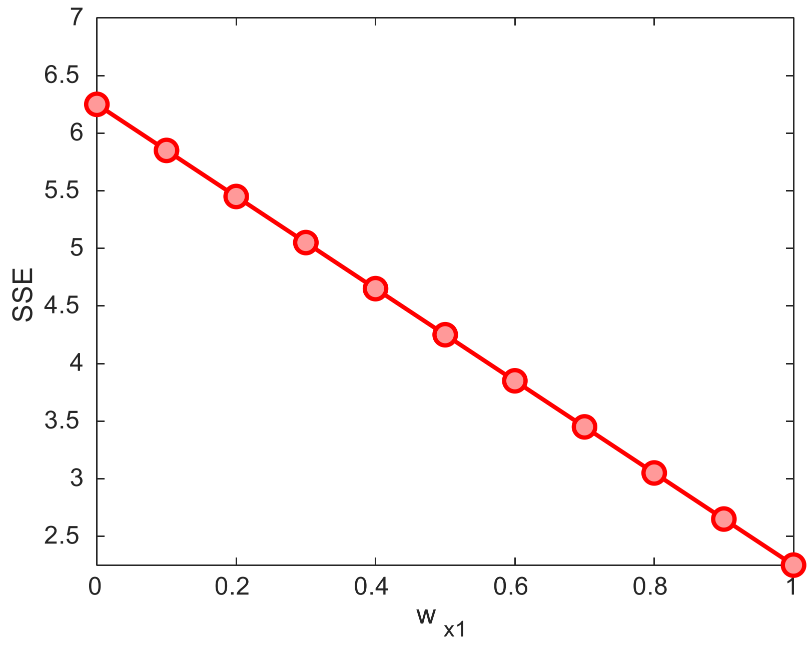
w_{ij}: weight with which object x_i belongs to cluster $\boldsymbol{c_j}$

- To minimize SSE, repeat the following steps:
  - Fix $\boldsymbol{c_j}$ and determine w_{ij} (cluster assignment)
  - Fix w_{ij} and recompute $\boldsymbol{c_j}$
- Hard clustering: $w_{ij} \in \{0,1\}$

Soft (Fuzzy) Clustering: Estimating Weights



$$SSE = w_{x1}(2.5-1)^2 + w_{x2}(5-2.5)^2$$
$$= 2.25w_{x1} + 6.25w_{x2}$$



SSE(x) has a minimum value of 2.25 when w_{x1} = 1, w_{x2} = 0

Fuzzy C-means

Objective function

p: fuzzifier (p > 1)

$$SSE = \sum_{j=1}^{k}\sum_{i=1}^{m} w_{ij}^{p}\, dist(\boldsymbol{x}_i, \boldsymbol{c}_j)^2 \qquad \sum_{j=1}^{k} w_{ij} = 1$$

- w_{ij}: weight with which object $\boldsymbol{x}_i$ belongs to cluster $\boldsymbol{c}_j$
- p: is a power for the weight not a superscript and controls how "fuzzy" the clustering is

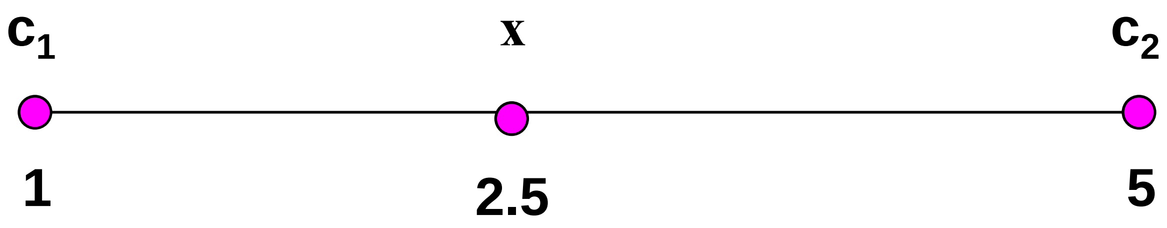
– To minimize objective function, repeat the following:

- Fix $\boldsymbol{c}_j$ and determine w_{ij}
- Fix w_{ij} and recompute $\boldsymbol{c}$

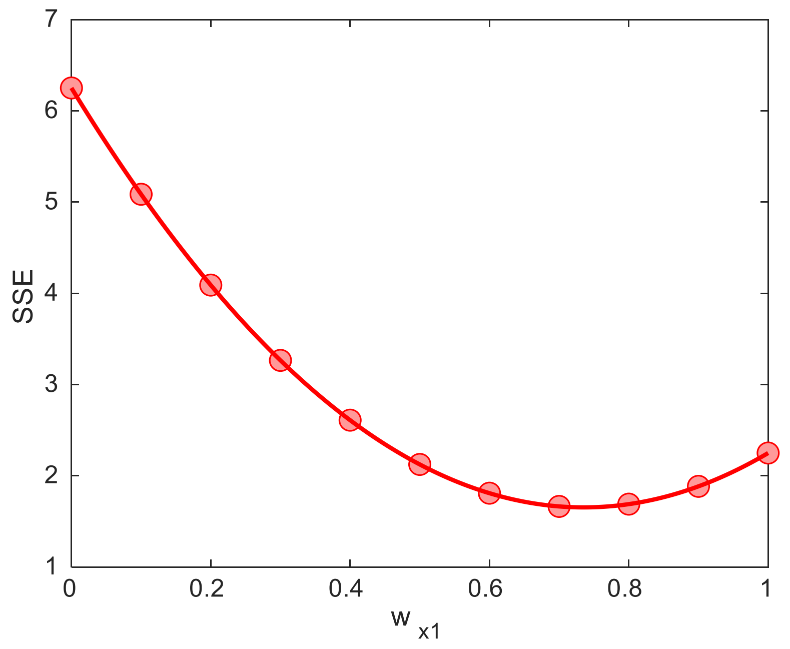
– Fuzzy c-means clustering: $w_{ij} \in [0,1]$

Bezdek, James C. *Pattern recognition with fuzzy objective function algorithms*. Kluwer Academic Publishers, 1981.

Fuzzy C-means



$$SSE = w_{x1}^2(2.5-1)^2 + w_{x2}^2(5-2.5)^2$$
$$= 2.25w_{x1}^2 + 6.25w_{x2}^2$$



SSE(x) has a minimum value of 1.654 when w_{x1} = 0.74, w_{x2} = 0.36

Fuzzy C-means

Objective function:

p: fuzzifier (p > 1)

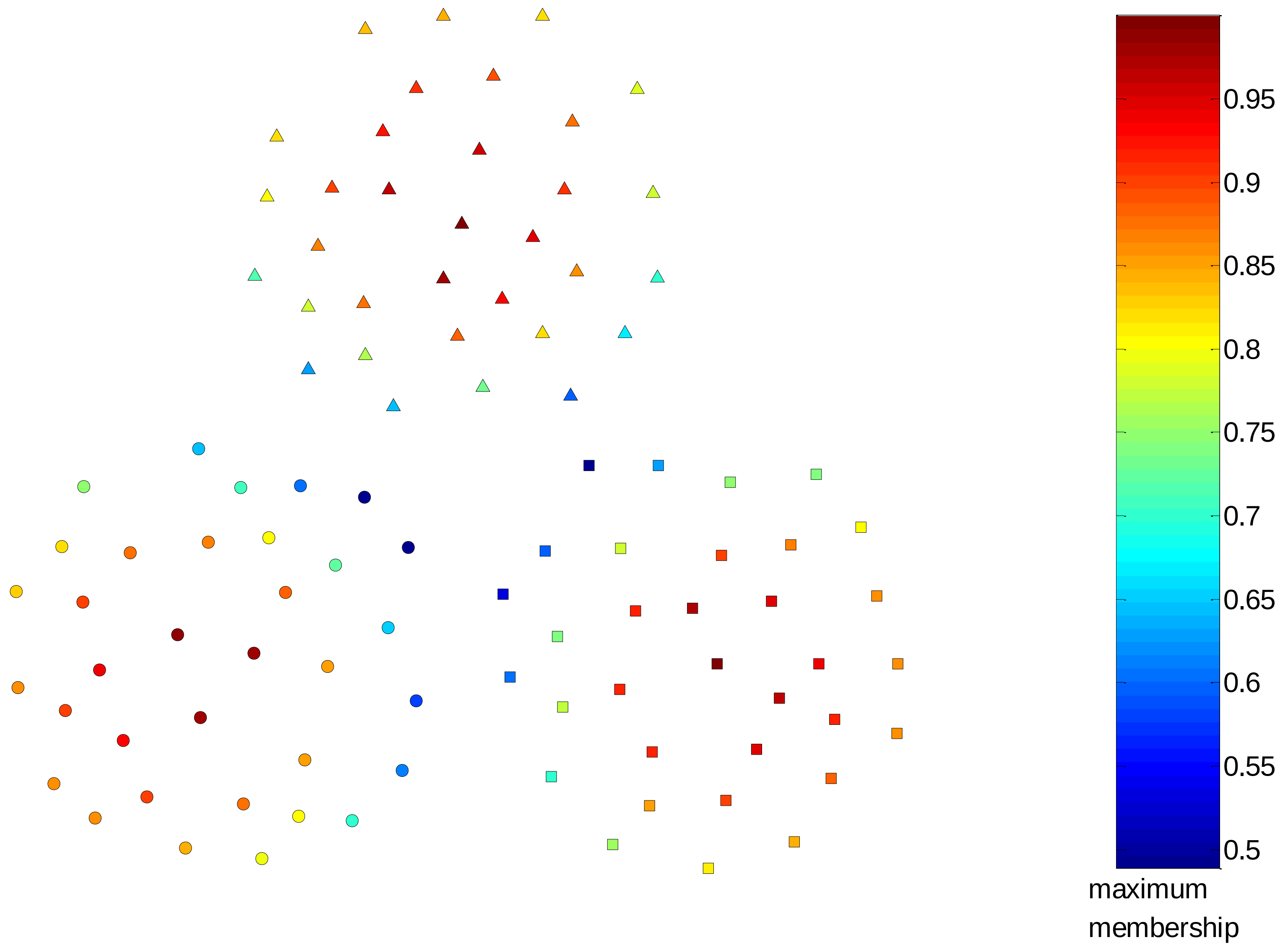
$$SSE = \sum_{j=1}^{k} \sum_{i=1}^{m} w_{ij}^{p} \, dist(\boldsymbol{x}_i, \boldsymbol{c}_j)^2 \qquad \sum_{j=1}^{k} w_{ij} = 1$$

Initialization: choose the weights w_{ij} randomly subject to the constraint that $\sum_{j=1}^{k} w_{ij} = 1$

Repeat:

- Update centroids: $\boldsymbol{c}_j = \sum_{i=1}^{m} w_{ij} \boldsymbol{x}_i \,/ \sum_{i=1}^{m} w_{ij}$
- Update weights: $w_{ij} = (1/dist(\boldsymbol{x}_i, \boldsymbol{c}_j)^2)^{\frac{1}{p-1}} / \sum_{q=1}^{k} (1/dist(\boldsymbol{x}_i, \boldsymbol{c}_q)^2)^{\frac{1}{p-1}}$

Fuzzy K-means Applied to Sample Data



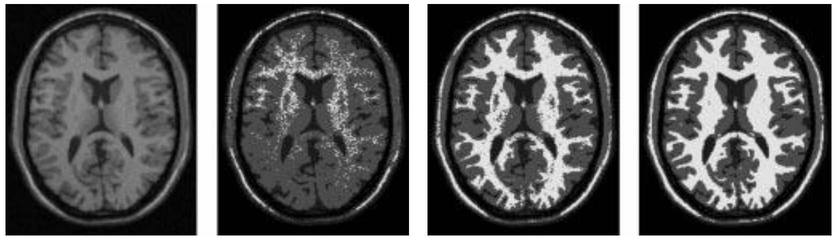
An Example Application: Image Segmentation

Modified versions of fuzzy c-means have been used for image segmentation

– Especially fMRI images (functional magnetic resonance images)

References

– Gong, Maoguo, Yan Liang, Jiao Shi, Wenping Ma, and Jingjing Ma. "Fuzzy c-means clustering with local information and kernel metric for image segmentation." *Image Processing, IEEE Transactions on* 22, no. 2 (2013): 573-584.



From left to right: original images, fuzzy c-means, EM, BCFCM

– Ahmed, Mohamed N., Sameh M. Yamany, Nevin Mohamed, Aly A. Farag, and Thomas Moriarty. "A modified fuzzy c-means algorithm for bias field estimation and segmentation of MRI data." *Medical Imaging, IEEE Transactions on* 21, no. 3 (2002): 193-199.

Clustering Using Mixture Models

Idea is to model the set of data points as arising from a mixture of distributions

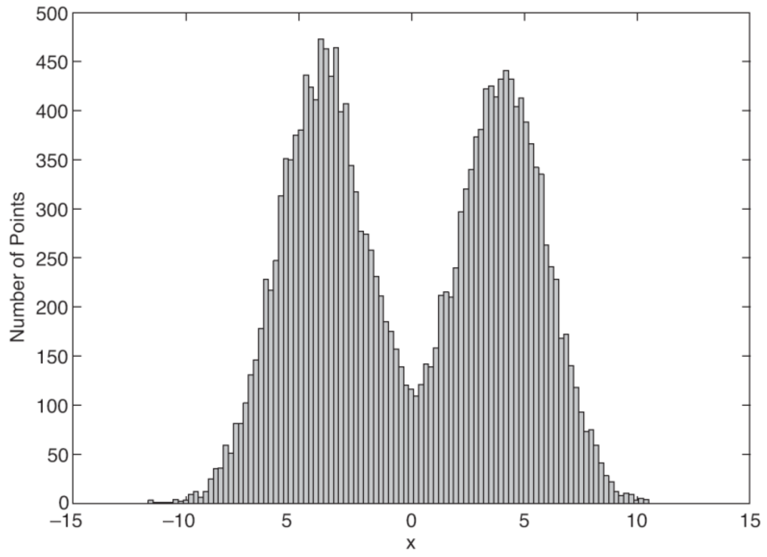
- Typically, normal (Gaussian) distribution is used
- But other distributions have been very profitably used

Clusters are found by estimating the parameters of the statistical distributions using the Expectation-Maximization (EM) algorithm

- k-means is a special case of this approach
- Provides a compact representation of clusters
- The probabilities with which point belongs to each cluster provide a functionality similar to fuzzy clustering.

Probabilistic Clustering: Example

Informal example: consider modeling the points that generate the following histogram.



Looks like a combination of two normal (Gaussian) distributions

Suppose we can estimate the mean and standard deviation of each normal distribution.

- This completely describes the two clusters
- We can compute the probabilities with which each point belongs to each cluster
- Can assign each point to the cluster (distribution) for which it is most probable.

$$prob(x_i|\Theta) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Probabilistic Clustering: EM Algorithm

```
Initialize the parameters
Repeat
    For each point, compute its probability under each distribution
    Using these probabilities, update the parameters of each distribution
Until there is no change
```

- Very similar to of K-means
- Consists of assignment and update steps
- Can use random initialization
  - Problem of local minima
- For normal distributions, typically use K-means to initialize
- If using normal distributions, can find elliptical as well as spherical shapes.

Probabilistic Clustering: Updating Centroids

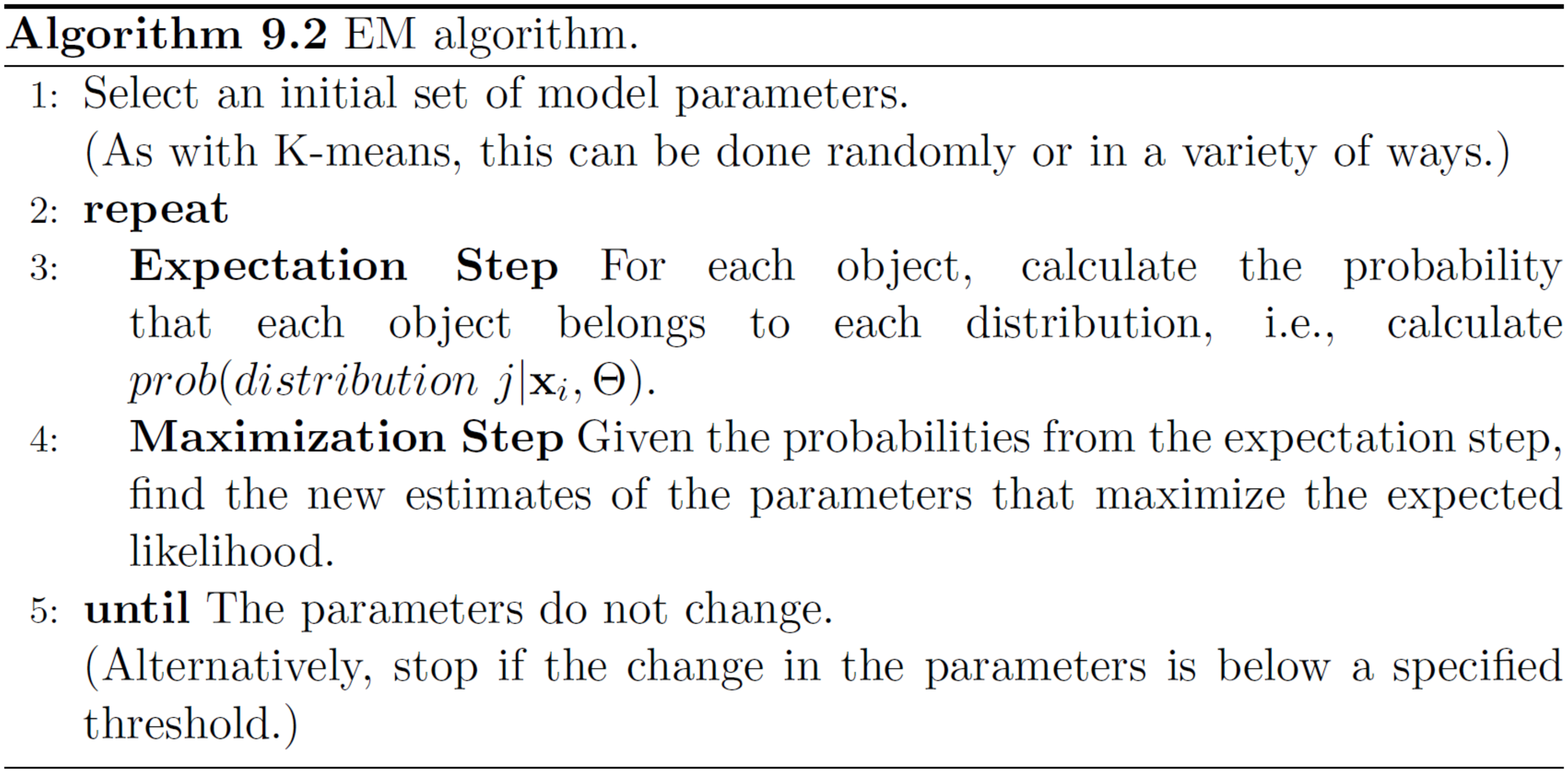
Update formula for weights assuming an estimate for statistical parameters

$$\boldsymbol{c}_j = \sum_{i=1}^{m} \mathbf{x}_i p(C_j \mid \boldsymbol{x}_i) \Big/ \sum_{i=1}^{m} p(C_j \mid \boldsymbol{x}_i)$$

$\boldsymbol{x_i}$ *is a data point*
$C_{\boldsymbol{j}}$ *is a cluster*
$\boldsymbol{c_j}$ *is a centroid*

- Very similar to the fuzzy k-means formula
  - Weights are probabilities
  - Weights are not raised to a power
  - Probabilities calculated using Bayes rule: $p(C_{\boldsymbol{j}} | \boldsymbol{x_i}) = \dfrac{p(\boldsymbol{x_i} | C_{\boldsymbol{j}}) p(C_{\boldsymbol{j}})}{\sum_{l=1}^{k} p(\boldsymbol{x_i} | C_l) p(C_{\boldsymbol{l}})}$
- Need to assign weights to each cluster
  - Weights may not be equal
  - Similar to prior probabilities
  - Can be estimated: $p(C_j) = \dfrac{1}{m} \sum_{i=1}^{m} p(C_j \mid \boldsymbol{x}_i)$

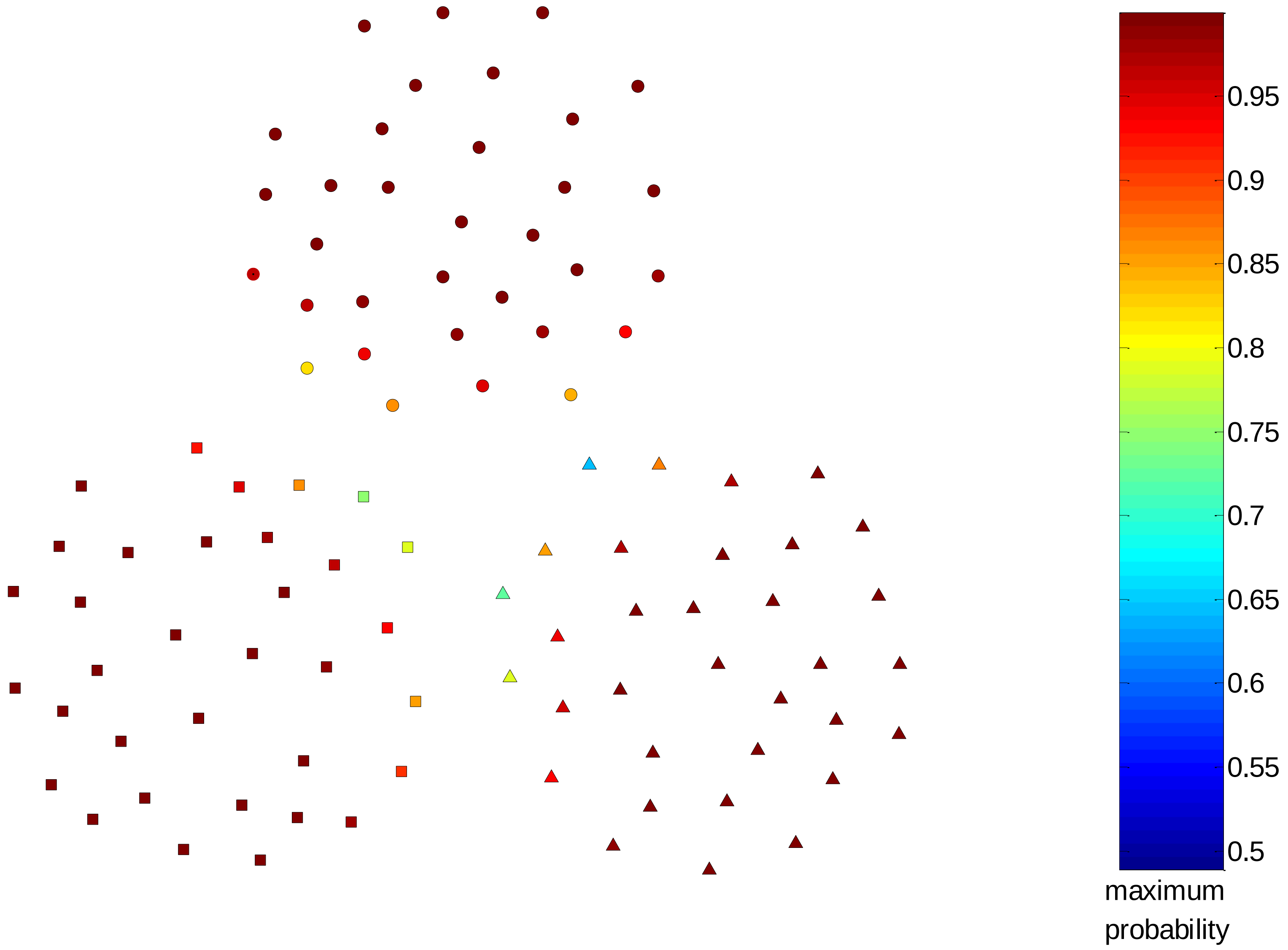
More Detailed EM Algorithm



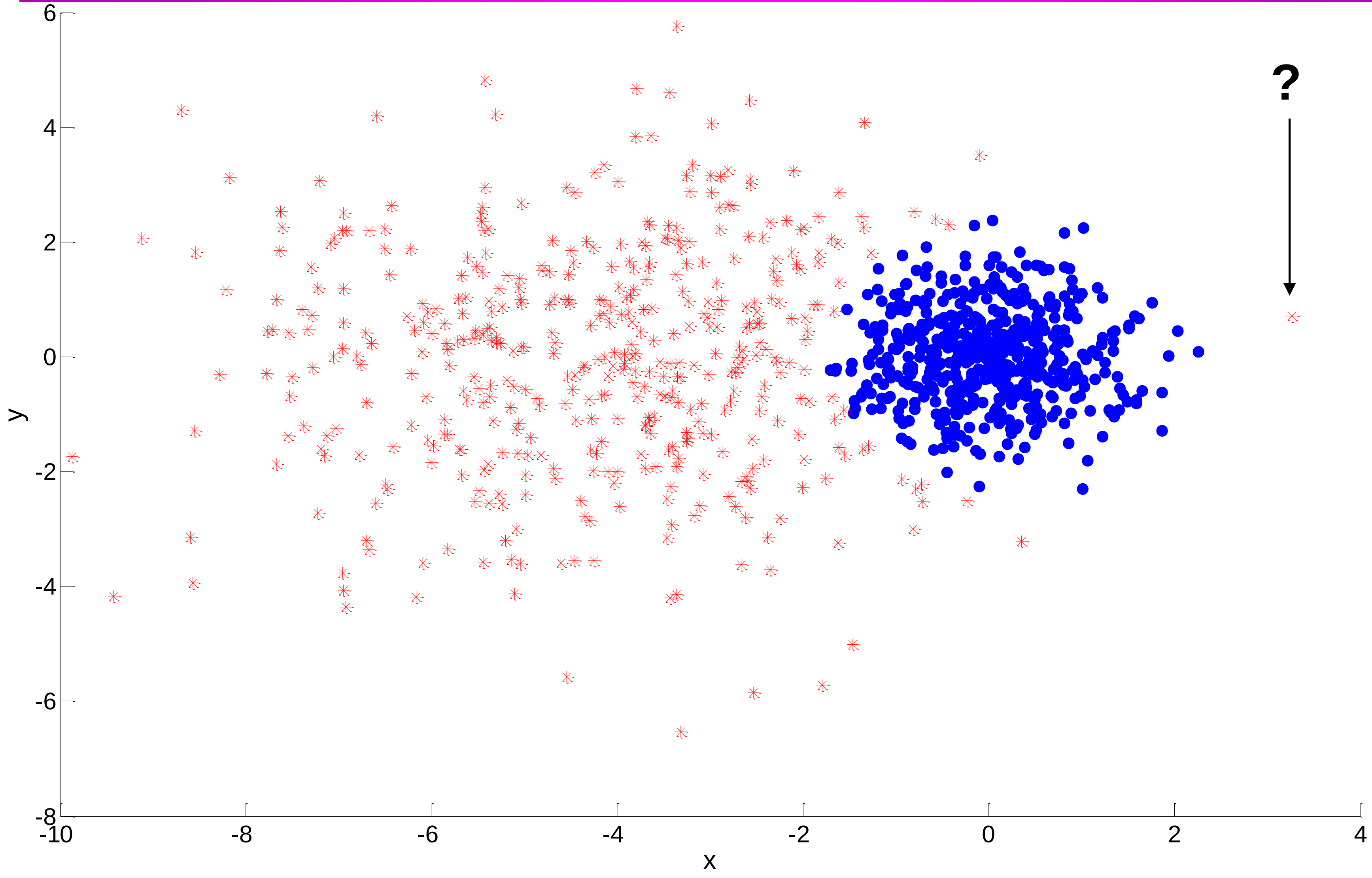
Algorithm 9.2 EM algorithm.

```
1: Select an initial set of model parameters.
   (As with K-means, this can be done randomly or in a variety of ways.)
2: repeat
3:    Expectation Step For each object, calculate the probability
      that each object belongs to each distribution, i.e., calculate
      $prob(distribution\ j|\mathbf{x}_i, \Theta)$.
4:    Maximization Step Given the probabilities from the expectation step,
      find the new estimates of the parameters that maximize the expected
      likelihood.
5: until The parameters do not change.
   (Alternatively, stop if the change in the parameters is below a specified
   threshold.)
```

Probabilistic Clustering Applied to Sample Data



Probabilistic Clustering: Dense and Sparse Clusters



Problems with EM

Convergence can be slow

Only guarantees finding local maxima

Makes some significant statistical assumptions

Number of parameters for Gaussian distribution grows as O(d^2), d the number of dimensions

- Parameters associated with covariance matrix
- K-means only estimates cluster means, which grow as O(d)

Alternatives to EM

Method of moments / Spectral methods

- ICML 2014 workshop bibliography

  https://sites.google.com/site/momentsicml2014/bibliography

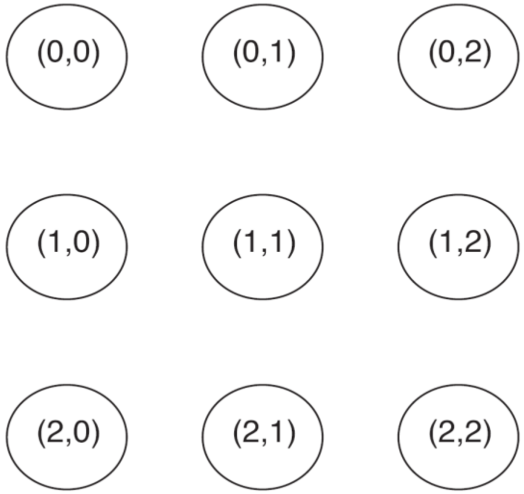
Markov chain Monte Carlo (MCMC)

Other approaches

SOM: Self-Organizing Maps

Self-organizing maps (SOM)

- Centroid based clustering scheme
- Like K-means, a fixed number of clusters are specified
- However, the **spatial relationship of clusters is also specified**, typically as a grid
- Points are considered one by one
- Each point is assigned to the closest centroid, and this centroid is updated
- **Other centroids are updated based on their spatial proximity to the closest centroid**



Kohonen, Teuvo, and Self-Organizing Maps. "Springer series in information sciences." *Self-organizing maps* 30 (1995).

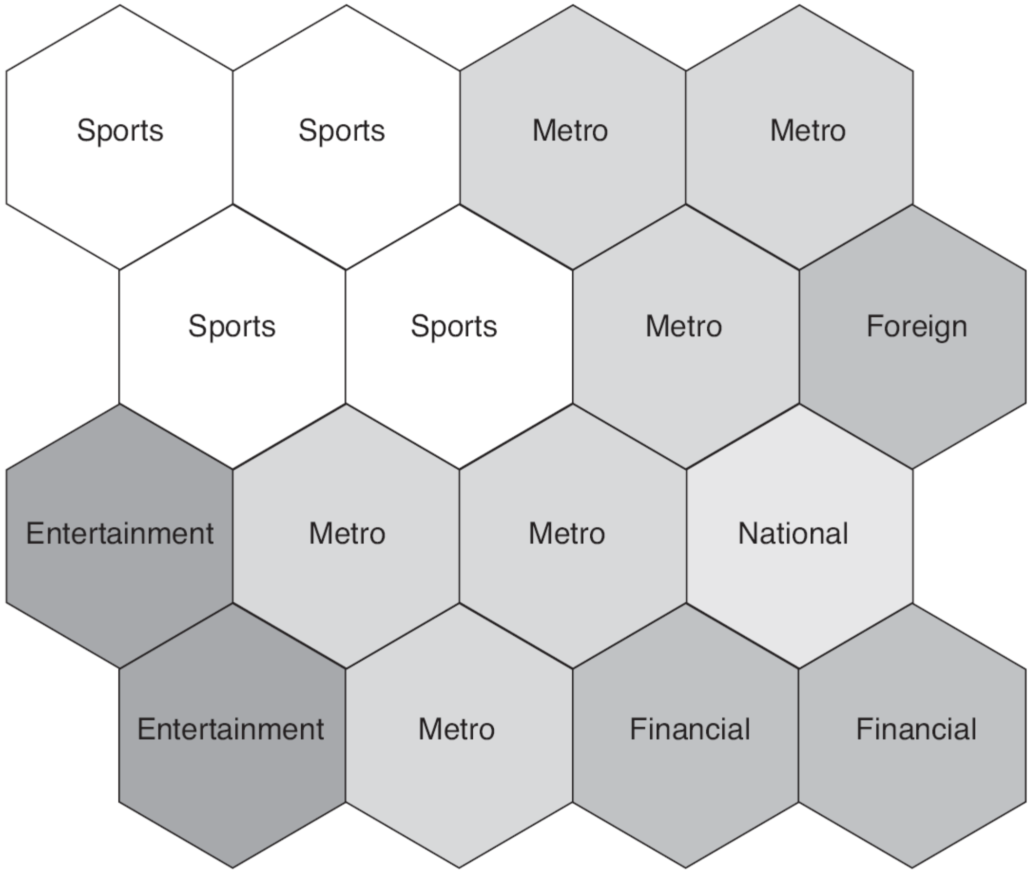
SOM: Self-Organizing Maps

Algorithm 9.3 Basic SOM Algorithm.

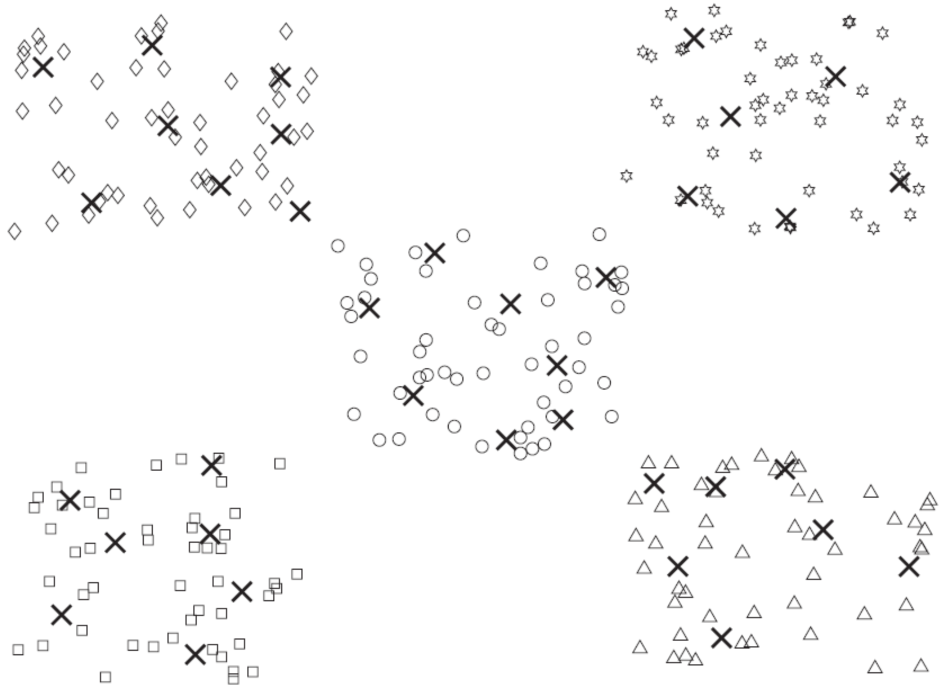
```
1: Initialize the centroids.
2: repeat
3:     Select the next object.
4:     Determine the closest centroid to the object.
5:     Update this centroid and the centroids that are close, i.e., in a specified neigh-
       borhood.
6: until The centroids don't change much or a threshold is exceeded.
7: Assign each object to its closest centroid and return the centroids and clusters.
```

- Updates are weighted by distance
  - Centroids farther away are affected less
- The impact of the updates decreases with each time
  - At some point the centroids will not change much
- SOM can be viewed as a type of dimensionality reduction
  - If a 2D (3D) grid is used, the results can be easily visualized, and it can facilitate the interpretation of clusters

SOM Clusters of LA Times Document Data



Another SOM Example: 2D Points


(a) Distribution of SOM reference vectors (**X**'s) for a two-dimensional point set.

diamond	diamond	diamond	hexagon	hexagon	hexagon
diamond	diamond	diamond	circle	hexagon	hexagon
diamond	diamond	circle	circle	circle	hexagon
square	square	circle	circle	triangle	triangle
square	square	circle	circle	triangle	triangle
square	square	square	triangle	triangle	triangle

(b) Classes of the SOM centroids.

Issues with SOM

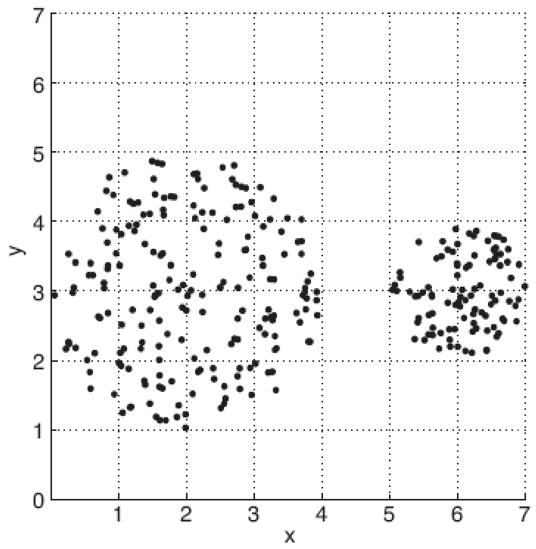
- High computational complexity
- No guarantee of convergence
- Choice of grid and other parameters is somewhat arbitrary
- Lack of a specific objective function

Grid-based Clustering

A type of density-based clustering

Algorithm 9.4 Basic grid-based clustering algorithm.

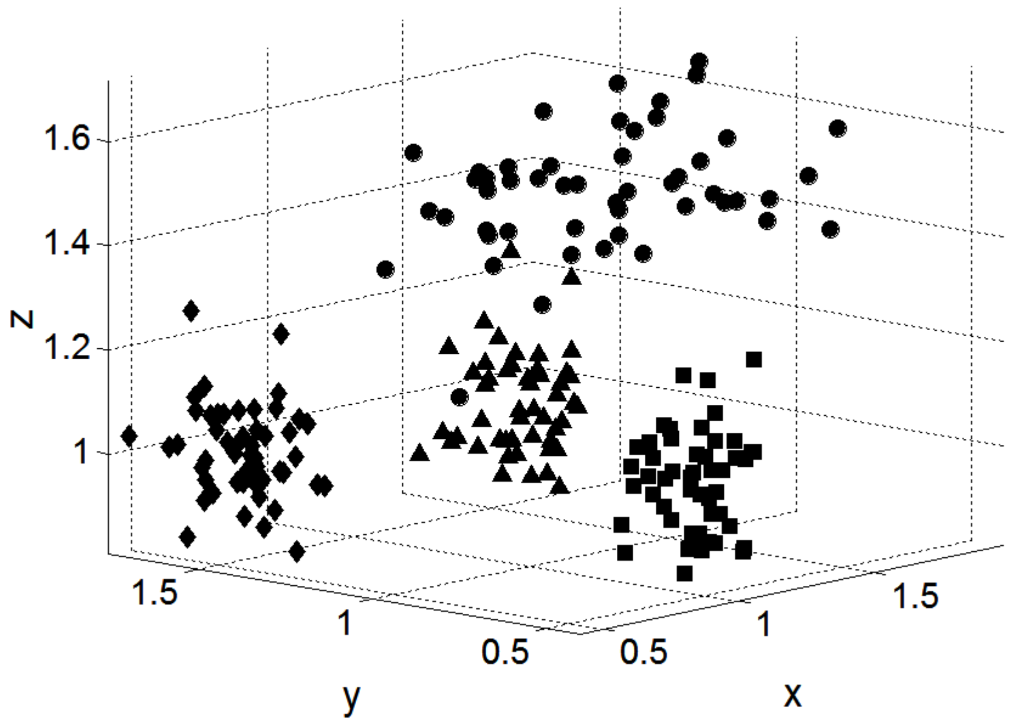
```
1: Define a set of grid cells.
2: Assign objects to the appropriate cells and compute the density of each cell.
3: Eliminate cells having a density below a specified threshold, $\tau$.
4: Form clusters from contiguous (adjacent) groups of dense cells.
```



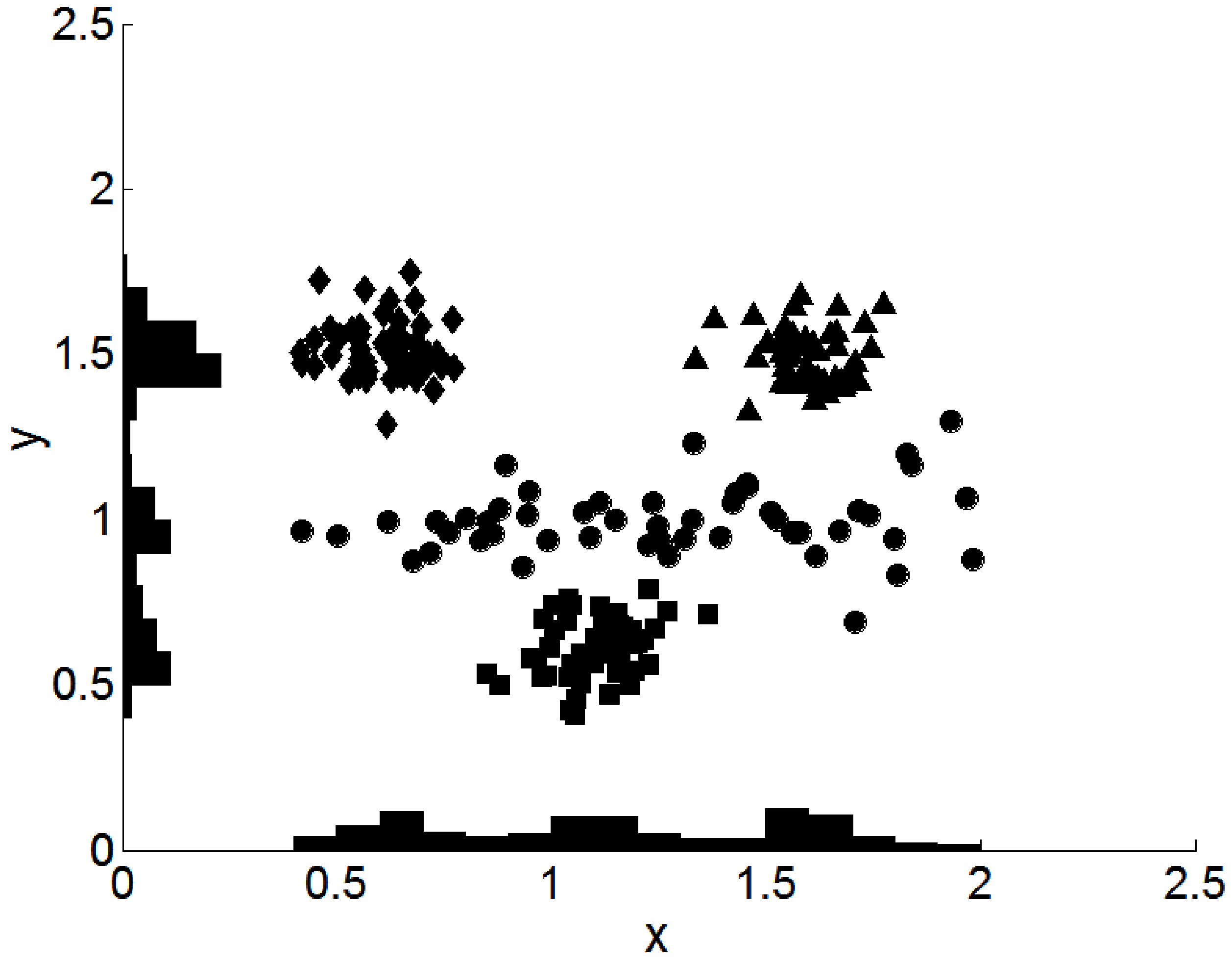
0	0	0	0	0	0	0
0	0	0	0	0	0	0
4	**17**	**18**	**6**	0	0	0
14	**14**	**13**	**13**	0	**18**	**27**
11	**18**	**10**	**21**	0	**24**	**31**
3	**20**	**14**	**4**	0	0	0
0	0	0	0	0	0	0

Subspace Clustering

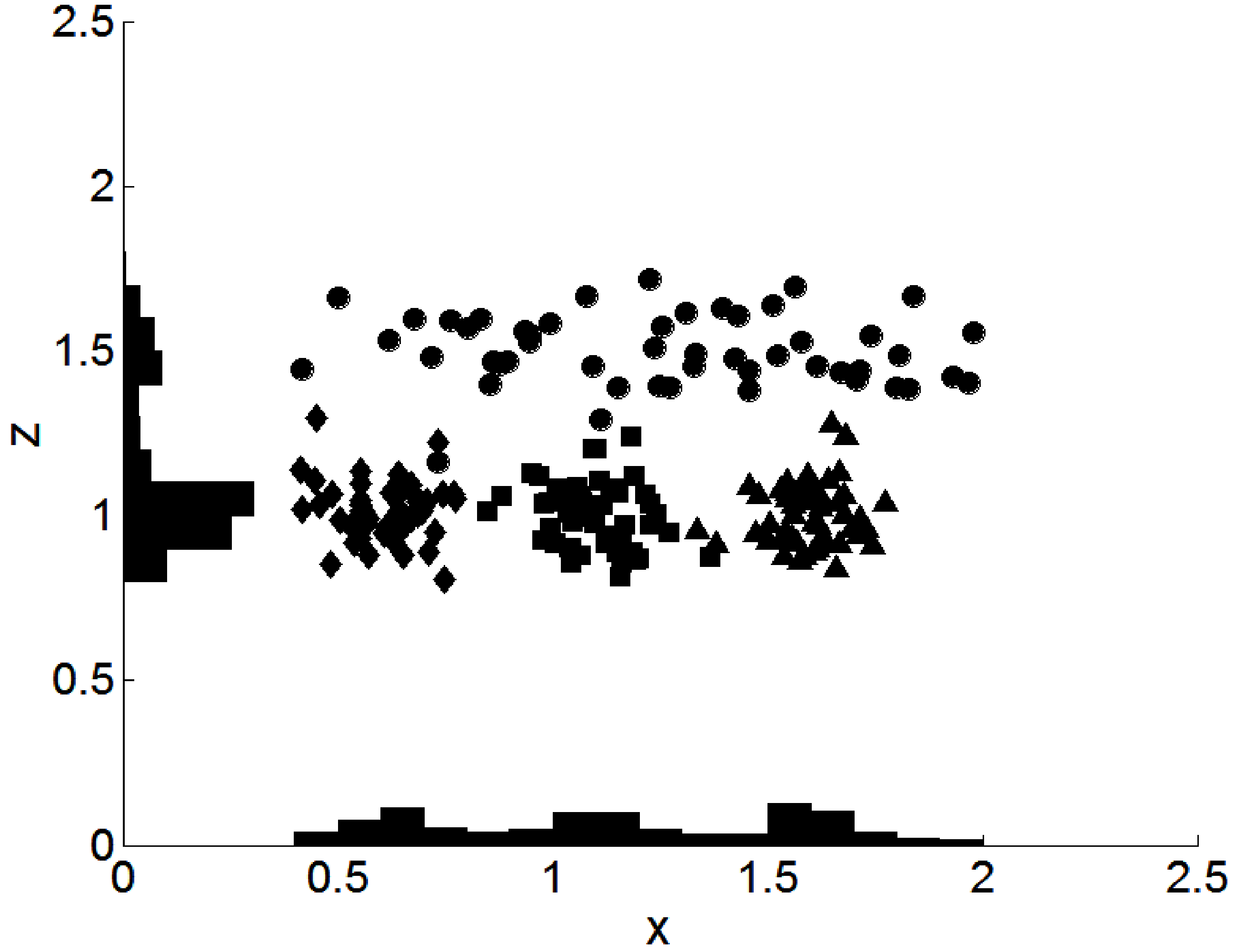
- Until now, we found clusters by considering all of the attributes
- Some clusters may involve only a subset of attributes, i.e., subspaces of the data



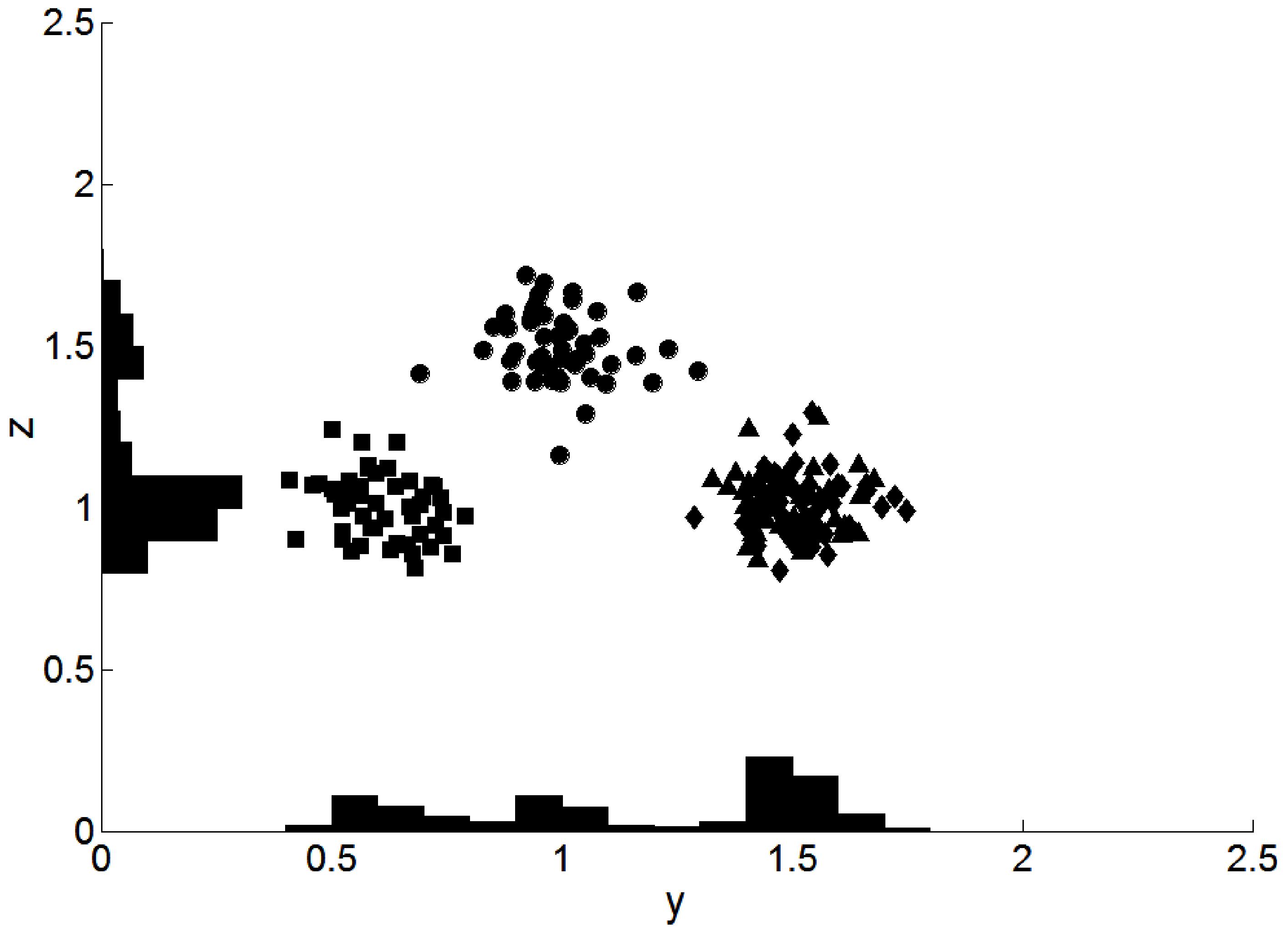
Clusters in subspaces



Clusters in subspaces



Clusters in subspaces



Clique – A Subspace Clustering Algorithm

A grid-based clustering algorithm that methodically finds subspace clusters

- Partitions the data space into rectangular units of equal volume in all possible subspaces
- Measures the density of each unit by the fraction of points it contains
- A unit is dense if the fraction of overall points it contains is above a user specified threshold, τ
- A cluster is a set of contiguous (touching) dense units

Clique Algorithm

- It is impractical to check volume units in all possible subspaces, since there is an exponential number of such units

- Monotone property of density-based clusters:
  - If a set of points cannot form a density based cluster in k dimensions, then the same set of points cannot form a density based cluster in all possible supersets of those dimensions

- Very similar to Apriori algorithm

- Can find overlapping clusters

Clique Algorithm

Algorithm 9.5 CLIQUE.

```
1: Find all the dense areas in the one-dimensional spaces corresponding to each
   attribute. This is the set of dense one-dimensional cells.
2: $k \leftarrow 2$
3: **repeat**
4:    Generate all candidate dense $k$-dimensional cells from dense $(k-1)$-dimensional
      cells.
5:    Eliminate cells that have fewer than $\xi$ points.
6:    $k \leftarrow k + 1$
7: **until** There are no candidate dense $k$-dimensional cells.
8: Find clusters by taking the union of all adjacent, high-density cells.
9: Summarize each cluster using a small set of inequalities that describe the attribute
   ranges of the cells in the cluster.
```

Limitations of Clique

- Time complexity is exponential in number of dimensions
  - Especially if “too many” dense units are generated at lower stages

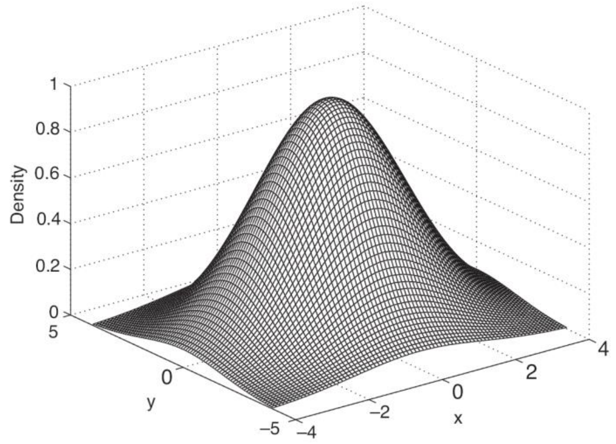
- May fail if clusters are of widely differing densities, since the threshold is fixed
  - Determining appropriate threshold and unit interval length can be challenging

Denclue (DENsity CLUstering)

Based on the notion of kernel-density estimation

– Contribution of each point to the density is given by an influence or kernel function

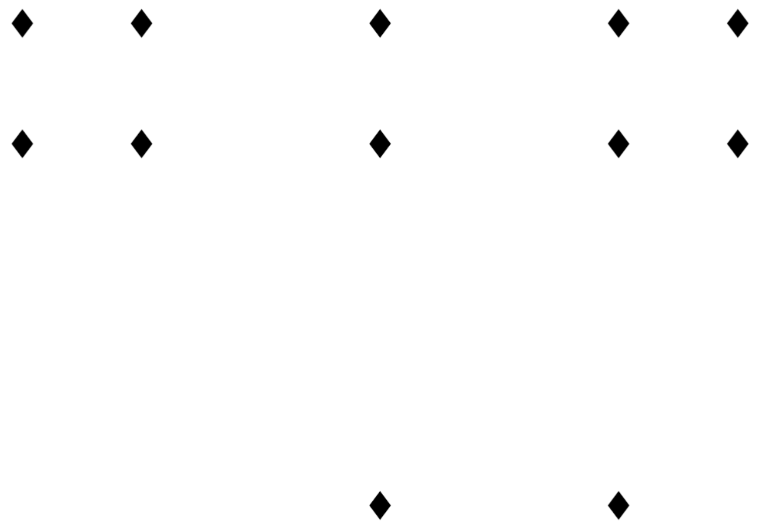
$$K(y) = e^{-distance(\mathbf{x},\mathbf{y})^2/2\sigma^2}$$

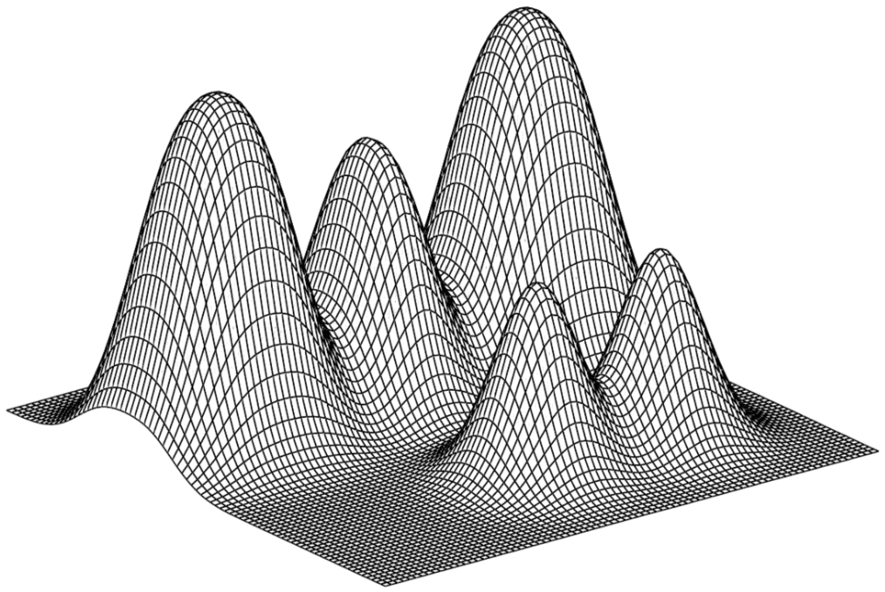


Formula and plot of Gaussian Kernel

– Overall density is the sum of the contributions of all points

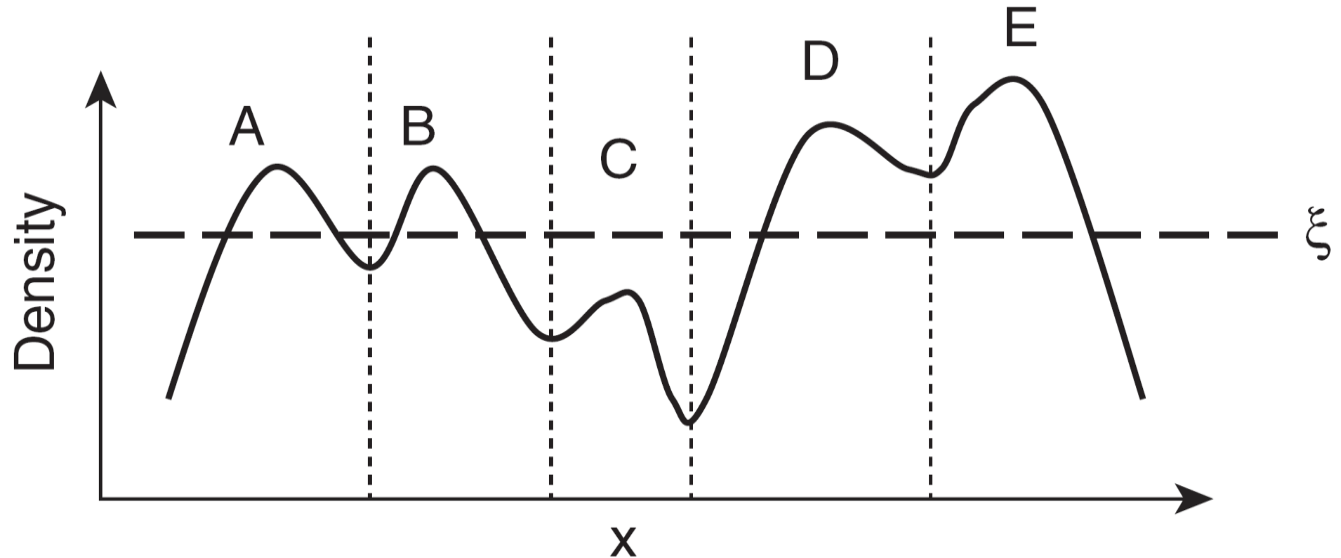
Example of Density from Gaussian Kernel


Set of 12 points.


Overall density—surface plot.

DENCLUE Algorithm

- Find the density function
- Identify local maxima (density attractors)
- Assign each point to the density attractor
  - Follow direction of maximum increase in density
- Define clusters as groups consisting of points associated with density attractor
- Discard clusters whose density attractor has a density less than a user specified minimum, ξ
- Combine clusters connected by paths of points that are connected by points with density above ξ



Graph-Based Clustering: General Concepts

Graph-Based clustering needs a proximity graph

- Each edge between two nodes has a weight which is the proximity between the two points

Many hierarchical clustering algorithms can be viewed in graph terms

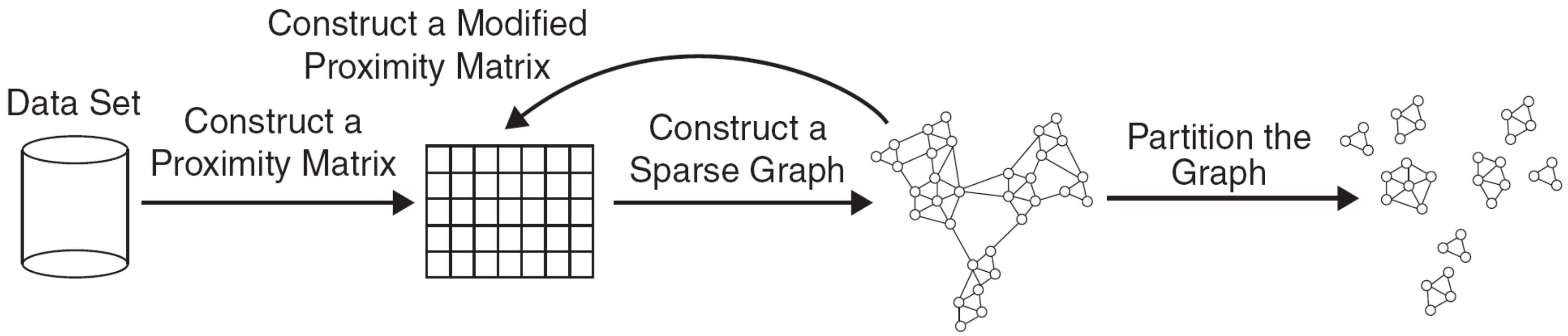
- MIN (single link) merges subgraph pairs connected with a lowest weight edge
- Group Average merges subgraph pairs that have high average connectivity

In the simplest case, clusters are connected components in the graph.

Graph-Based Clustering: Chameleon

Based on several key ideas

- Sparsification of the proximity graph
- Partitioning the data into clusters that are relatively pure subclusters of the "true" clusters
- Merging based on preserving characteristics of clusters


Data Set
Construct a Proximity Matrix
Construct a Modified Proximity Matrix
Construct a Sparse Graph
Partition 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.
- 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)

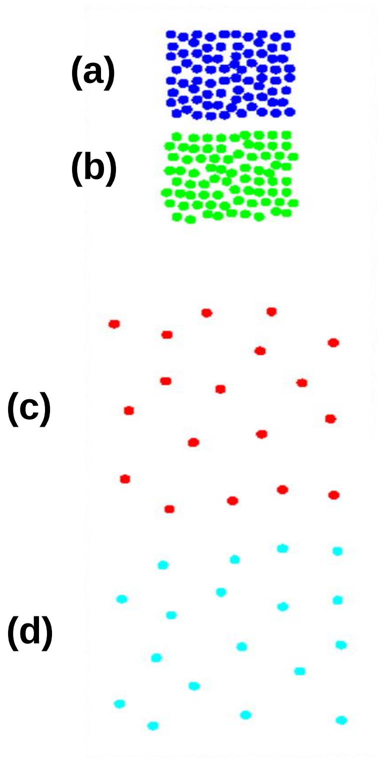
- Chameleon and Hypergraph-based Clustering

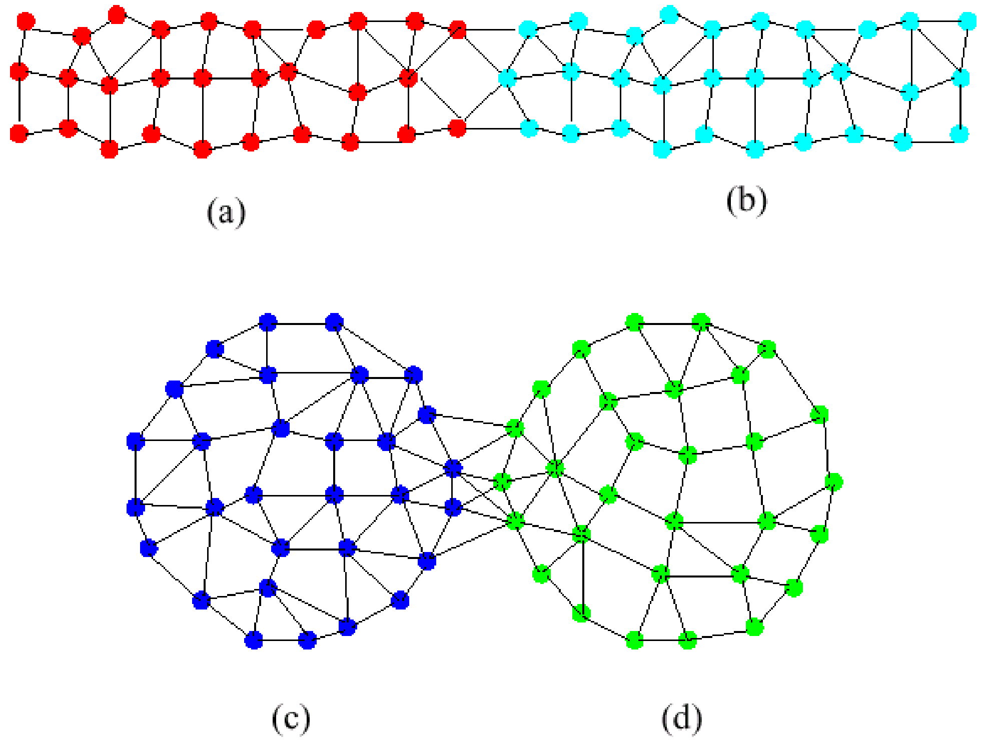
Limitations of Current Merging Schemes

Existing merging schemes in hierarchical clustering algorithms are static in nature

- MIN:
  - 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)

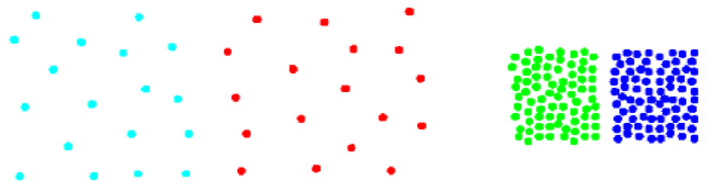
Average connectivity schemes will merge (c) and (d)

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 properties are 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*



Relative Interconnectivity

- **Relative Interconnectivity (RI)** is the absolute interconnectivity of two clusters normalized by the internal connectivity of the clusters. Two clusters are combined if the points in the resulting cluster are almost as strongly connected as points in each of the original clusters. Mathematically,

$$RI = \frac{EC(C_i, C_j)}{\frac{1}{2}(EC(C_i) + EC(C_j))}, \tag{9.18}$$

where $EC(C_i, C_j)$ is the sum of the edges (of the k-nearest neighbor graph) that connect clusters C_i and C_j; $EC(C_i)$ is the minimum sum of the cut edges if we bisect cluster C_i; and $EC(C_j)$ is the minimum sum of the cut edges if we bisect cluster C_j.

Relative Closeness

- **Relative Closeness (RC)** is the absolute closeness of two clusters normalized by the internal closeness of the clusters. Two clusters are combined only if the points in the resulting cluster are almost as close to each other as in each of the original clusters. Mathematically,

$$RC = \frac{\bar{S}_{EC}(C_i, C_j)}{\frac{m_i}{m_i+m_j}\bar{S}_{EC}(C_i) + \frac{m_j}{m_i+m_j}\bar{S}_{EC}(C_j)}, \qquad (9.17)$$

  where m_i and m_j are the sizes of clusters C_i and C_j, respectively, $\bar{S}_{EC}(C_i, C_j)$ is the average weight of the edges (of the k-nearest neighbor graph) that connect clusters C_i and C_j; $\bar{S}_{EC}(C_i)$ is the average weight of edges if we bisect cluster C_i; and $\bar{S}_{EC}(C_j)$ is the average weight of edges if we bisect cluster C_j. (EC stands for edge cut.)

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., be a sub-cluster of a “real” cluster

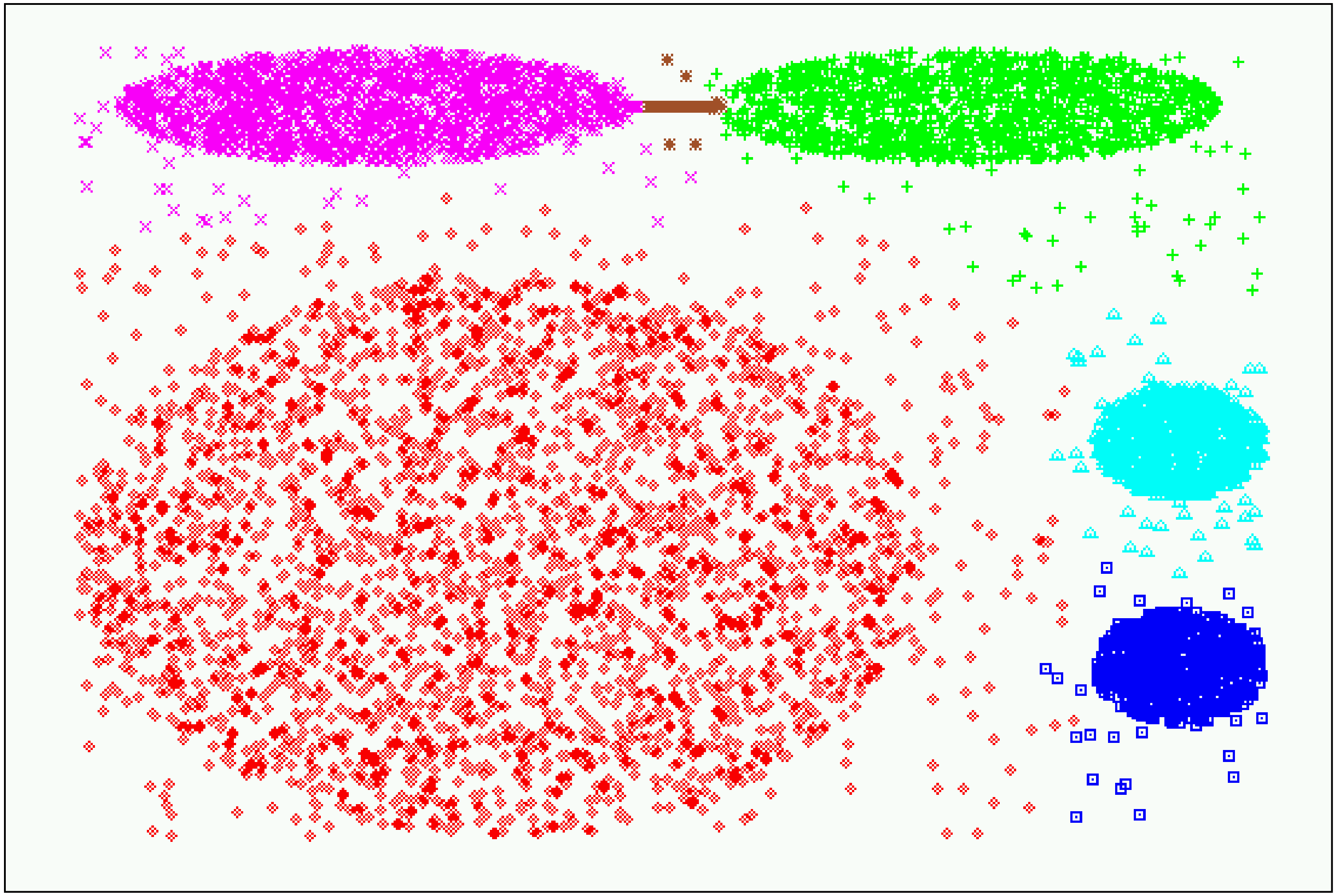
Chameleon: Steps …

Phase 2: Use Hierarchical Agglomerative Clustering to merge sub-clusters

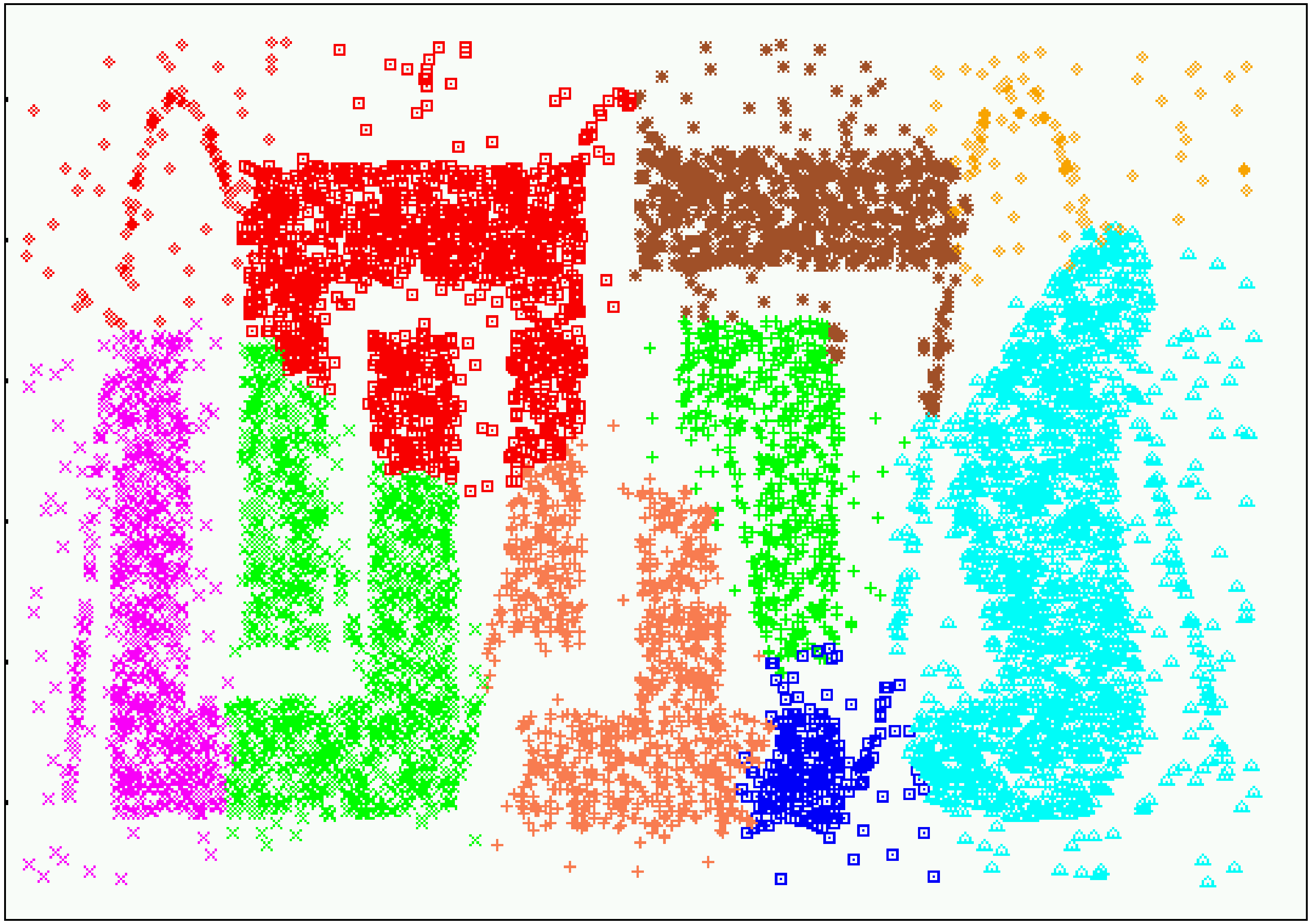
- Two clusters are combined if the *resulting cluster shares certain properties with the constituent clusters*

- 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

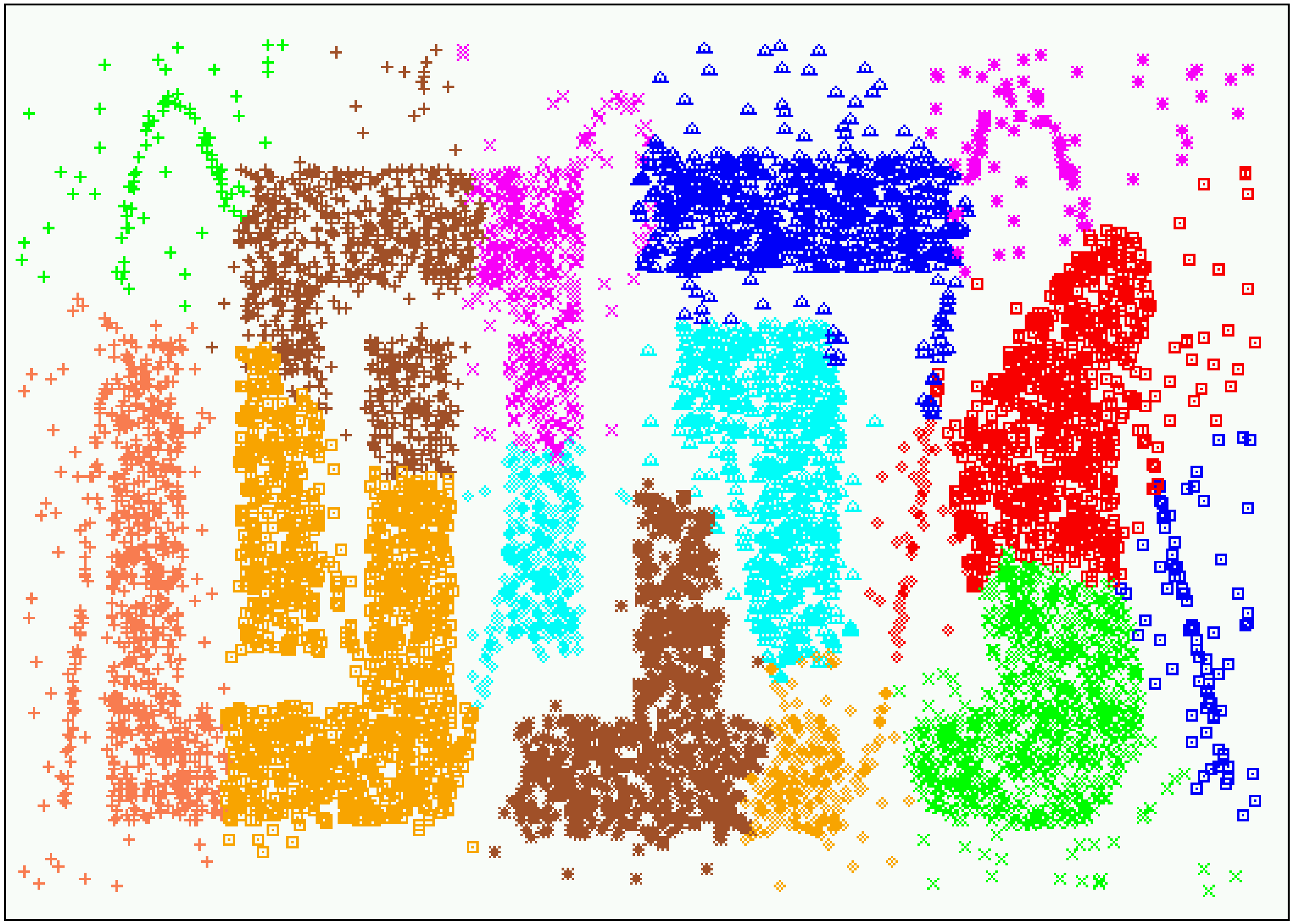
Experimental Results: CHAMELEON



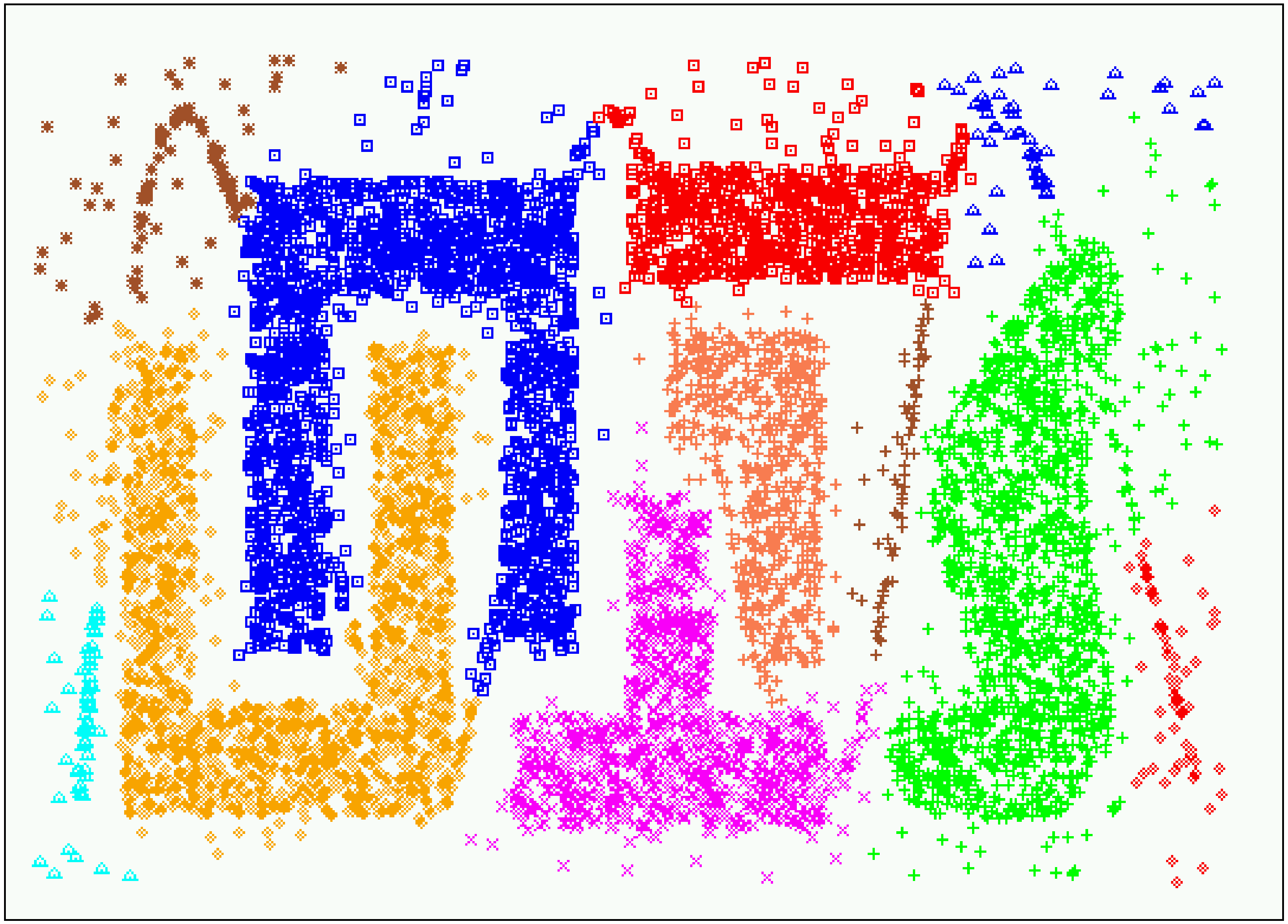
Experimental Results: CURE (*10 clusters*)



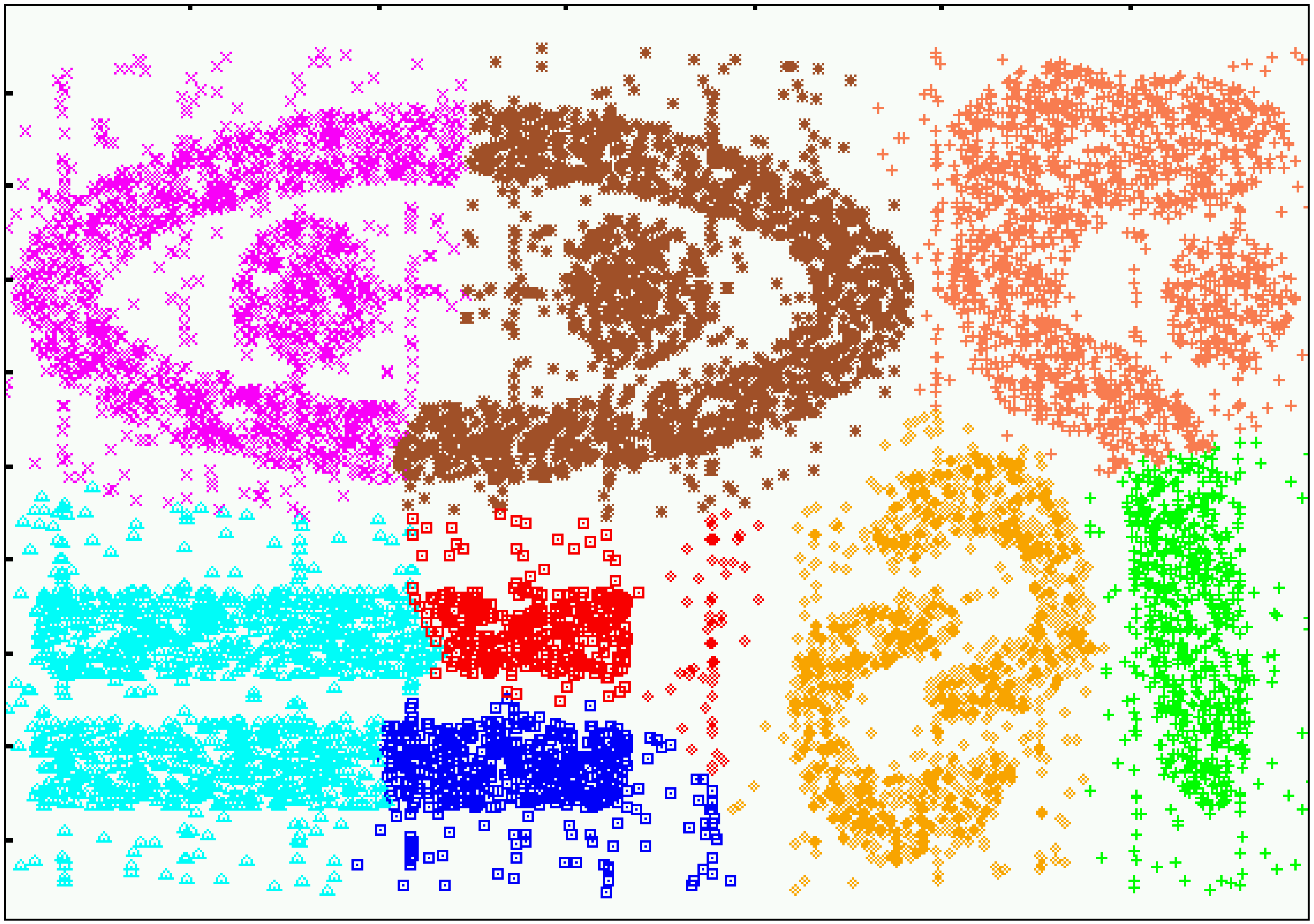
Experimental Results: CURE (*15 clusters*)



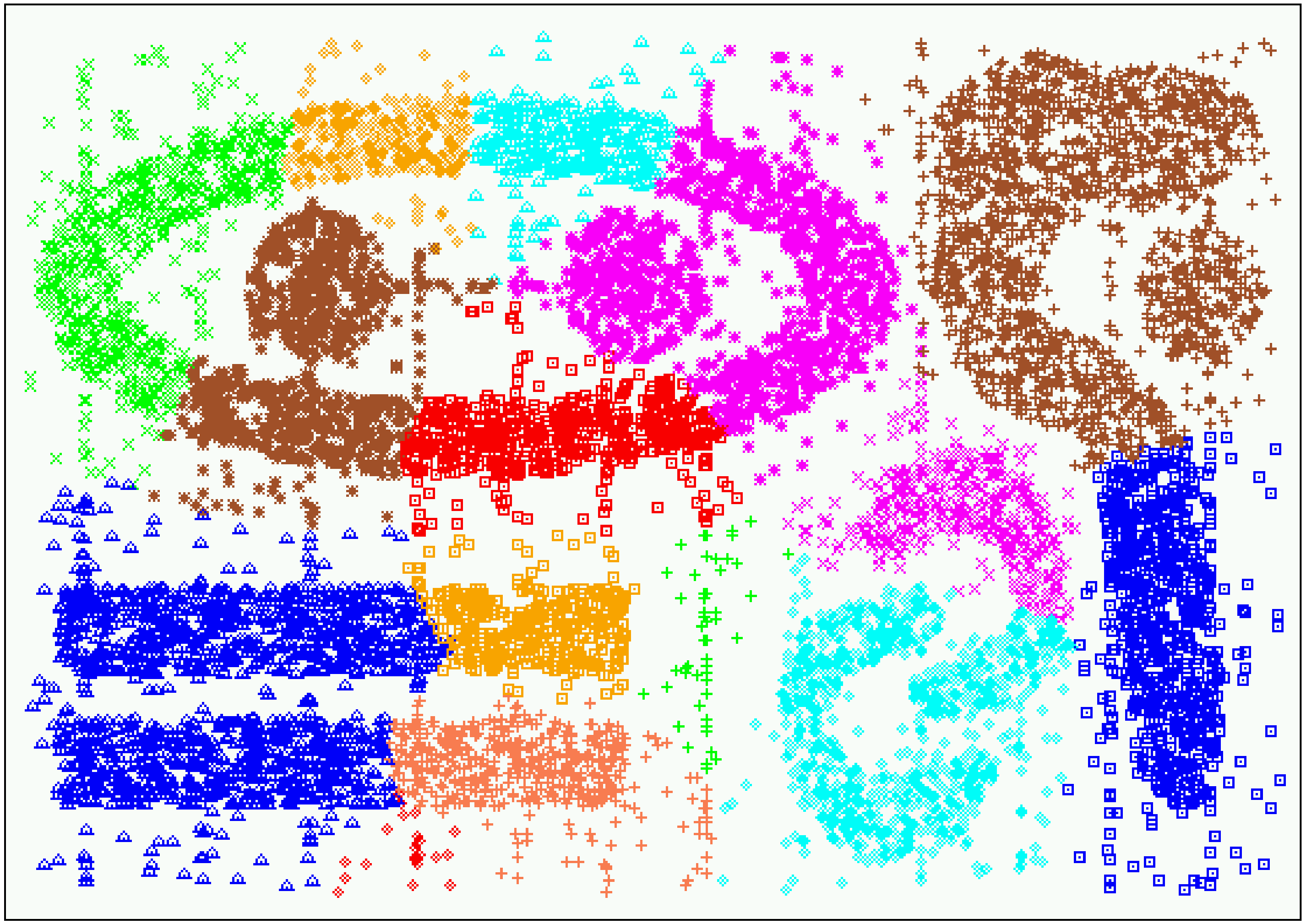
Experimental Results: CHAMELEON



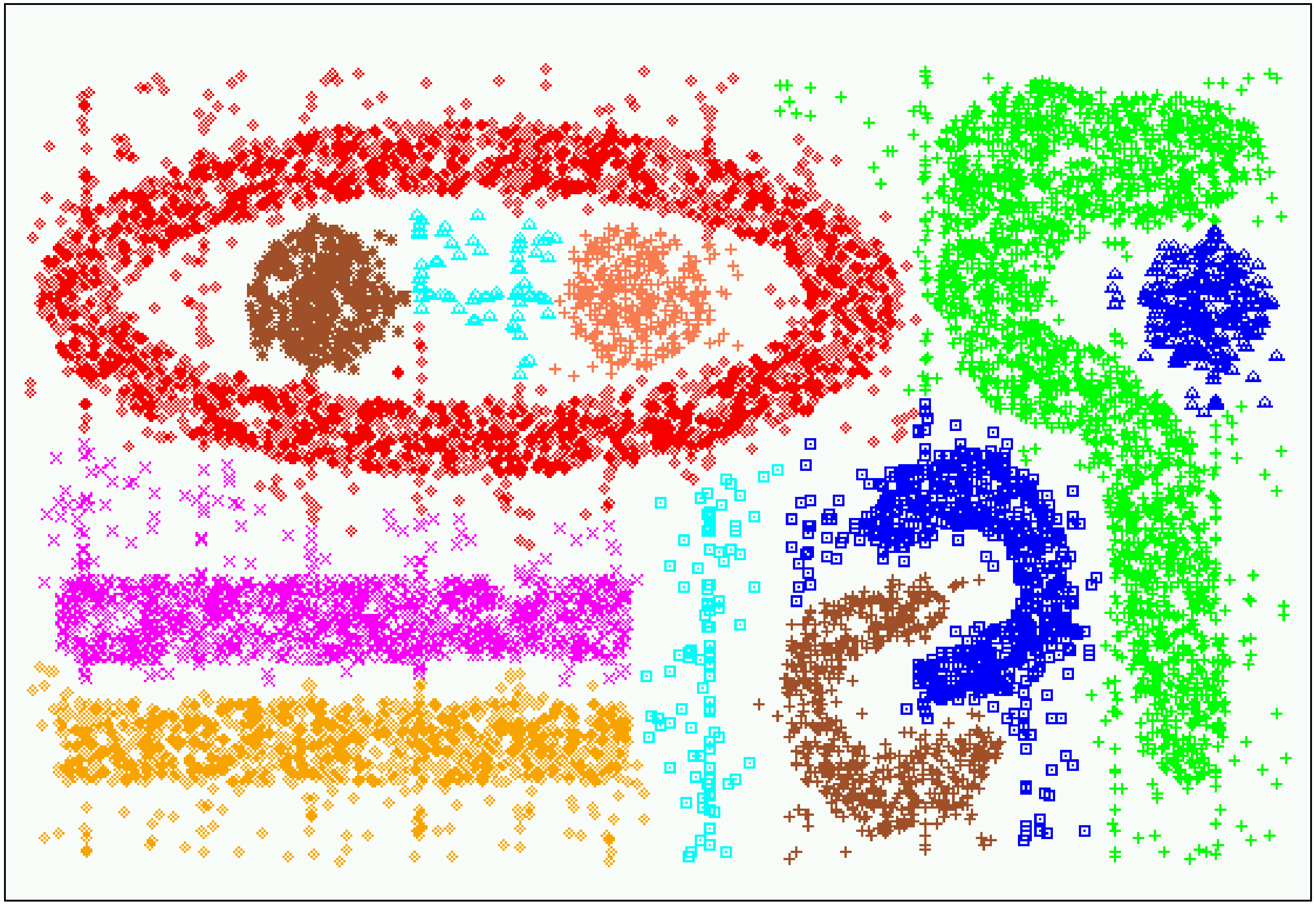
Experimental Results: CURE (*9 clusters*)



Experimental Results: CURE (*15 clusters*)



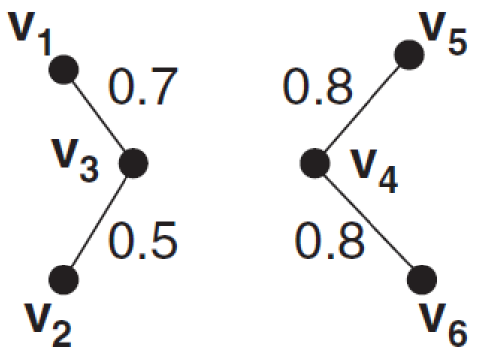
Experimental Results: CHAMELEON



Spectral Clustering

Spectral clustering is a graph-based clustering approach

- Does a graph partitioning of the proximity graph of a data set
- Breaks the graph into components, such that
  - ◆ The nodes in a component are strongly connected to other nodes in the component
  - ◆ The nodes in a component are weakly connected to nodes in other components
  - ◆ See simple example below (**W** is the proximity matrix)



$$W = \begin{bmatrix} 0 & 0 & 0.7 & 0 & 0 & 0 \\ 0 & 0 & 0.5 & 0 & 0 & 0 \\ 0.7 & 0.5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.8 & 0.8 \\ 0 & 0 & 0 & 0.8 & 0 & 0 \\ 0 & 0 & 0 & 0.8 & 0 & 0 \end{bmatrix}$$

Spectral Clustering

For the simple graph below, the proximity matrix can be written as

$$\mathbf{W} = \begin{pmatrix} \mathbf{W}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{W}_2 \end{pmatrix}$$

- Because the graph consists of two connected components finding clusters is easy.
- More generally, we need an automated approach
  - Must be able to handle graphs where the components are not completely separate
  - Spectral graph partitioning provides such an approach
  - Based on eigenvalue decomposition of a slight modification of the proximity matrix.

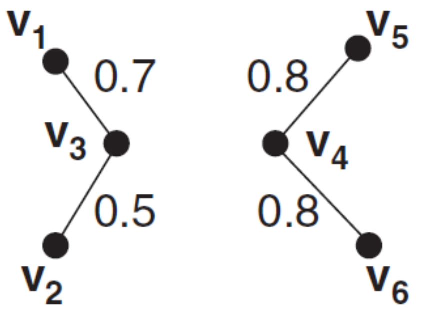
Clustering via Spectral Graph Partitioning

- Uses an eigenvalue based approach to do the graph portioning
  - Based on the Laplacian matrix (**L**) of a graph, which is derived from the proximity matrix (**W**)
    - **W** is also known as the weighted adjacency matrix

- Define a diagonal matrix **D**

$$D_{ij} = \begin{cases} \sum_k W_{ij}, & if\ i = j \\ 0, & otherwise \end{cases}$$

- k^{th} diagonal entry of D is the sum of the edges of the k^{th} node of **W**
  - See example below



$$W = \begin{bmatrix} 0 & 0 & 0.7 & 0 & 0 & 0 \\ 0 & 0 & 0.5 & 0 & 0 & 0 \\ 0.7 & 0.5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.8 & 0.8 \\ 0 & 0 & 0 & 0.8 & 0 & 0 \\ 0 & 0 & 0 & 0.8 & 0 & 0 \end{bmatrix} \quad D = \begin{bmatrix} 0.7 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1.2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1.6 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.8 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.8 \end{bmatrix}$$

Clustering via Spectral Graph Partitioning …

Define a matrix called the Laplacian of the graph, **L**

$$\mathbf{L} = \mathbf{D} - \mathbf{W}$$

L has the following properties:

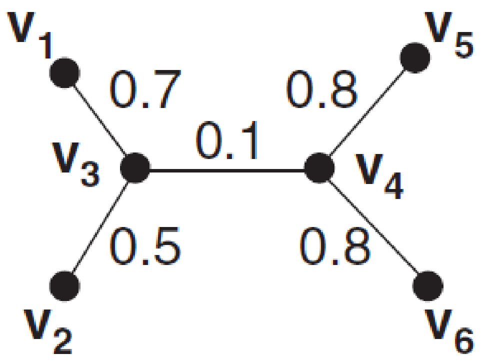
- It is symmetric
- $\mathbf{v}^T\mathbf{L}\mathbf{v} \geq 0$, i.e., the matrix is positive semi-definite and all eigenvalues are positive.
- Last eigenvalue is 0

Can write **L** in terms of its eigenvalue decomposition as

- $\mathbf{L} = \mathbf{\Lambda V}$, where $\mathbf{\Lambda}$ is a diagonal matrix of the eigenvalues and $\mathbf{V}$ is the matrix of eigenvectors

$$L = \begin{bmatrix} 0.7 & 0 & -0.7 & 0 & 0 & 0 \\ 0 & 0.5 & -0.5 & 0 & 0 & 0 \\ -0.7 & -0.5 & 1.2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1.6 & -0.8 & -0.8 \\ 0 & 0 & 0 & -0.8 & 0.8 & 0 \\ 0 & 0 & 0 & -0.8 & 0 & 0.8 \end{bmatrix} \quad \Lambda = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.58 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.8 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1.8 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2.4 \end{bmatrix} \quad V = \begin{bmatrix} 0.58 & 0 & 0.64 & 0 & -0.50 & 0 \\ 0.58 & 0 & -0.76 & 0 & -0.31 & 0 \\ 0.58 & 0 & 0.11 & 0 & 0.81 & 0 \\ 0 & -0.58 & 0 & 0 & 0 & -0.82 \\ 0 & -0.5 & 0 & -0.71 & 0 & 0.41 \\ 0 & -0.5 & 0 & 0.71 & 0 & 0.41 \end{bmatrix}$$

A Slightly More Complicated Example



$$W = \begin{bmatrix} 0 & 0 & 0.7 & 0 & 0 & 0 \\ 0 & 0 & 0.5 & 0 & 0 & 0 \\ 0.7 & 0.5 & 0 & 0.1 & 0 & 0 \\ 0 & 0 & 0.1 & 0 & 0.8 & 0.8 \\ 0 & 0 & 0 & 0.8 & 0 & 0 \\ 0 & 0 & 0 & 0.8 & 0 & 0 \end{bmatrix} \quad D = \begin{bmatrix} 0.7 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1.3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1.7 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.8 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.8 \end{bmatrix}$$

$$L = \begin{bmatrix} 0.7 & 0 & -0.7 & 0 & 0 & 0 \\ 0 & 0.5 & -0.5 & 0 & 0 & 0 \\ -0.7 & -0.5 & 1.3 & -0.1 & 0 & 0 \\ 0 & 0 & -0.1 & 1.7 & -0.8 & -0.8 \\ 0 & 0 & 0 & -0.8 & 0.8 & 0 \\ 0 & 0 & 0 & -0.8 & 0 & 0.8 \end{bmatrix} \quad \Lambda = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.06 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.58 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.80 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1.88 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2.47 \end{bmatrix}$$

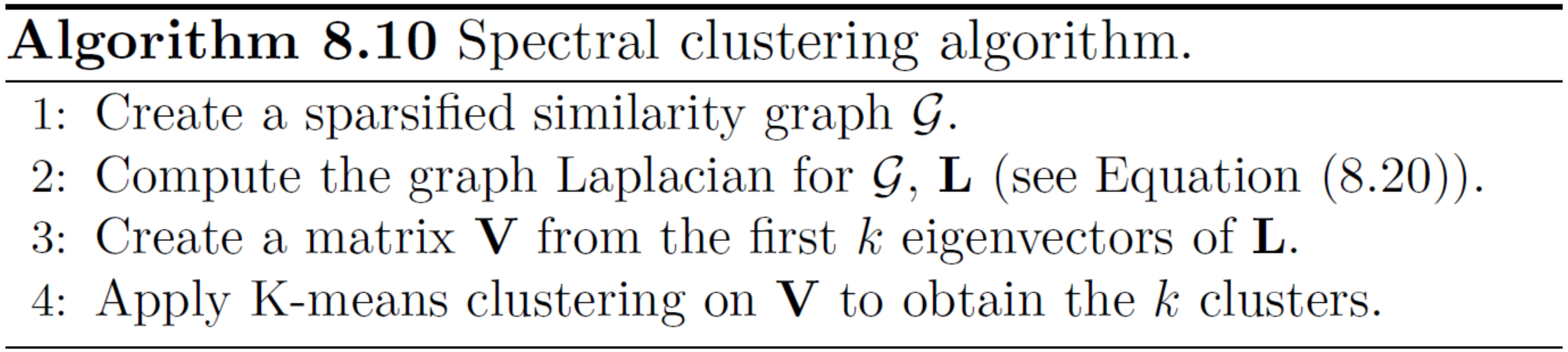
$$V = \begin{bmatrix} 0.41 & -0.41 & 0.65 & 0 & 0.48 & -0.04 \\ 0.41 & -0.43 & -0.75 & 0 & 0.29 & 0.03 \\ 0.41 & -0.38 & 0.11 & 0 & -0.81 & 0.10 \\ 0.41 & 0.38 & 0 & 0 & 0.08 & -0.82 \\ 0.41 & 0.42 & 0 & -0.71 & 0.06 & 0.39 \\ 0.41 & 0.42 & 0 & 0.71 & 0.06 & 0.39 \end{bmatrix}$$

Spectral Graph Clustering Algorithm

Given the Laplacian of a graph, it is easy to define a spectral graph clustering algorithm

We simply apply k-means to the matrix consisting of the first k eignenvectors of L

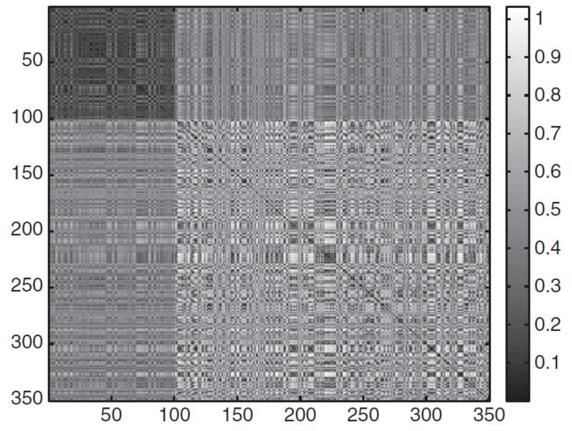
Note that we cluster the rows of that matrix



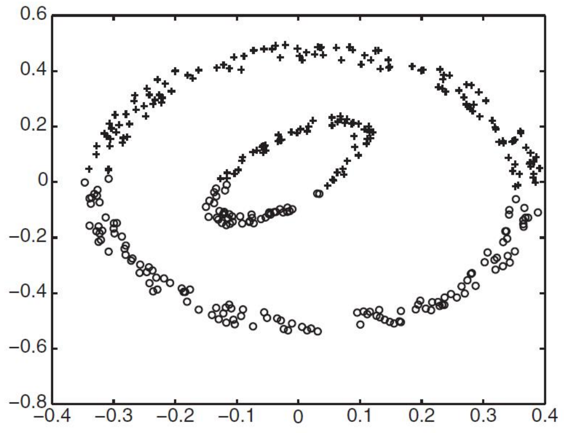
Algorithm 8.10 Spectral clustering algorithm.

```
1: Create a sparsified similarity graph $\mathcal{G}$.
2: Compute the graph Laplacian for $\mathcal{G}$, $\mathbf{L}$ (see Equation (8.20)).
3: Create a matrix $\mathbf{V}$ from the first $k$ eigenvectors of $\mathbf{L}$.
4: Apply K-means clustering on $\mathbf{V}$ to obtain the $k$ clusters.
```

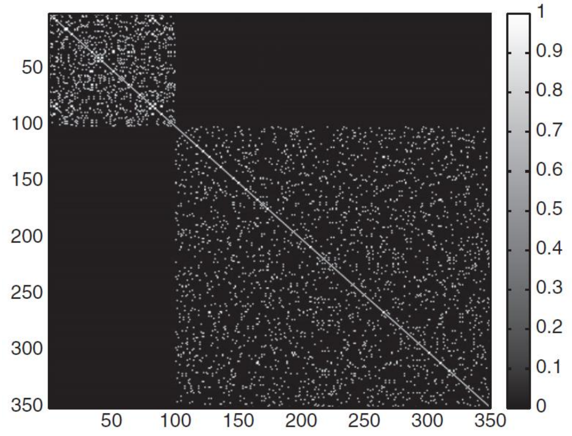
Application of K-means and Spectral Clustering to a 2-D Ring



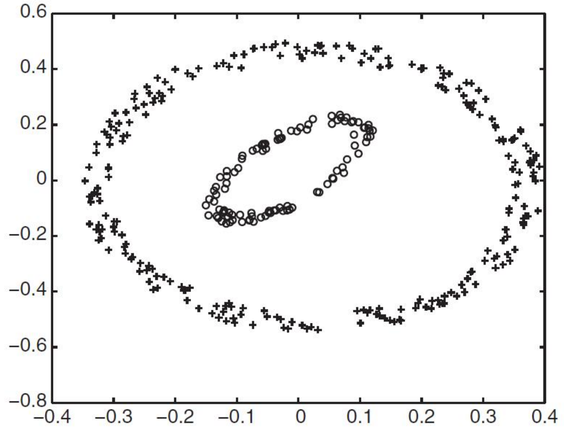
(a) Heat map of Euclidean distance.



(b) Results of K-means clustering.



(c) Heat map of sparsified similarity.



(d) Results of spectral clustering.

Strengths and Limitations

Can detect clusters of different shape and sizes

Sensitive to how graph is created and sparsified

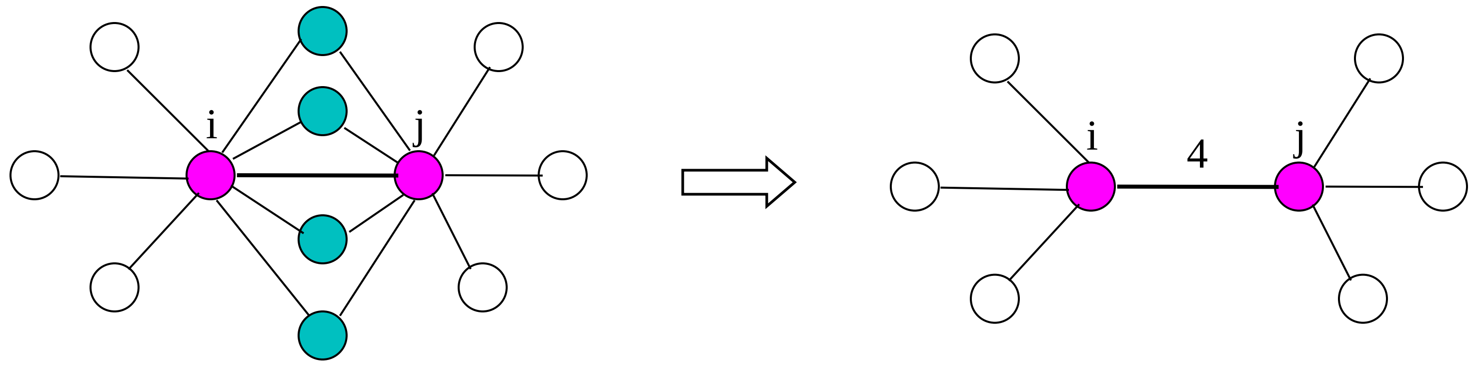
Sensitive to outliers

Time complexity depends on the sparsity of the data matrix

- Improved by sparsification

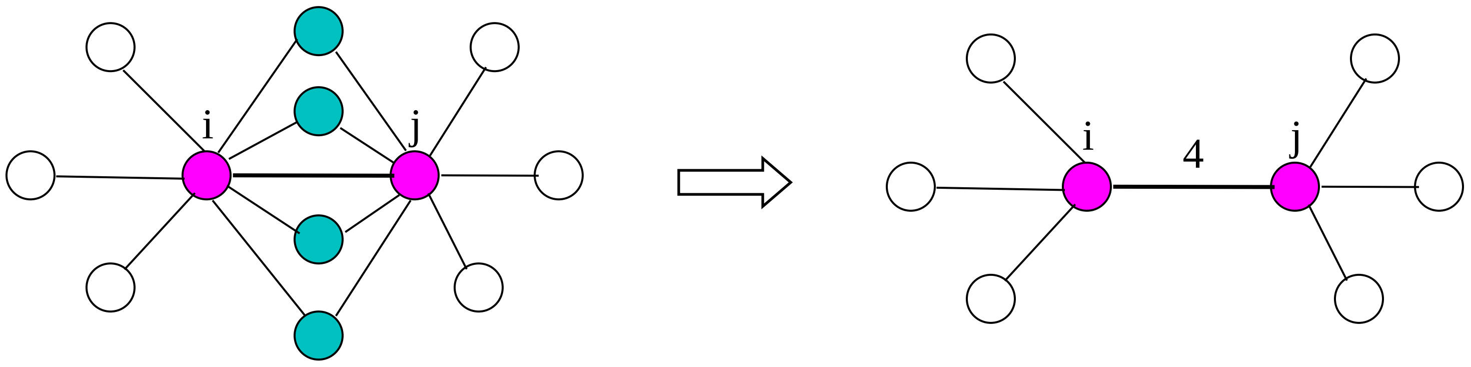
Graph-Based Clustering: SNN Approach

Shared Nearest Neighbor (SNN) graph: the weight of an edge is the number of shared nearest neighbors between vertices given that the vertices are connected



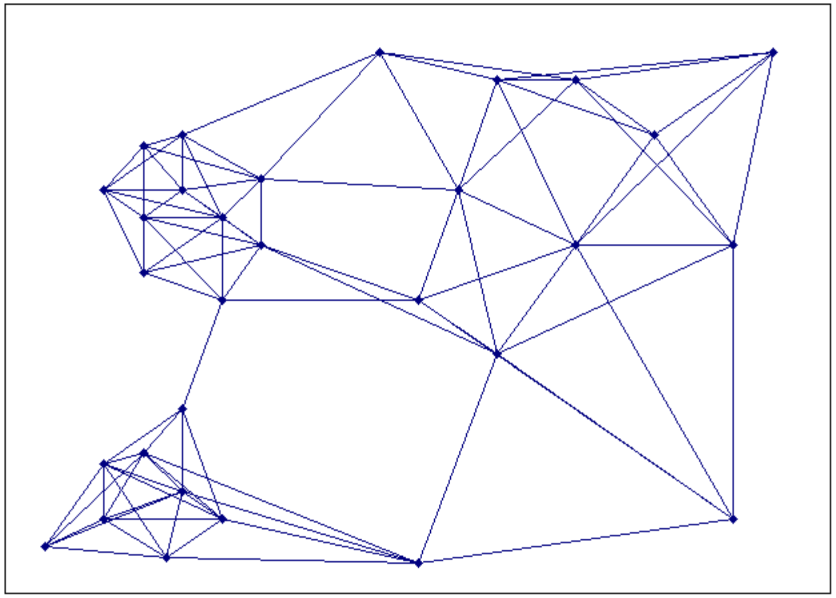
Graph-Based Clustering: SNN Approach

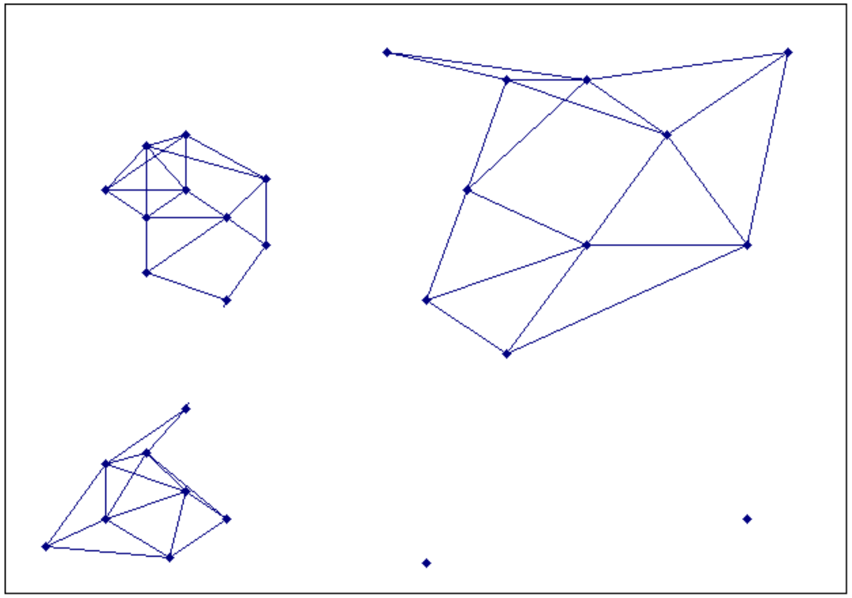
Shared Nearest Neighbor (SNN) graph: the weight of an edge is the number of shared neighbors between vertices given that the vertices are connected



If two points are similar to many of the same points, then they are likely similar to one another, even if a direct measurement of similarity does not indicate this.

Creating the SNN Graph





Sparse Graph

Shared Near Neighbor Graph

Link weights are similarities between neighboring points

Link weights are number of Shared Nearest Neighbors

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

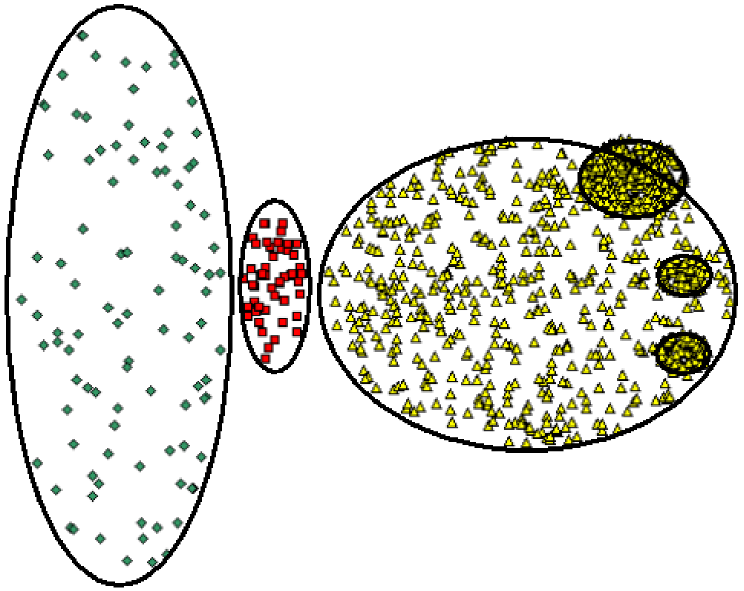
A pair of points is put in the same cluster if

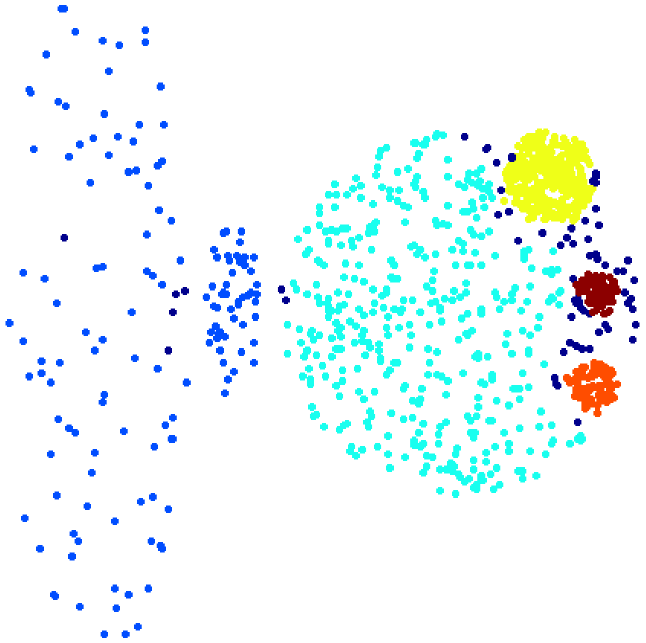
- any two points share more than T neighbors and
- the two points are in each others k nearest neighbor list

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

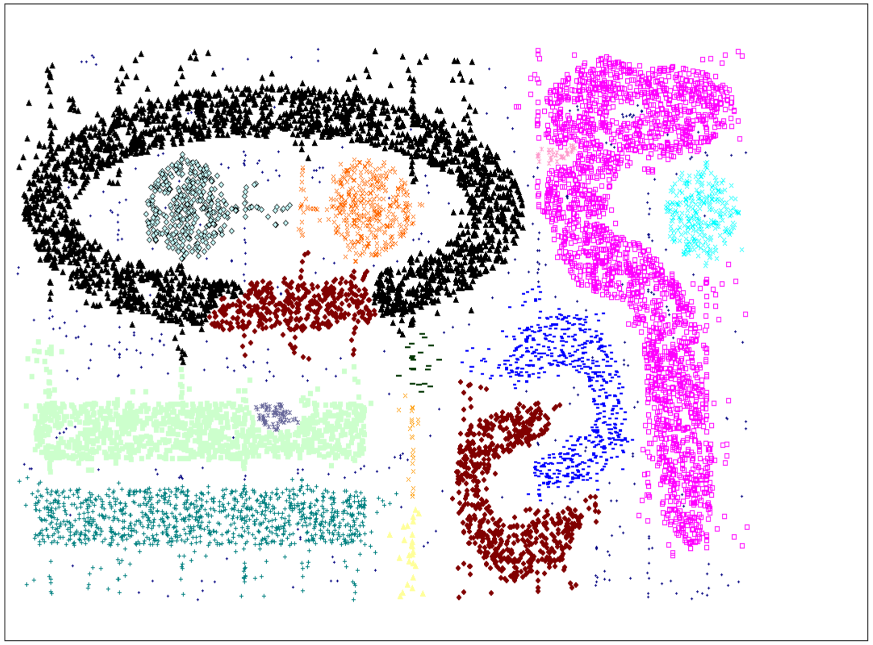
When Jarvis-Patrick Works Reasonably Well

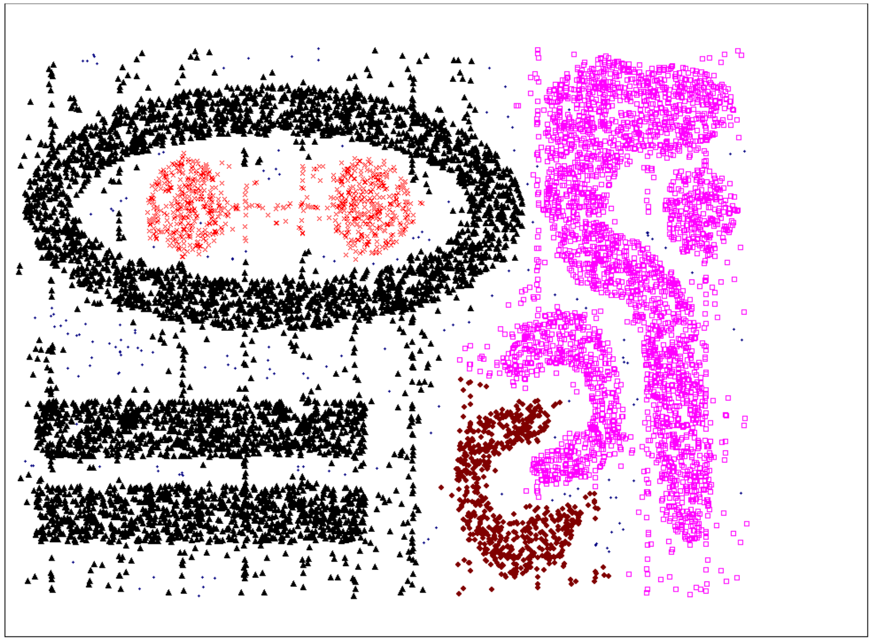

Original Points


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 Density-Based Clustering

Combines:

- Graph based clustering (similarity definition based on number of shared nearest neighbors)
- Density based clustering (DBScan-like approach)

SNN density measures whether a point is surrounded by similar points (with respect to its nearest neighbors)

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*

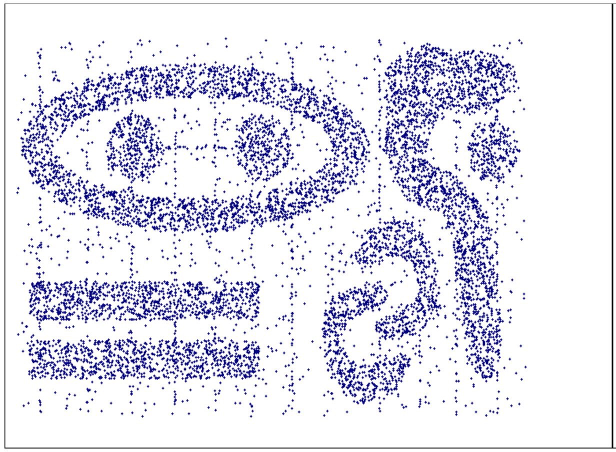
6. **Form clusters from the core points**
   If two core points are within a “radius”, *Eps*, of each other they are placed in the same cluster

7. **Discard all noise points**
   All non-core points that are not within a “radius” of *Eps* of a core point are discarded

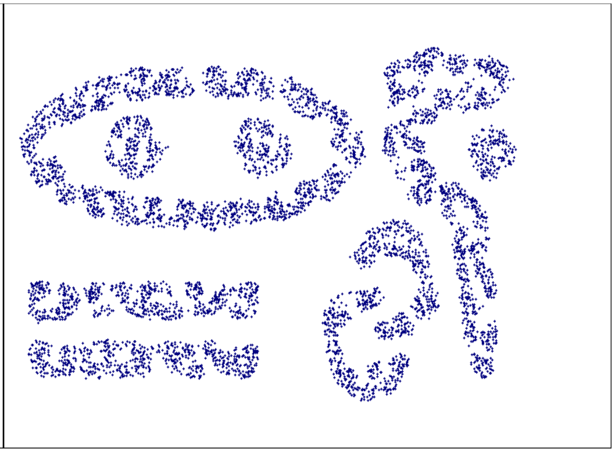
8. **Assign all non-noise, non-core points to clusters**
   This can be done by assigning such points to the nearest core point

   (Note that steps 4-8 are DBSCAN)

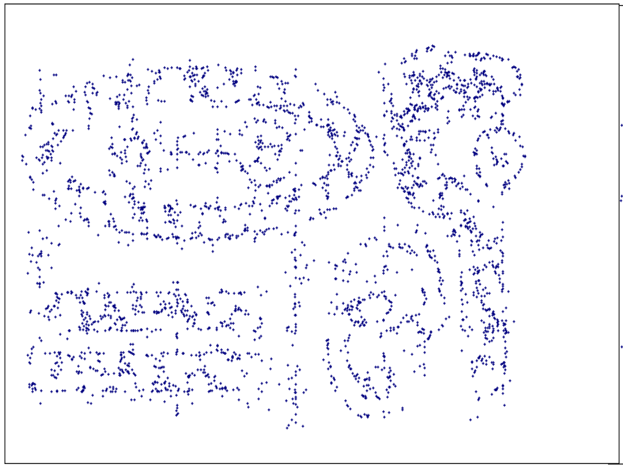
SNN Density



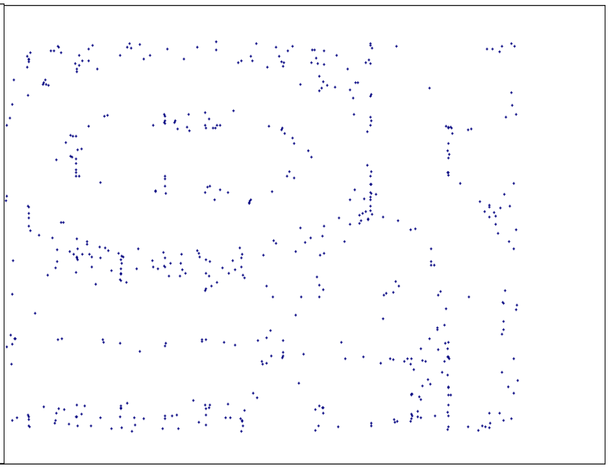
a) All Points



b) High SNN Density

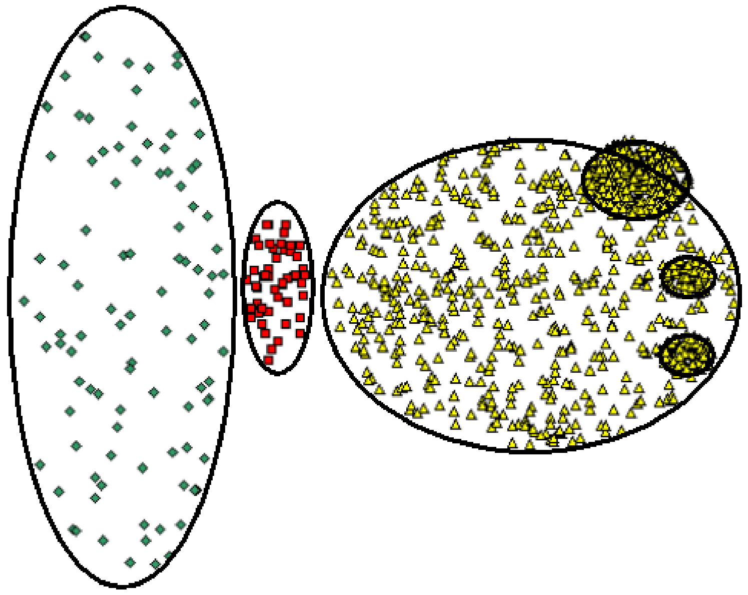


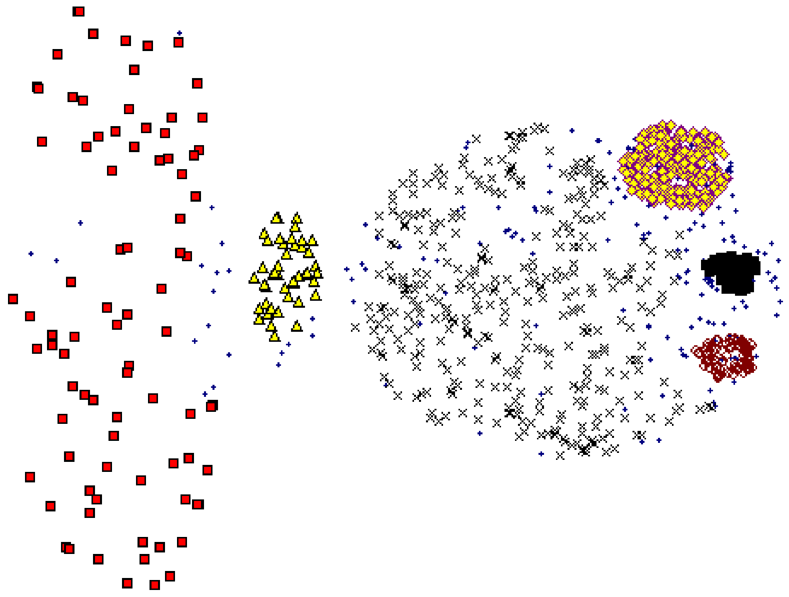
c) Medium SNN Density



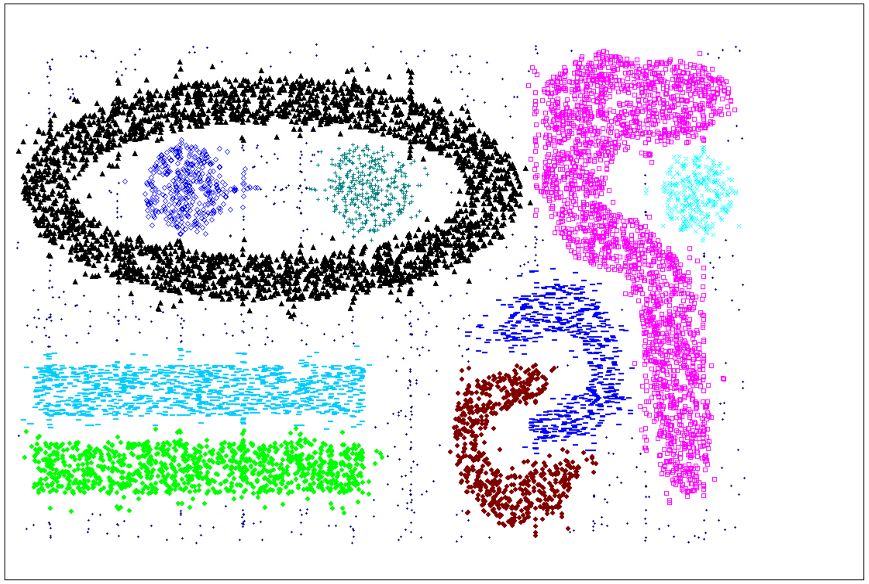
d) Low SNN Density

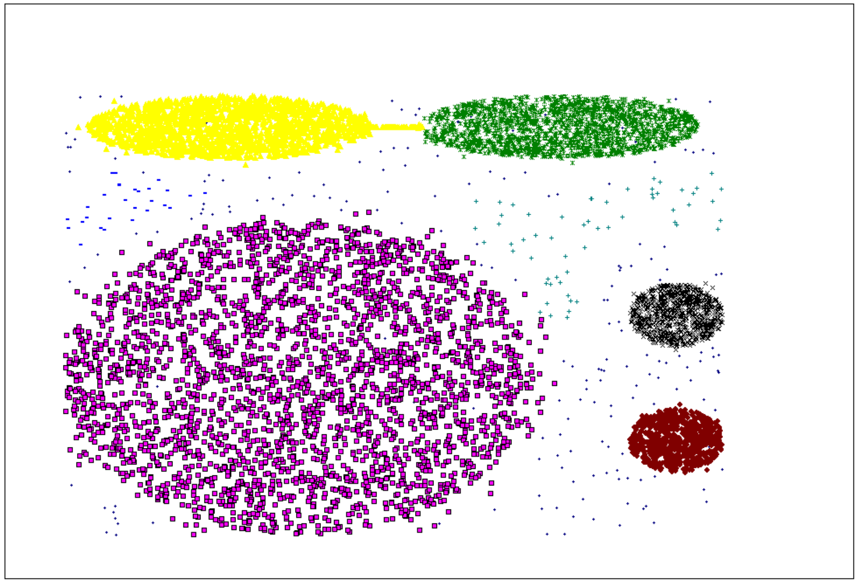
SNN Clustering Can Handle Differing Densities


Original Points


SNN Clustering

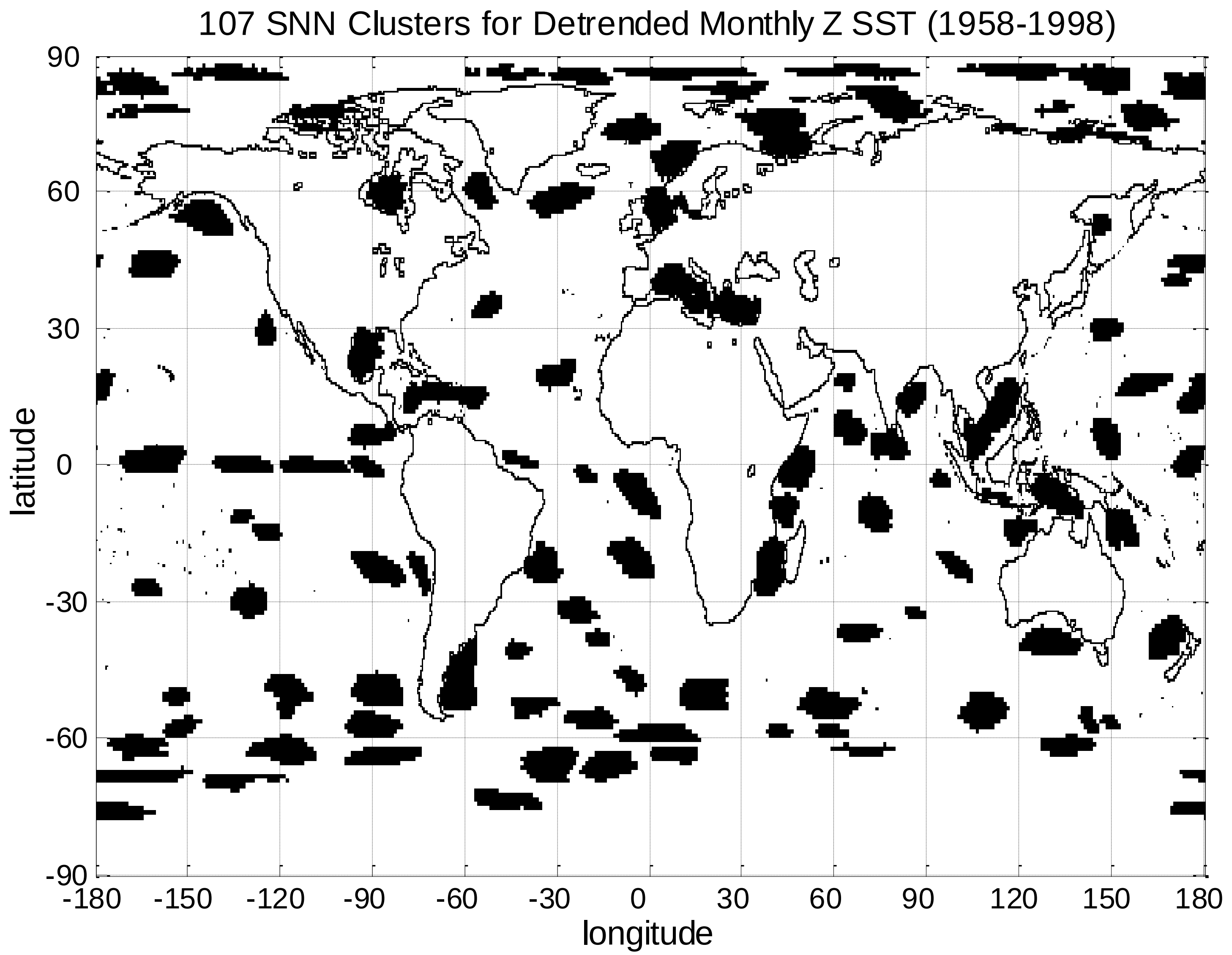
SNN Clustering Can Handle Other Difficult Situations





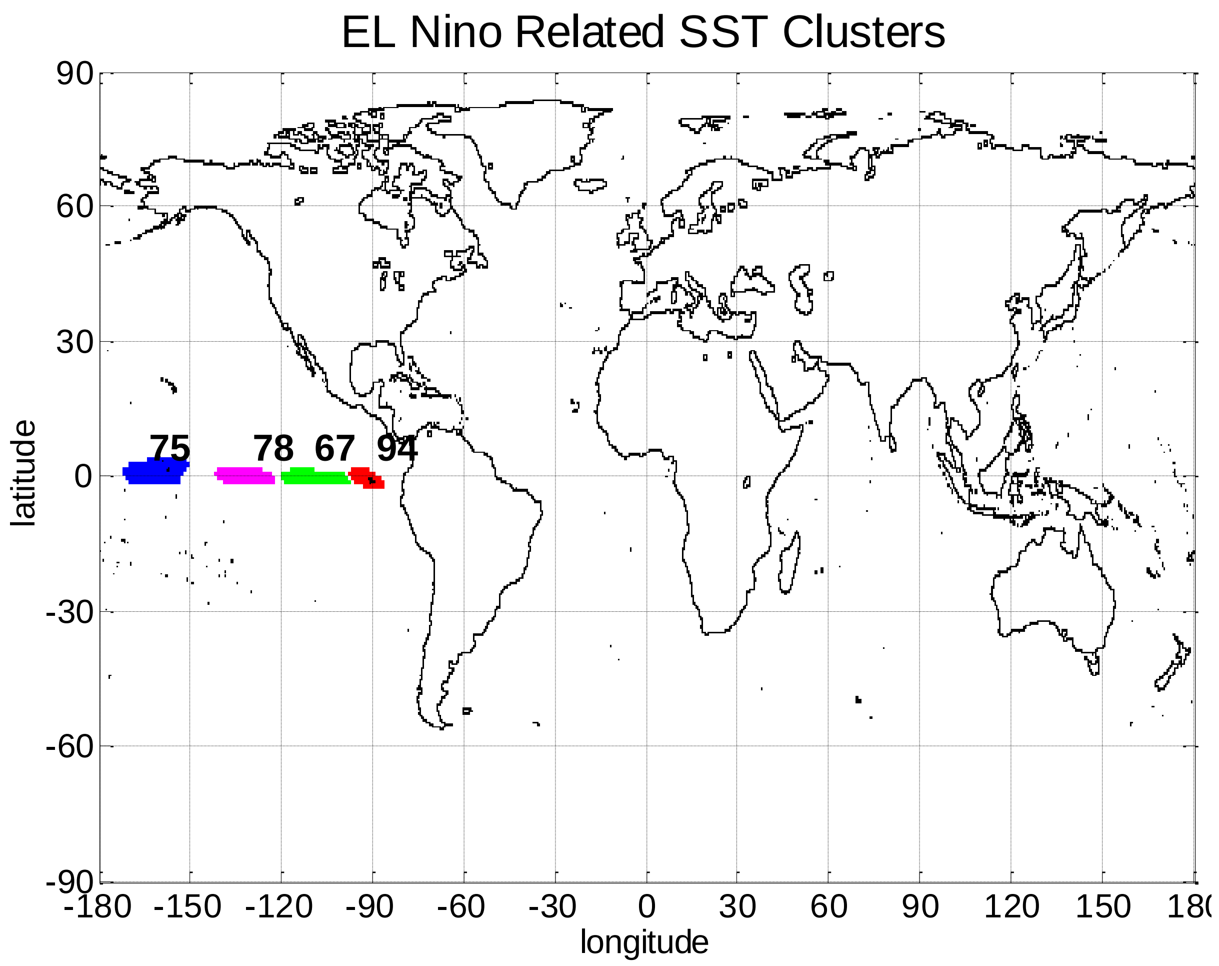
SNN Clusters in Sea-Surface Temperature (SST) Time-series Data

Steinbach, et al (KDD 2003)



SST Clusters that Correspond to El Nino Climate Indices

Steinbach, et al (KDD 2003)



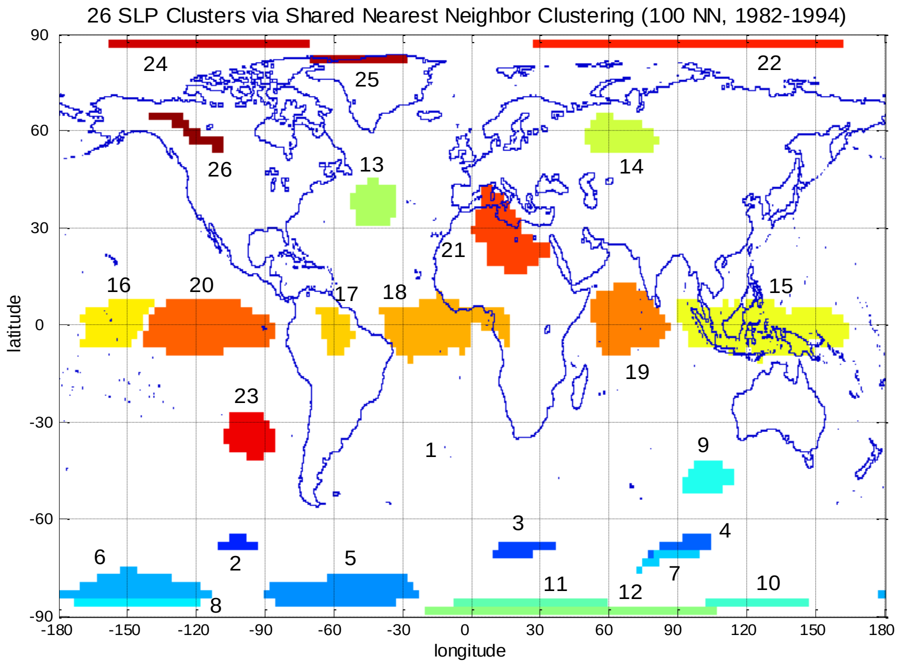
Niño Region	Range Longitude	Range Latitude
1+2 (94)	90°W-80°W	10°S-0°
3 (67)	150°W-90°W	5°S-5°N
3.4 (78)	170°W-120°W	5°S-5°N
4 (75)	160°E-150°W	5°S-5°N

El Nino Regions Defined by Earth Scientists

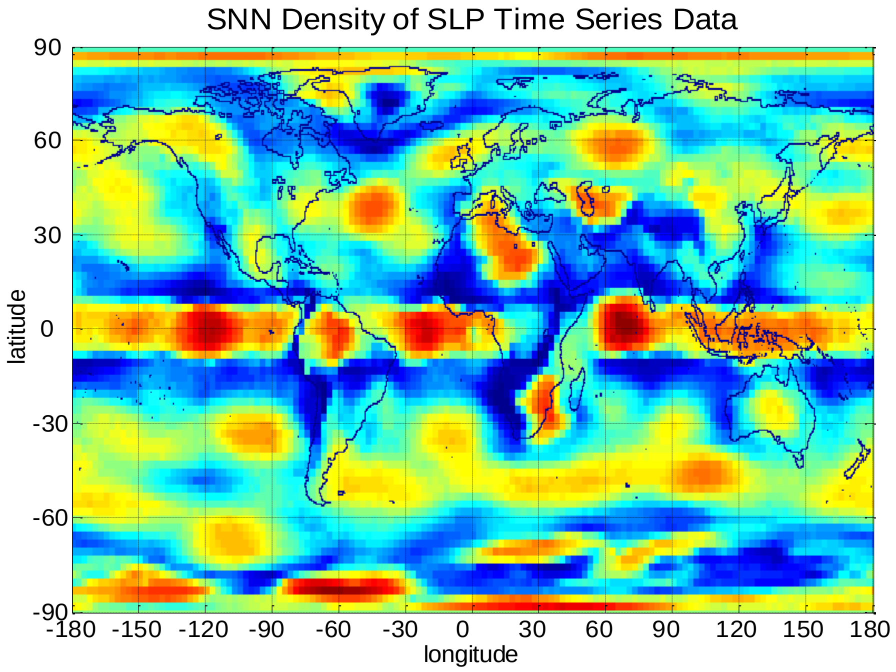
SNN clusters of SST that are highly correlated with El Nino indices, ~ 0.93 correlation.

SNN Clusters in Sea-Level-Pressure (SLP) Time-series Data

Steinbach, et al (KDD 2003)

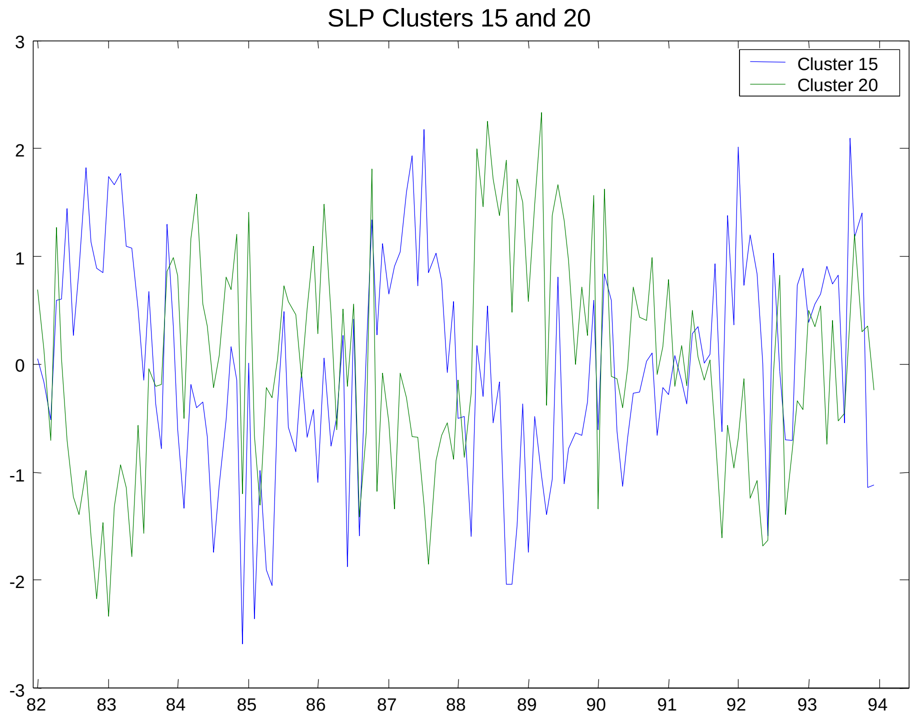

26 SLP Clusters via Shared Nearest Neighbor Clustering (100 NN, 1982-1994)
latitude
longitude
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26

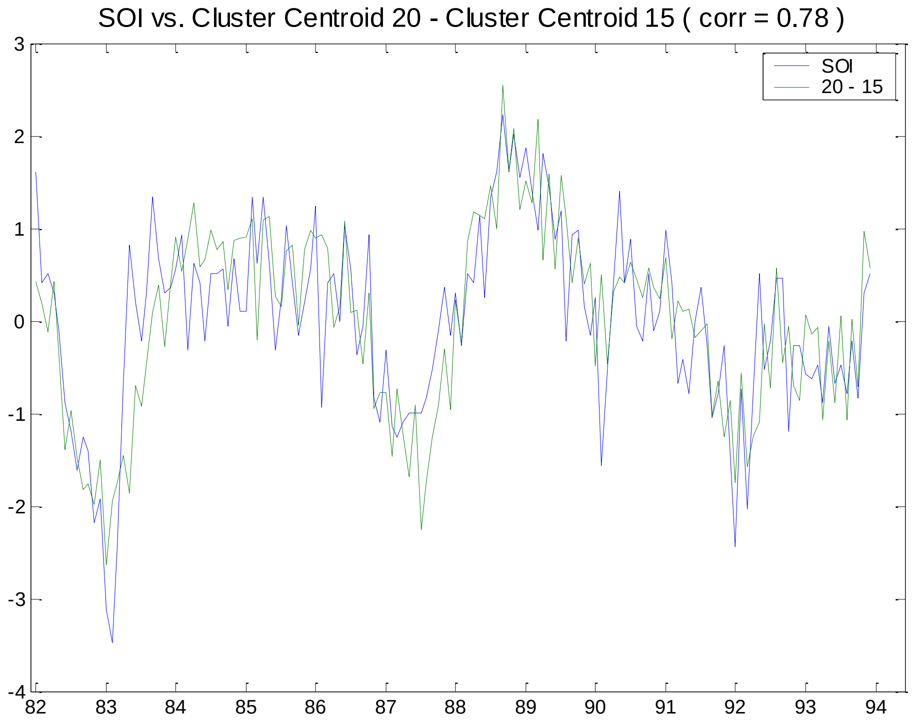
SNN Clusters of SLP.



SNN Density of Points on the Globe.

Pairs of SLP Clusters that Correspond to El-Nino SOI





Centroids of SLP clusters 15 and 20 (near Darwin, Australia and Tahiti) 1982-1993.

Centroid of cluster 20 – Centroid of cluster 15 versus SOI

Limitations of SNN Clustering

- 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
- Parameterization is not easy

Characteristics of Data, Clusters, and Clustering Algorithms

A cluster analysis is affected by characteristics of

- Data
- Clusters
- Clustering algorithms

Looking at these characteristics gives us a number of dimensions that you can use to describe clustering algorithms and the results that they produce

Characteristics of Data

- High dimensionality
  - Dimensionality reduction
- Types of attributes
  - Binary, discrete, continuous, asymmetric
  - Mixed attribute types (e.g., some are continuous, others nominal)
- Differences in attribute scales
  - Normalization techniques
- Size of data set
- Noise and Outliers
- Properties of the data space
  - Can you define a meaningful centroid or a meaningful notion of density

Characteristics of Clusters

- Data distribution
  - Parametric models
- Shape
  - Globular or arbitrary shape
- Differing sizes
- Differing densities
- Level of separation among clusters
- Relationship among clusters
- Subspace clusters

Characteristics of Clustering Algorithms

- Order dependence
- Non-determinism
- Scalability
- Number of parameters

Which Clustering Algorithm?

- Type of Clustering
  - Taxonomy vs flat
- Type of Cluster
  - Prototype vs connected reguions vs density-based
- Characteristics of Clusters
  - Subspace clusters, spatial inter-relationships
- Characteristics of Data Sets and Attributes
- Noise and Outliers
- Number of Data Objects
- Number of Attributes
- Algorithmic Considerations

Comparison of MIN and EM-Clustering

We assume EM clustering using the Gaussian (normal) distribution.

MIN is hierarchical, EM clustering is partitional.

Both MIN and EM clustering are complete.

MIN has a graph-based (contiguity-based) notion of a cluster, while EM clustering has a prototype (or model-based) notion of a cluster.

MIN will not be able to distinguish poorly separated clusters, but EM can manage this in many situations.

MIN can find clusters of different shapes and sizes; EM clustering prefers globular clusters and can have trouble with clusters of different sizes.

Min has trouble with clusters of different densities, while EM can often handle this.

Neither MIN nor EM clustering finds subspace clusters.

Comparison of MIN and EM-Clustering

- MIN can handle outliers, but noise can join clusters; EM clustering can tolerate noise, but can be strongly affected by outliers.
- EM can only be applied to data for which a centroid is meaningful; MIN only requires a meaningful definition of proximity.
- EM will have trouble as dimensionality increases and the number of its parameters (the number of entries in the covariance matrix) increases as the square of the number of dimensions; MIN can work well with a suitable definition of proximity.
- EM is designed for Euclidean data, although versions of EM clustering have been developed for other types of data. MIN is shielded from the data type by the fact that it uses a similarity matrix.
- MIN makes no distribution assumptions; the version of EM we are considering assumes Gaussian distributions.

Comparison of MIN and EM-Clustering

- EM has an O(n) time complexity; MIN is $O(n^2 \log(n))$.
- Because of random initialization, the clusters found by EM can vary from one run to another; MIN produces the same clusters unless there are ties in the similarity matrix.
- Neither MIN nor EM automatically determine the number of clusters.
- MIN does not have any user-specified parameters; EM has the number of clusters and possibly the weights of the clusters.
- EM clustering can be viewed as an optimization problem; MIN uses a graph model of the data.
- Neither EM or MIN are order dependent.

Comparison of DBSCAN and K-means

Both are partitional.

K-means is complete; DBSCAN is not.

K-means has a prototype-based notion of a cluster; DB uses a density-based notion.

K-means can find clusters that are not well-separated. DBSCAN will merge clusters that touch.

DBSCAN handles clusters of different shapes and sizes; K-means prefers globular clusters.

Comparison of DBSCAN and K-means

DBSCAN can handle noise and outliers; K-means performs poorly in the presence of outliers

K-means can only be applied to data for which a centroid is meaningful; DBSCAN requires a meaningful definition of density

DBSCAN works poorly on high-dimensional data; K-means works well for some types of high-dimensional data

Both techniques were designed for Euclidean data, but extended to other types of data

DBSCAN makes no distribution assumptions; K-means is really assuming spherical Gaussian distributions

Comparison of DBSCAN and K-means

K-means has an O(n) time complexity; DBSCAN is O(n^2)

Because of random initialization, the clusters found by K-means can vary from one run to another; DBSCAN always produces the same clusters

DBSCAN automatically determines the number of clusters; K-means does not

K-means has only one parameter, DBSCAN has two.

K-means clustering can be viewed as an optimization problem and as a special case of EM clustering; DBSCAN is not based on a formal model.