BBS654 Data Mining

Pinar Duygulu

Slides are adapted from

Nazli Ikizler

Density-Based Clustering Methods

- Clustering based on density (local cluster criterion), such as density-connected points
- Major features:
  - Discover clusters of arbitrary shape
  - Handle noise
  - One scan
  - Need density parameters as termination condition
- Several interesting studies:
  - DBSCAN: Ester, et al. (KDD'96)
  - OPTICS: Ankerst, et al (SIGMOD'99).
  - DENCLUE: Hinneburg & D. Keim (KDD'98)
  - CLIQUE: Agrawal, et al. (SIGMOD'98) (more grid-based)

DBSCAN

- DBSCAN is a density based clustering algorithm

Density = number of points within a specified radius (*Eps*)

- A point is a *core point* if it has more than specified number of points (*MinPts*) within *Eps*
- Core point is in the interior of a cluster

- A *border point* has fewer than *MinPts* within *Eps* but is in neighborhood of a core point

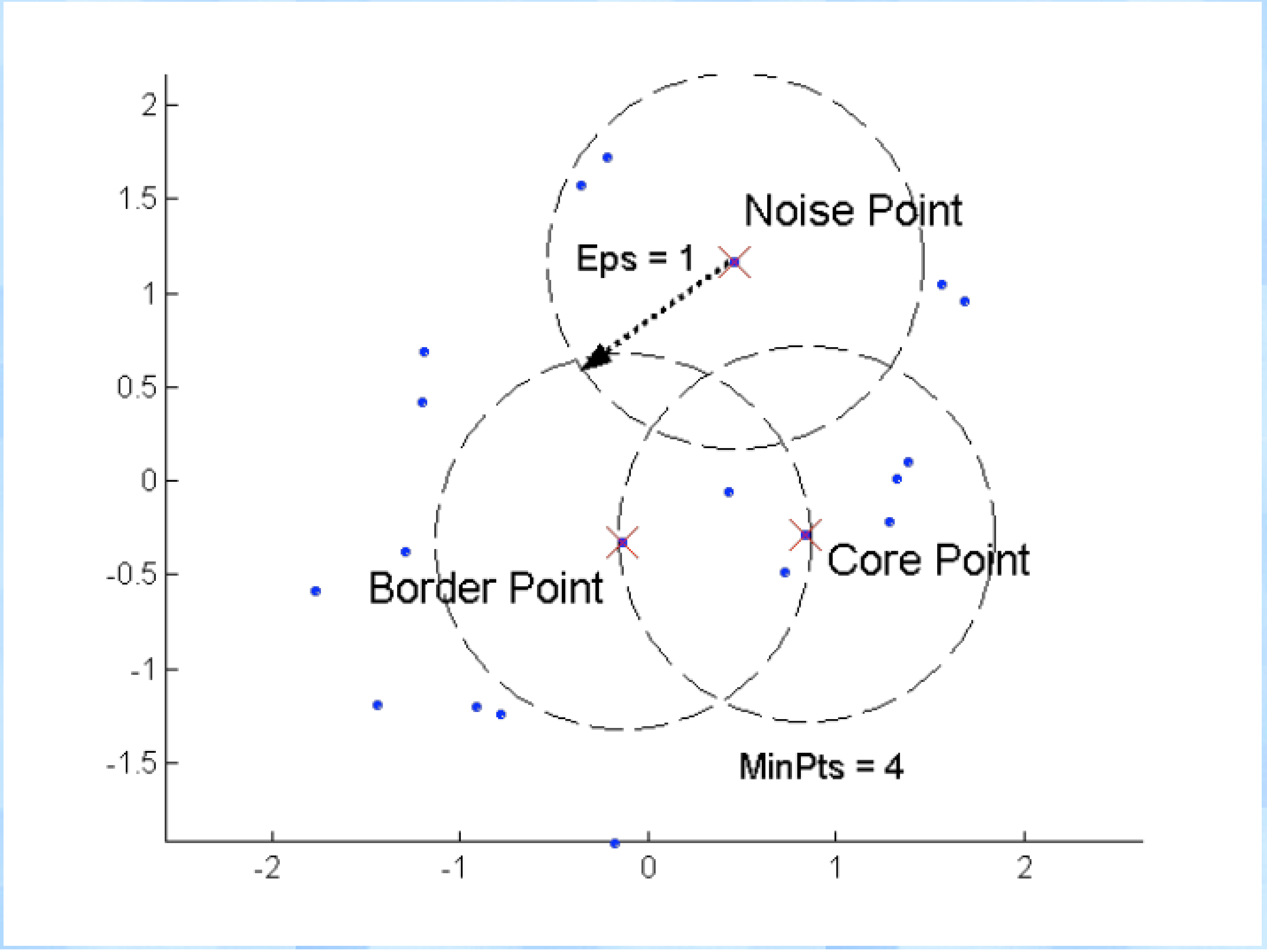
- A *noise point* is any point that is neither a core point nor a border point

Classification of points in density-based clustering

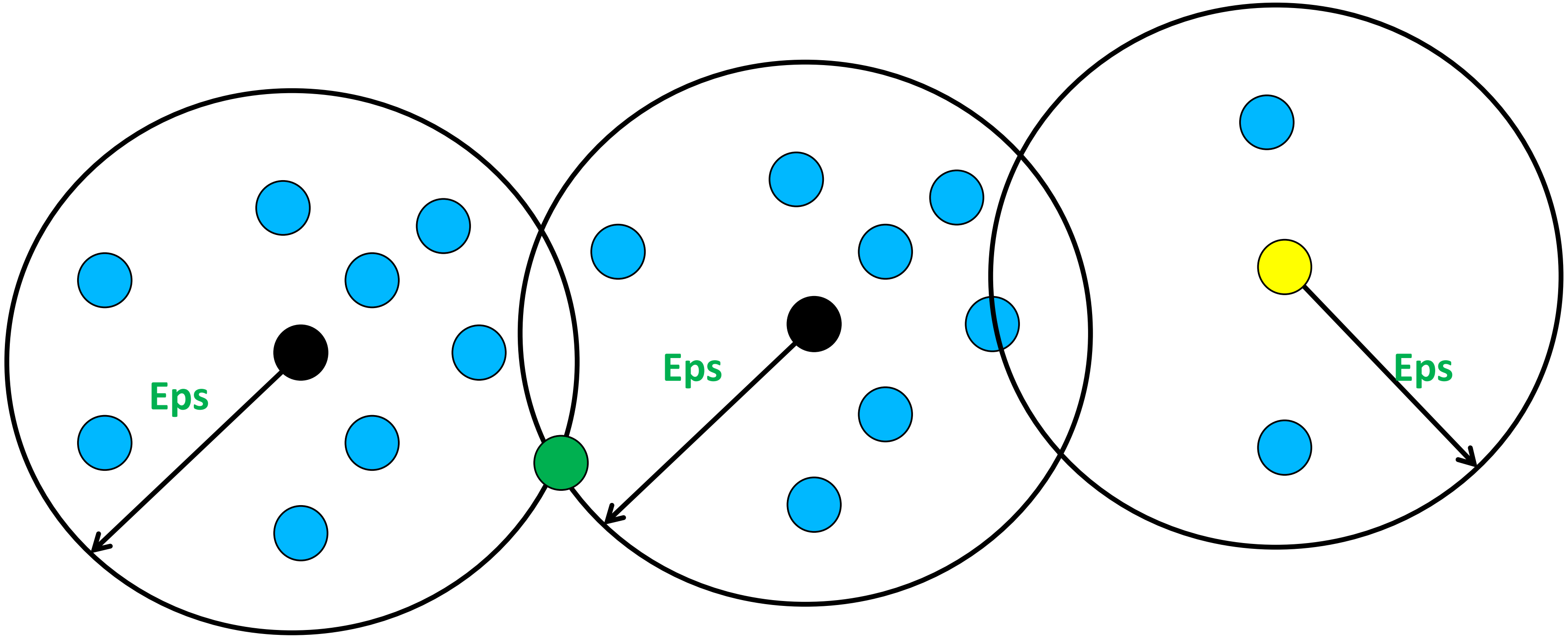
- **Core points**: Interior points of a density-based cluster. A point **p** is a core point if for distance **Eps** :
    - **$|N_{Eps}(p)=\{q \mid dist(p,q) <= \varepsilon \}| \geq MinPts$**

- **Border points:** Not a core point but within the neighborhood of a core point (it can be in the neighborhoods of many core points)

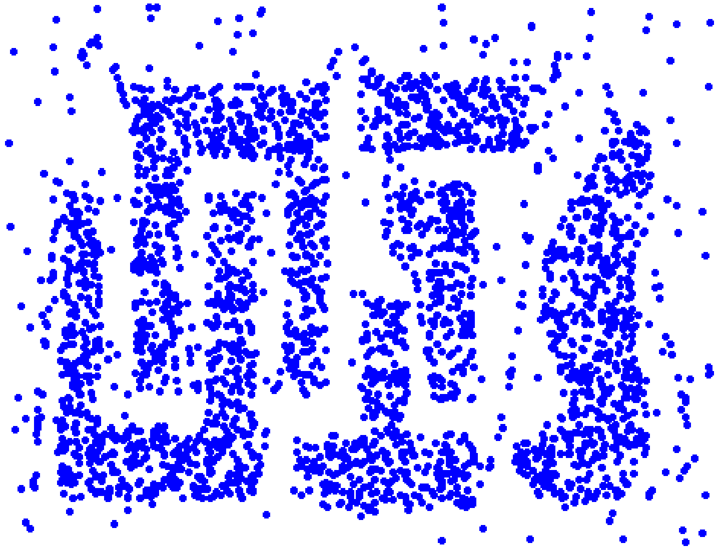
- **Noise points:** Not a core or a border point


Noise Point
Eps = 1
Border Point
Core Point
MinPts = 4
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-1
0
1
2

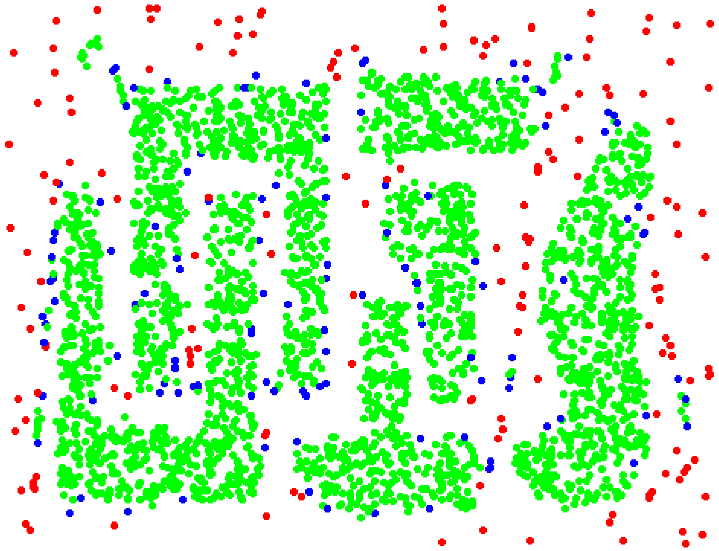
Core, Border and Noise points



Core, Border and Noise points



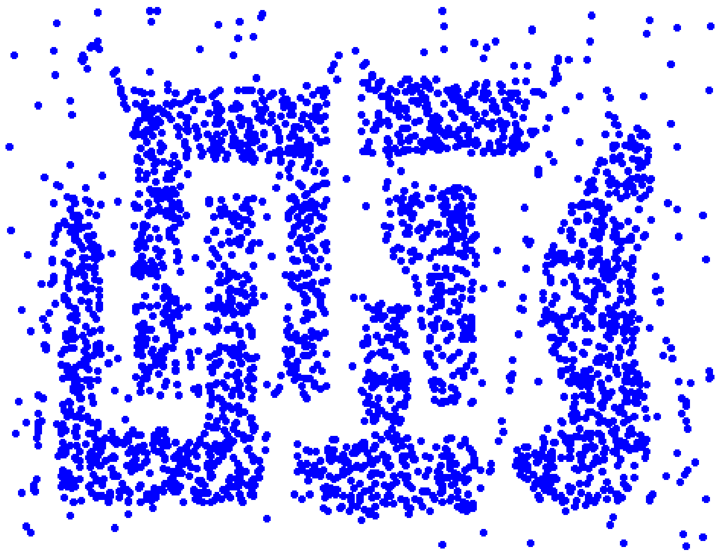
Original Points



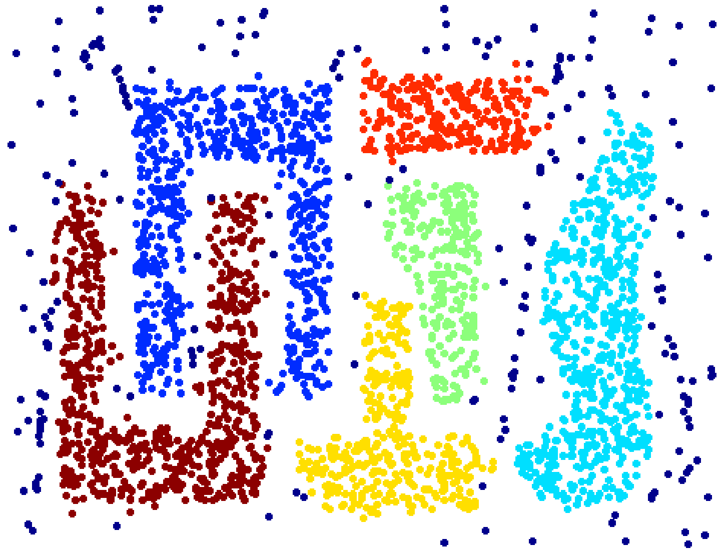
Point types: core, border and noise

Eps = 10, MinPts = 4

Clusters output by DBScan



Original Points



Clusters

- Resistant to Noise
- Can handle clusters of different shapes and sizes

DBSCAN: The Algorithm

- Label all points as ***core***, ***border***, or ***noise*** points

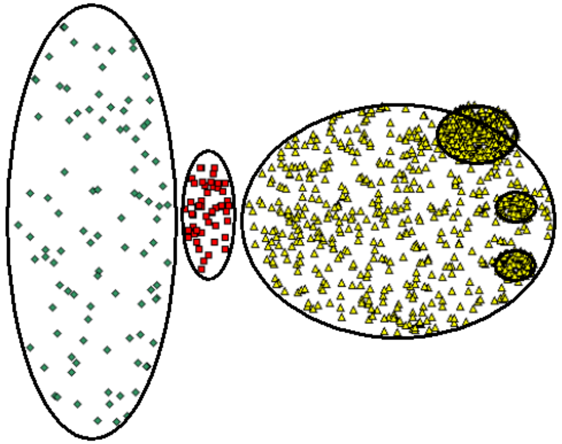
- Eliminate noise points

- Put an edge between all core points that are within ***Eps*** of each other

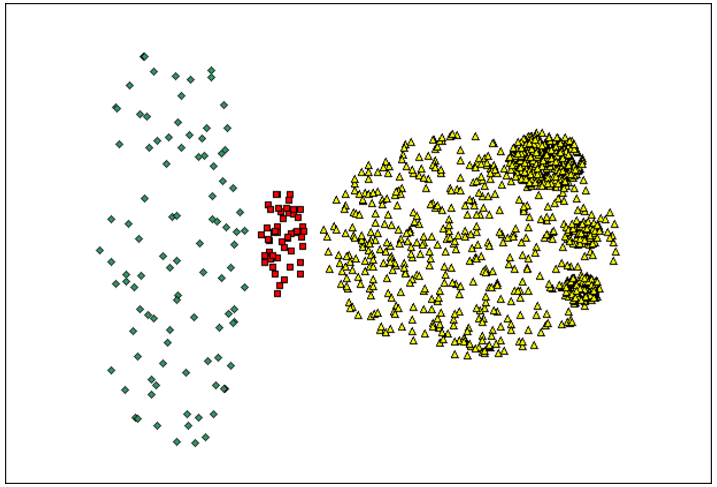
- Make each group of connected core points into a separate cluster

- Assign each border point to one of the cluster of its associated core points

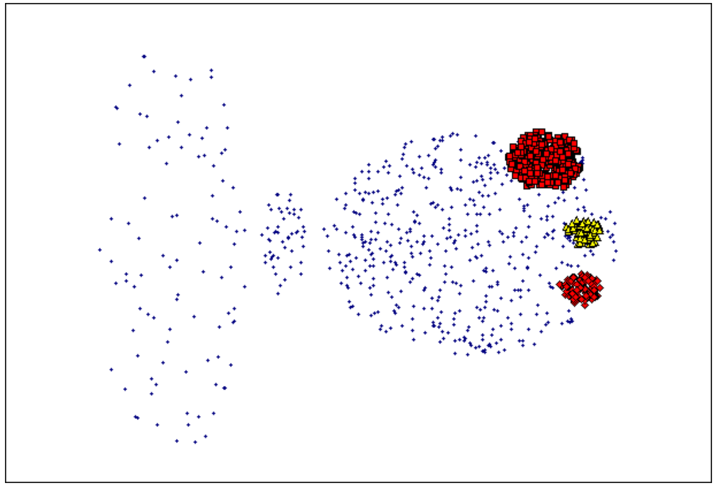
When DBSCAN Does NOT Work Well



Original Points



(MinPts=4, Eps=9.75).

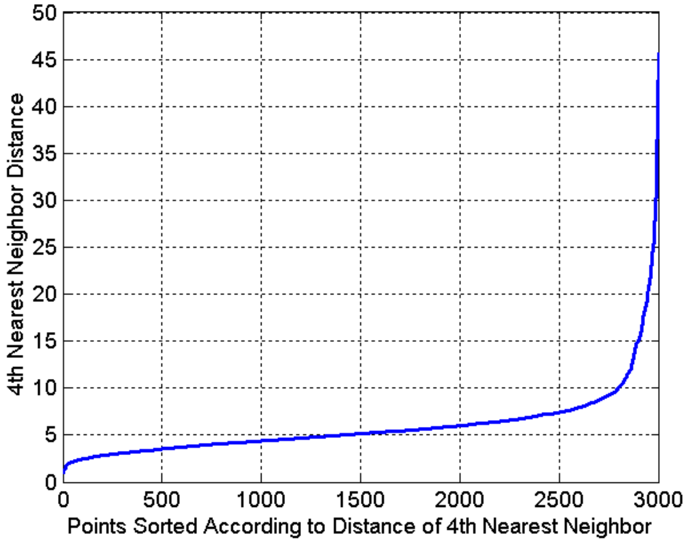


(MinPts=4, Eps=9.92)

DBScan can fail to identify clusters of varying densities

Determining EPS and MinPts

- Idea is that for points in a cluster, their kth nearest neighbors are at roughly the same distance
- Noise points have the kth nearest neighbor at farther distance
- So, plot sorted distance of every point to its kth nearest neighbor



Strengths and Weaknesses of DBSCAN

- Resistant to noise

- Finds clusters of arbitrary shapes and sizes

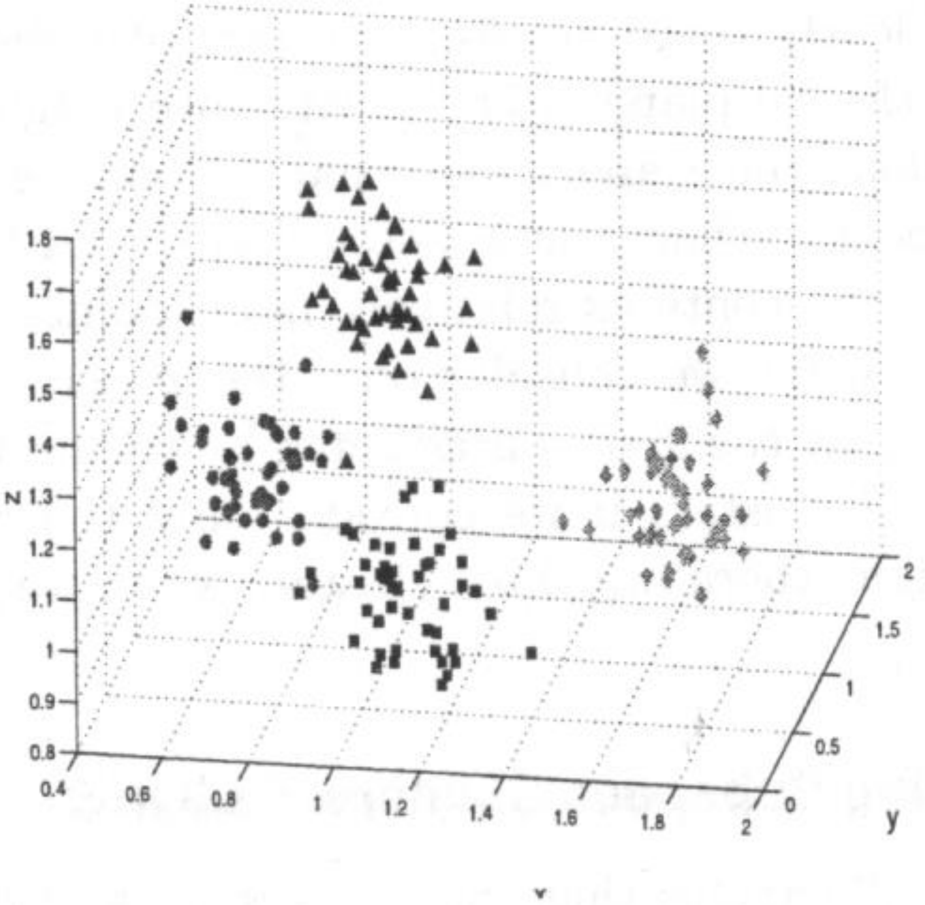
- Difficulty in identifying clusters with varying densities

- Problems in high-dimensional spaces; notion of density unclear

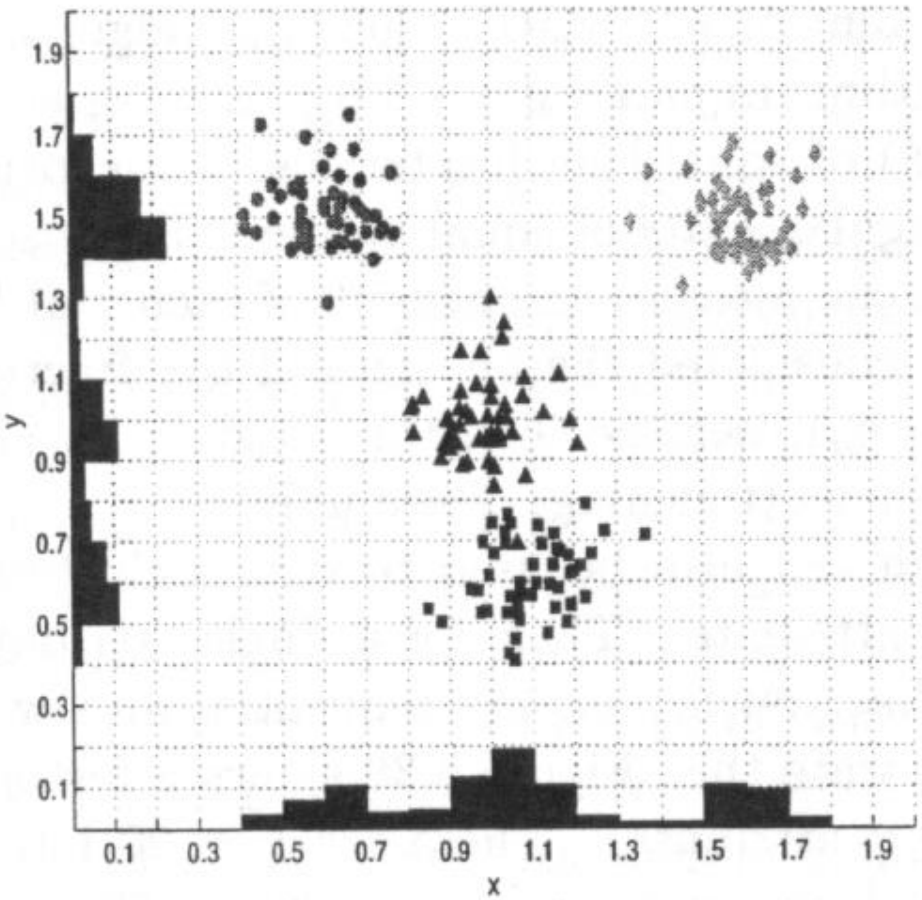
- Can be computationally expensive when the computation of nearest neighbors is expensive

Subspace clustering

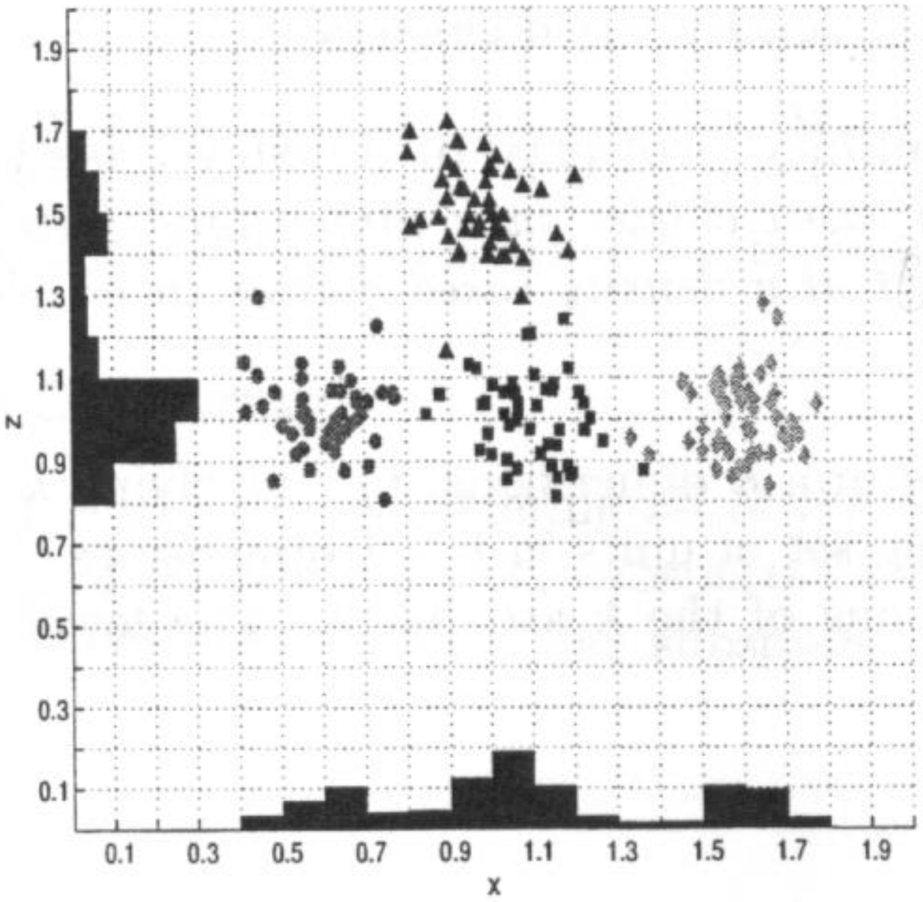
- Instead of using all the attributes (features) of a dataset, if we consider only subset of the features (subspace of the data), then the clusters that we find can be quite different from one subspace to another
- The clusters we find depend on the subset of the attributes that we consider



(a) Four clusters in three dimensions.



(b) View in the XY plane.

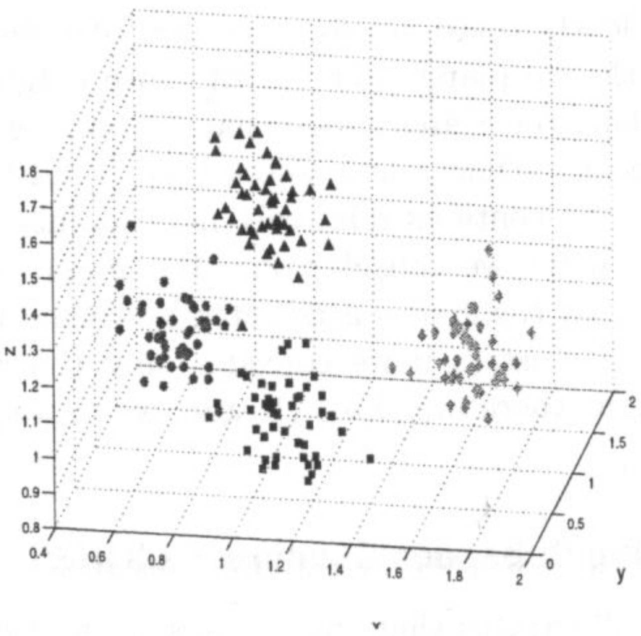


(c) View in the XZ plane.

CLIQUE

CLIQUE is a grid based clustering algorithm

- CLIQUE splits each dimension (attribute) in to a fixed number of equal length intervals. This partitions the data space in to rectangular *units* of equal volume
- We can measure the density of each unit by the fraction of points it contains
- A unit is considered dense if its density > user specified threshold
- A cluster is a group of contiguous (touching) dense units



(a) Four clusters in three dimensions.

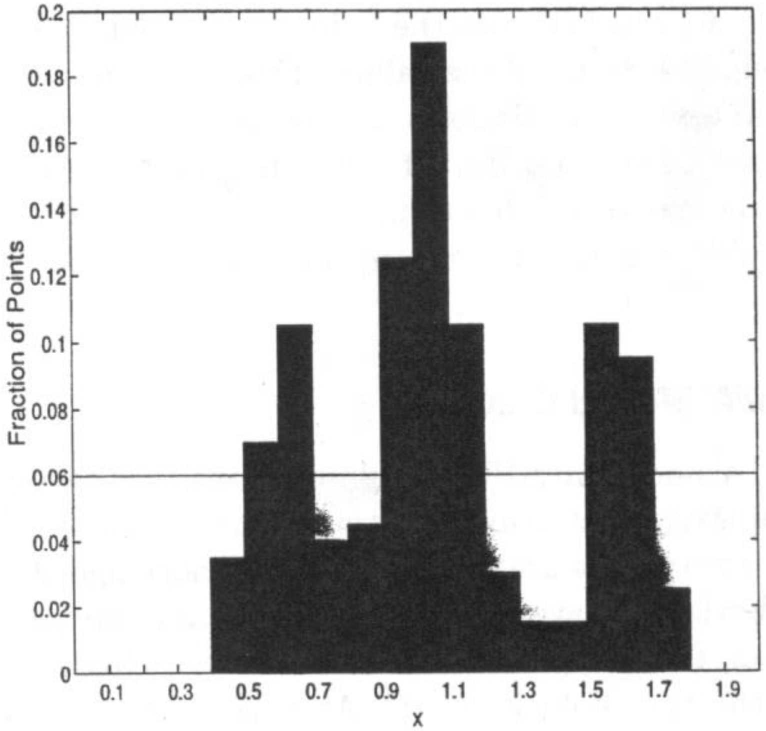


Figure 5.36. Histogram showing distribution of points for the X attribute.

CLIQUE

CLIQUE starts by finding all the dense areas in the one dimensional spaces associated with each Attribute

- Then it generates the set of two dimensional cells that might possibly be dense by looking at pairs of dense one dimensional cells

- In general, CLIQUE generates the possible set of k-dimensional cells that might possibly be dense by looking at dense (k-1)-dimensional cells.

- It then finds clusters finds clusters by taking union of all adjacent high density cells

Limitations

Time complexity is exponential in the number of dimensions

- Will have difficulty 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 the appropriate parameters for a variety of data sets can be challenging
- It is not typically possible to find all clusters using the same threshold

Clustering Scalability for Large Datasets

- One very common solution is sampling, but the sampling could miss small clusters.

– Data is sometimes not organized to make valid sampling easy or efficient.

- Another approach is to compress the data or portions of the data.

– Any such approach must ensure that not too much information is lost. (*Scaling Clustering Algorithms to Large Databases*, Bradley, Fayyad and Reina.)

Scalable Clustering: CURE

Clustering Using Representatives

Uses a number of points to represent a cluster

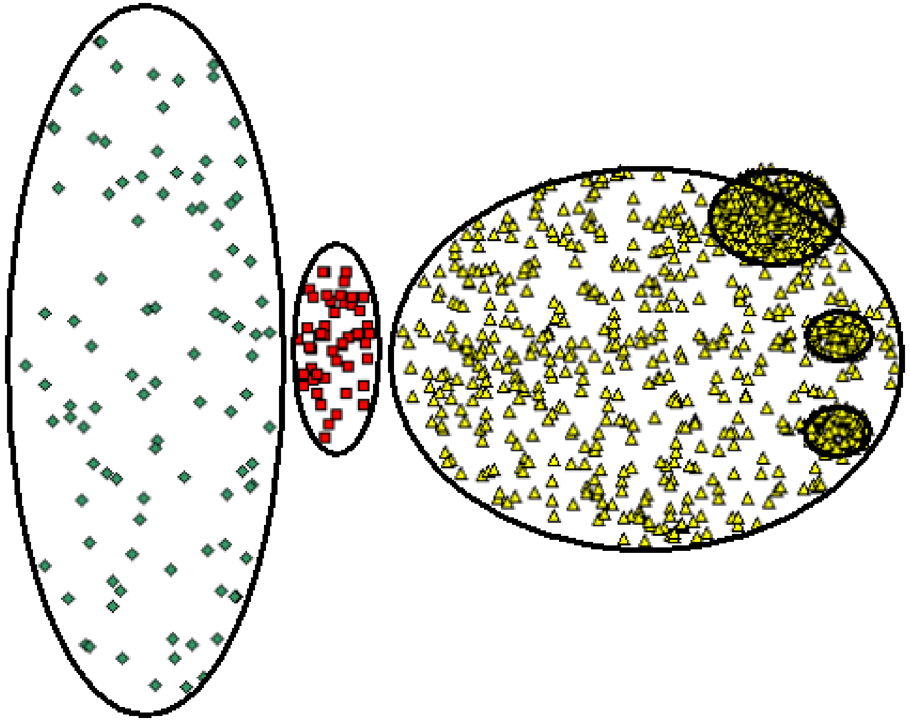
• Representative points are found by selecting a constant number of points from a cluster and then “shrinking” them toward the center of the cluster

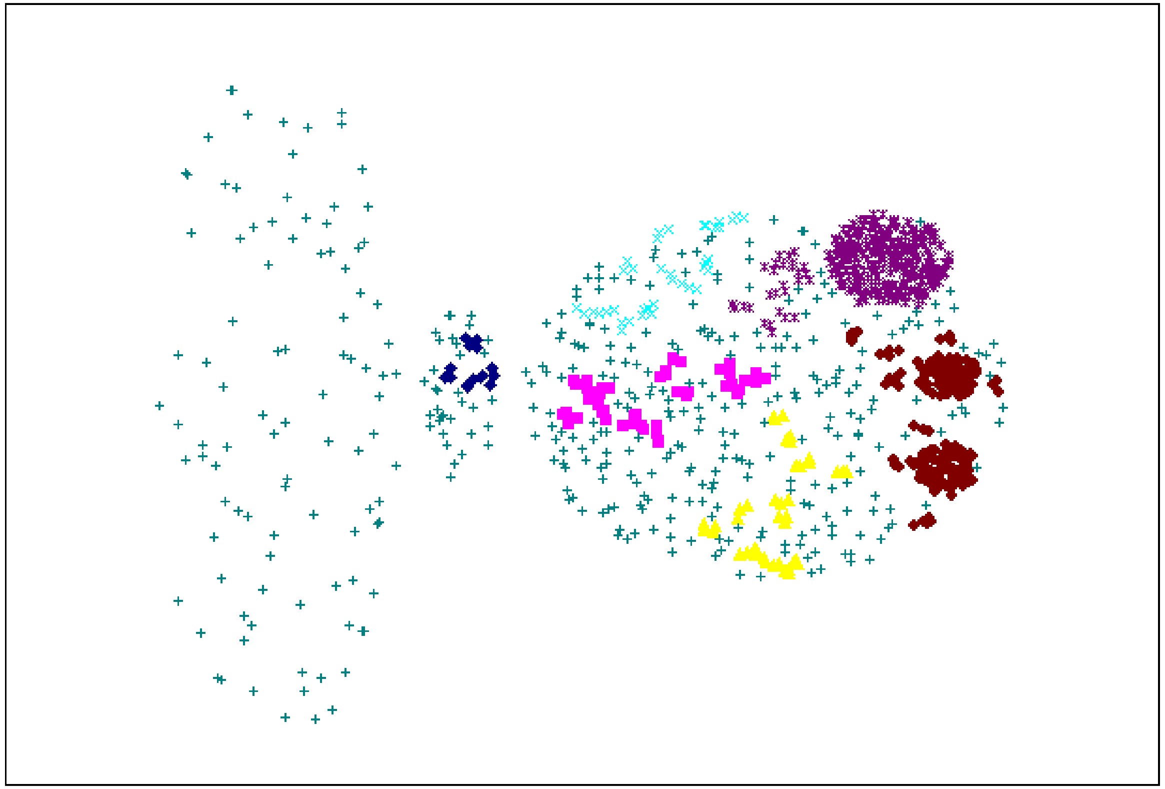
• Cluster similarity is the similarity of the closest pair of representative points from different clusters

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

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

CURE can not handle differing densities





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

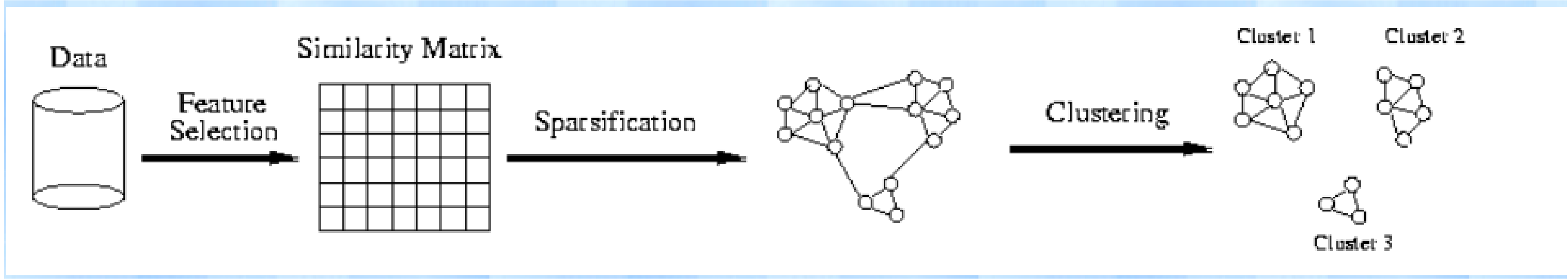
• In the most simple case, clusters are connected components in the graph

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 similarity matrix
- – The amount of time required to cluster the data is drastically reduced
- – The size of the problems that can be handled is increased

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


Data
Feature Selection
Similarity Matrix
Sparsification
Clustering
Cluster 1
Cluster 2
Cluster 3

Chameleon: Clustering Using Dynamic Modeling

- Adapt to the characteristics of the data set to 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*.

Chamelon

- Preprocessing Step: Represent the Data by a Graph
- – Given a set of points, we construct the *k-nearestneighbor* (*k-NN*) graph to capture the relationship between a point and its *k* nearest neighbors.

• 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.
- – Graph algorithms take into account global structure.

Chamelon

- 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 are 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.

SNN Approach to Clustering

Ordinary distance measures have problems

– Euclidean distance is less appropriate in high dimensions

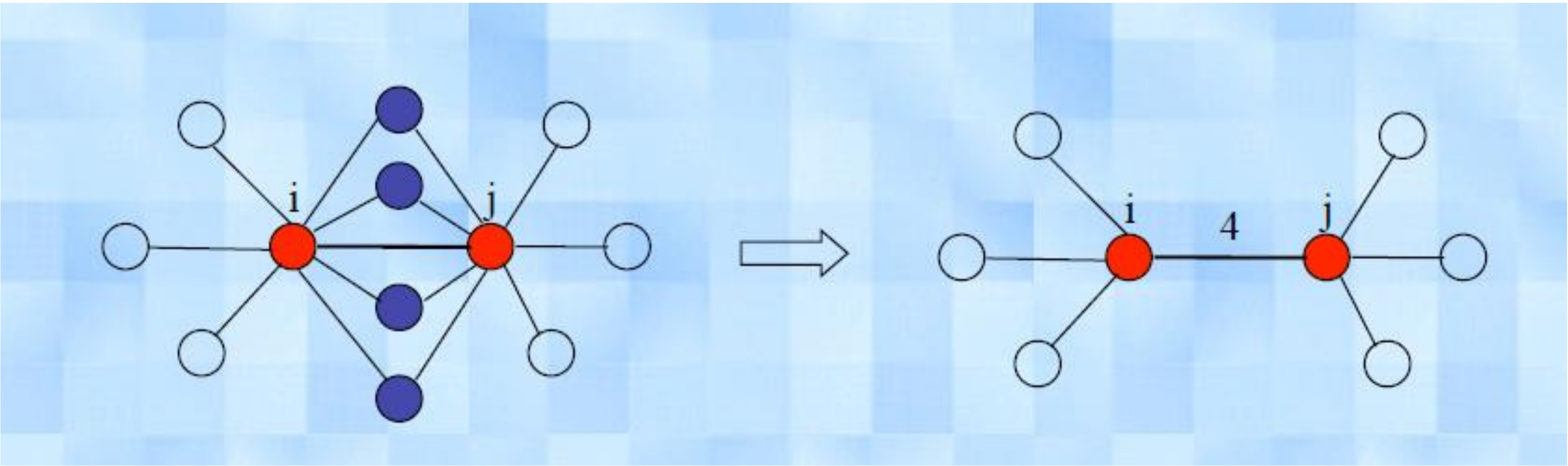
• Presences are more important than absences

– Cosine and Jaccard measure take in to account presences, but do not satisfy the triangle inequality

• SNN distance is more appropriate in these cases

Shared Near Neighbor Graph

- In the SNN graph, the strength of a link is the number of shared neighbors between documents given that the documents are connected



SNN Approach: Density

Ordinary density measures have problems

– Typical Euclidean density is number of points per unit volume

– As dimensionality increases, density goes to 0

• Can estimate the relative density, i.e., probability density, in a region

– Look at the distance to the kth nearest neighbor, or

– Look at the number of points within a fixed radius

– However, since distances become uniform in high dimensions, this does not work well either

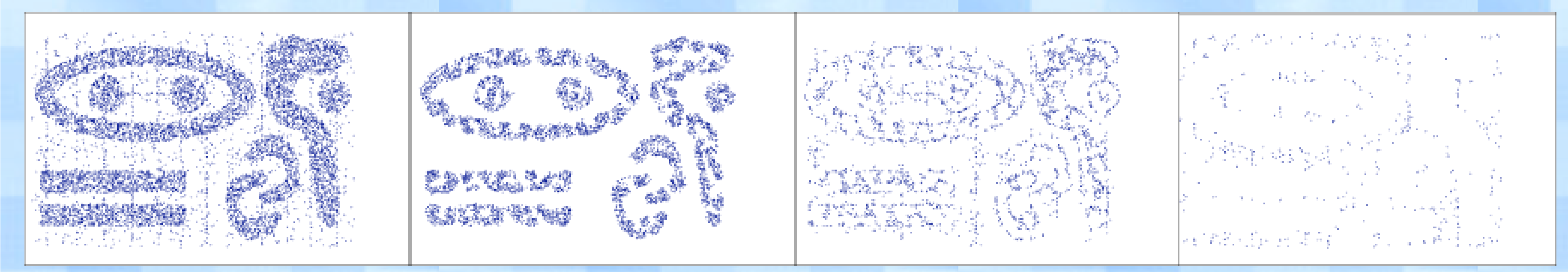
• If we use SNN similarity then we can obtain a more robust definition of density

– Relatively insensitive to variations in normal density

– Relatively insensitive to high dimensionality

– Uniform regions are dense, gradients are not

- SNN Density can identify Core, Border and Noise points
- Assume a DBSCAN definition of density
- – Number of points within *Eps*



a) All Points Density b) High SNN Density c) Medium SNN Density d) Low SNN Density

ROCK

ROCK (RObust Clustering using linKs)

– Clustering algorithm for data with categorical and Boolean attributes

• It redefines the distances between points to be the number of shared neighbors whose strength is greater than a given threshold

• Then uses a hierarchical clustering scheme to cluster the data

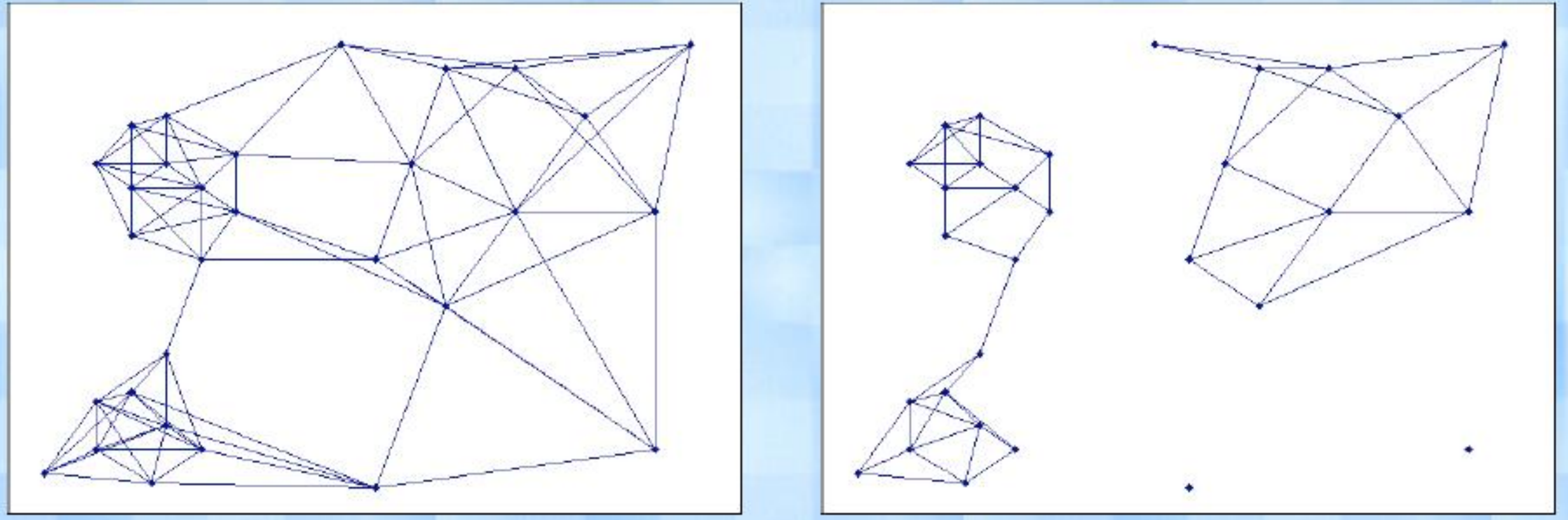
1. Obtain a sample of points from the data set

2. Compute the link value for each set of points, i.e., transform the original similarities (computed by the 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” similarities and the “maximize the shared neighbors” objective function

4. Assign the remaining points to the clusters that have been found

Creating the SNN Graph



5 Near neighbor graph

Shared near neighbor graph

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

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 parameter, *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.

5. Find the core points

Using 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

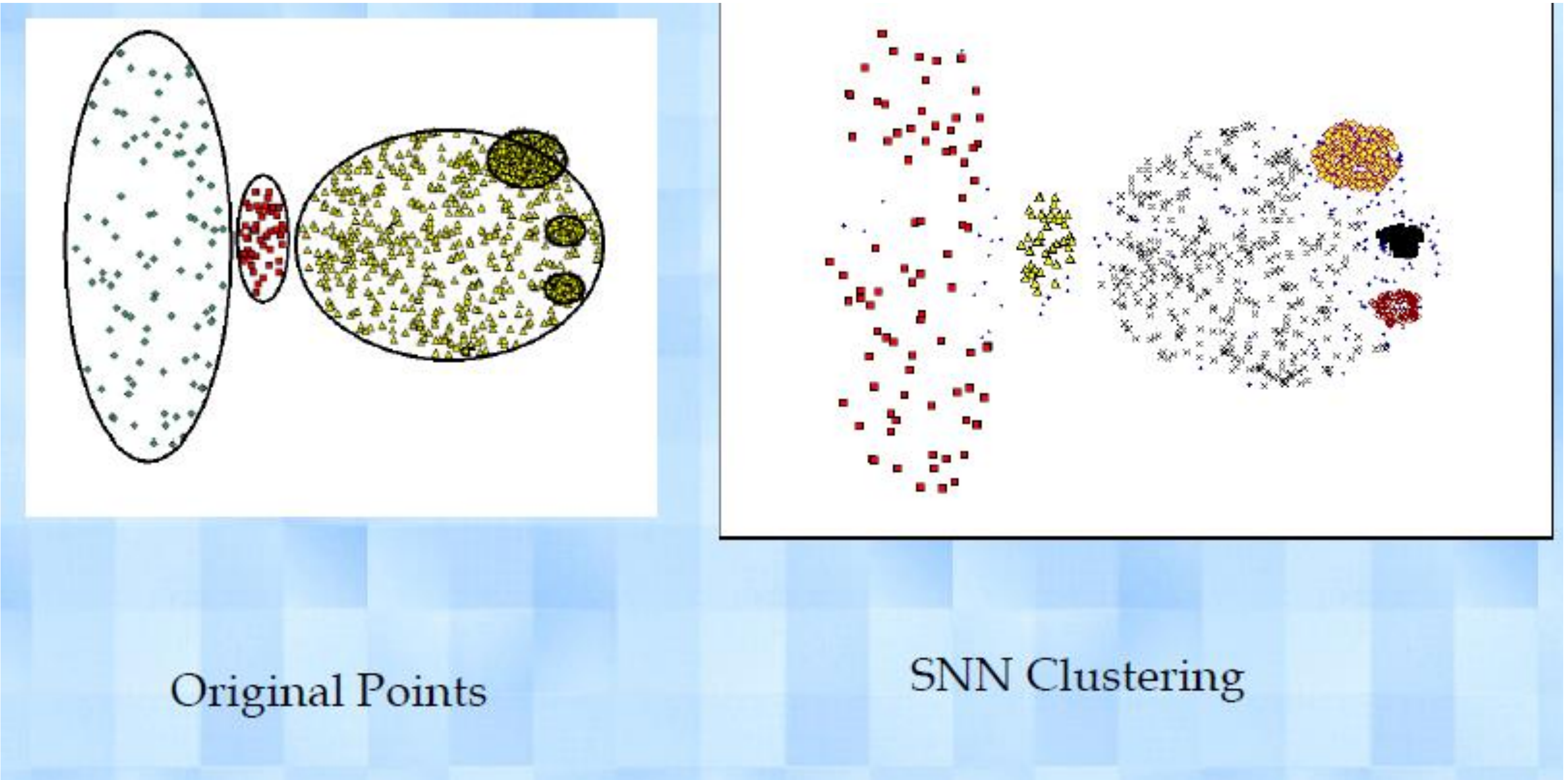
All non-core points that are not within a radius of *Eps* of a core point are discarded.

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

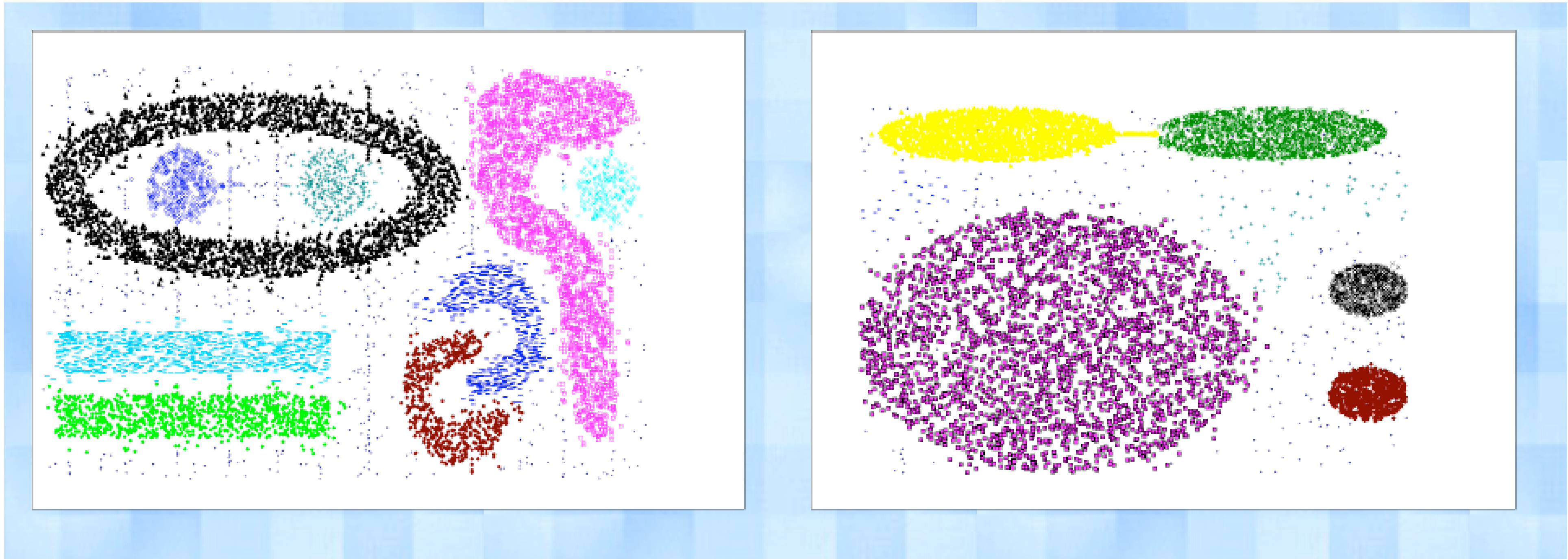
SNN Clustering Can Handle Differing Densities



Original Points

SNN Clustering

SNN Can Handle Other Difficult Situations



Model-based Clustering

Assume data generated from **k** probability distributions

Goal: find the distribution parameters

Algorithm: Expectation Maximization (EM)

Output: Distribution parameters and a **soft** assignment of points to clusters

Mixture Model-Based Clustering

- A set *C* of *k* probabilistic clusters $C_1, \ldots, C_k$ with probability density functions $f_1, \ldots, f_k$, respectively, and their probabilities $\omega_1, \ldots, \omega_k$.
- Probability of an object *o* generated by cluster C_j is

$$P(o|C_j) = \omega_j f_j(o)$$

- Probability of *o* generated by the set of cluster ***C*** is

$$P(o|\boldsymbol{C}) = \sum_{j=1}^{k} \omega_j f_j(o)$$

- Since objects are assumed to be generated independently, for a data set D = $\{o_1, \ldots, o_n\}$, we have,

$$P(D|\boldsymbol{C}) = \prod_{i=1}^{n} P(o_i|\boldsymbol{C}) = \prod_{i=1}^{n} \sum_{j=1}^{k} \omega_j f_j(o_i)$$

- Task: Find a set *C* of *k* probabilistic clusters s.t. *P*(*D*|***C***) is maximized

The EM (Expectation Maximization) Algorithm

- **The (EM) algorithm:** A framework to approach maximum likelihood or maximum a posteriori estimates of parameters in statistical models.
  - **E-step** assigns objects to clusters according to the current fuzzy clustering or parameters of probabilistic clusters
  - **M-step** finds the new clustering or parameters that minimize the sum of squared error (SSE) or the expected likelihood
    - Under uni-variant normal distribution assumptions:
- More about mixture model and EM algorithms: http://www.stat.cmu.edu/~cshalizi/350/lectures/29/lecture-29.pdf

EM Algorithm (Mixture Model)

- Initialize K cluster centers
- Iterate between two steps
  - **E**xpectation step: assign points to clusters (Bayes)

probability that d_i is in class c_j

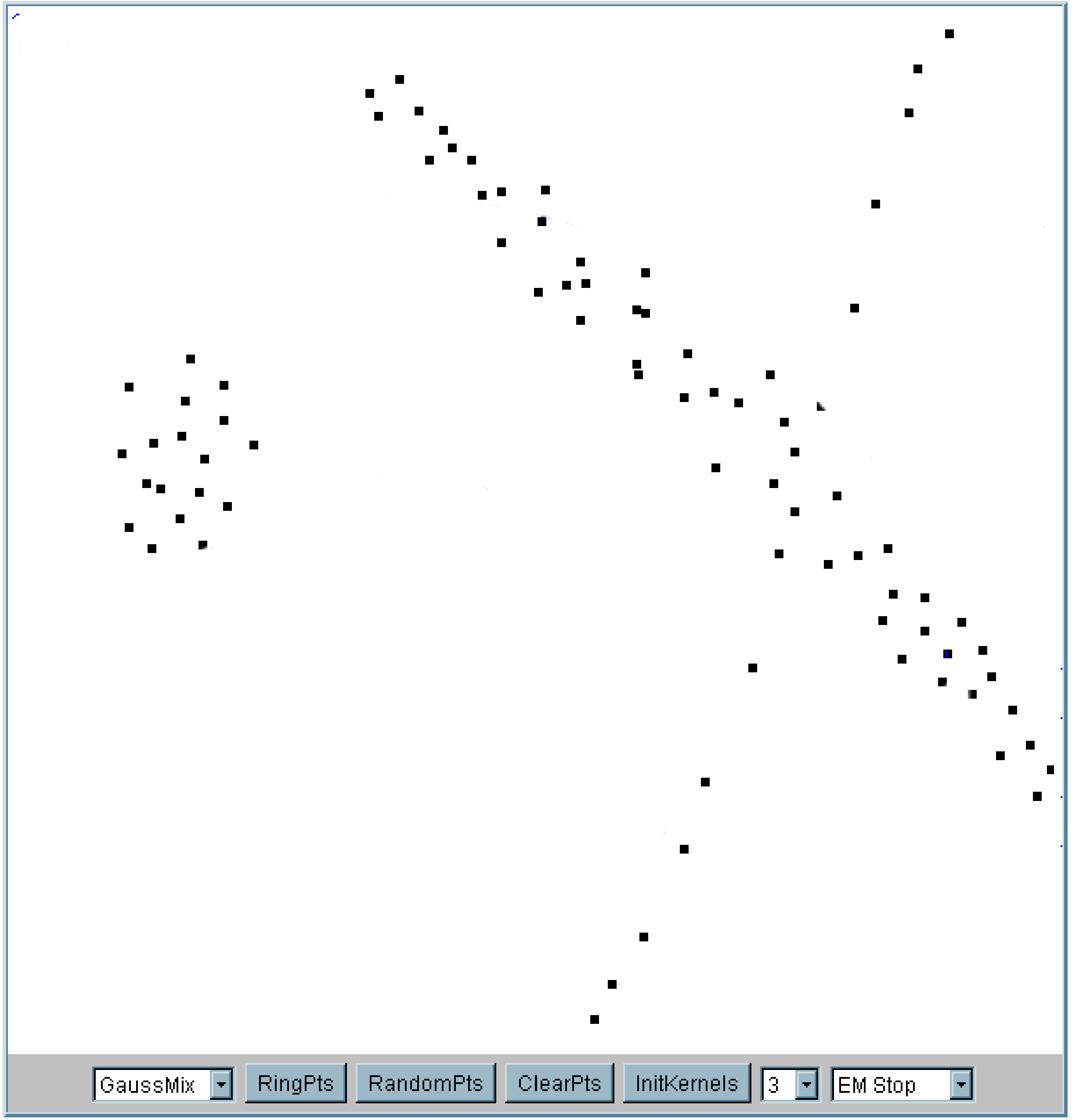
$$P(d_i \in c_k) = w_k \Pr(d_i | c_k) \Big/ \sum_j w_j \Pr(d_i | c_j)$$

$$w_k = \frac{\sum_i \Pr(d_i \in c_k)}{N}$$

= probability of class c_k

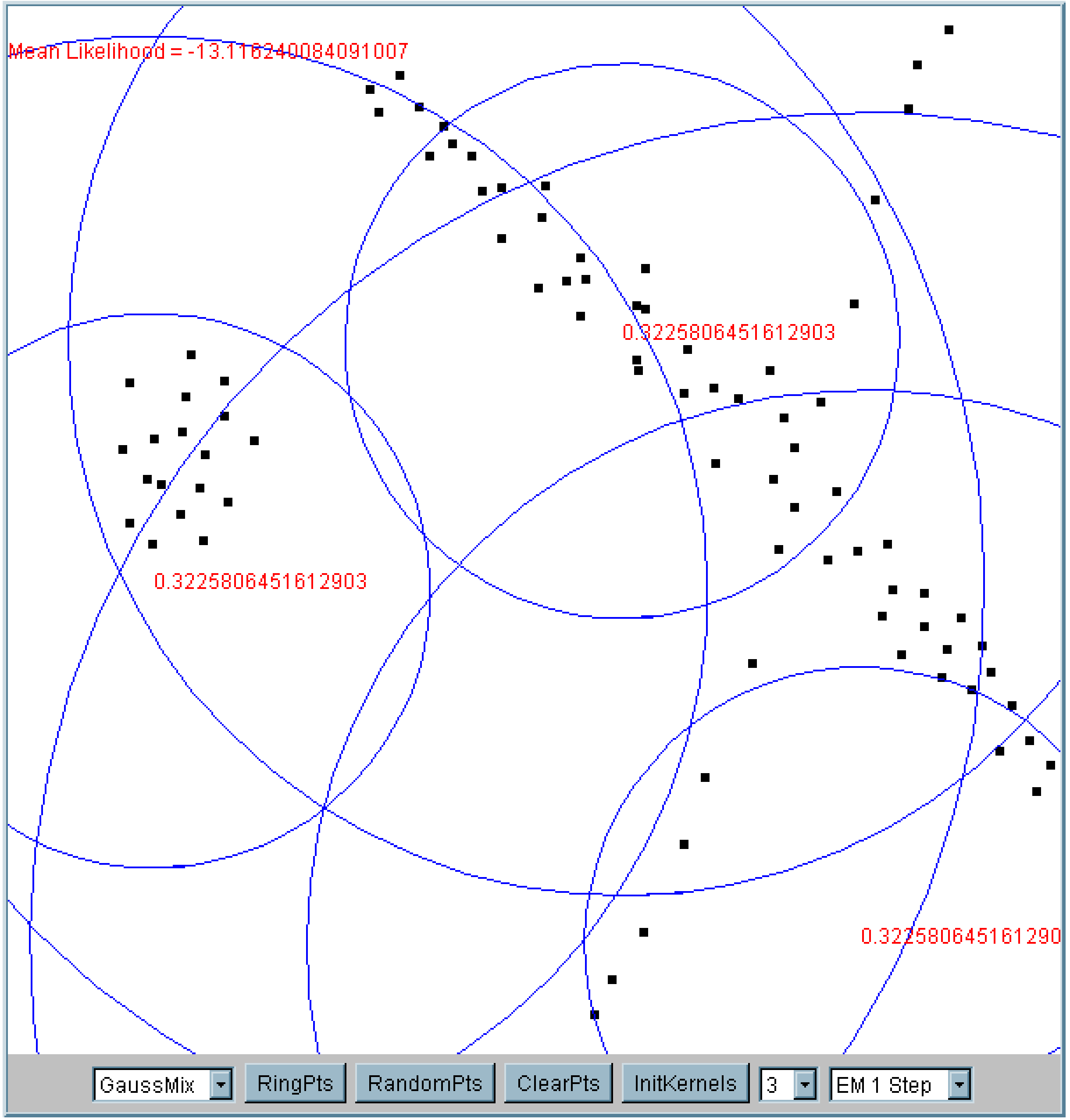
  - **M**aximation step: estimate model parameters (optimization)

$$\mu_k = \frac{1}{m} \sum_{i=1}^{m} \frac{d_i P(d_i \in c_k)}{\sum_k P(d_i \in c_j)}$$


GaussMix
RingPts
RandomPts
ClearPts
InitKernels
3
EM Stop

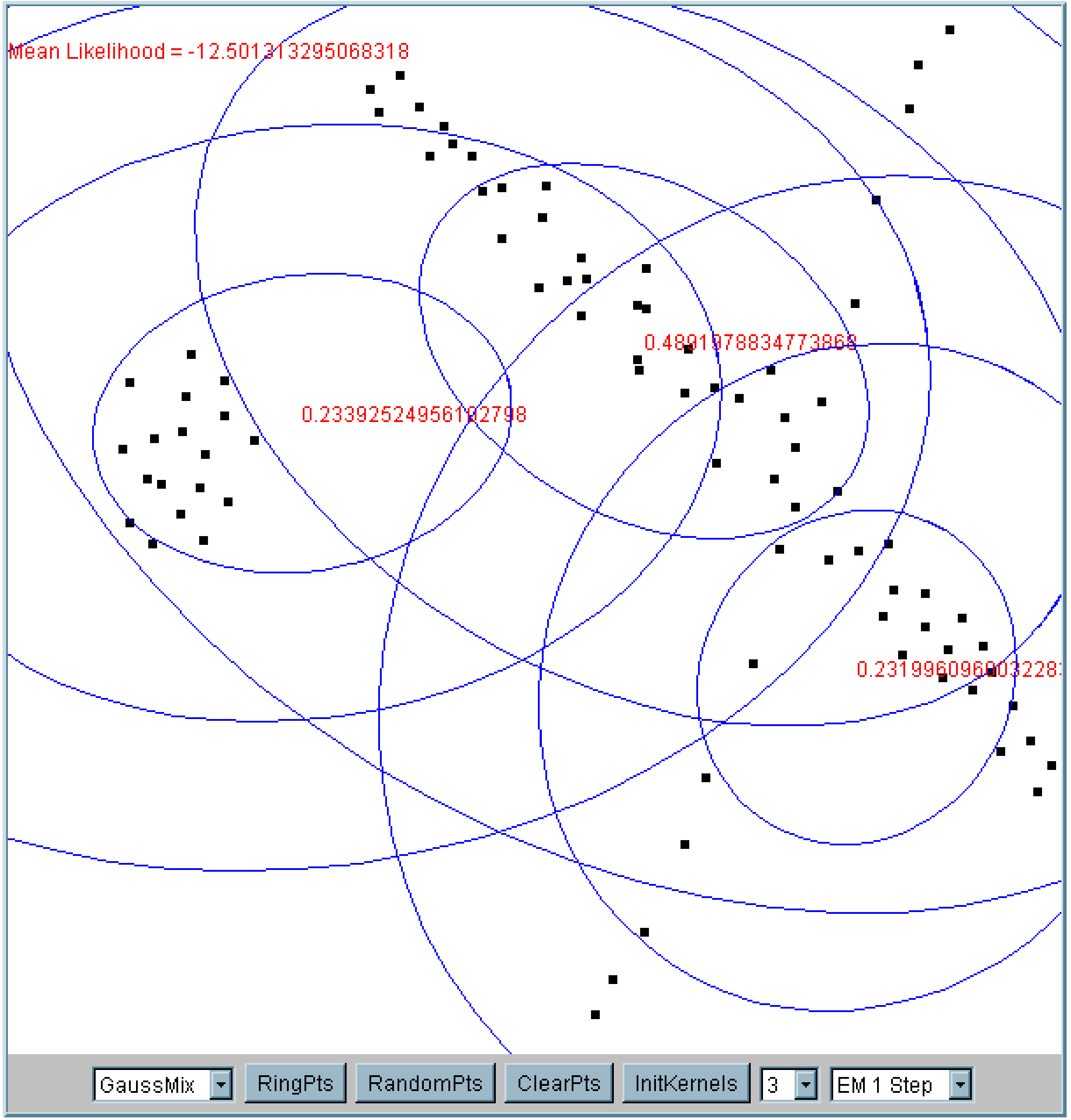
Iteration 1

The cluster means are randomly assigned



Mean Likelihood = -13.116240084091007

Iteration 2



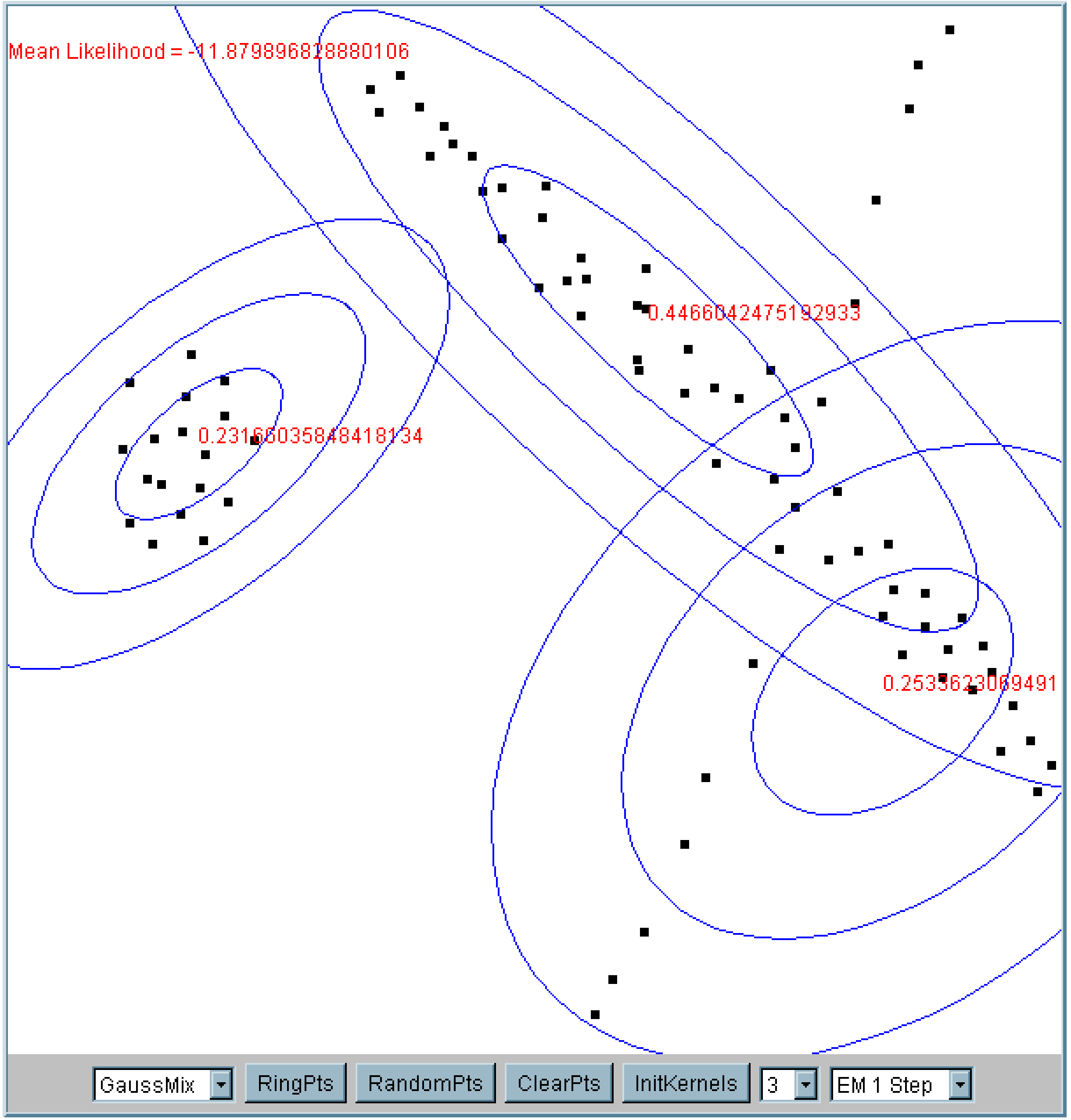
Mean Likelihood = -12.501313295068318

0.2339252495610279 8

0.4891978834773869

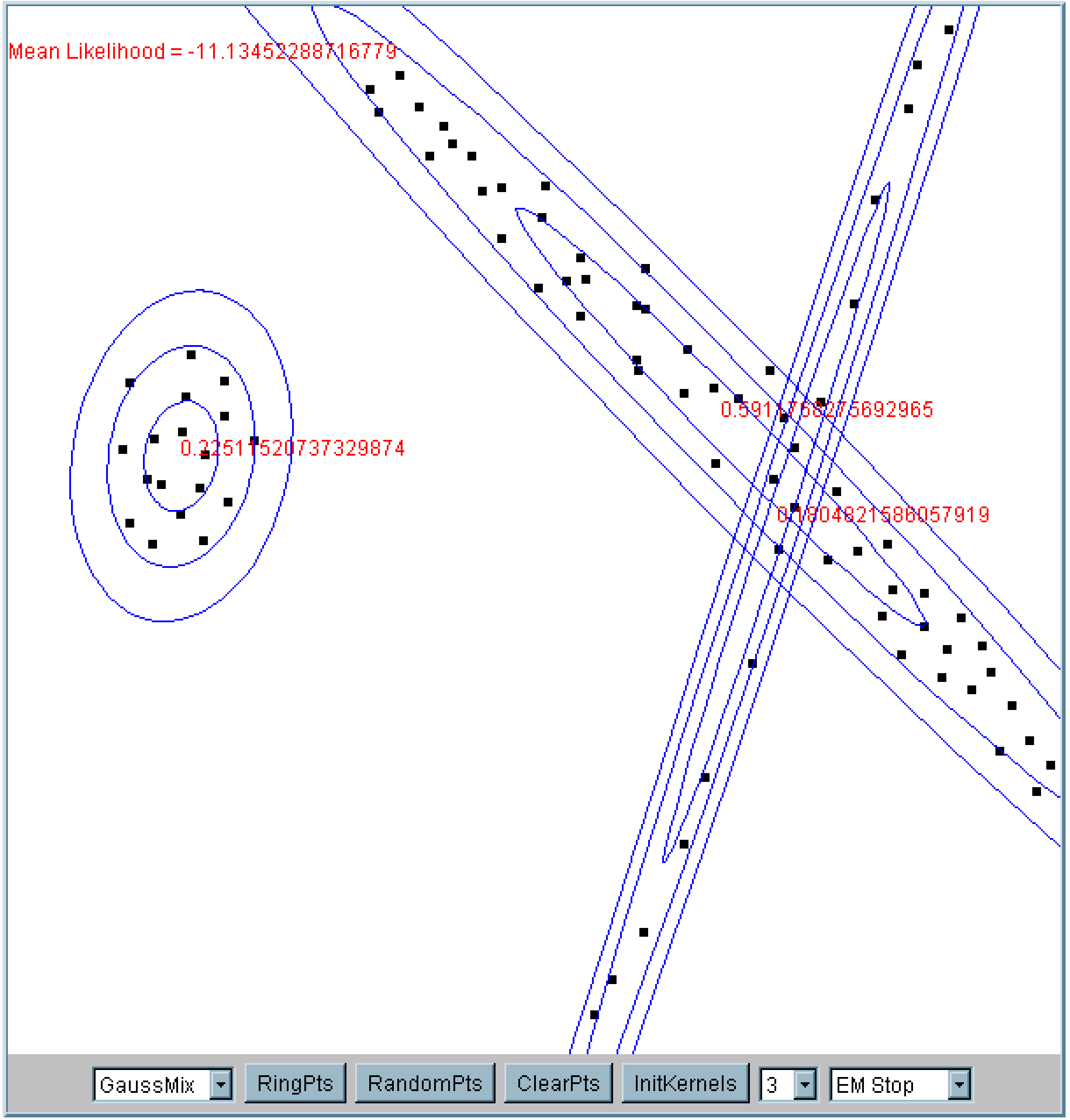
0.231996096003228

Iteration 5



Mean Likelihood = -11.879896828880106

Iteration 25



Mean Likelihood = -11.13452288716779

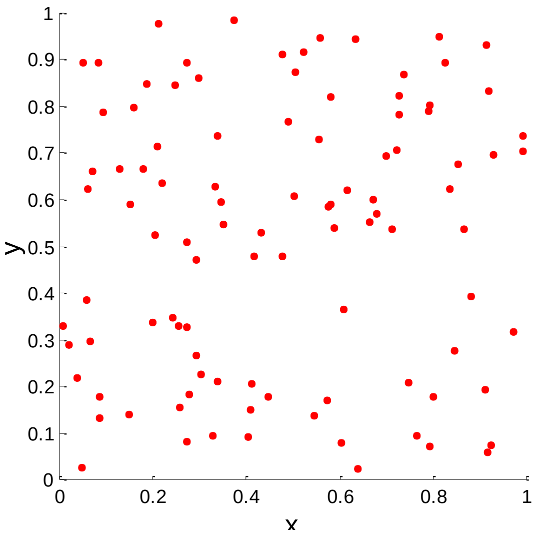
Different Aspects of Cluster Validation

1. Determining the clustering tendency of a set of data, i.e., distinguishing whether non-random structure actually exists in the data.
2. Comparing the results of a cluster analysis to externally known results, e.g., to externally given class labels.
3. Evaluating how well the results of a cluster analysis fit the data *without* reference to external information.
   - Use only the data
4. Comparing the results of two different sets of cluster analyses to determine which is better.
5. Determining the ‘correct’ number of clusters.

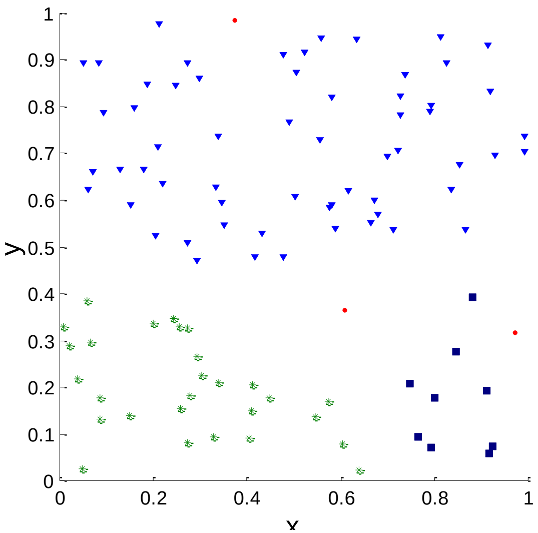
For 2, 3, and 4, we can further distinguish whether we want to evaluate the entire clustering or just individual clusters.

Clusters found in Random Data

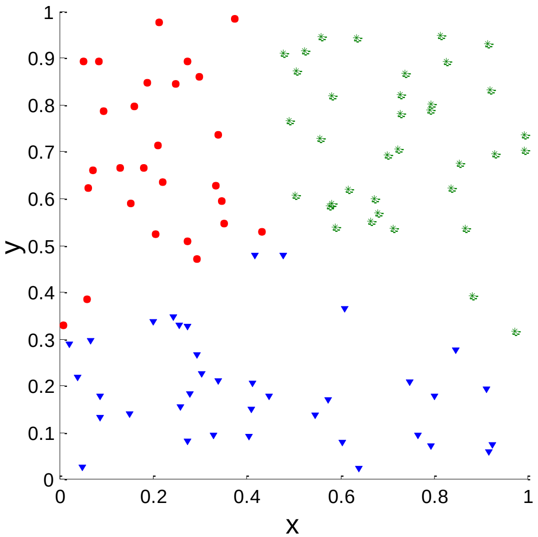
Random Points



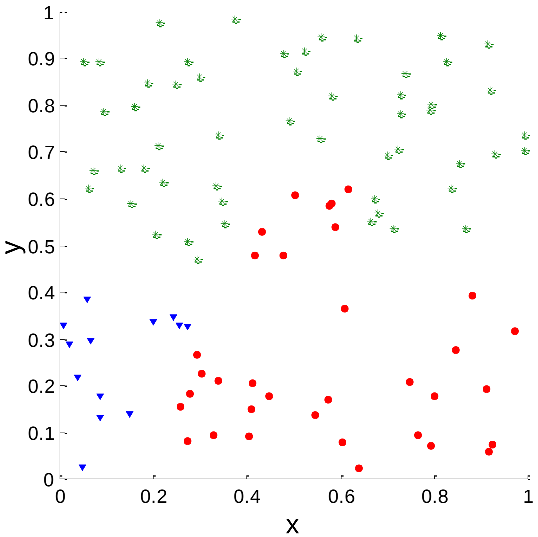
DBSCAN



K-means



Complete Link



Measures of Cluster Validity

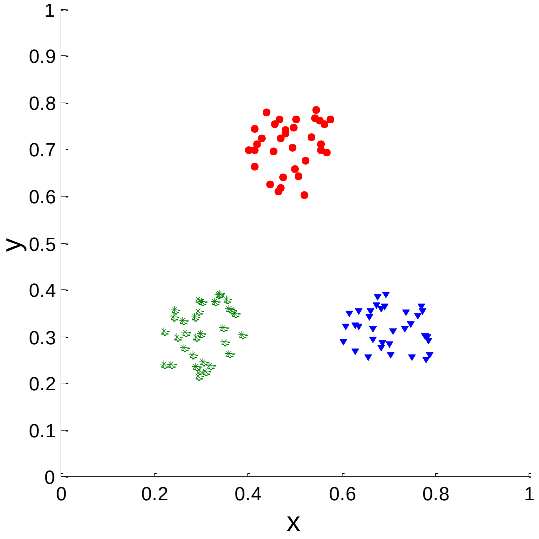
- Numerical measures that are applied to judge various aspects of cluster validity, are classified into the following three types.
  - External Index: Used to measure the extent to which cluster labels match externally supplied class labels.
    - Precision and Recall
  - Internal Index: Used to measure the goodness of a clustering structure *without* respect to external information.
    - Sum of Squared Error (SSE)
  - Relative Index: Used to compare two different clusterings or clusters.
    - Often an external or internal index is used for this function, e.g., SSE or entropy

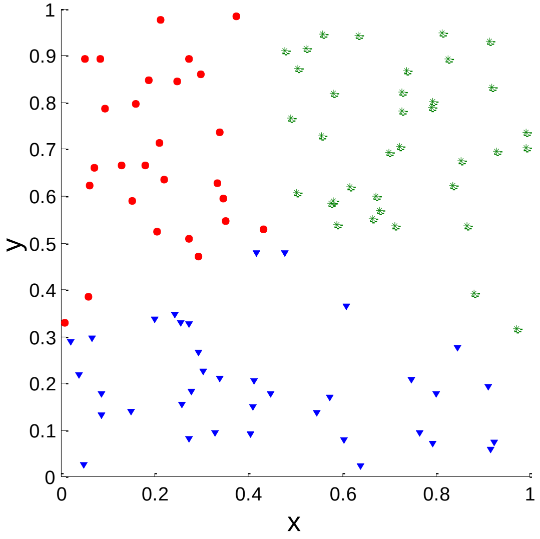
Measuring Cluster Validity Via Correlation

- Two matrices
  - Proximity Matrix
  - “Incidence” Matrix
    - One row and one column for each data point
    - An entry is 1 if the associated pair of points belong to the same cluster
    - An entry is 0 if the associated pair of points belongs to different clusters
- Compute the correlation between the two matrices
  - Since the matrices are symmetric, only the correlation between n(n-1) / 2 entries needs to be calculated.
- High correlation indicates that points that belong to the same cluster are close to each other.
- Not a good measure for some density or contiguity based clusters.

Measuring Cluster Validity Via Correlation

- Correlation of incidence and proximity matrices for the K-means clusterings of the following two data sets.



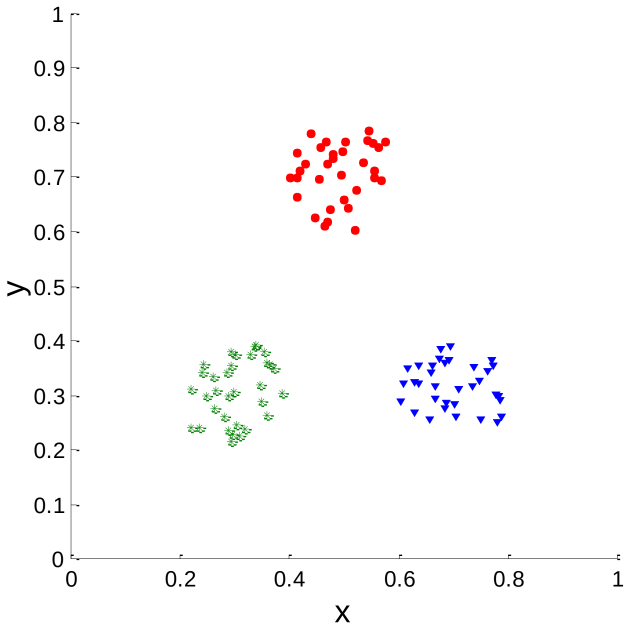


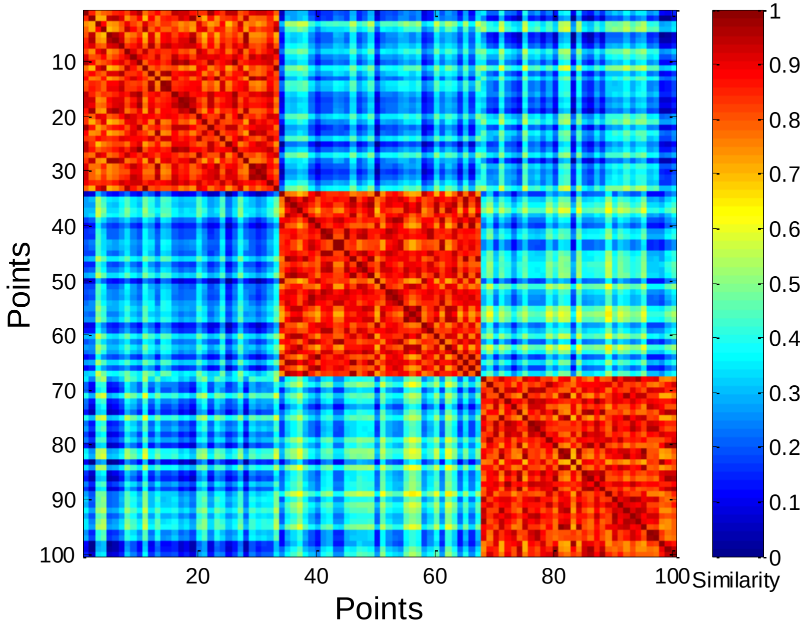
Corr = -0.9235

Corr = -0.5810

Using Similarity Matrix for Cluster Validation

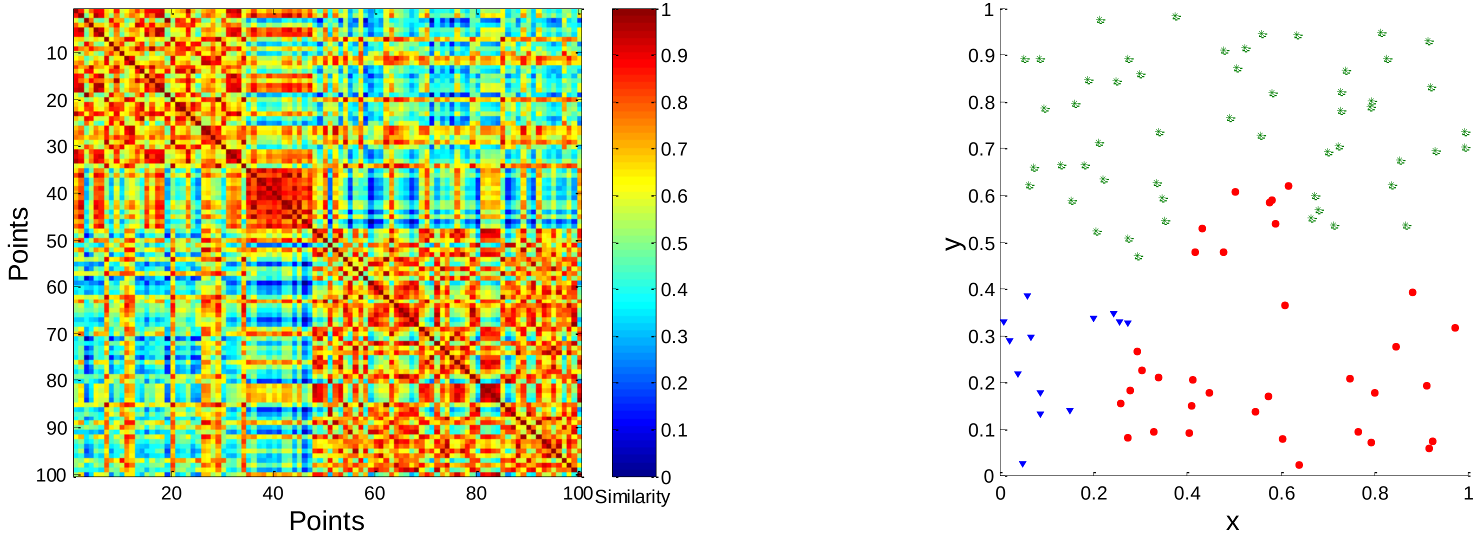
- Order the similarity matrix with respect to cluster labels and inspect visually.





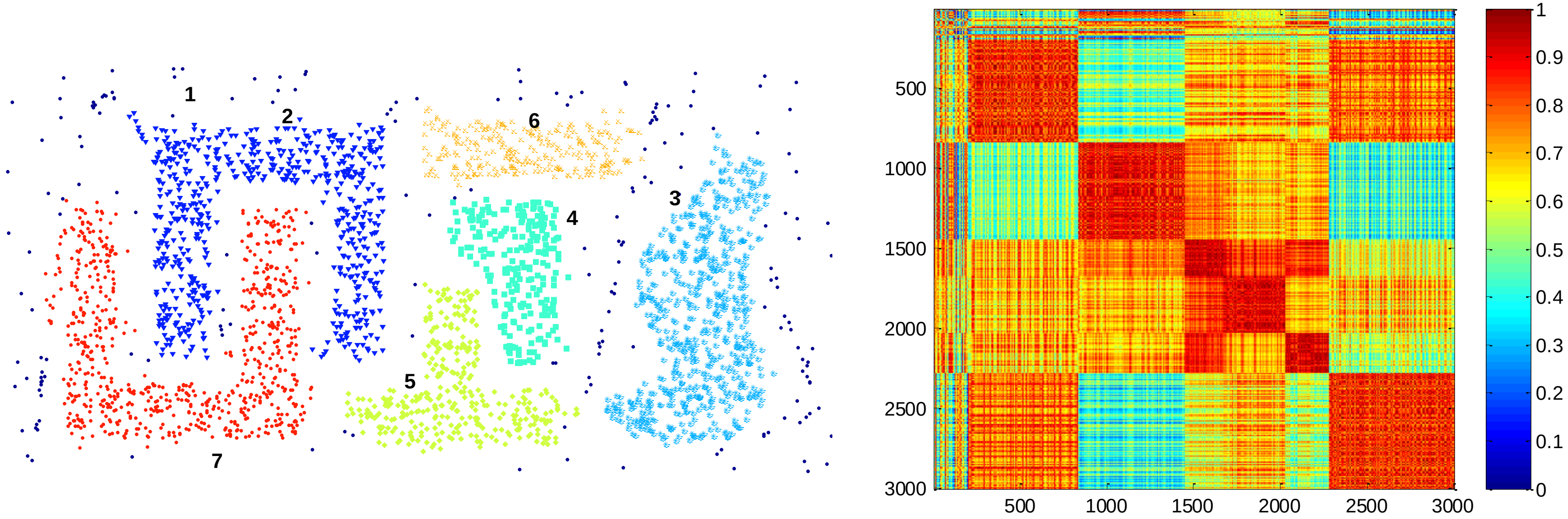
Using Similarity Matrix for Cluster Validation

- Clusters in random data are not so crisp



Complete Link

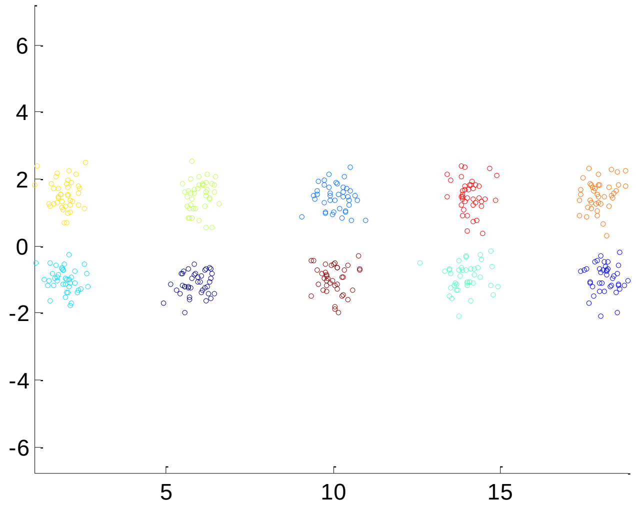
Using Similarity Matrix for Cluster Validation

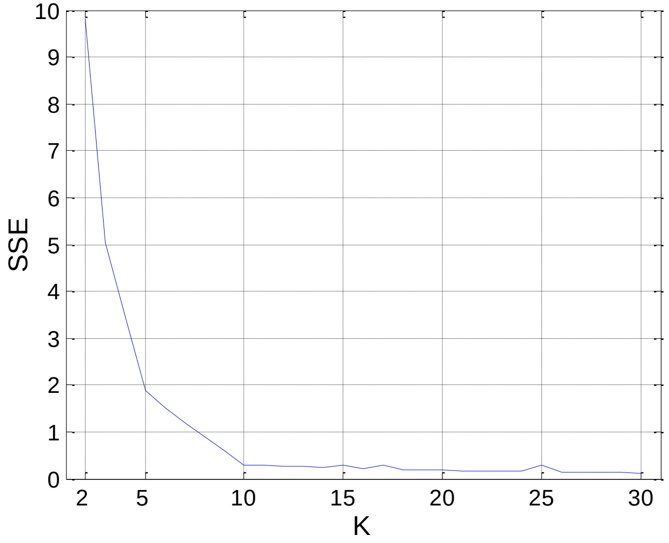


DBSCAN

Internal Measures: SSE

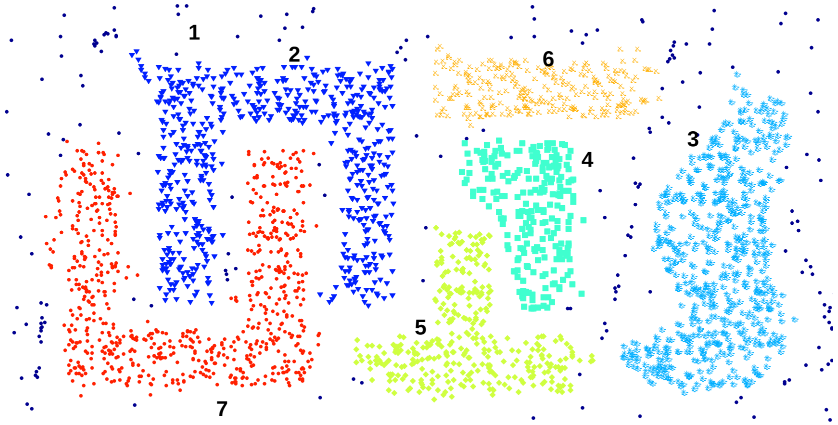
- Clusters in more complicated figures aren't well separated
- Internal Index: Used to measure the goodness of a clustering structure without respect to external information
  - SSE
- SSE is good for comparing two clusterings or two clusters (average SSE).
- Can also be used to estimate the number of clusters

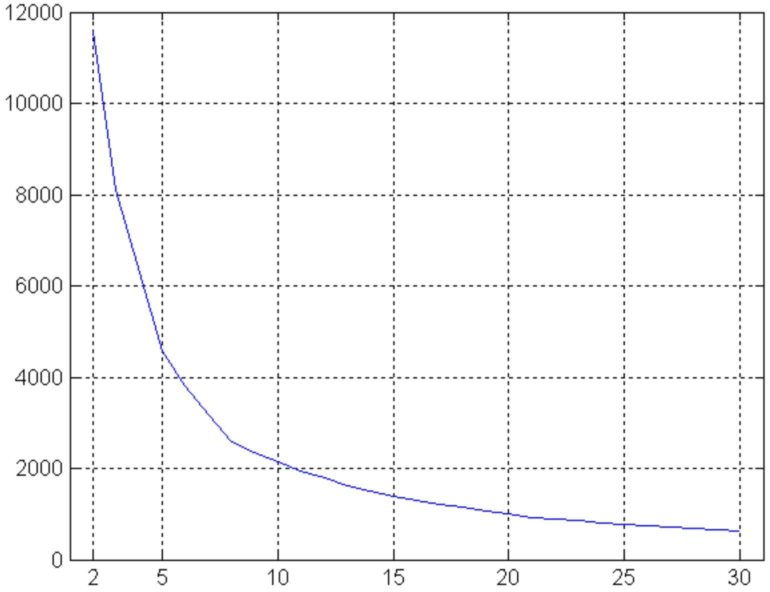




Internal Measures: SSE

- SSE curve for a more complicated data set

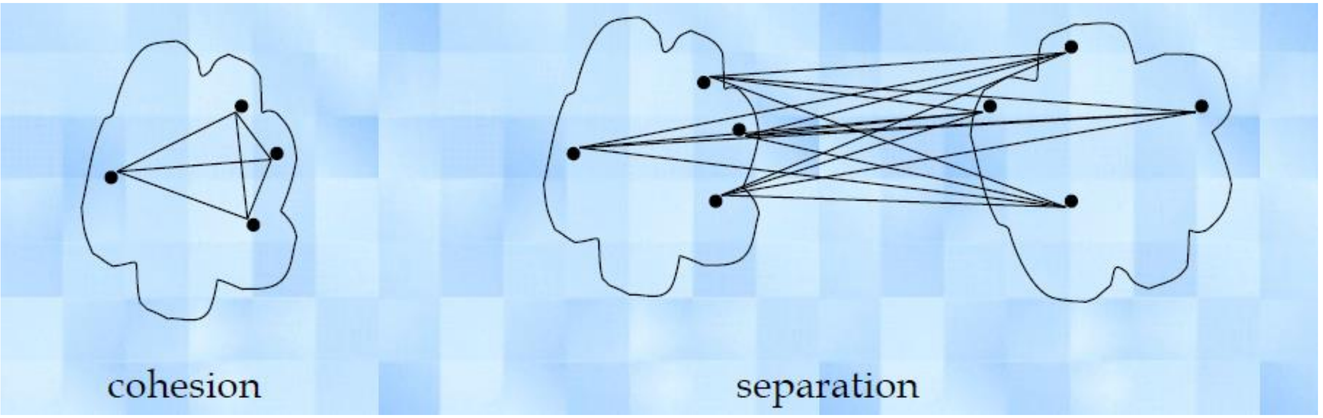




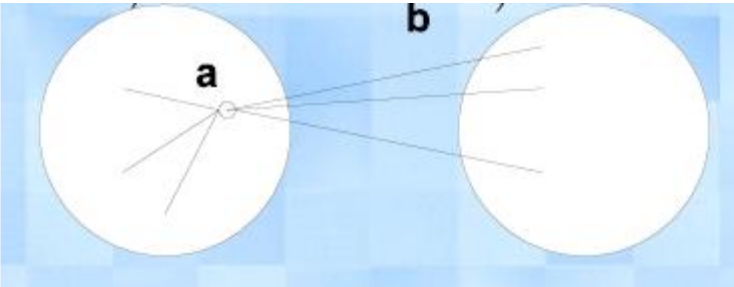
SSE of clusters found using K-means

Internal Measures: Cohesion and Separation

- Cluster Cohesion: Measures how closely related are objects in a cluster
- Cluster cohesion is the sum of the weight of all links within a Cluster
- Cluster Separation: Measure how distinct or well-separated a cluster is from other clusters
- Cluster separation is the sum of the weights between nodes in the cluster and nodes outside the cluster



Internal Measures of Cluster Validity: Silhouette Coefficient



Silhouette Coefficient combine ideas of both cohesion
and separation, but for individual points, as well as clusters and clusterings

- For an individual point, *i*

– Calculate ***a*** = average distance of *i* to the points in its cluster

– Calculate ***b*** = min (average distance of *i* to points in another cluster)

– The silhouette coefficient for a point is then given by

s = 1 – a/b if a < b, (or s = b/a - 1 if a ≥ b, not the usual case)

– Typically between 0 and 1

– The closer to 1 the better

- Can calculate the Average Silhouette width for a cluster or a clustering

External Measures of Cluster Validity: Entropy and Purity

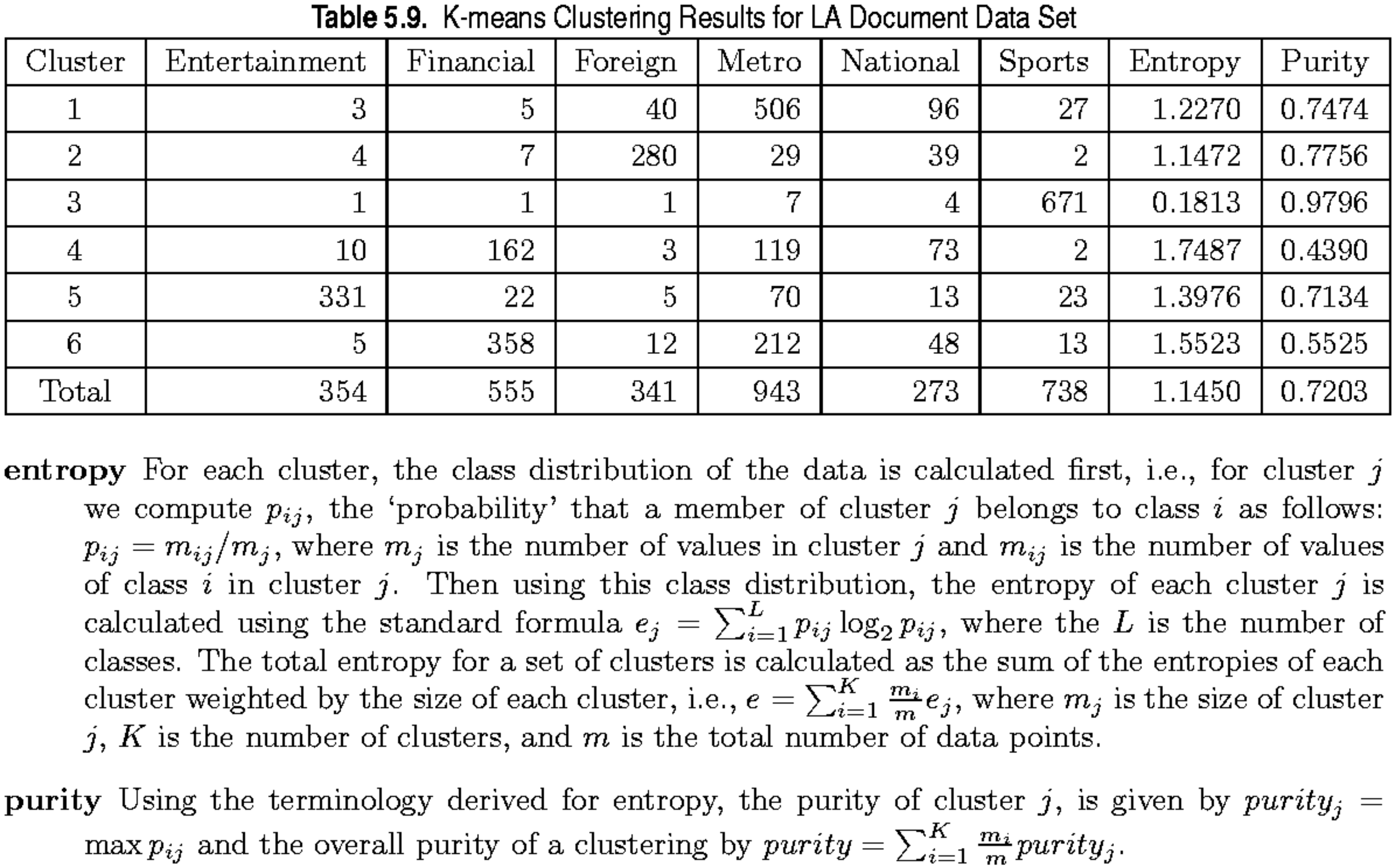


Table 5.9. K-means Clustering Results for LA Document Data Set

Cluster	Entertainment	Financial	Foreign	Metro	National	Sports	Entropy	Purity
1	3	5	40	506	96	27	1.2270	0.7474
2	4	7	280	29	39	2	1.1472	0.7756
3	1	1	1	7	4	671	0.1813	0.9796
4	10	162	3	119	73	2	1.7487	0.4390
5	331	22	5	70	13	23	1.3976	0.7134
6	5	358	12	212	48	13	1.5523	0.5525
Total	354	555	341	943	273	738	1.1450	0.7203

entropy For each cluster, the class distribution of the data is calculated first, i.e., for cluster j we compute p_{ij}, the 'probability' that a member of cluster j belongs to class i as follows: $p_{ij} = m_{ij}/m_j$, where m_j is the number of values in cluster j and m_{ij} is the number of values of class i in cluster j. Then using this class distribution, the entropy of each cluster j is calculated using the standard formula $e_j = \sum_{i=1}^{L} p_{ij} \log_2 p_{ij}$, where the L is the number of classes. The total entropy for a set of clusters is calculated as the sum of the entropies of each cluster weighted by the size of each cluster, i.e., $e = \sum_{i=1}^{K} \frac{m_i}{m} e_j$, where m_j is the size of cluster j, K is the number of clusters, and m is the total number of data points.

purity Using the terminology derived for entropy, the purity of cluster j, is given by $purity_j = \max p_{ij}$ and the overall purity of a clustering by $purity = \sum_{i=1}^{K} \frac{m_i}{m} purity_j$.

Final Comment on Cluster Validity

"The validation of clustering structures is the most difficult and frustrating part of cluster analysis.

Without a strong effort in this direction, cluster analysis will remain a black art accessible only to those true believers who have experience and great courage."

Algorithms for Clustering Data, Jain and Dubes