Cluster Analysis: Advanced Concepts and Algorithms

Dr. Meng Qu
Rutgers University

Outline

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

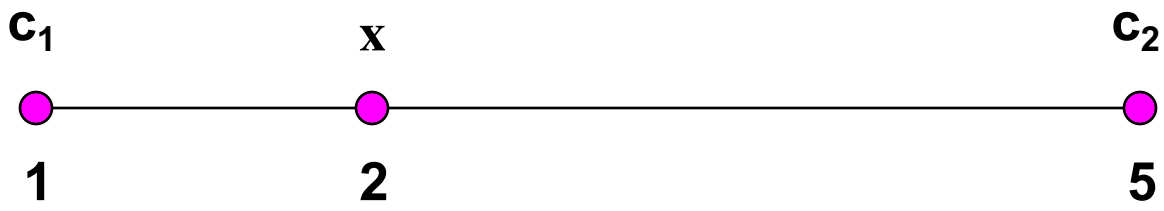
Hard (Crisp) vs Soft (Fuzzy) Clustering

- Hard (Crisp) vs. Soft (Fuzzy) clustering
  - Generalize K-means objective function

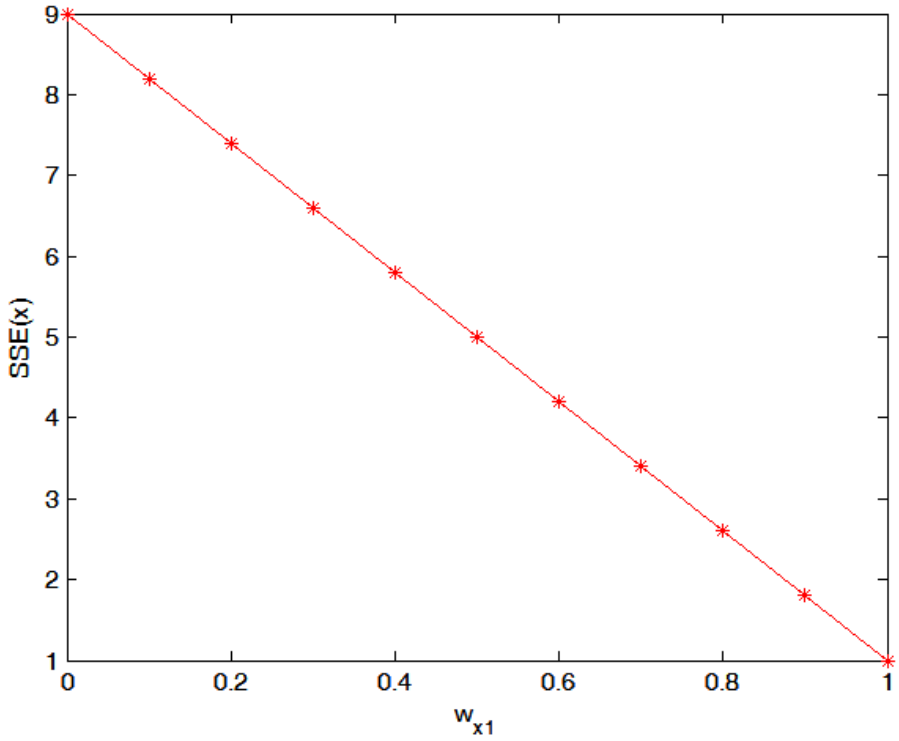
$$SSE = \sum_{j=1}^{k} \sum_{i=1}^{N} w_{ij} \left(x_i - c_j \right)^2, \qquad \sum_{j=1}^{k} w_{ij} = 1$$

    - w_{ij} : weight with which object x_i belongs to cluster C_j
  - To minimize SSE, repeat the following steps:
    - Fixed c_j and determine w_{ij} (cluster assignment)
    - Fixed w_{ij} and recompute c_j
  - Hard clustering: $w_{ij} \in \{0,1\}$

Hard (Crisp) vs Soft (Fuzzy) Clustering



$$SSE(x) = w_{x1}(2-1)^2 + w_{x2}(5-2)^2$$
$$= w_{x1} + 9w_{x2}$$



SSE(x) is minimized when $w_{x1} = 1$, $w_{x2} = 0$

Fuzzy C-means

- Objective function

p: fuzzifier (p > 1)

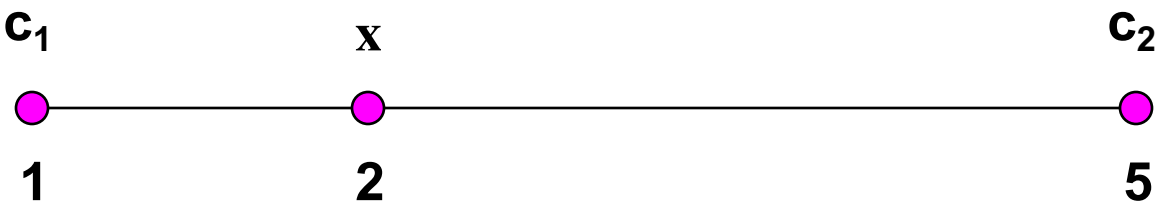
$$SSE = \sum_{j=1}^{k} \sum_{i=1}^{N} w_{ij}^{p} \left(x_i - c_j\right)^2, \qquad \sum_{j=1}^{k} w_{ij} = 1$$

  - w_{ij} : weight with which object x_i belongs to cluster C_j

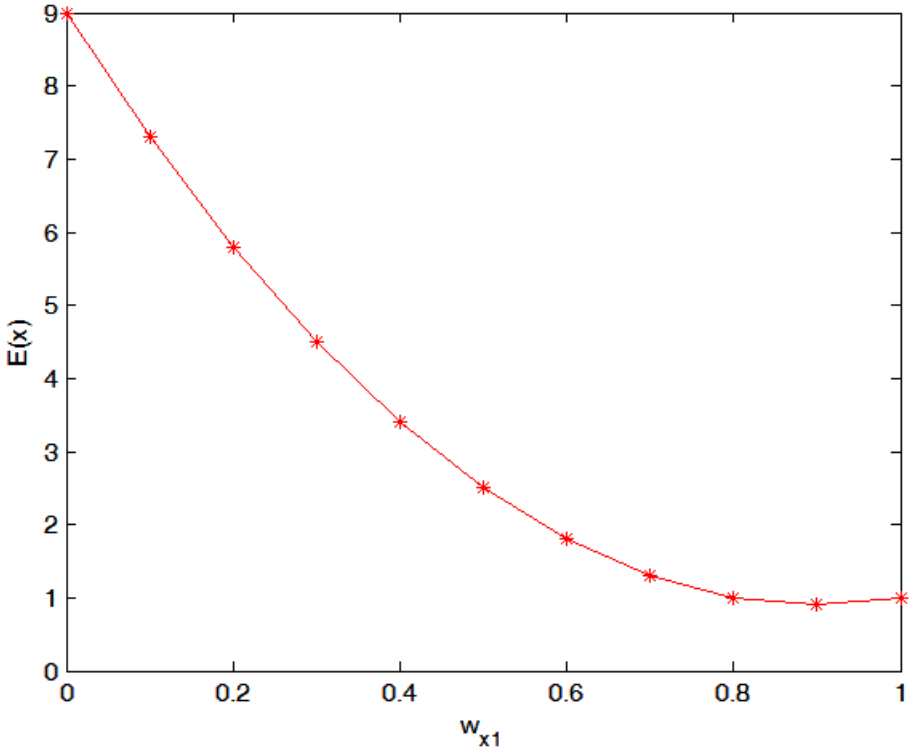
  - To minimize objective function, repeat the following:
    - Fixed c_j and determine w_{ij}
    - Fixed w_{ij} and recompute c_j

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

Fuzzy C-means



$$E(x) = w_{x1}^2(2-1)^2 + w_{x2}^2(5-2)$$
$$= w_{x1}^2 + 9w_{x2}^2$$



SSE(x) is minimized when $w_{x1} = 0.9$, $w_{x2} = 0.1$

Fuzzy C-means

- Objective function:

p: fuzzifier (p > 1)

$$SSE = \sum_{j=1}^{k}\sum_{i=1}^{N} w_{ij}^{p}\left(x_i - c_j\right)^2, \qquad \sum_{j=1}^{k} w_{ij} = 1$$

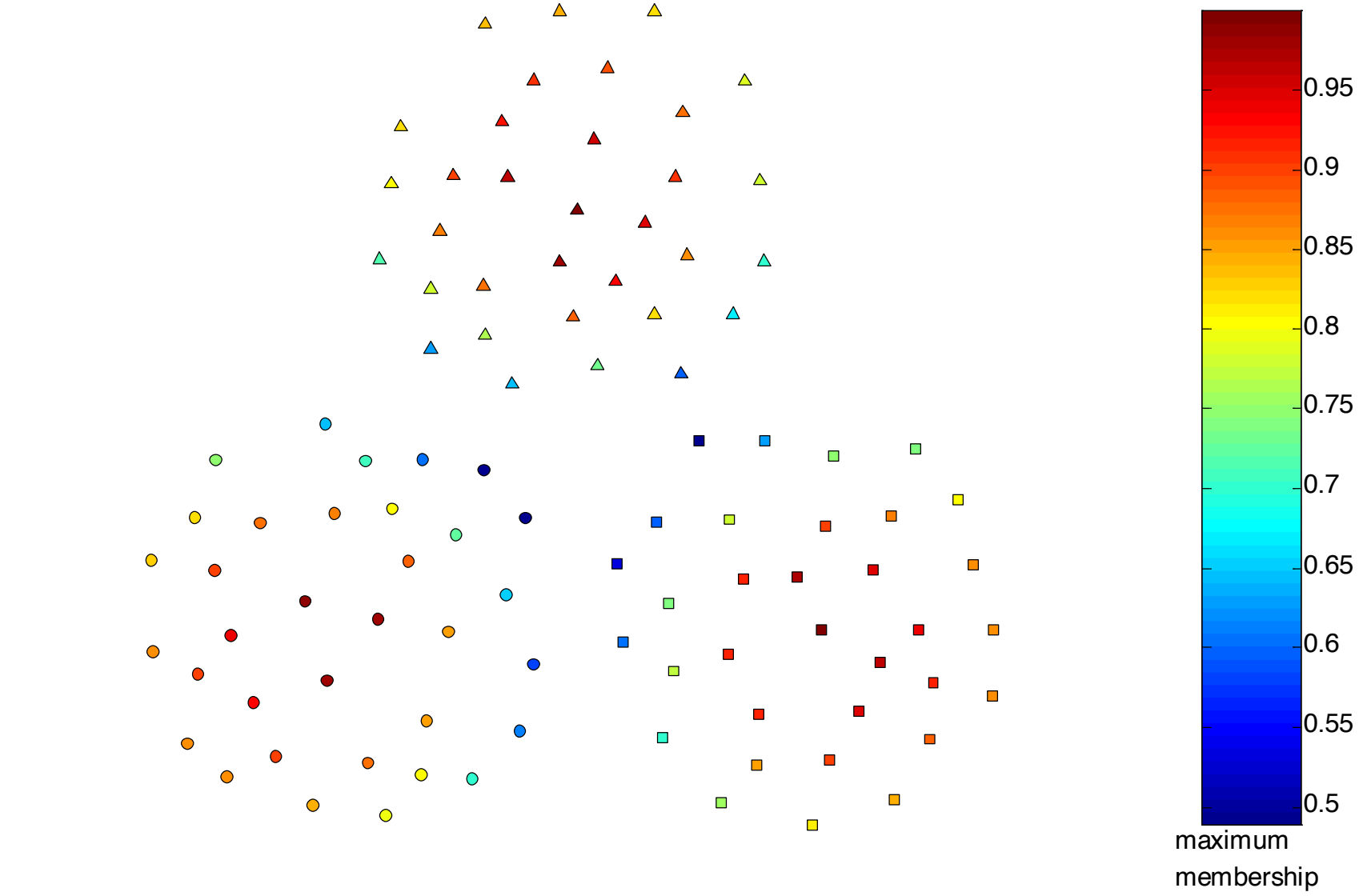
- Initialization: choose the weights w_{ij} randomly
- Repeat:
  - Update centroids:

$$c_j = \sum_{i=1}^{m} w_{ij}^{p}\mathbf{x}_i \Big/ \sum_{i=1}^{m} w_{ij}^{p}$$

  - Update weights:

$$w_{ij} = \left(1/dist(\mathbf{x}_i, \mathbf{c}_j)^2\right)^{\frac{1}{p-1}} \Big/ \sum_{q=1}^{k}\left(1/dist(\mathbf{x}_i, \mathbf{c}_q)^2\right)^{\frac{1}{p-1}}$$

Fuzzy K-means Applied to Sample Data

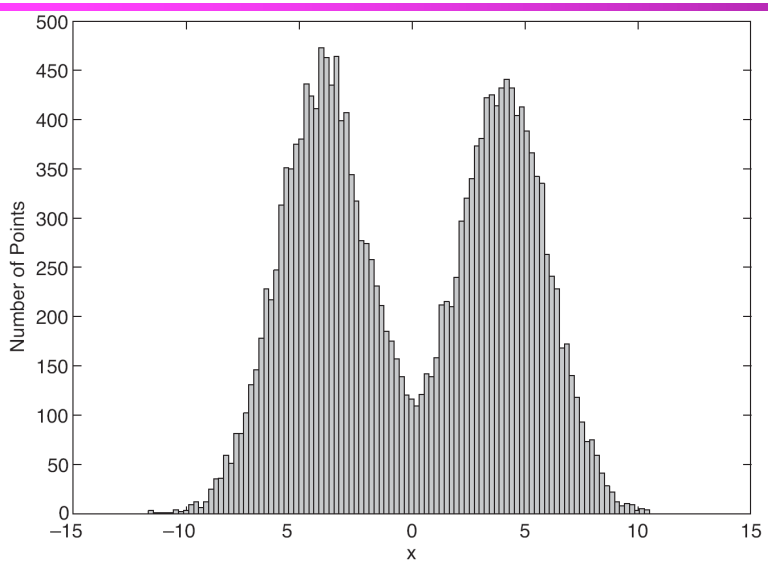


Hard (Crisp) vs Soft (Probabilistic) Clustering

- 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
  - Can use a k-means like algorithm, called the EM algorithm, to estimate these parameters
    - Actually, 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 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) in 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

Expectation-Maximization Algorithm

Initialize the parameters

Repeat

Expectation Step: For each point, calculate the probability that each object belongs to each distribution.

Maximization Step: Given the probabilities from the expectation step, find the new estimates of the parameters that maximize the expected likelihood.

Until the parameters do not change.

(Or, stop if the change in the parameters is below a specified threshold)

EM Algorithm

- 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.

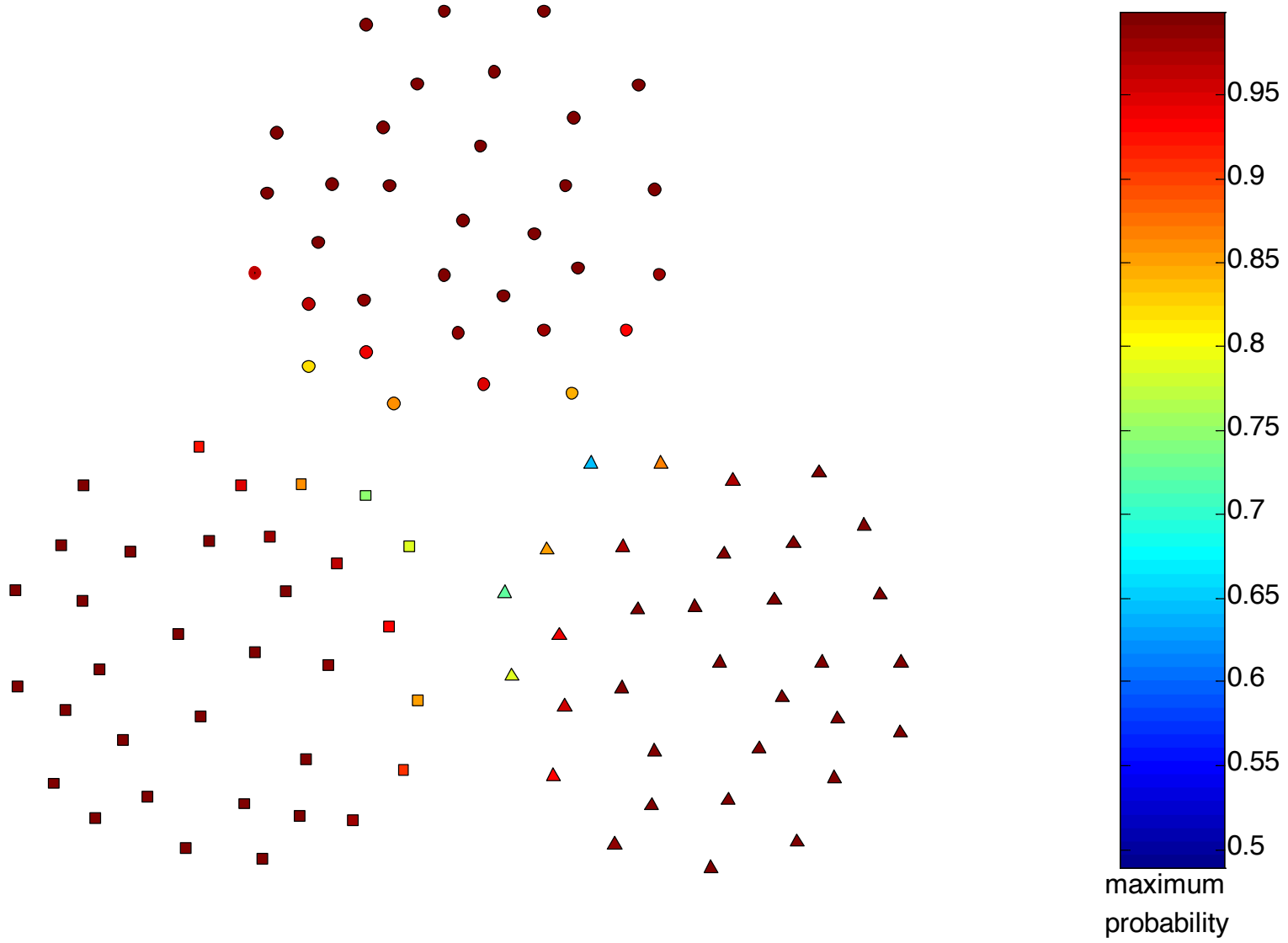
Probabilistic Clustering: Updating Centroids

Update formula for the centroids

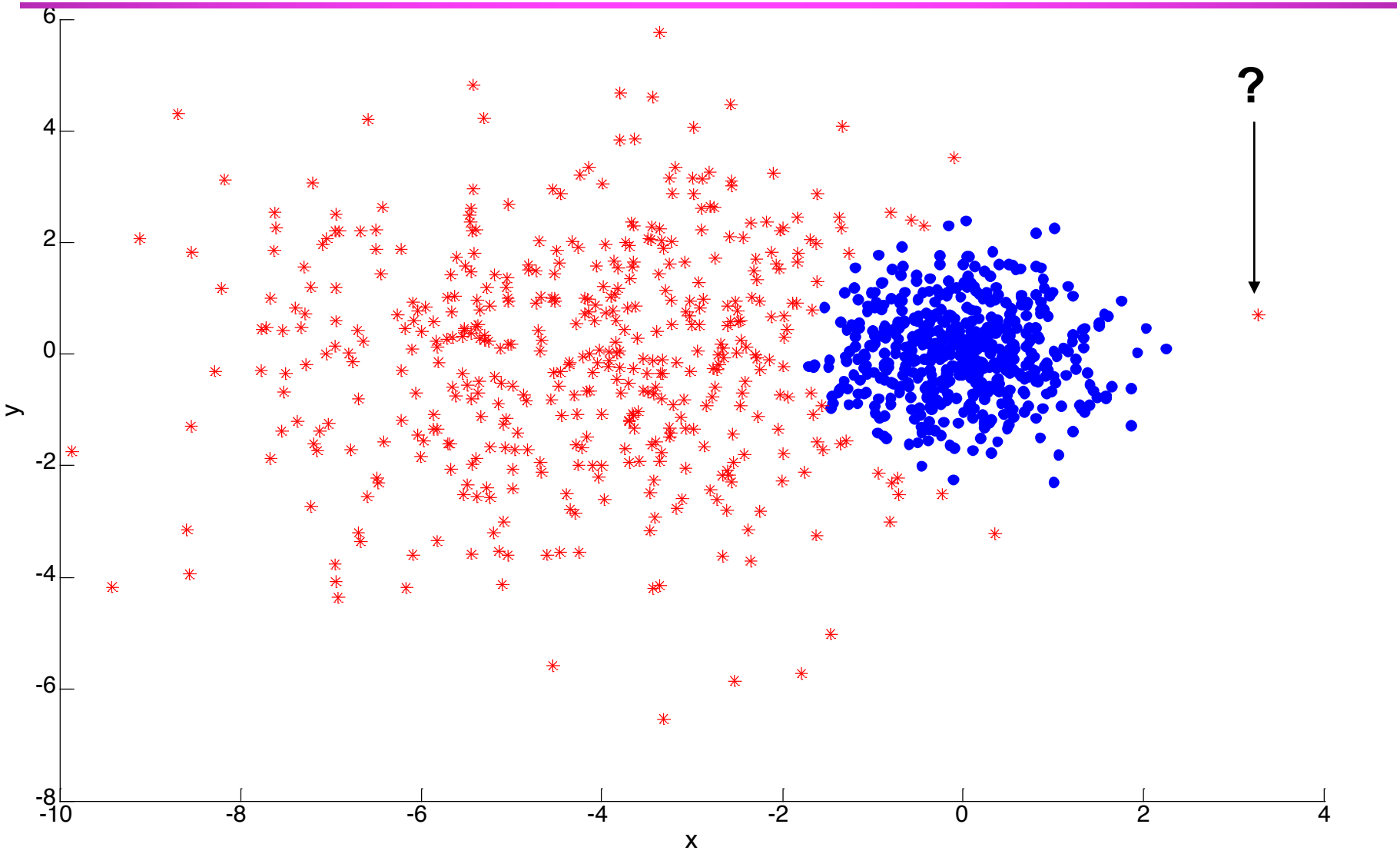
$$c_j = \sum_{i=1}^{m} x_i p(j \mid x_i) / \sum_{i=1}^{m} p(j \mid x_i)$$

- Very similar to the fuzzy k-means formula
  - Weights are not raised to a power
  - Weights are probabilities
  - These probabilities are calculated using Bayes rule
- Need to assign weights to each cluster
  - Weights may not be equal
  - Similar to prior probabilities
  - Can be estimated, if desired

Probabilistic Clustering Applied to Sample Data



Probabilistic Clustering: Dense and Sparse Clusters



Advantages and Limitations of Mixture Model Clustering Using EM

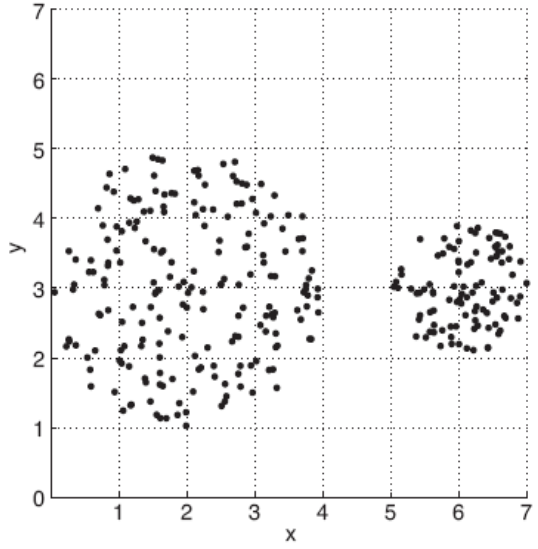
- Advantages:
  - More general than K-means and can find clusters of different sizes and elliptical shapes.
  - Easy to characterize the clusters produced, since they can be described by a small number of parameters.
- Limitations:
  - EM is slow
  - Difficult to estimate the number of clusters, difficult to choose the parameters

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, τ.
4: Form clusters from contiguous (adjacent) groups of dense cells.

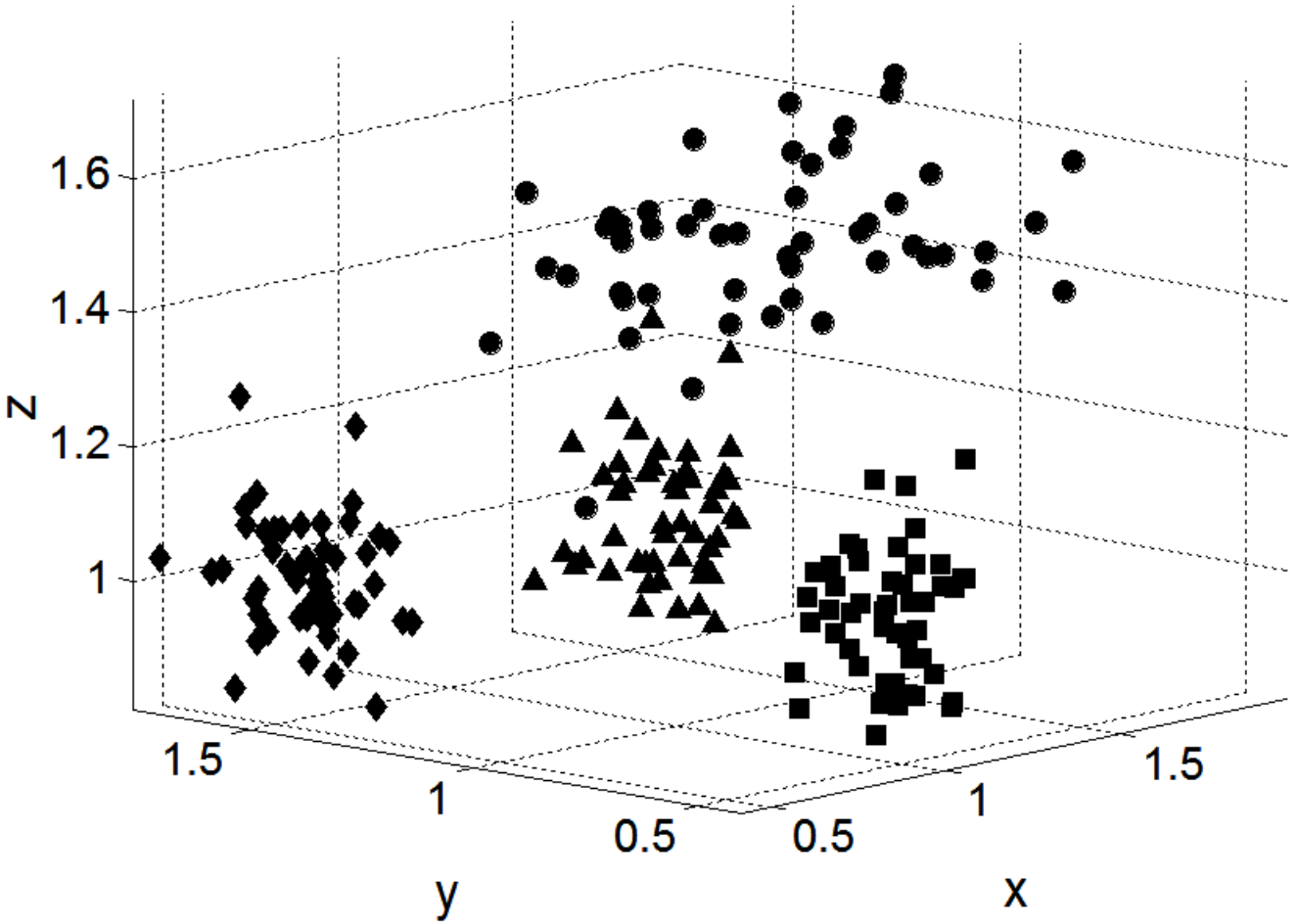


0	0	0	0	0	0	0
0	0	0	0	0	0	0
4	**17**	**18**	**6**	0	0	0
14	**14**	**13**	**13**	0	**18**	**27**
11	**18**	**10**	**21**	0	**24**	**31**
3	**20**	**14**	**4**	0	0	0
0	0	0	0	0	0	0

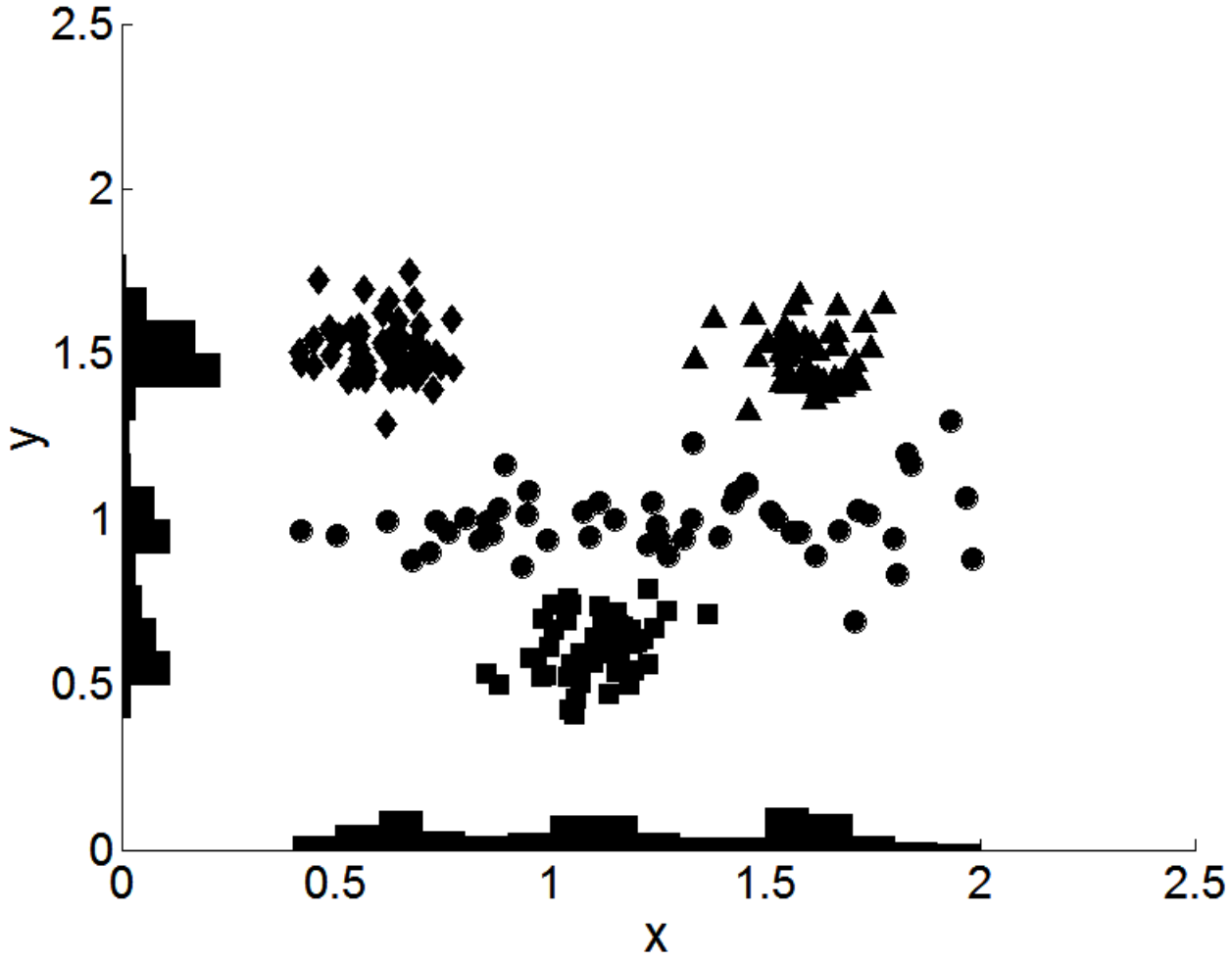
Subspace Clustering

- 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
  - Example:
    - When k-means is used to find document clusters, the resulting clusters can typically be characterized by 10 or so terms

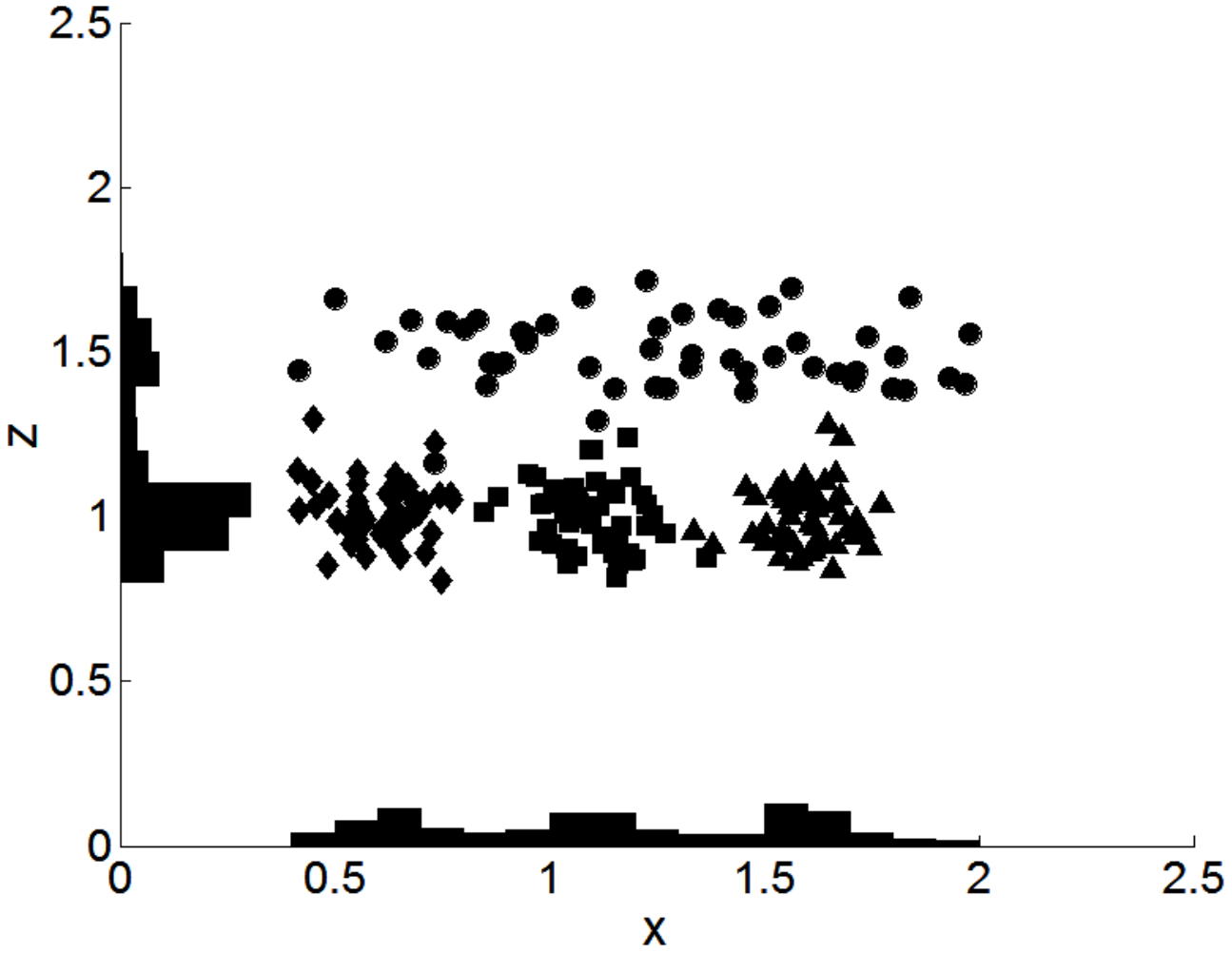
Example



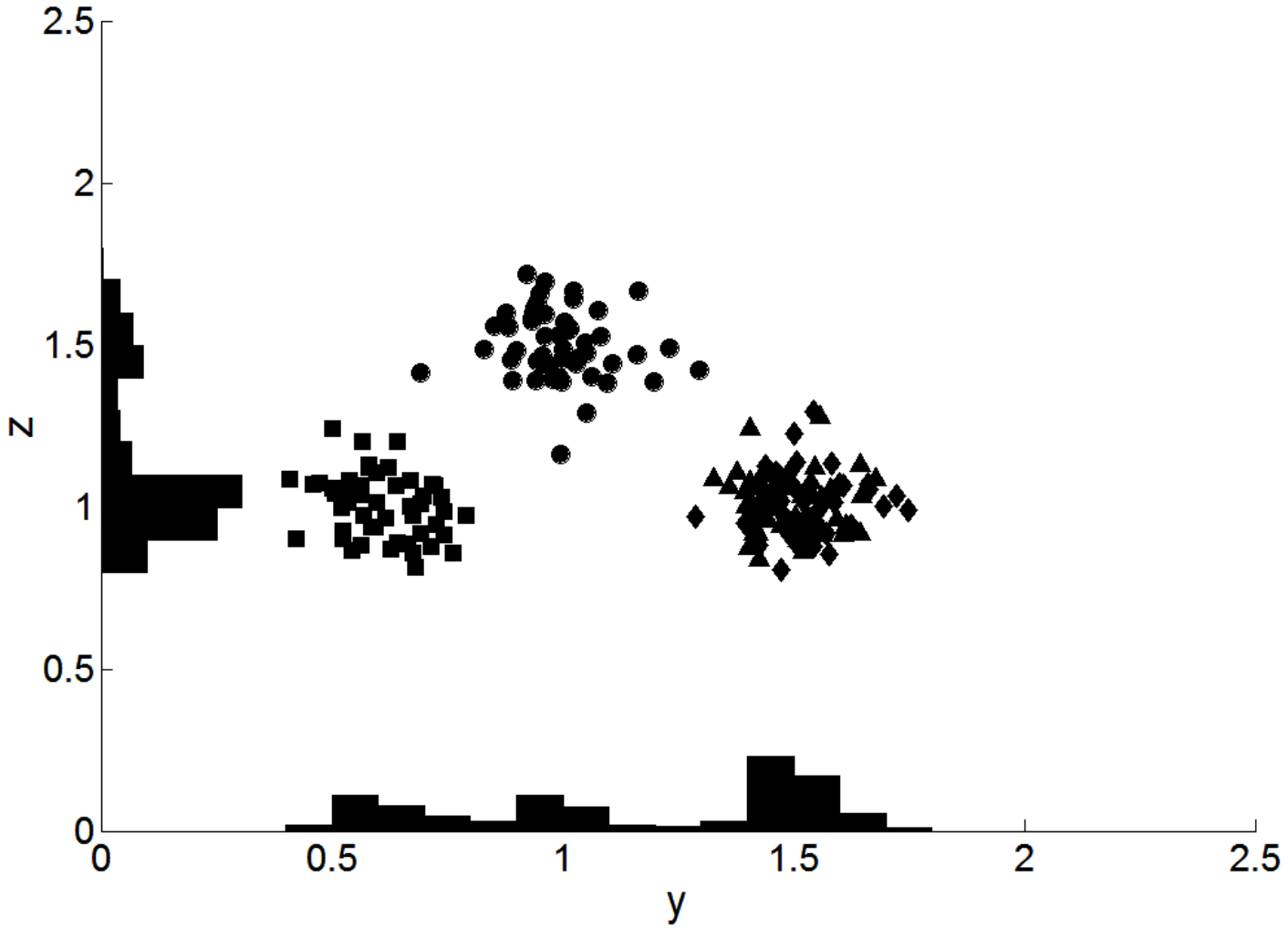
Example



Example



Example



Clique Algorithm - Overview

- A grid-based clustering algorithm that methodically finds subspace clusters
  - Partitions the data space into rectangular units of equal volume
  - 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 group of collections of contiguous (touching) dense units

Clique Algorithm

- It is impractical to check each volume unit to see if it is dense since there is exponential number of such units
- Monotone property of density-based clusters:
  - If a set of points forms a density based cluster in k dimensions, then the same set of points is also part of a density based cluster in all possible subsets 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 ξ 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

Graph-Based Clustering: General Concepts

- 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.

CURE Algorithm: Graph-Based Clustering

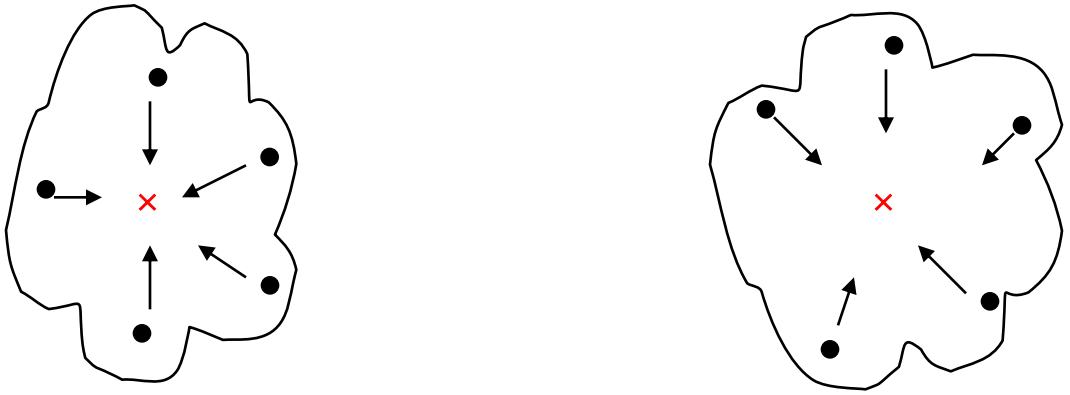
- Agglomerative hierarchical clustering algorithms vary in terms of how the proximity of two clusters are computed
  - MIN (single link)
    - susceptible to noise/outliers
  - MAX (complete link)/GROUP AVERAGE/Centroid/Ward's:
    - may not work well with non-globular clusters
- CURE algorithm tries to handle both problems
  - Starts with a proximity matrix/proximity graph

CURE Algorithm

- Represents a cluster using multiple representative points
  - Representative points are found by selecting a constant number of points from a cluster
    - First representative point is chosen to be the point furthest from the center of the cluster
    - Remaining representative points are chosen so that they are farthest from all previously chosen points

CURE Algorithm

- "Shrink" representative points toward the center of the cluster by a factor, α



- Shrinking representative points toward the center helps avoid problems with noise and outliers

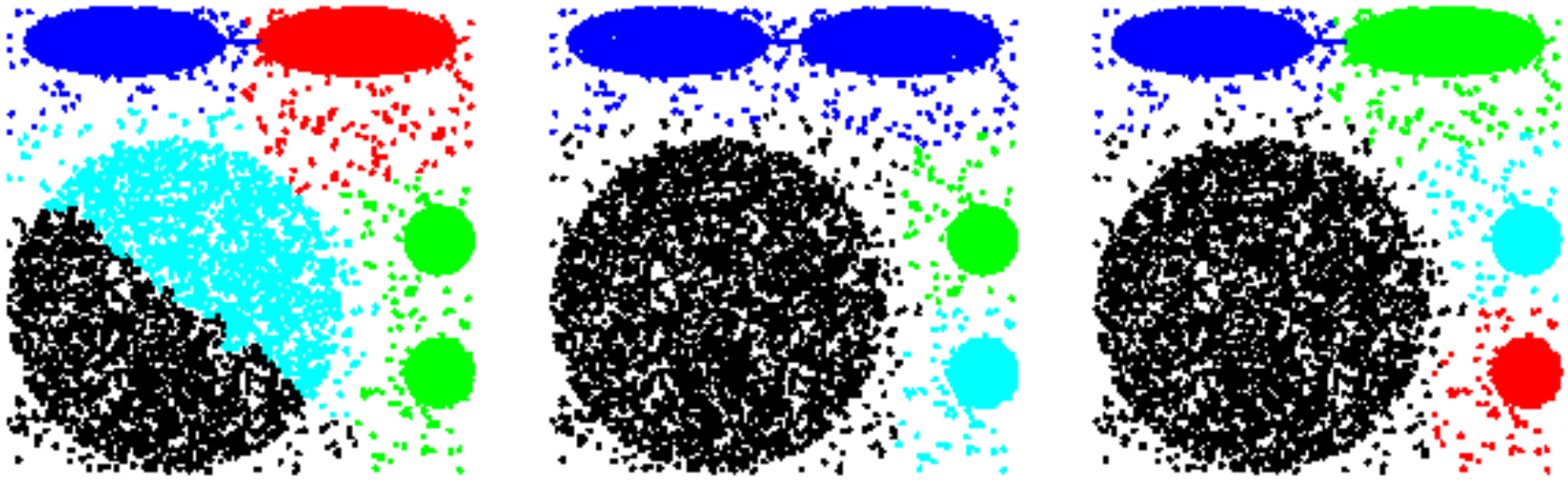
- Cluster similarity is the similarity of the closest pair of representative points from different clusters

CURE Algorithm

- Uses agglomerative hierarchical scheme to perform clustering;
  - α = 0: similar to centroid-based
  - α = 1: somewhat similar to single-link

- CURE is better able to handle clusters of arbitrary shapes and sizes

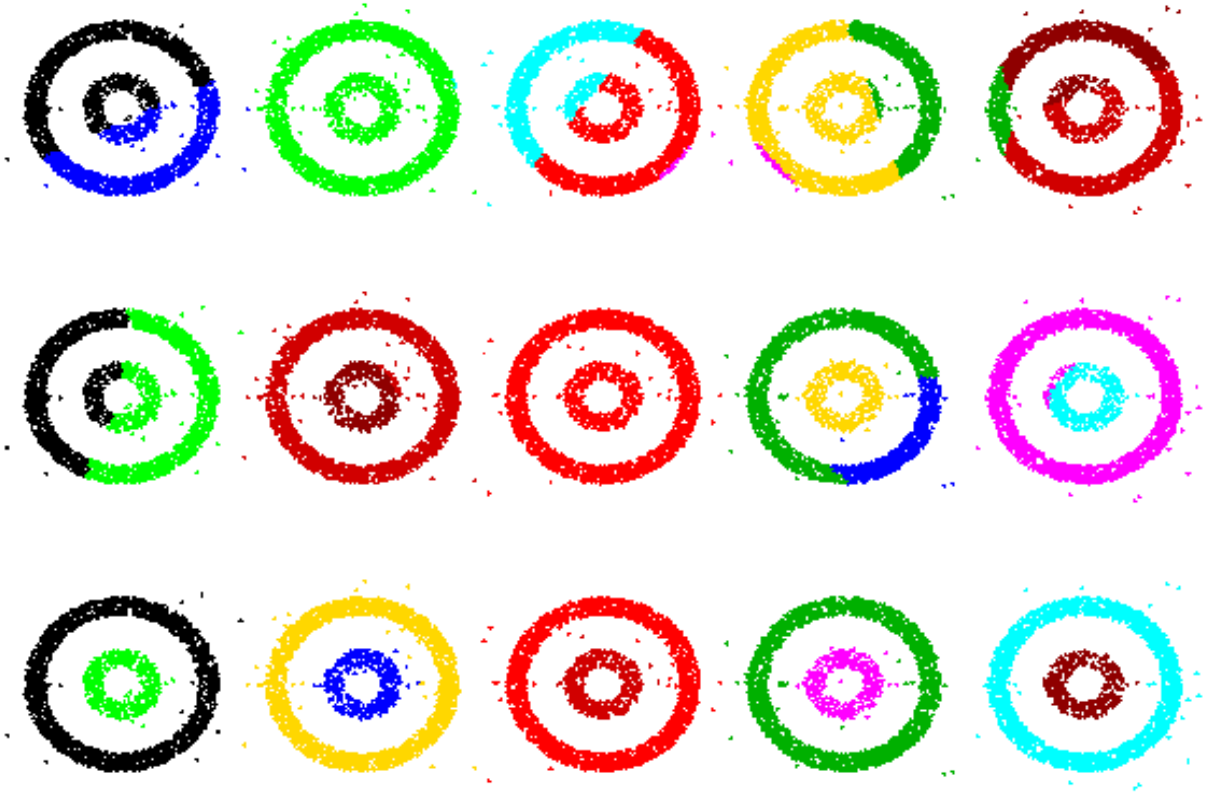
Experimental Results: CURE



a) BIRCH b) MST METHOD c) CURE

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

Experimental Results: CURE



a) BIRCH

(centroid)

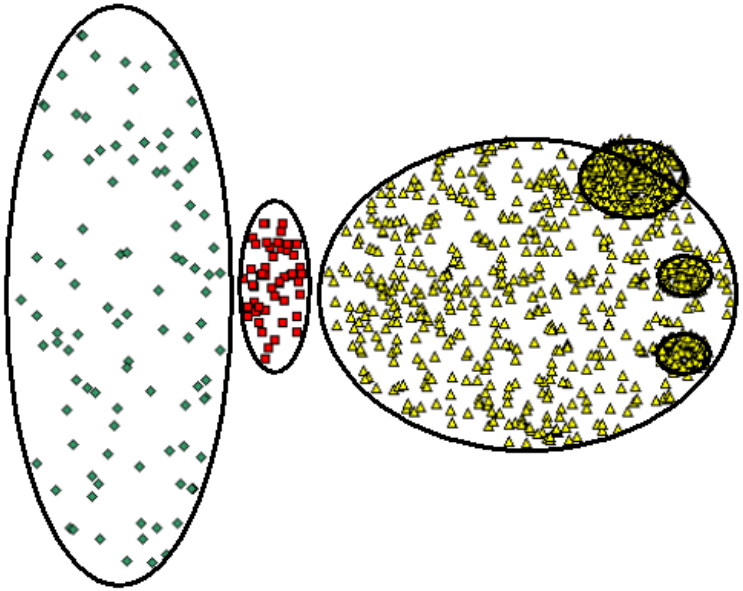
b) MST METHOD

(single link)

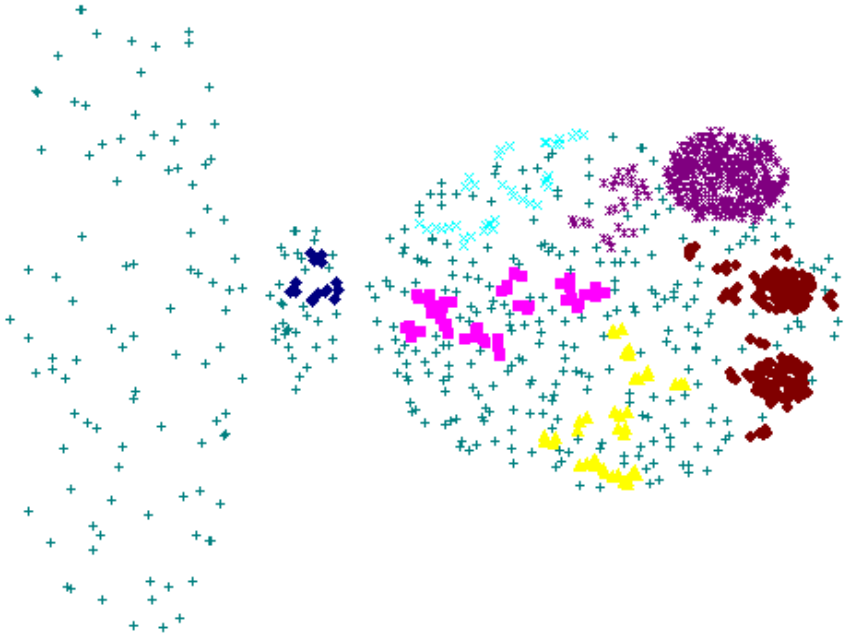
c) CURE

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

CURE Cannot Handle Differing Densities



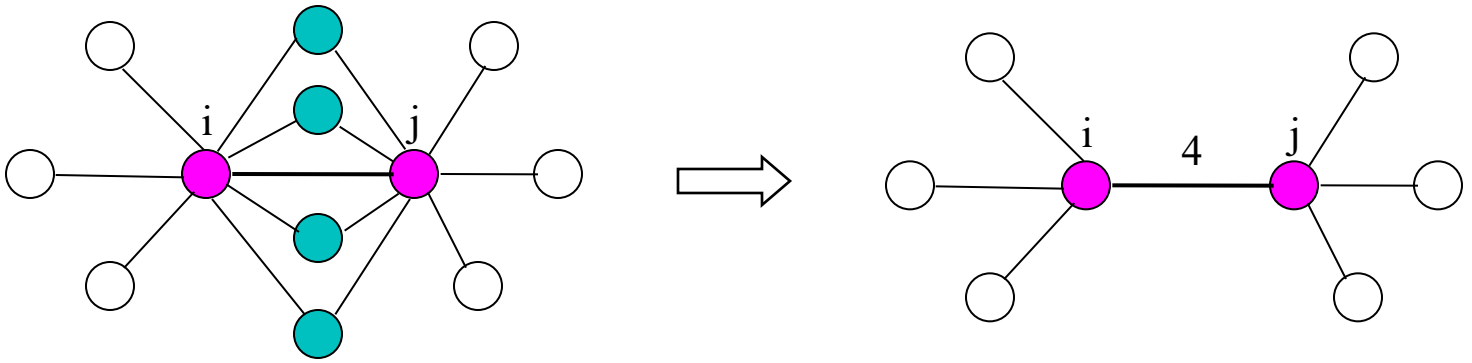
Original Points



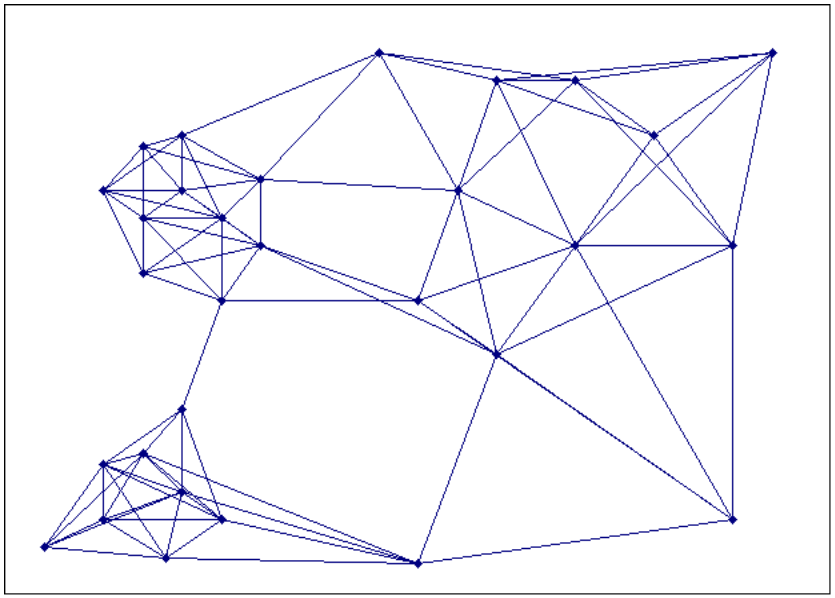
CURE

Graph-Based Clustering: 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

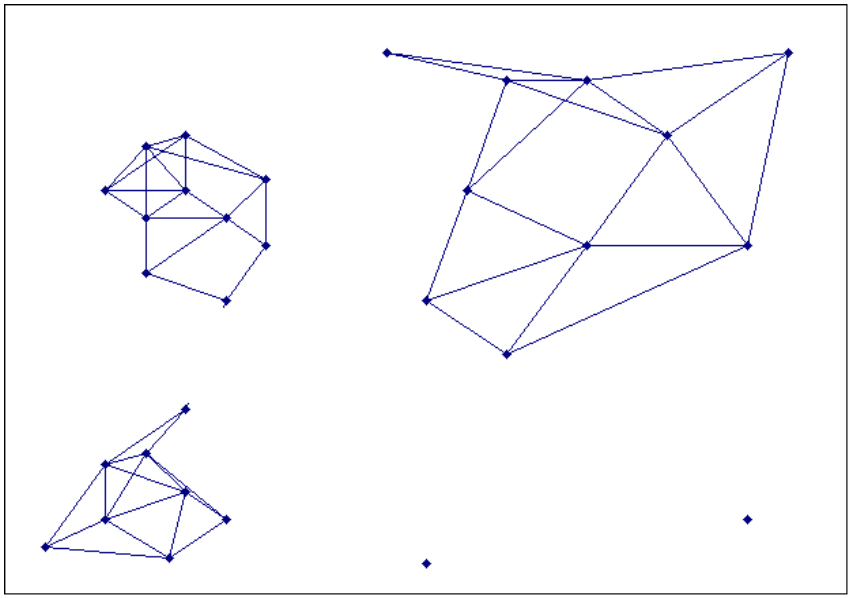


Creating the SNN Graph



Sparse Graph

Link weights are similarities between neighboring points



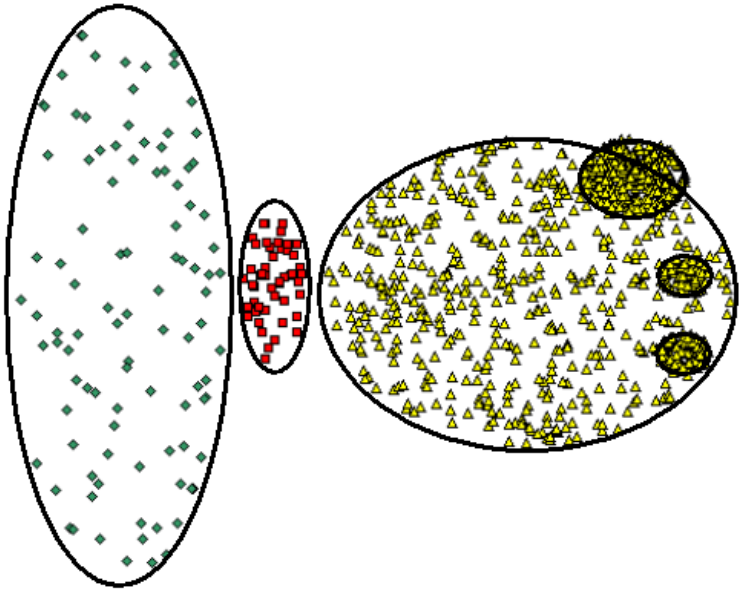
Shared Near Neighbor Graph

Link weights are number of Shared Nearest Neighbors

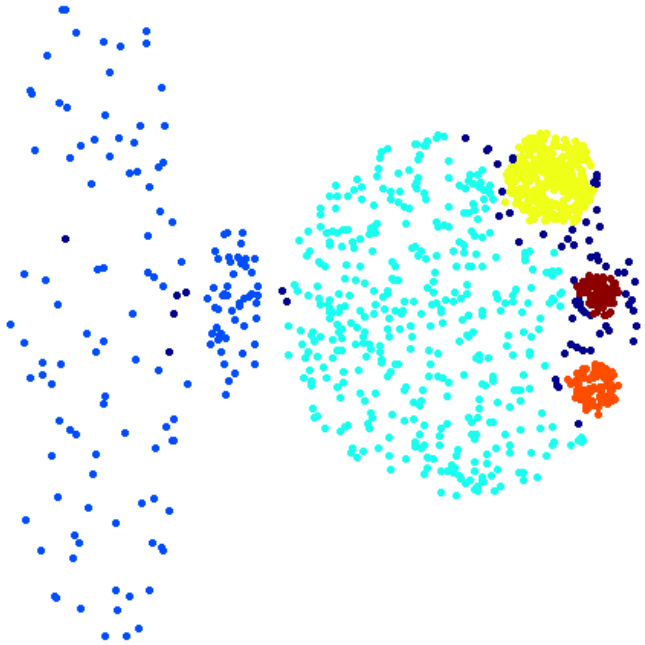
Jarvis-Patrick Clustering

- 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



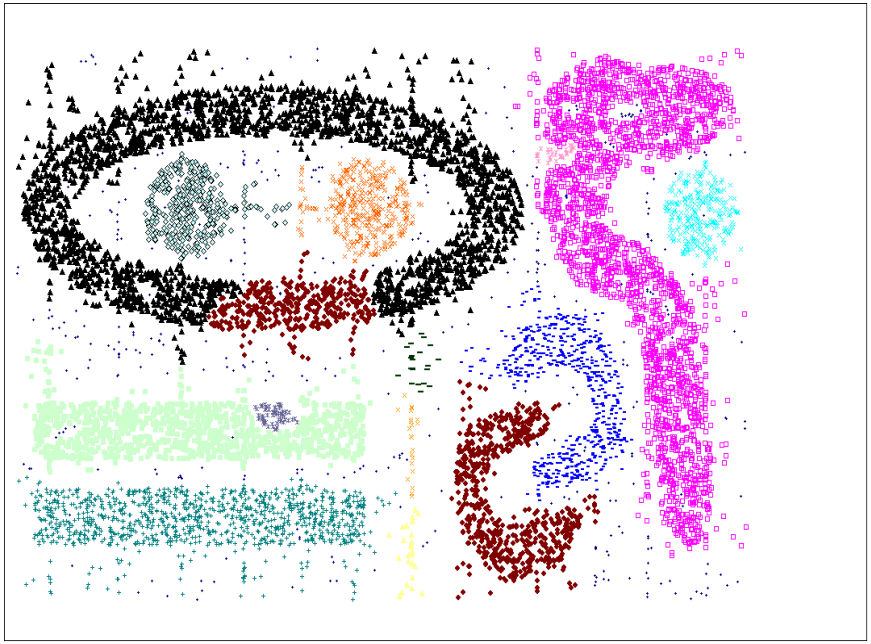
Original Points



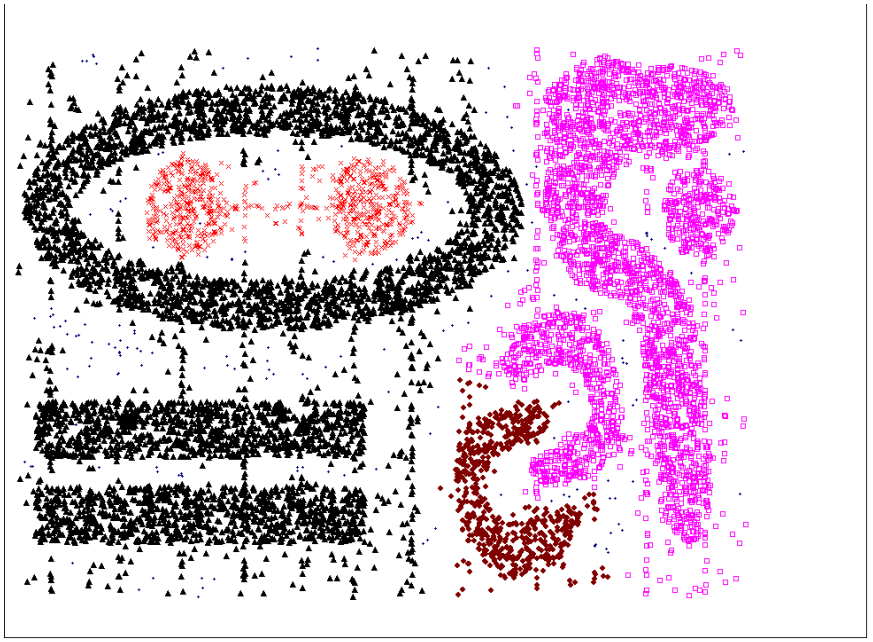
Jarvis Patrick Clustering

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*

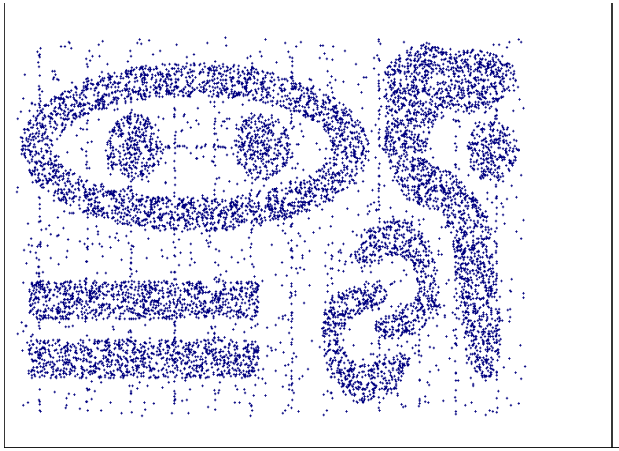
6. **Form clusters from the core points**
   If two core points are within a "radius", *Eps*, of each other they are place in the same cluster

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

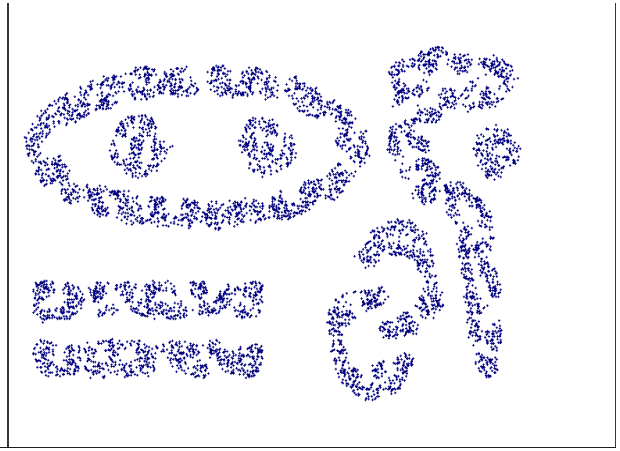
8. **Assign all non-noise, non-core points to clusters**
   This can be done by assigning such points to the nearest core point

   (Note that steps 4-8 are DBSCAN)

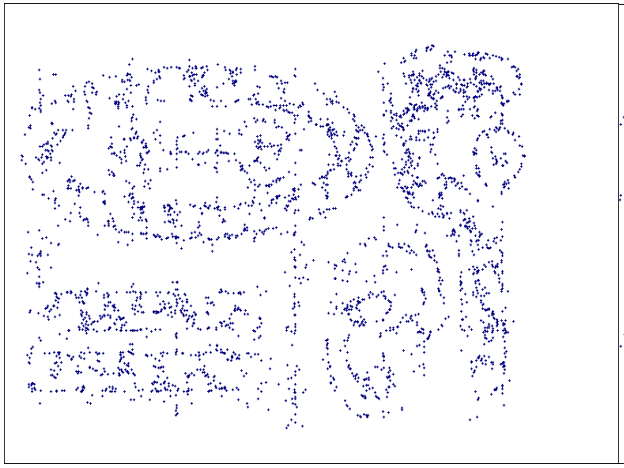
SNN Density



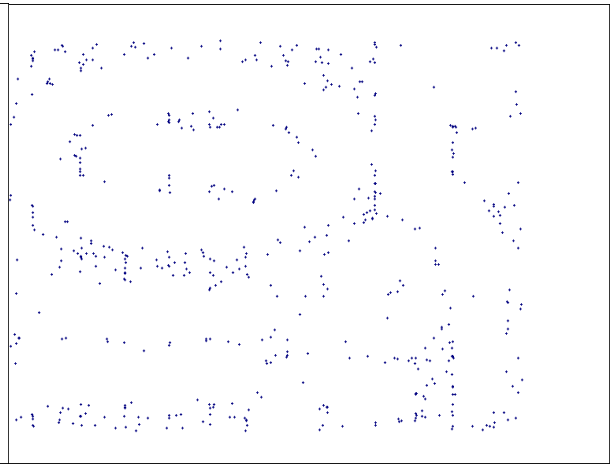
a) All Points



b) High SNN Density

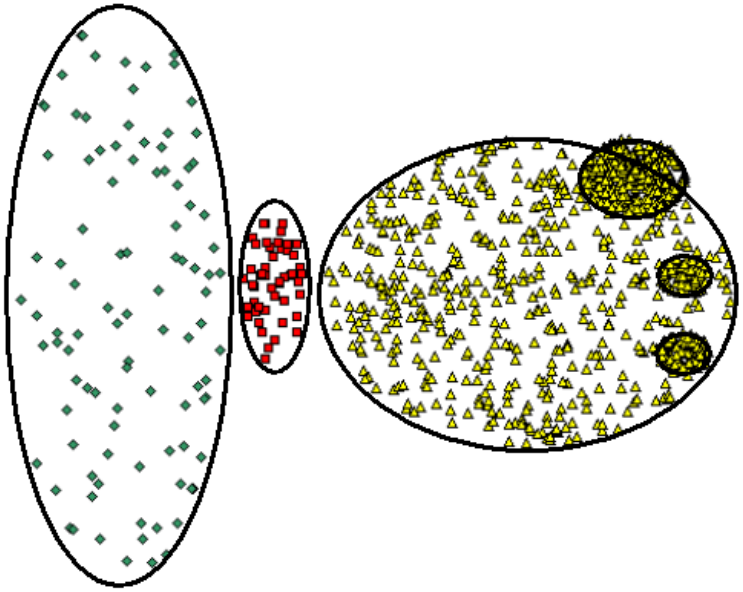


c) Medium SNN Density

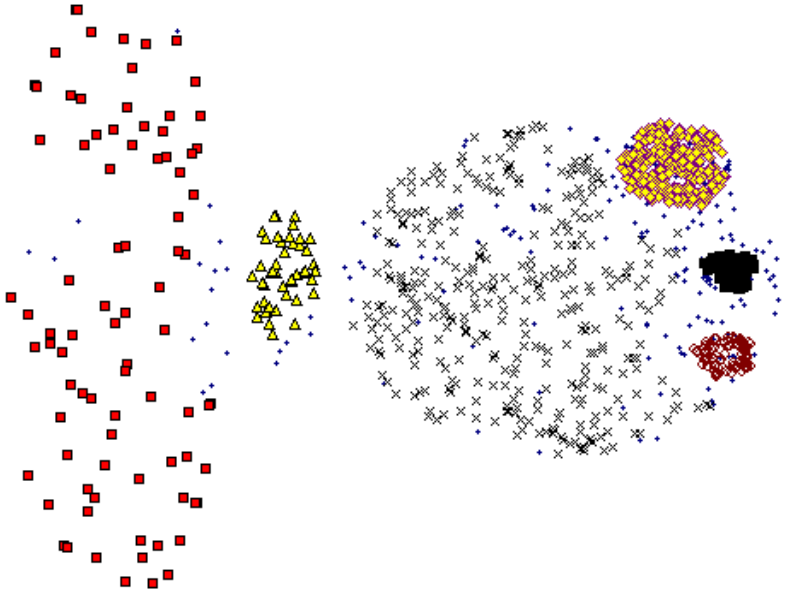


d) Low SNN Density

SNN Clustering Can Handle Differing Densities

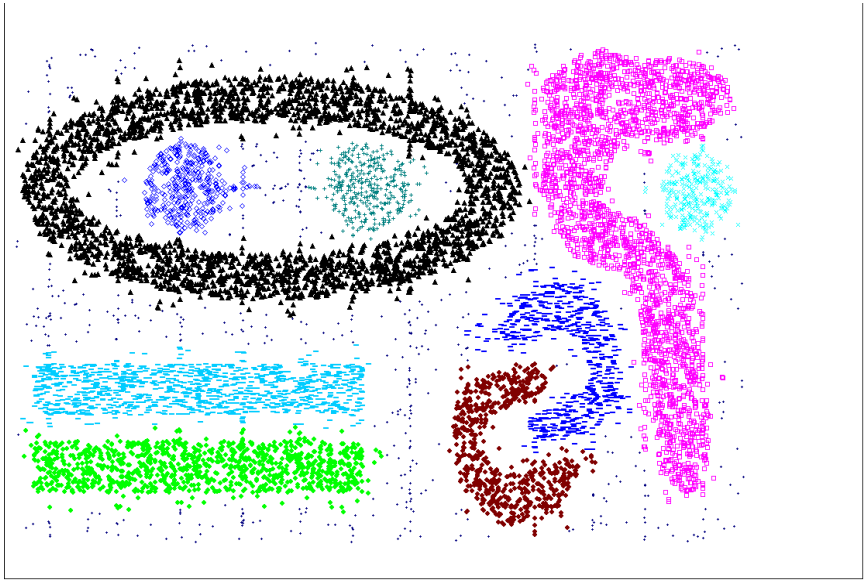


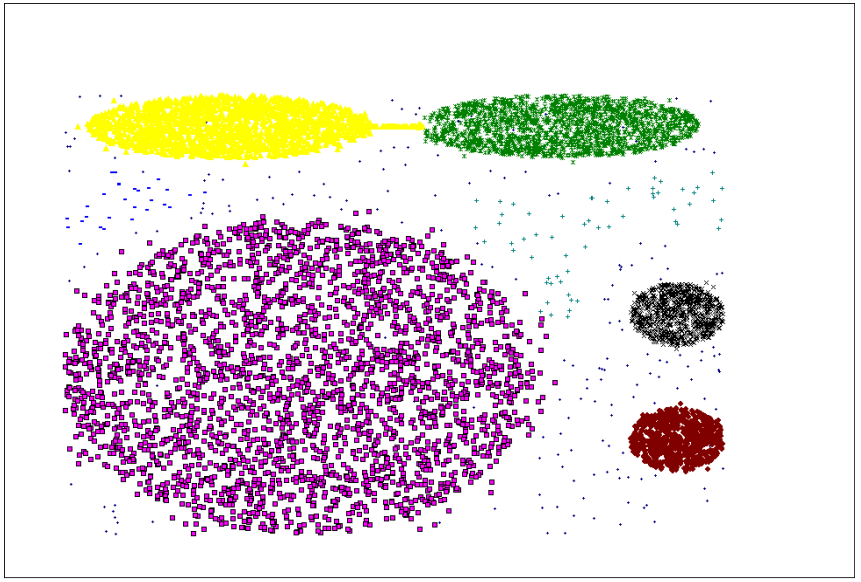
Original Points



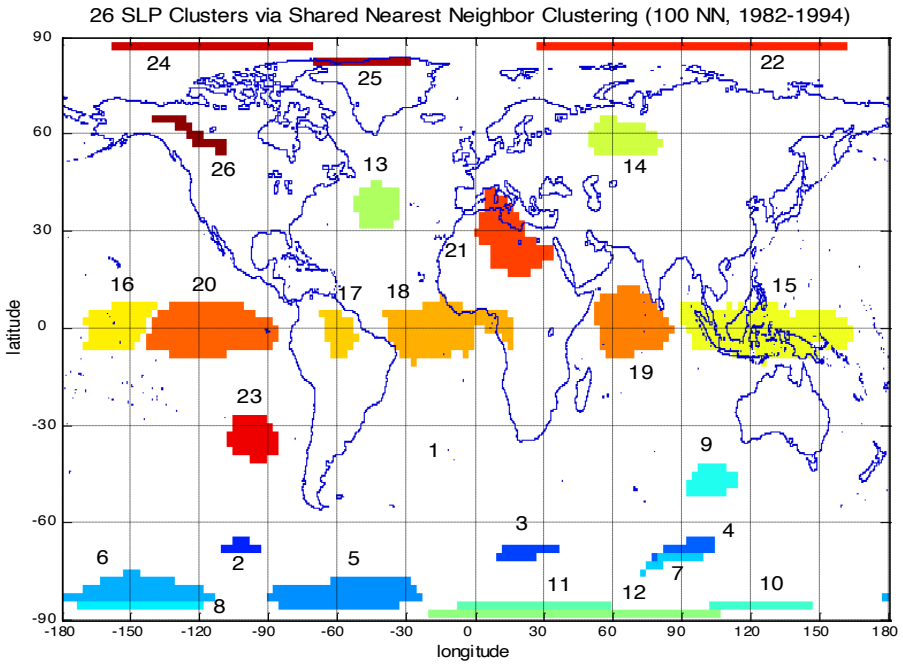
SNN Clustering

SNN Clustering Can Handle Other Difficult Situations

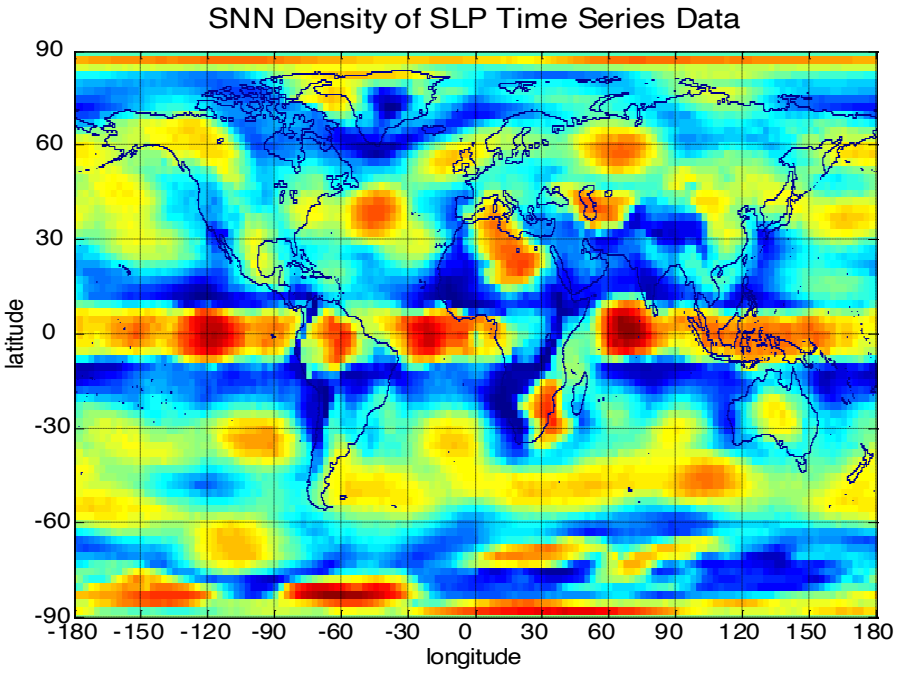




Finding Clusters of Time Series In Spatio-Temporal Data



SNN Clusters of SLP.



SNN Density of Points on the Globe.

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

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
- Size of data set
- Sparsity of attribute values
- Noise and Outliers
- Types of attributes and type of data sets
- Differences in attribute scale
- Properties of the data space
  - Can you define a meaningful centroid

Characteristics of Clusters

- Data distribution
- Shape
- Differing sizes
- Differing densities
- Poor separation
- Relationship of clusters
- Subspace clusters

Characteristics of Clustering Algorithms

- Order dependence
- Non-determinism
- Parameter selection
- Scalability
- Underlying model
- Optimization based approach

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 not 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 not distribution assumptions; K-means is really assuming spherical Gaussian distributions

Comparison of DBSCAN and K-means

- 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 cluster; 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.