Data Mining:

4. Clusteranalyse

Nachtrag) Similarity, Dissimilarity

B) More Clustering Algorithms, Cluster Evaluation

Similarity and Dissimilarity

- Similarity
  - Numerical measure of how alike two data objects are
  - is higher when objects are more alike
  - often falls in the range [0,1]
- Dissimilarity (or Distance)
  - Numerical measure of how different two data objects are
  - is lower when objects are more alike
  - Minimum dissimilarity is often 0
  - Upper limit varies
- “Proximity” refers to similarity or dissimilarity (!)

Similarity/Dissimilarity for Simple Attributes

p and *q* are the attribute values for two data objects.

Attribute Type	Dissimilarity	Similarity
Nominal (Categorial)	$d = \begin{cases} 0 & \text{if } p = q \\ 1 & \text{if } p \neq q \end{cases}$	$s = \begin{cases} 1 & \text{if } p = q \\ 0 & \text{if } p \neq q \end{cases}$
Ordinal (Categorial and ordered)	$d = \frac{\lvert p-q \rvert}{n-1}$ (values mapped to integers 0 to $n-1$, where n is the number of values)	$s = 1 - d$
Interval or Ratio (Continuous)	$d = \lvert p - q \rvert$	$s = 1-d$, $s = \frac{1}{1+d}$ or $s = 1 - \frac{d - min\_d}{max\_d - min\_d}$ or $s = e^{-d}$

Table 5.1. Similarity and dissimilarity for simple attributes

Euclidean Distance between Data Objects

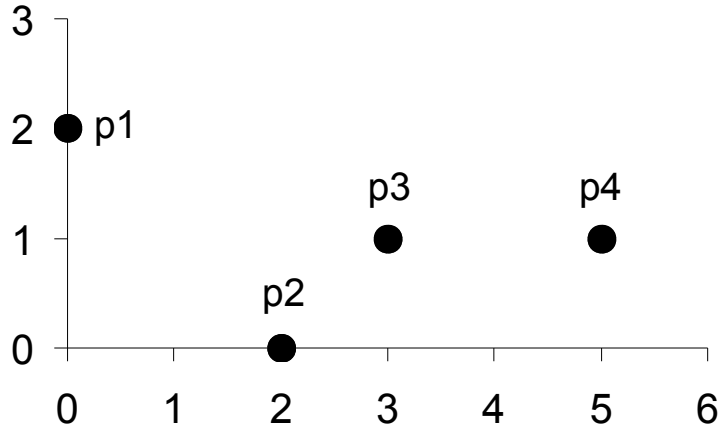
- Euclidean Distance ***dist(p,q)***

$$= \sqrt{\sum_{k=1}^{n} (p_k - q_k)^2}$$

where n is the number of dimensions (or attributes) and p_k and q_k are, respectively, the k^{th} attributes (or components) of data objects p and q.
Of course, these must be continuous attributes.

- Normalization of different dimensions is needed, if scales differ.

Euclidean Distance between Data Objects



point	x	y
p1	0	2
p2	2	0
p3	3	1
p4	5	1

	p1	**p2**	**p3**	**p4**
p1	0	2,828	3,162	5,099
p2	2,828	0	1,414	3,162
p3	3,162	1,414	0	2
p4	5,099	3,162	2	0

Distance Matrix

Minkowski Distance between Data Objects

- Minkowski Distance is a generalization of Euclidean Distance; also called L_r norm

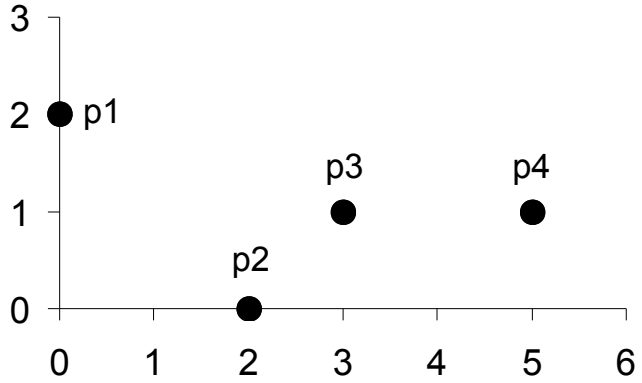
$$dist_r = \left(\sum_{k=1}^{n} | p_k - q_k |^r \right)^{\frac{1}{r}}$$

Where r is a parameter,
n is the number of dimensions (attributes)
and p_k and q_k are, respectively, the k^{th} attributes (components) of data objects p and q.

Minkowski Distance: Examples

- *r* = 1. City block (Manhattan, taxicab, L_1 norm) distance.
  - A common example of this is the Hamming distance, which is just the number of bits that are different between two binary vectors
- *r* = 2. Euclidean distance
- $lim_{r \to \infty} dist_r$. "supremum" ($L_{max}$ or L_∞ norm) distance.
  - This is the maximum difference between any component of the vectors
- *Do not confuse r with n, i.e., all these distances are defined for all numbers of dimensions.*

Minkowski Distance: Examples



point	x	y
p1	0	2
p2	2	0
p3	3	1
p4	5	1

L1	p1	p2	p3	p4
p1	0	4	4	6
p2	4	0	2	4
p3	4	2	0	2
p4	6	4	2	0

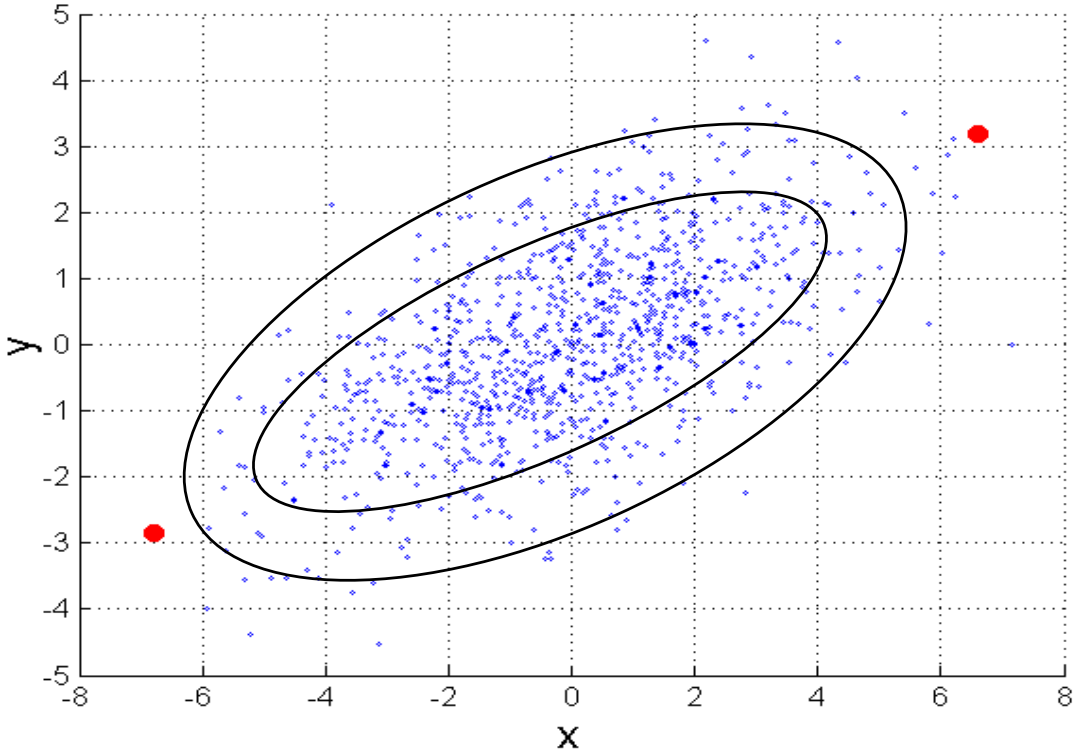
L2	p1	p2	p3	p4
p1	0	2,828	3,162	5,099
p2	2,828	0	1,414	3,162
p3	3,162	1,414	0	2
p4	5,099	3,162	2	0

L_∞	p1	p2	p3	p4
p1	0	2	3	5
p2	2	0	1	3
p3	3	1	0	2
p4	5	3	2	0

Distance Matrix

A Distance Considering Correlation of Attributes

$$mahalanobis(p,q) = (p-q)\sum^{-1}(p-q)^{T}$$



Σ is the covariance matrix of the input data *X*

$$\Sigma_{j,k} = \frac{1}{n-1}\sum_{i=1}^{n}(X_{ij} - \overline{X}_j)(X_{ik} - \overline{X}_k)$$

Idea:
distance acc.to prediction ellipses

For red points, the Euclidean distance is 14.7, Mahalanobis distance is 6.

Common Properties of a Distance

- Distances, such as the Euclidean distance, often have some well known properties.

  1. $d(p, q) \geq 0$ for all p and q and $d(p, q) = 0$ only if $p = q$. (Positive definiteness)
  2. $d(p, q) = d(q, p)$ for all p and q. (Symmetry)
  3. $d(p, r) \leq d(p, q) + d(q, r)$ for all points p, q, and r. (Triangle Inequality)

  where $d(p,q)$ is the distance between points (data objects) p and q.

- A distance that satisfies these properties is a metric

Common Properties of a Similarity

- Similarities also have some well known properties.

  1. $s(p, q) = 1$ (or maximum similarity) only if $p = q$.

  2. $s(p, q) = s(q, p)$ for all p and q. (Symmetry)

  where $s(p, q)$ is the similarity between points (data objects) p, q.

Similarity Coefficients Between Binary Vectors

- Another common situation is that objects, *p* and *q*, have only binary attributes
- Compute similarities using the following quantities

  M_{01} = the number of attributes where p was 0 and q was 1

  M_{10} = the number of attributes where p was 1 and q was 0

  M_{00} = the number of attributes where p was 0 and q was 0

  M_{11} = the number of attributes where p was 1 and q was 1
- Simple Matching Coefficient and Jaccard Coefficient:

  for symmetric binary attributes:

  SMC = number of matches / number of attributes

  $= (M_{11} + M_{00}) / (M_{01} + M_{10} + M_{11} + M_{00})$

  for asymmetric binary attributes:

  J = number of 11 matches / number of not-both-zero attributes values

  $= (M_{11}) / (M_{01} + M_{10} + M_{11})$

SMC versus Jaccard: Example

p = 1 0 0 0 0 0 0 0 0 0

q = 0 0 0 0 0 0 1 0 0 1

M_{01} = 2 (the number of attributes where p was 0 and q was 1)

M_{10} = 1 (the number of attributes where p was 1 and q was 0)

M_{00} = 7 (the number of attributes where p was 0 and q was 0)

M_{11} = 0 (the number of attributes where p was 1 and q was 1)

SMC = $(M_{11} + M_{00})/(M_{01} + M_{10} + M_{11} + M_{00})$ = (0+7) / (2+1+0+7) = 0.7

J = (M_{11}) / $(M_{01} + M_{10} + M_{11})$ = 0 / (2 + 1 + 0) = 0

Cosine Similarity of Termcount Vectors

- If d_1 and d_2 are two document vectors of term frequencies, then

  $$\cos(d_1, d_2) = (d_1 / ||d_1||) \bullet (d_2 / ||d_1||),$$

  where • indicates vector dot product and $||d||$ is the length of vector d.

- Explanation:

  The vectors are normalized to length 1.
  The result is the cosine of the angle between the vectors.
  ($\cos(0^o)=1$ maximal similarity, $\cos(90^o)=0$)

- Example:

  d_1 = **3 2 0 5 0 0 0 2 0 0**

  d_2 = **1 0 0 0 0 0 0 1 0 2**

  $d_1 \bullet d_2 = 3*1 + 2*0 + 0*0 + 5*0 + 0*0 + 0*0 + 0*0 + 2*1 + 0*0 + 0*2 = 5$

  $||d_1|| = (3*3+2*2+0*0+5*5+0*0+0*0+0*0+2*2+0*0+0*0)^{0.5} = (42)^{0.5} = 6.481$

  $||d_2|| = (1*1+0*0+0*0+0*0+0*0+0*0+0*0+1*1+0*0+2*2)^{0.5} = (6)^{0.5} = 2.245$

  $\cos(d_1, d_2) = 0.3150$

General Approach for Combining Similarities

- Sometimes attributes are of many different types, but an overall similarity is needed.
- A typical calculation:
  take (weighted) average of similarity values
- Averaging in the presence of asymmetric attributes:

1. For the k^{th} attribute, compute a similarity, s_k, in the range $[0, 1]$.
2. Define an indicator variable, δ_k, for the k_{th} attribute as follows:

$$\delta_k = \begin{cases} 0 & \text{if the } k^{th} \text{ attribute is a binary asymmetric attribute and both objects have} \\ & \text{a value of 0, or if one of the objects has a missing values for the } k^{th} \text{ attribute} \\ 1 & \text{otherwise} \end{cases}$$

3. Compute the overall similarity between the two objects using the following formula:

$$similarity(p, q) = \frac{\sum_{k=1}^{n} \delta_k s_k}{\sum_{k=1}^{n} \delta_k}$$

Density

- Density-based clustering requires a notion of density

- Examples:
  - Euclidean density
    = number of points per unit volume
    - cell/grid- or center-based
  - Probability density
  - Graph-based density

Euclidean Density – Cell-based

- Simplest approach is to divide region into a number of rectangular cells of equal volume and define density as # of points the cell contains

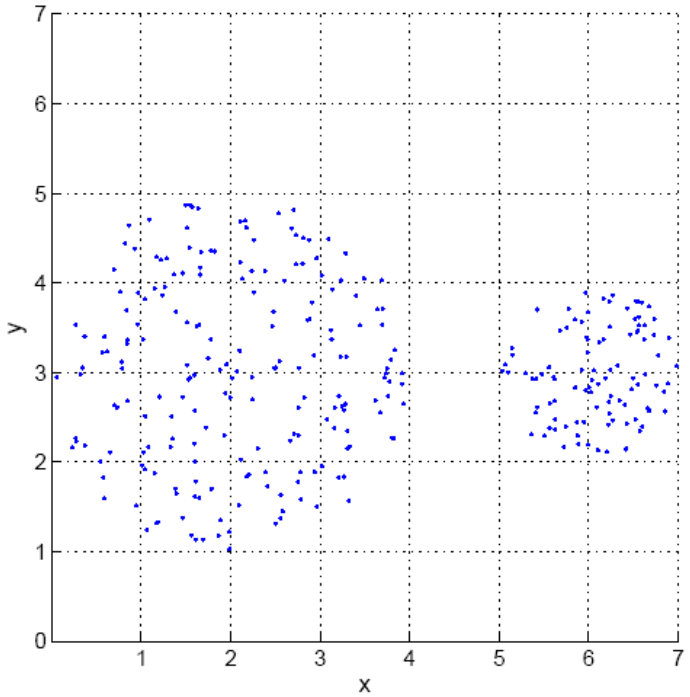


Figure 7.13. Cell-based density.

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

Table 7.6. Point counts for each grid cell.

Euclidean Density – Center-based

- Euclidean density is the number of points within a specified radius of the point

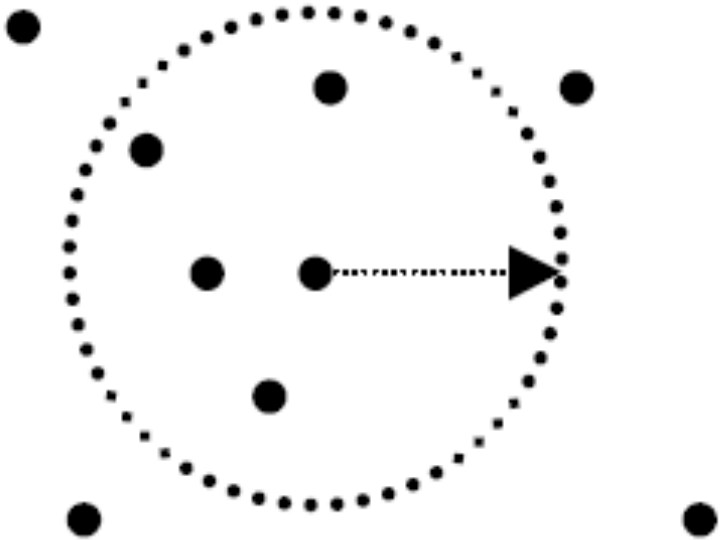
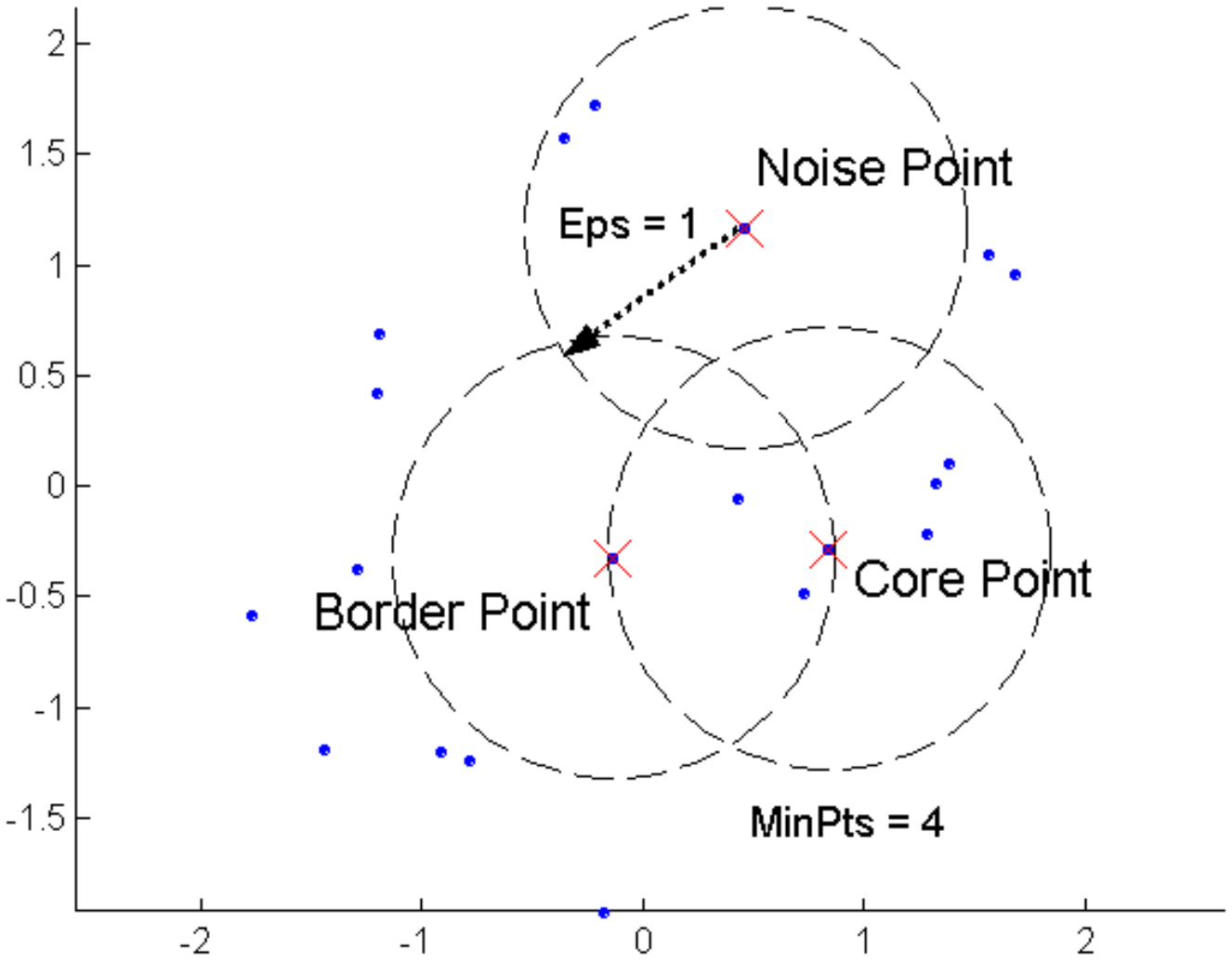


Figure 7.14. Illustration of center-based density.

Density-based Clustering: DBSCAN

- DBSCAN is a density-based algorithm.
  - Density = number of points within a specified radius (Eps)
  - A point is a core point if it has more than a specified number of points (MinPts) within Eps
    - These are points that are at the interior of a cluster
  - A border point has fewer than MinPts within Eps, but is in the Eps-neighborhood of one or several core points.
  - A noise point is any point that is not a core point or a border point.

DBSCAN: Core, Border, and Noise Points



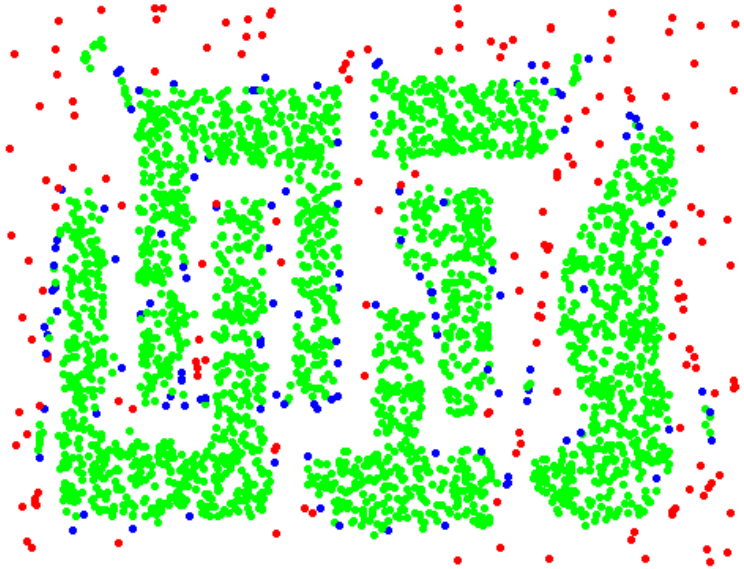
DBSCAN Algorithm

1. Label all points as core, border, or noise points.
2. Eliminate noise points
3. Put an edge between all core points that are within Eps of each other
4. Make each group of connected core points into a separate cluster
5. Assign each border point to the cluster of one of its associated core points.

- Space complexity: O(N)
- Time Complexity: O(N· time to find points in Eps-neighbourhood), worst case $O(N^2)$, with appropriate spatial data structures O(N·log N)

DBSCAN: Core, Border and Noise Points

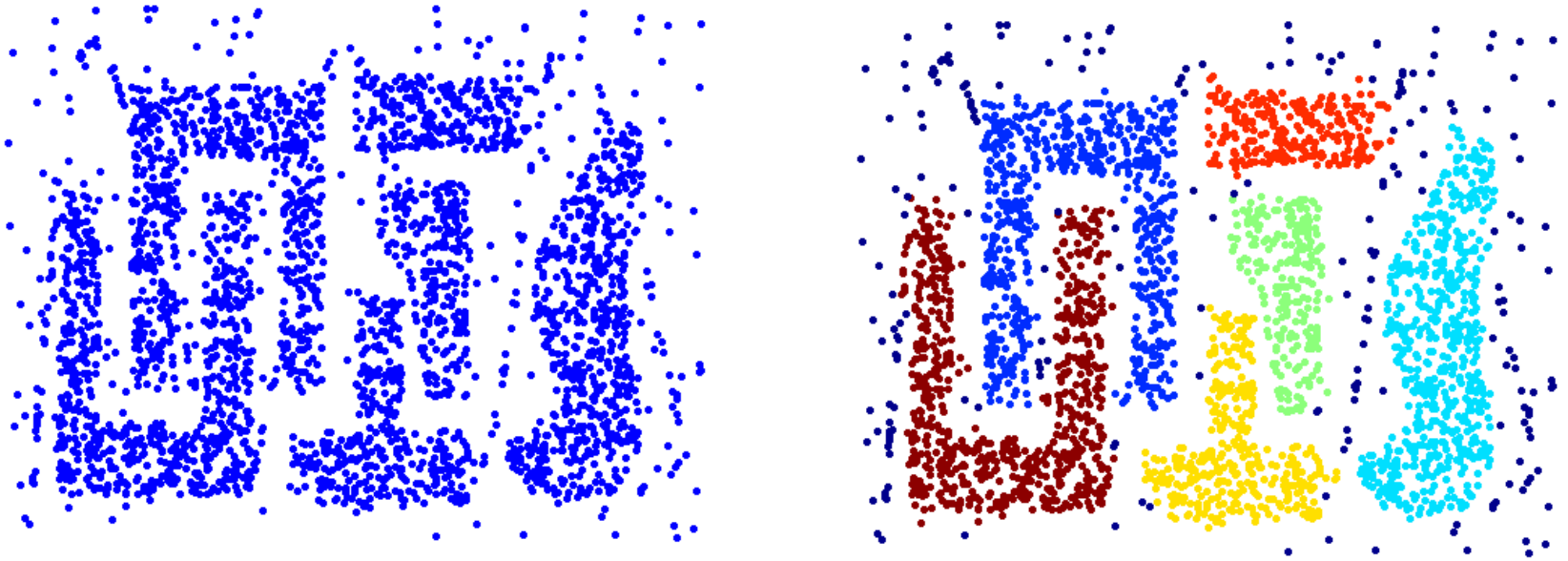
3000 Original Points



Point types: core, border and noise

Eps = 10, MinPts = 4

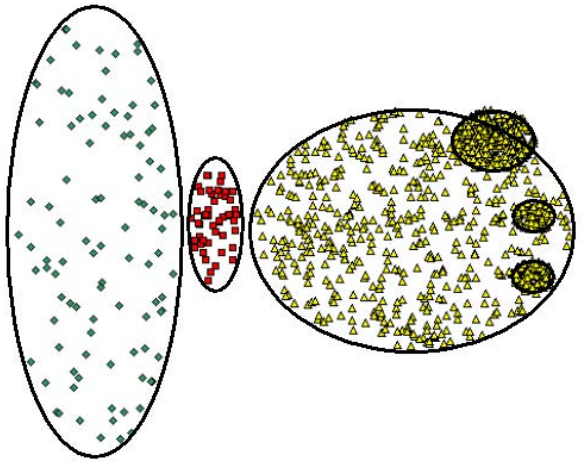
When DBSCAN Works Well (Example A)



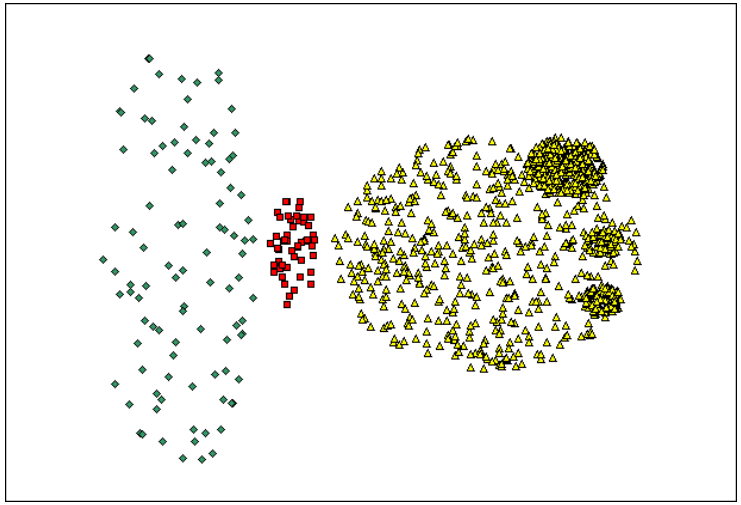
Clusters

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

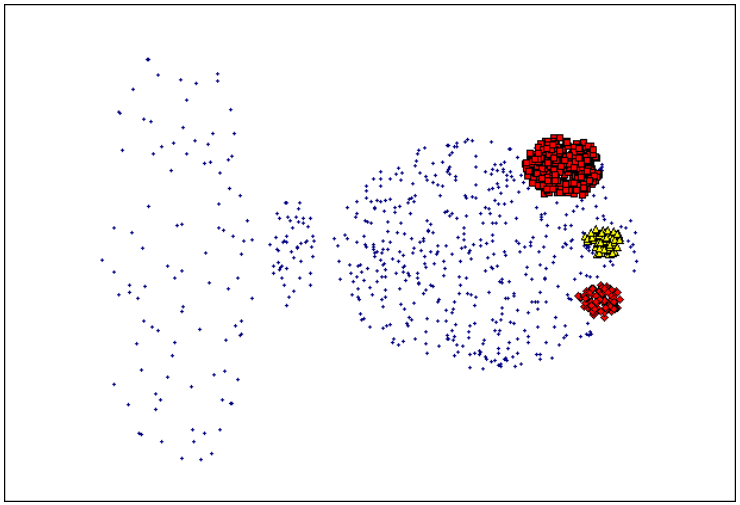
When DBSCAN Does NOT Work Well (Example B)



Original Points



(MinPts=4, Eps=9.75).

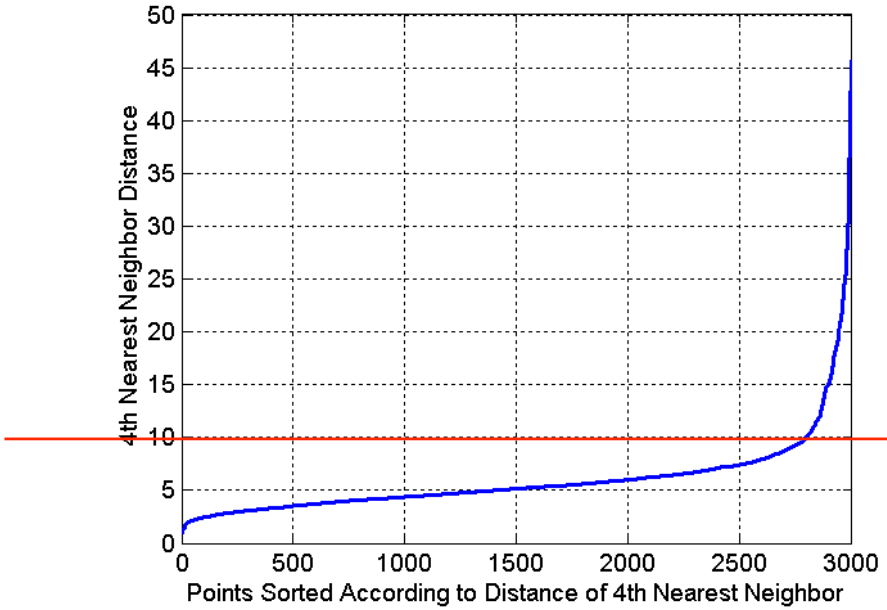


(MinPts=4, Eps=9.92)

- **Varying densities**
- **High-dimensional data**

DBSCAN: Determining Eps and MinPts

- Idea is that for points in a cluster, their k^{th} nearest neighbors are at roughly the same distance
- Noise points have the k^{th} nearest neighbor at farther distance
- So, plot sorted distance of every point to its k^{th} nearest neighbor; expect to see a sharp change at distance values; take this as Eps.



Parameters for (***):

Eps

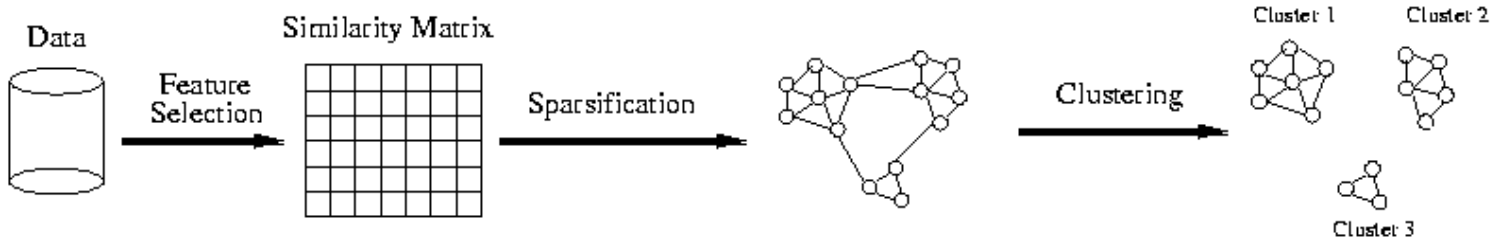
MinPts=k (here 4)

More Clustering Algorithms

- There are much more clustering algorithms
  - Prototype-based
    - Using fuzzy sets
    - Using statistical models
      - Data set seen as a set of of observations from a mixture of different probability distributions
    - Using self-organizing maps (neural networks)
  - Density-based
    - Grid-based
  - Graph-based
    - E.g. **CHAMELEON**
  - Scalable
    - E.g. CURE (clustering using (multiple) cluster representatives, sampling, aggl.hier.clustering)

Graph-Based, Hierarchical 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
- Needs sparsification
  - Omit edges between too unsimilar/distant neighbours (criterion?)
  - Can drastically reduce the amount of data to be processed
  - Facilitates use of graph partitioning algorithms

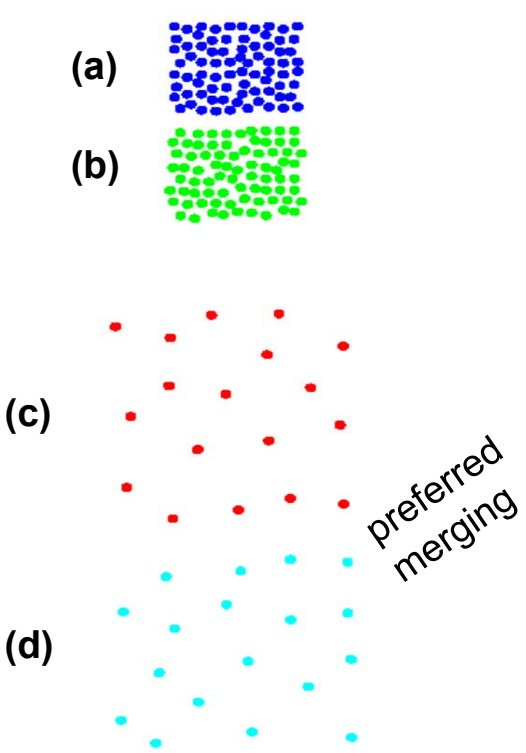


- In the simplest case, clusters are connected components in the graph.

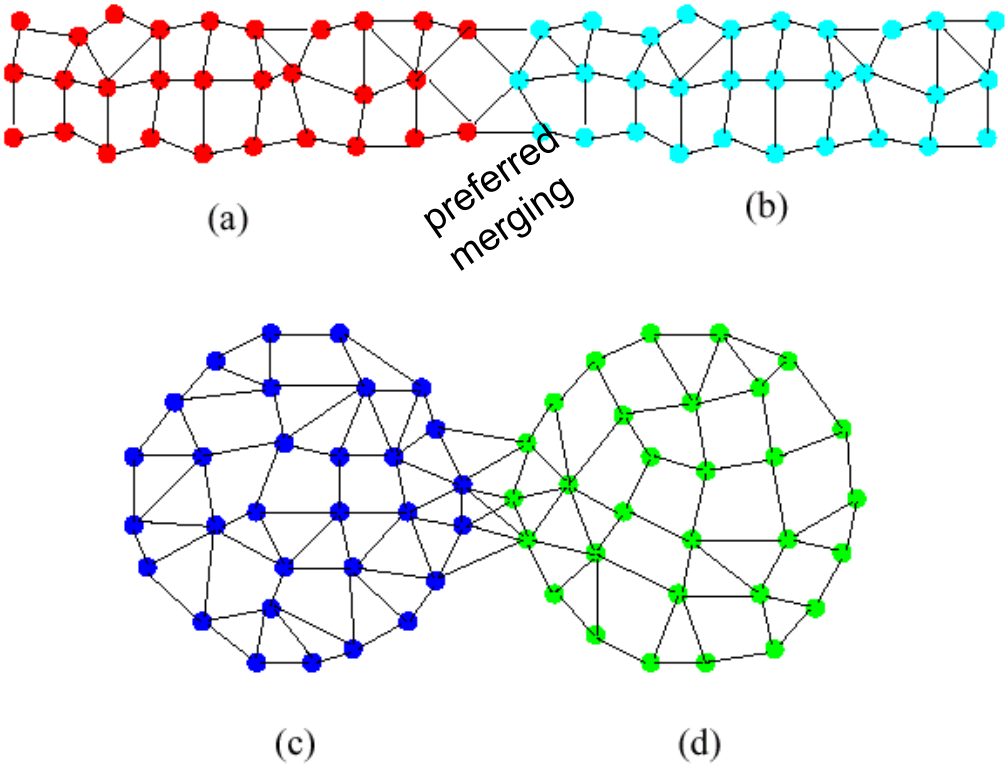
Limitations of Current Merging Schemes

- Existing merging schemes in hierarchical clustering algorithms are static in nature
  - MIN:
    - merge two clusters based on their *closeness* (or minimum distance)
    - susceptible to noise/outliers
  - GROUP-AVERAGE:
    - merge two clusters based on their average *connectivity*
    - may not work well with non-globular clusters (as MAX)
- **CHAMELEON** uses both schemes, dynamically.
- [CURE tries to to avoid the MIN and MAX disadvantages by using a choice of several representative points.]

Limitations of Current Merging Schemes



Closeness (MIN) schemes will merge (a) and (b)



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

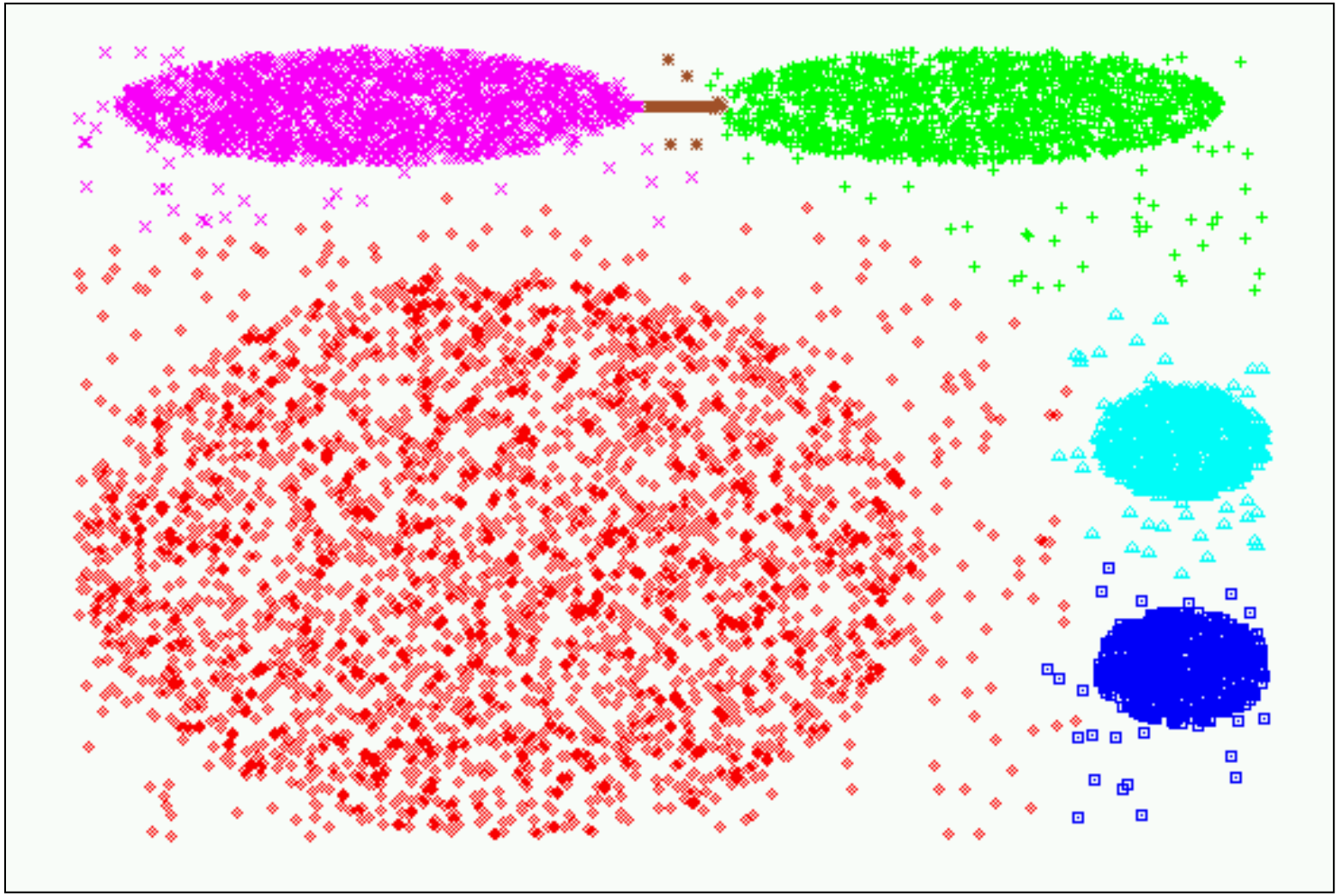
CHAMELEON : Clustering Using Dynamic Modeling

- Algorithm adapts to the characteristics of the data set to find the natural clusters
  - Two clusters are combined if the resulting cluster shares certain *properties* with the constituent clusters
  - The merging scheme should preserve *self-similarity*
- Uses a dynamic model to measure the similarity between clusters
  - Key properties to be maximized are: (compare “preferred matchings” above)
    - Relative Closeness: Absolute closeness of two clusters normalized by the internal closeness of the clusters
    - Relative Interconnectivity: Absolute interconnectivity of two clusters normalized by the internal connectivity of the clusters

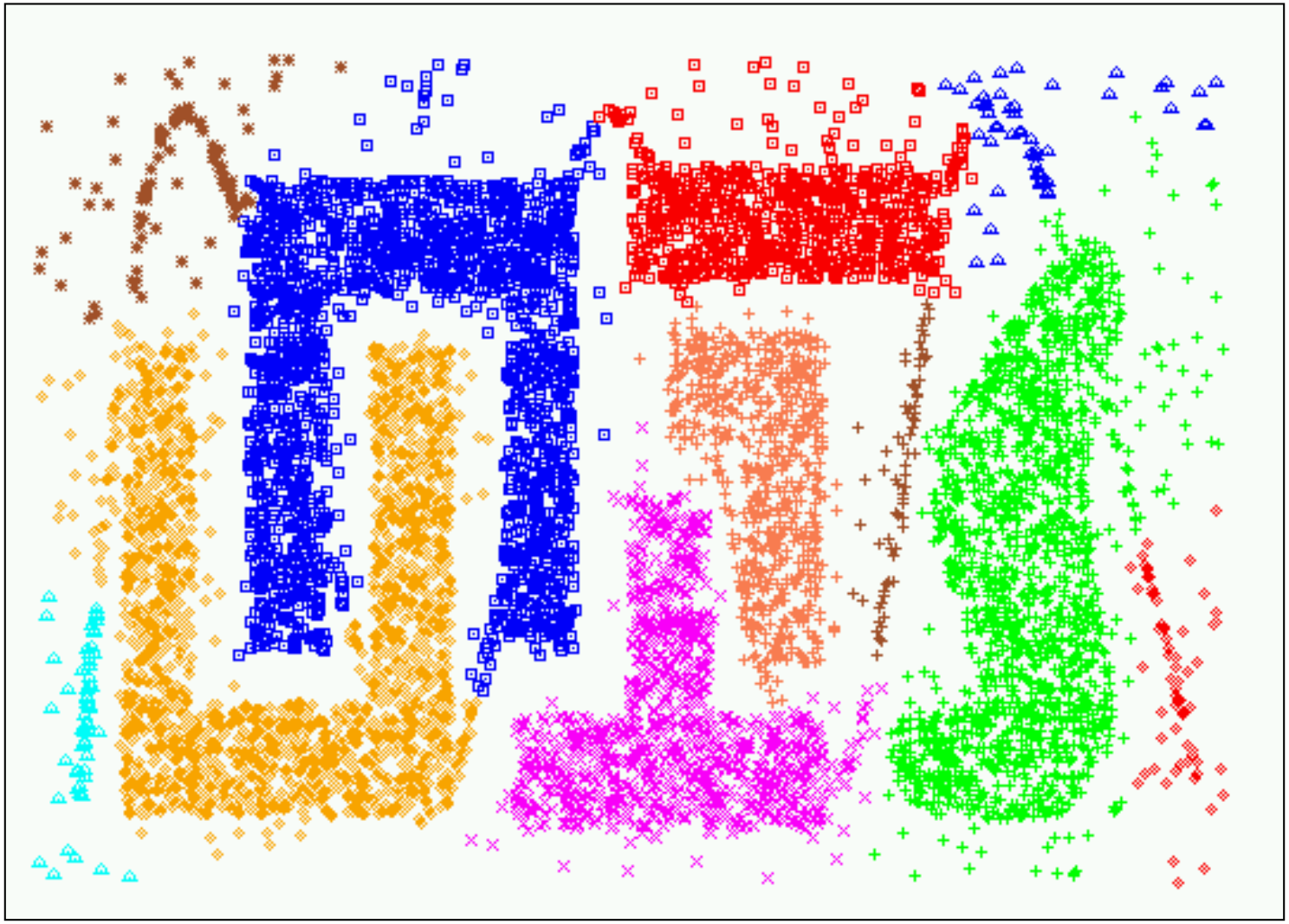
CHAMELEON : Steps

- Preprocessing Step: Represent the Data by a Graph
  - Given a set of points, construct the k-nearest-neighbor graph to capture the relationship between a point and its k nearest neighbors
    - Thus, the concept of neighborhood is captured dynamically (even if region is sparse)
- Phase 1: Use a multilevel graph partitioning algorithm
  - To find a large number of clusters (just below some min_size) of well-connected vertices
    - Each cluster should contain mostly points from one “true” cluster, i.e., is a sub-cluster of a “real” cluster
- Phase 2: Use Hierarchical Agglomerative Clustering with above properties to merge sub-clusters

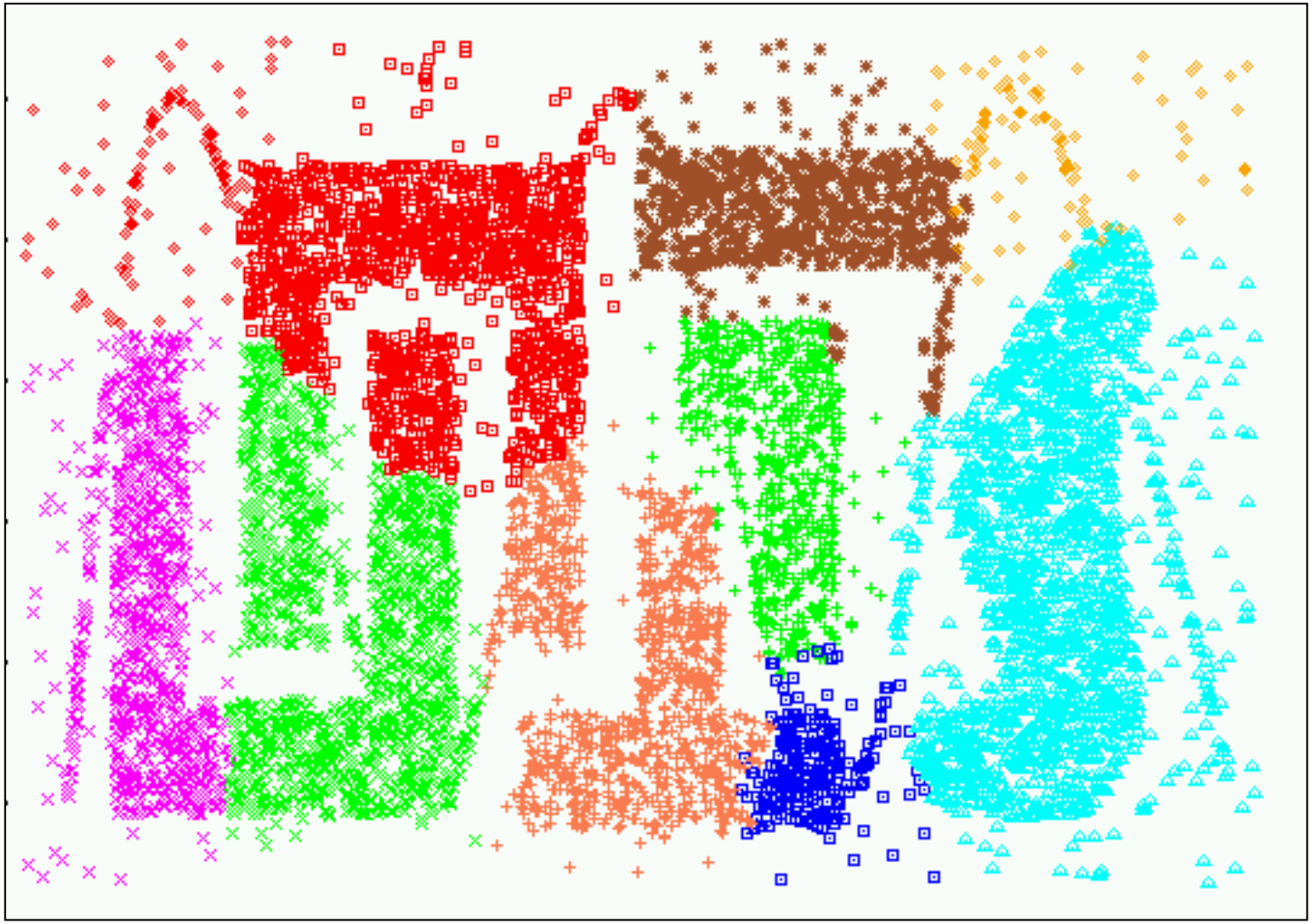
CHAMELEON : Experimental Results



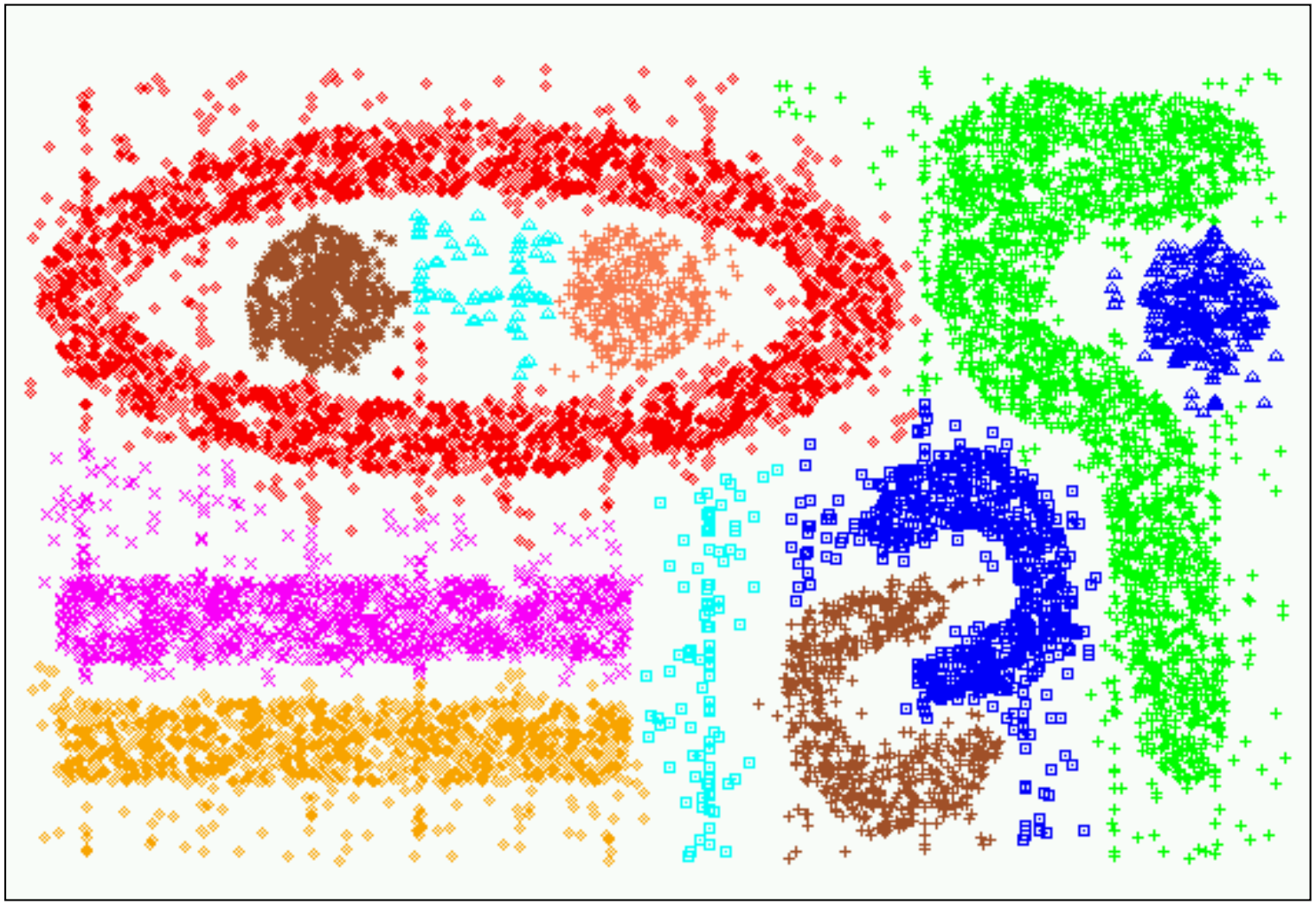
CHAMELEON : Experimental Results



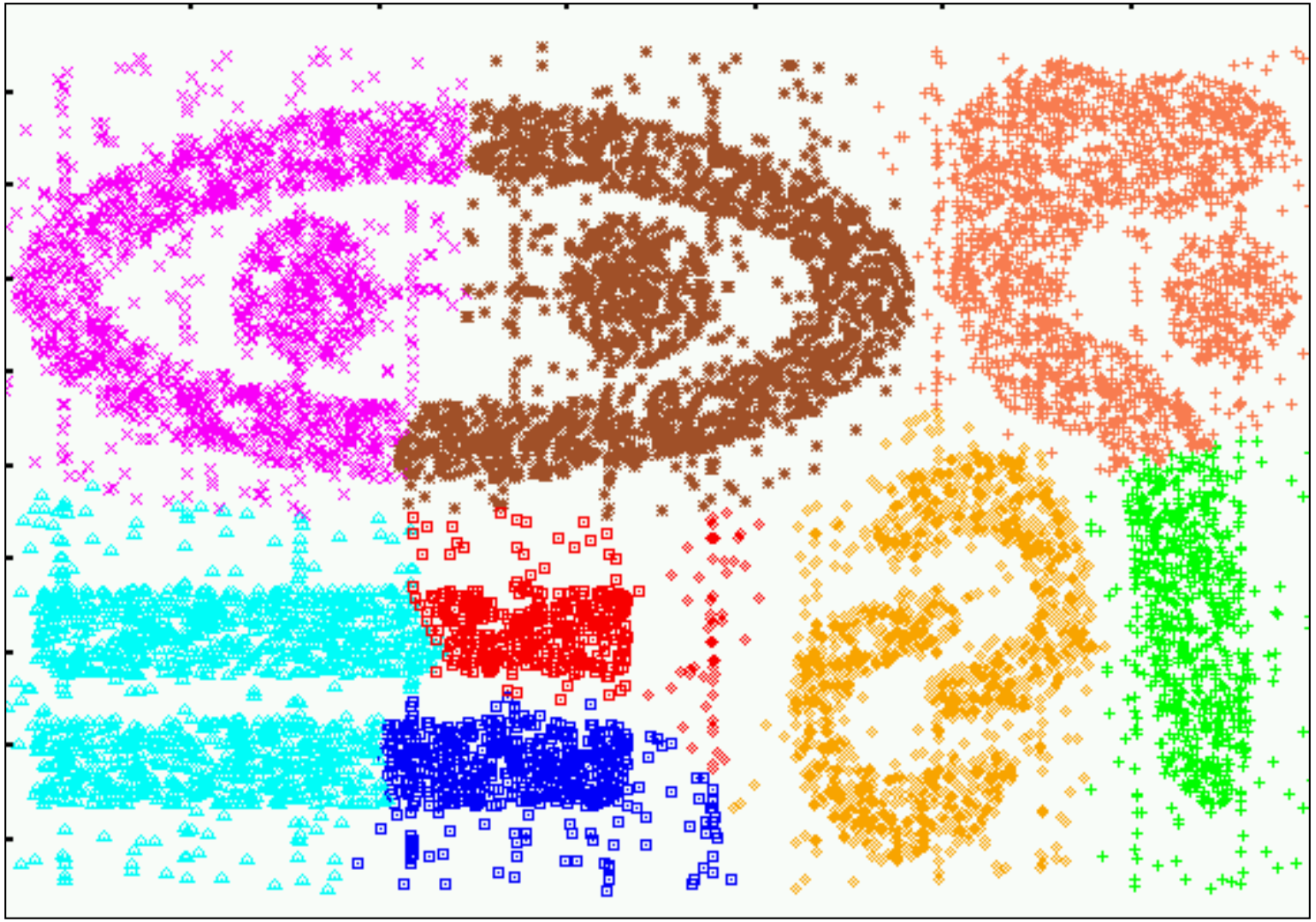
Compare CURE (10 clusters)



CHAMELEON : Experimental Results



Compare CURE (9 clusters)



Cluster Evaluation

- For supervised classification we have a variety of measures to evaluate how good our model is
  - Accuracy, precision, recall
- … and validation procedures

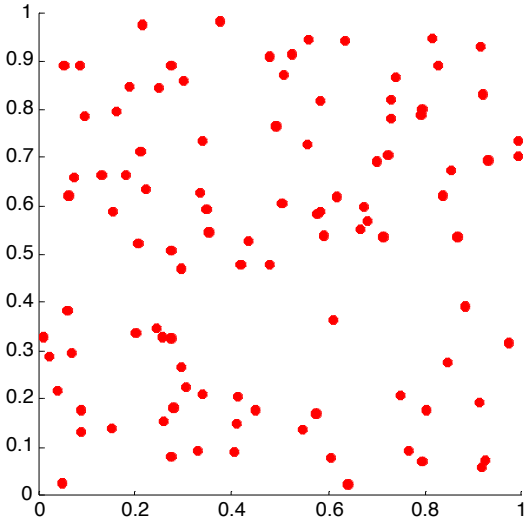
- For (unsupervised) cluster analysis, the analogous question is how to evaluate the “goodness” of the resulting clusters?

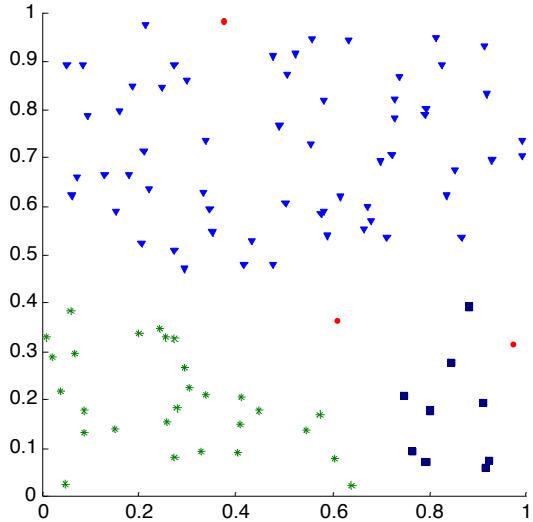
- But “clusters are in the eye of the beholder”!

- Then why do we want to evaluate them?
  - To avoid finding patterns in noise
  - To compare clustering algorithms
  - To compare two sets of clusters
  - To compare two clusters

Clusters found in Random Data

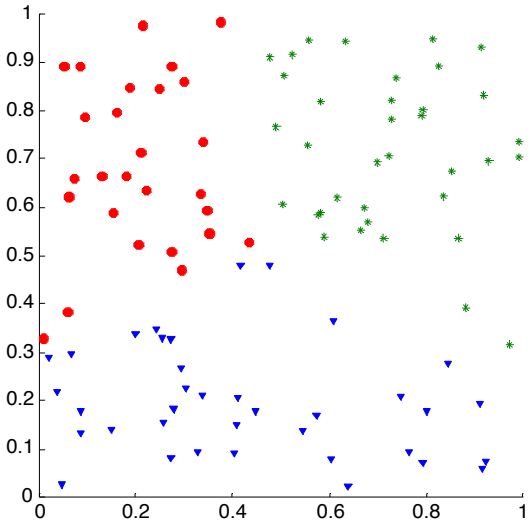
Random Points

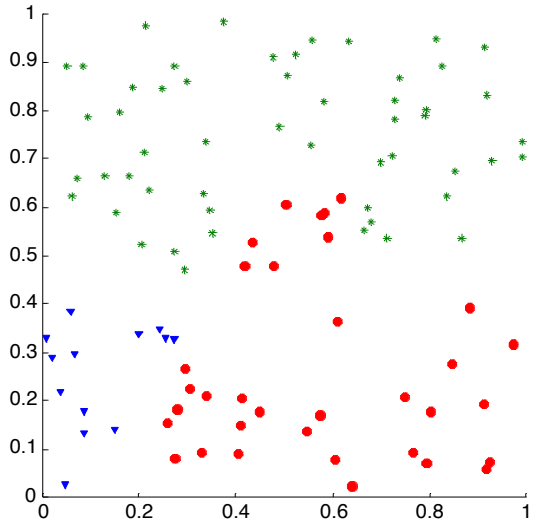




DBSCAN

K-means





Complete Link (MAX)

Different Aspects of "Cluster Validation"

1. Determine the clustering tendency of a set of data, i.e., distinguish whether non-random structure actually exists in the data.
2. Compare the results of a cluster analysis to externally known results, e.g., to externally given class labels.
3. Evaluate how well the results of a cluster analysis fit the data *without* reference to external information (internally).

   - Use only the data
4. Compare the results of two different sets of cluster analyses to determine which is better.
5. Determine 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.

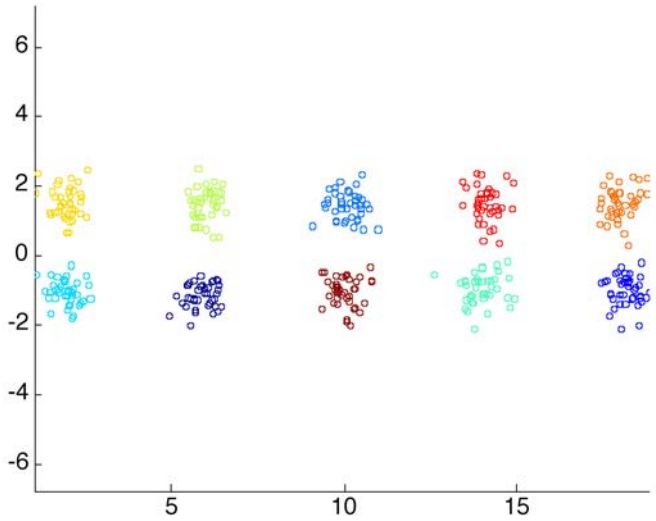
1, 3, and 5 are performed in an unsupervised manner; 2 supervised, 4 maybe both.

Measures for Cluster Validation

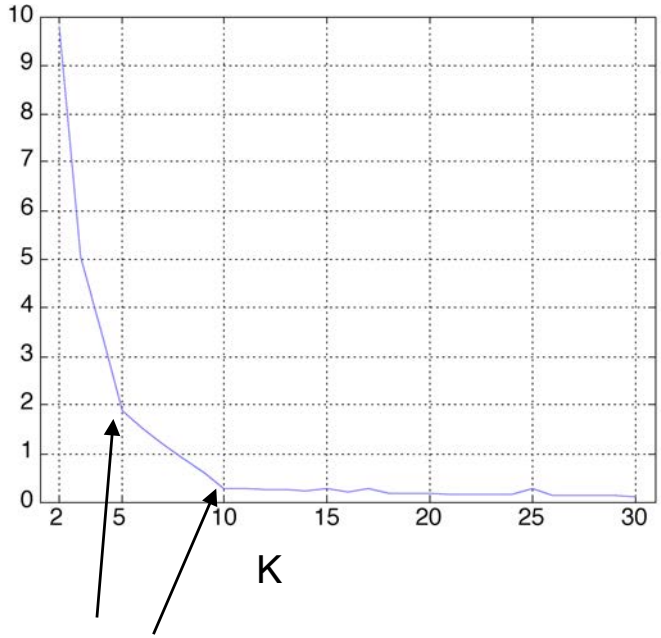
- Numerical measures that are applied to judge various aspects of cluster validity, are classified into the following three types.
  - External Criterion (or “Index”): Used to measure the extent to which cluster labels match externally supplied class labels.
    - E.g. Entropy
  - Internal Criterion: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

A Well-Known Internal Measure: SSE

- SSE is good for comparing two clusterings or two clusters. (total/local SSE).
- Can *often* be used to estimate the number of clusters

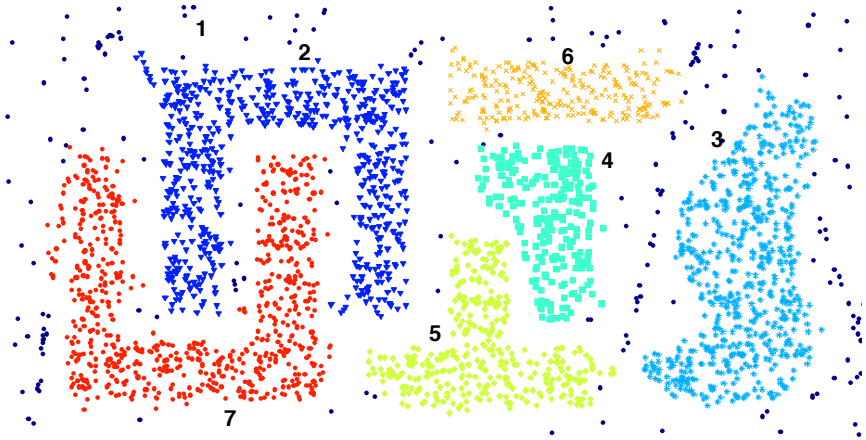


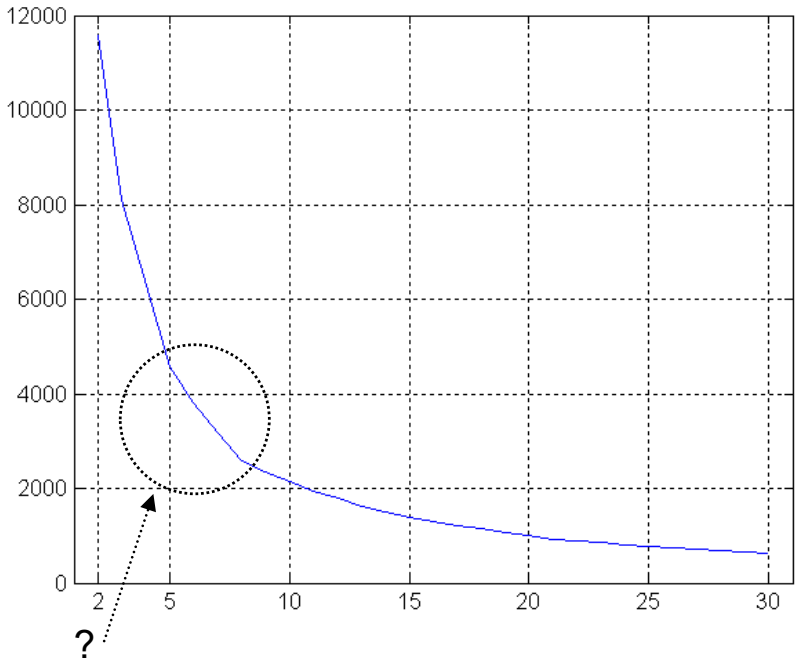
SSE



A Well-Known Internal Measure: SSE (contd.)

- SSE curve for a more complicated data set





SSE of clusters found using K-means

Basic Concepts of Many Internal Measures: Prototype-based View of Cohesion and Separation

- Cluster Cohesion: Measures how closely related are objects in a cluster
  - For example: SSE = WSS [**w**ithin cluster **s**um of **s**quared distances] (point - cluster centroid)

$$WSS = \sum_i \sum_{x \in C_i} (x - m_i)^2$$

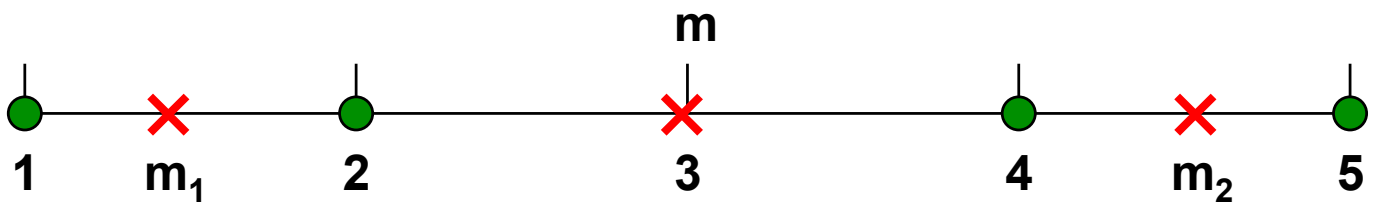
- Cluster Separation: Measure how distinct or well-separated a cluster is from other clusters
  - For example: BSS [**b**etween cluster **s**um of **s**quared distances] (overall mean m – cluster centroid m_i)

$$BSS = \sum_i |C_i| (m - m_i)^2$$

where $|C_i|$ is the size of cluster i

  - can be related to sum of cluster-pairwise squared distances $(m_i - m_j)^2$ between cluster centroids

BSS/WSS example



K=1 cluster:

$$WSS = (1-3)^2 + (2-3)^2 + (4-3)^2 + (5-3)^2 = 10$$

$$BSS = 4 \times (3-3)^2 = 0$$

$$Total = 10 + 0 = 10$$

K=2 clusters:

$$WSS = (1-1.5)^2 + (2-1.5)^2 + (4-4.5)^2 + (5-4.5)^2 = 1$$

$$BSS = 2 \times (3-1.5)^2 + 2 \times (4.5-3)^2 = 9$$

$$Total = 1 + 9 = 10$$

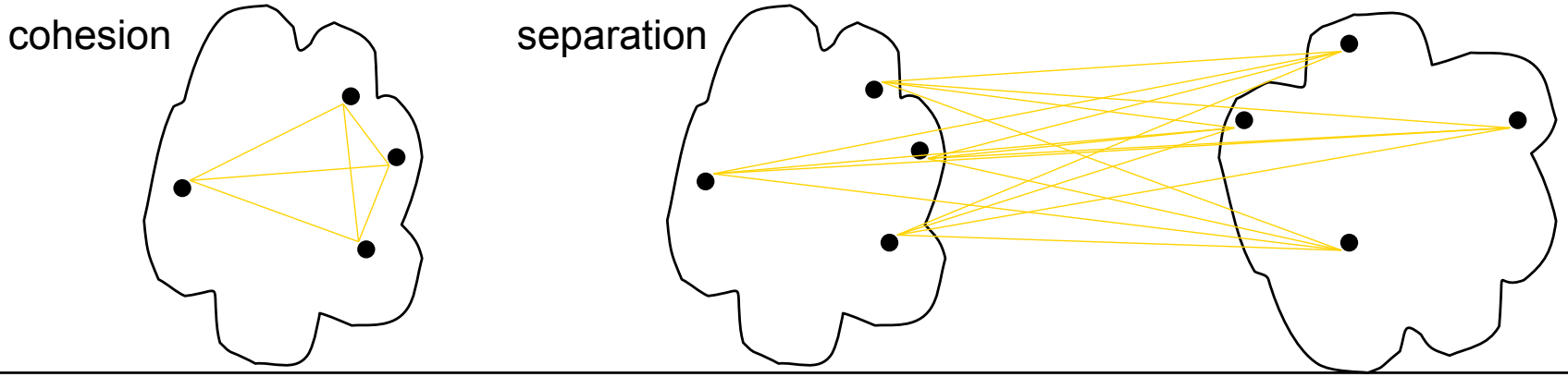
- *For sums of squared Euclidean distances:* BSS+WSS=constant
- Then, minimizing cohesion is equivalent to maximizing separation
- Or, use a weighted sum to stress cohesion or separation

Graph-based View of Cohesion and Separation

- A proximity graph based approach can also be used:
- Cluster Cohesion is the sum of the weights of all links within a cluster.

$$WSS' = \sum_i \sum_{x,y \in C_i} w(x,y)$$

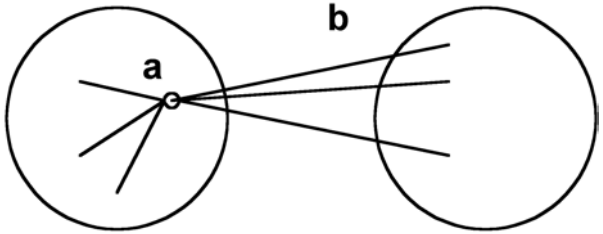
- For weights = squared Euclidean distances: $SSE = \sum_i \frac{1}{2|C_i|} \sum_{x,y \in C_i} w(x,y)$
  - i.e. per cluster: SSE=average weight of all links.
- Cluster Separation is the sum of the weights between nodes in the cluster and nodes outside the cluster.



Another Popular Internal Measure: Silhouette Coefficient

- The silhouette coefficient combines ideas of both cohesion and separation, but for individual points, as well as for clusters and clusterings
- For an individual point p in cluster C_i:
  - calculate $\boldsymbol{a}_p$ = average distance of p to the points in C_i
  - calculate $\boldsymbol{b}_p$ = min{average distance of p to points in $C_j \mid j \neq i$ }
  - The **silhouette coefficient** for a point is then given by

$$s_p = (b_p - a_p)/\max(a_p, b_p)$$



  - Varies between -1 and 1.
  - Negative values undesirable
  - The closer to 1 the better.

- Then calculate the average Silhouette coeffcient [for all points] of a cluster or of a clustering

Another Popular Internal Measure: Silhouette Coefficient (contd.)

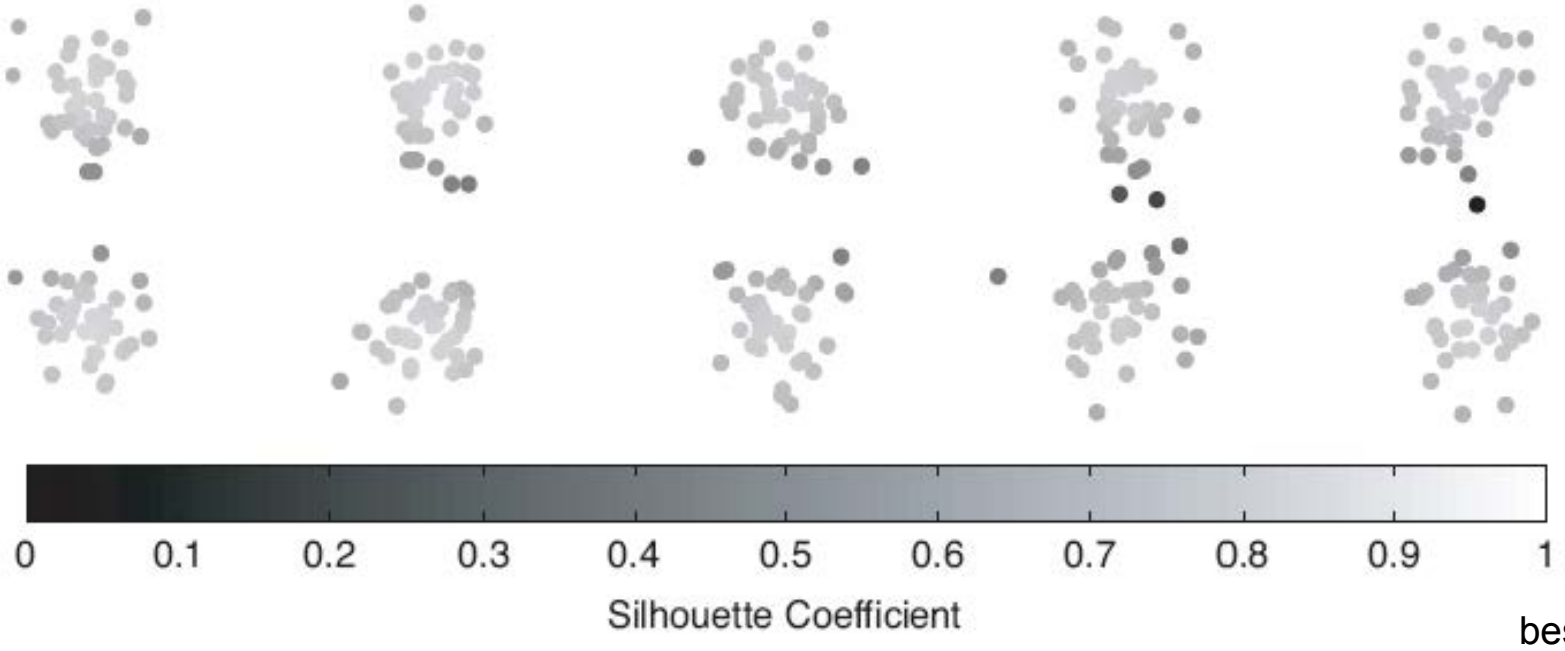


Figure 8.29. Silhouette coefficients for points in ten clusters.

Another Popular Internal Measure: Silhouette Coefficient (contd.)

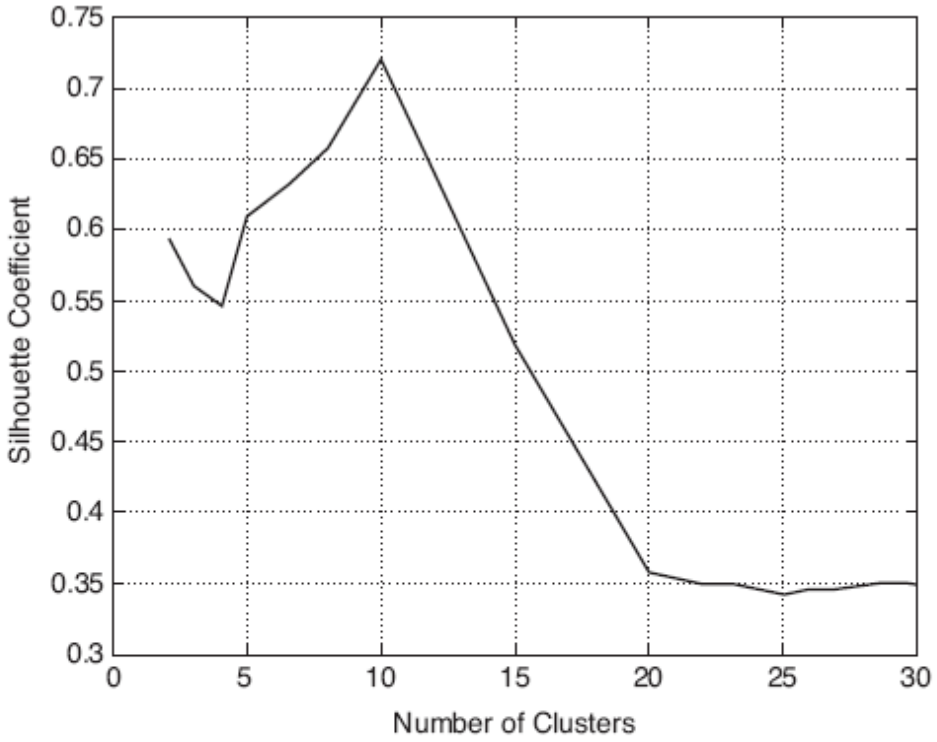


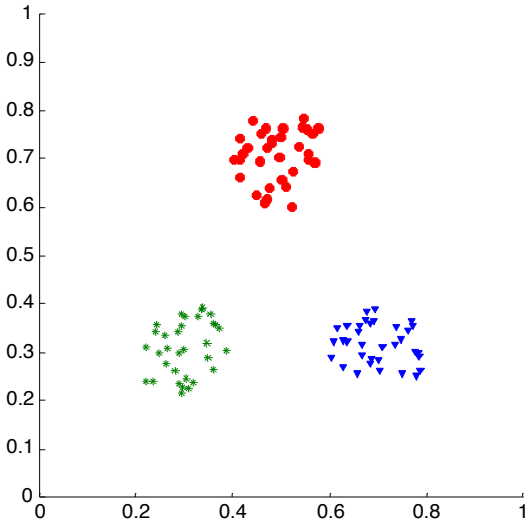
Figure 8.33. Average silhouette coefficient versus number of clusters for the data of Figure 8.29.

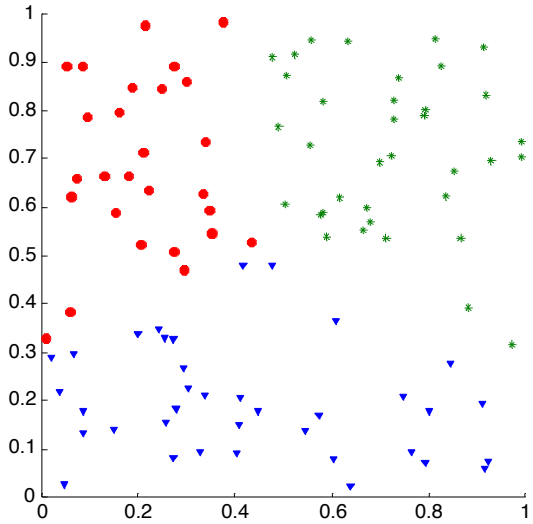
Internal Measuring Using Proximity Matrix: Correlation

- Two matrices
  - Proximity Matrix
  - "Cluster 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.

Internal Measuring Using Proximity Matrix: Correlation (contd.)

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



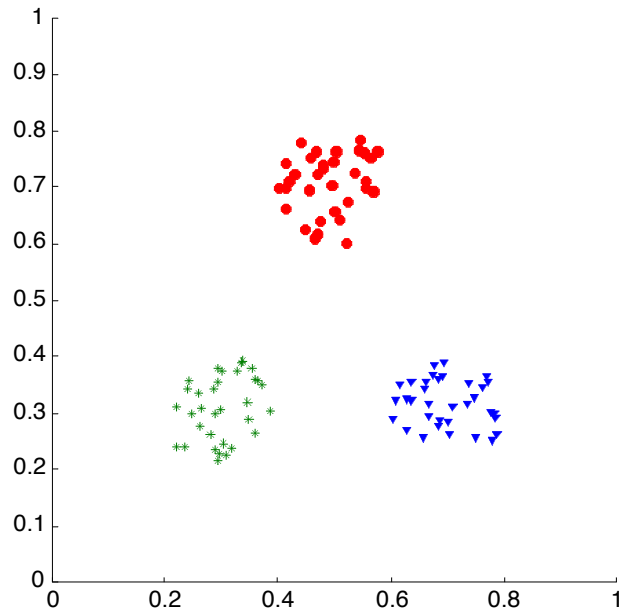


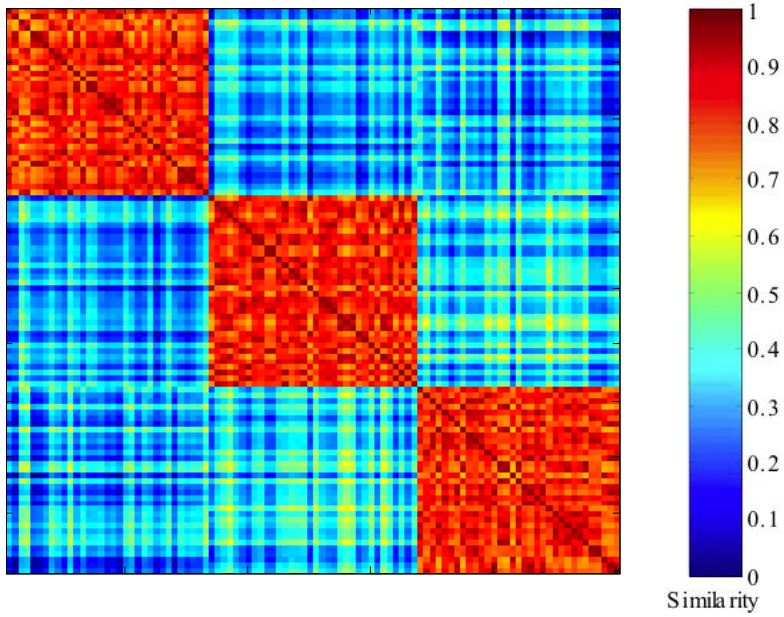
Corr = -0.9235 (better)

Corr = -0.5810 (worse)

Internal Measuring Using Proximity Matrix: Visual Inspection

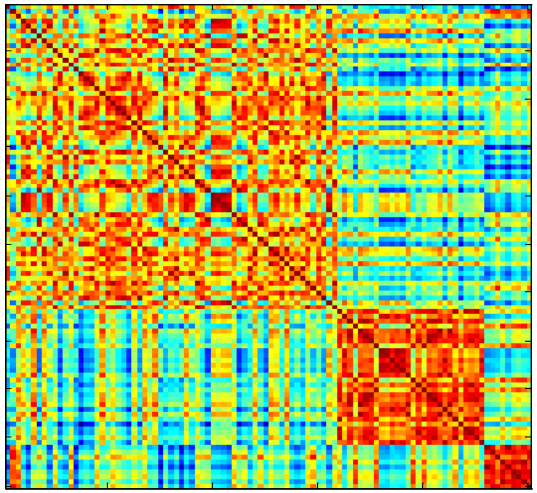
- Order the similarity matrix with respect to cluster labels and inspect visually.

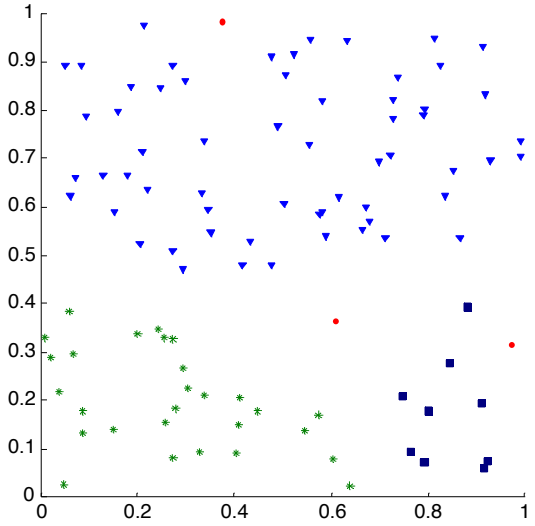




Internal Measuring Using Proximity Matrix: Visual Inspection (contd.)

- Clusters in random data are not so crisp

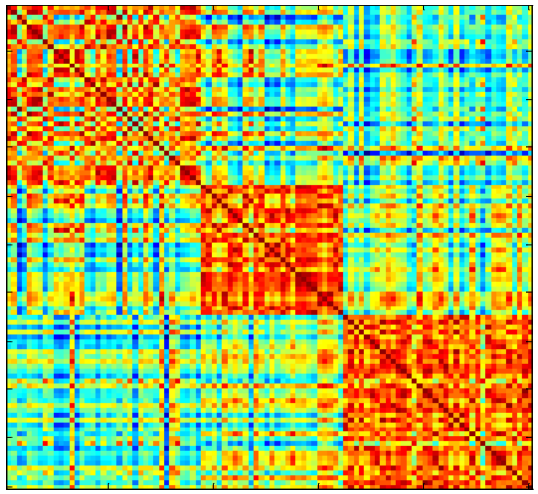


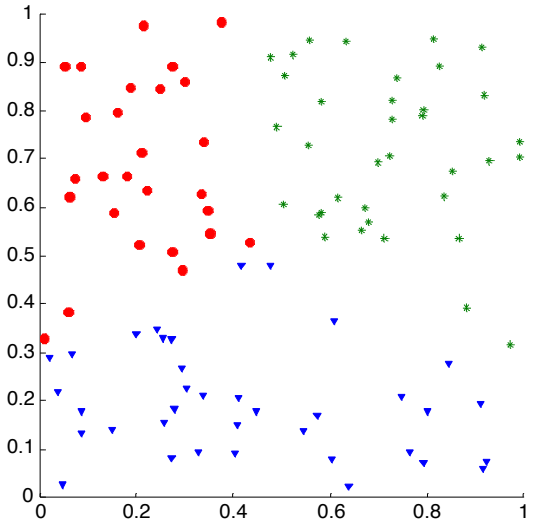


DBSCAN

Internal Measuring Using Proximity Matrix: Visual Inspection (contd.)

- Clusters in random data are not so crisp

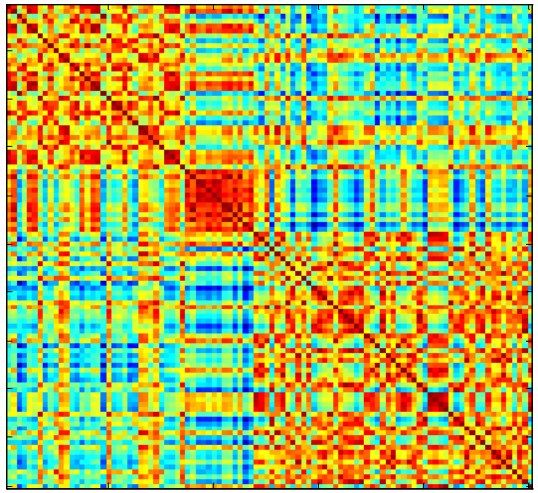


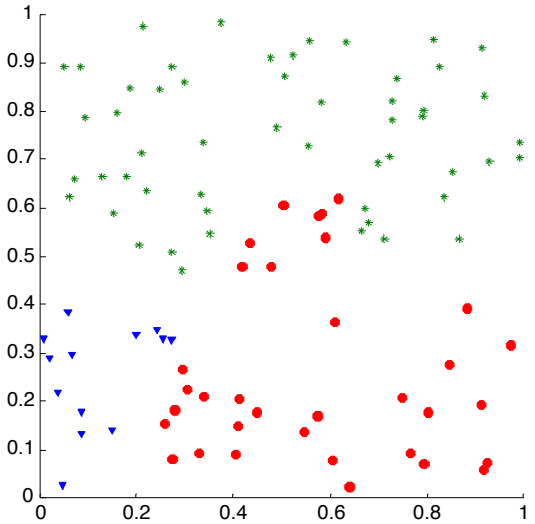


K-means

Internal Measuring Using Proximity Matrix: Visual Inspection (contd.)

- Clusters in random data are not so crisp

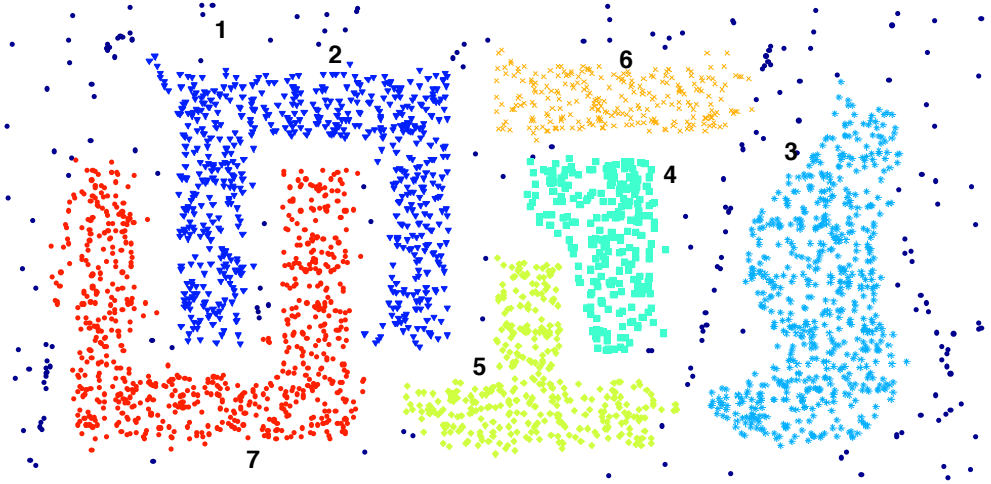


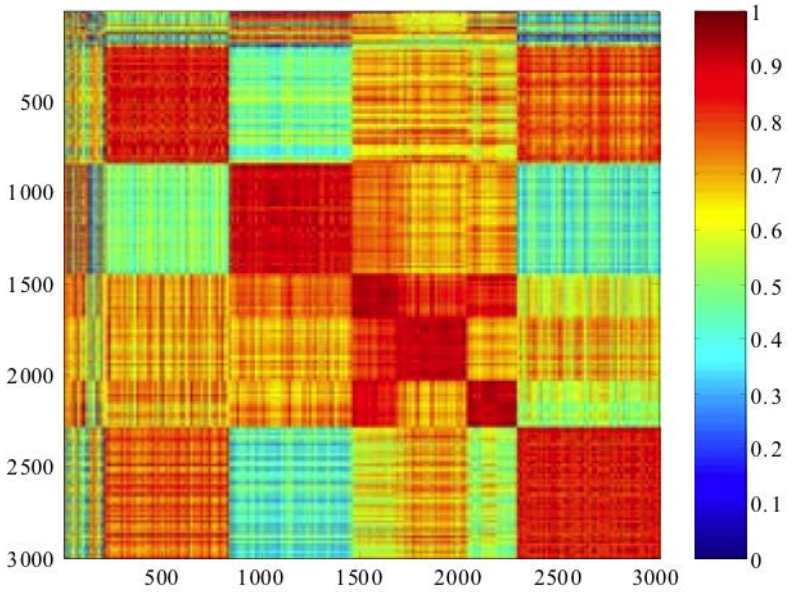


Complete Link (MAX Aggl. Hier. Clustering)

Die Buchautoren: „Just as people can find patterns in clouds, data mining algorithms can find clusters in random data.“

Internal Measuring Using Proximity Matrix: Visual Inspection (contd.)





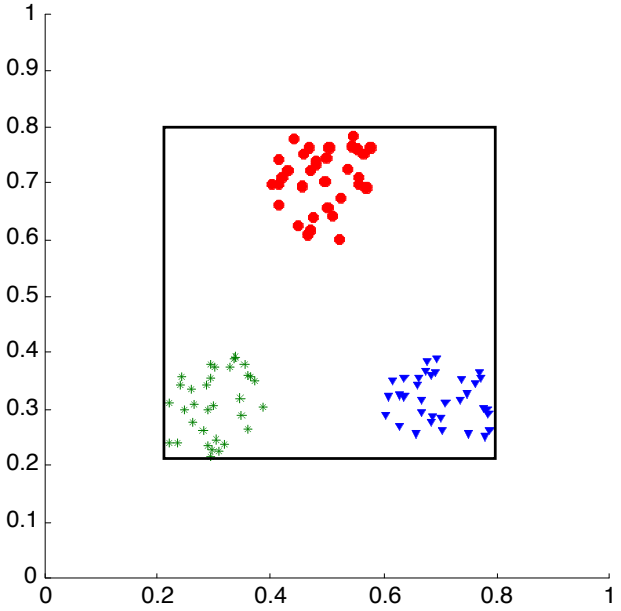
DBSCAN

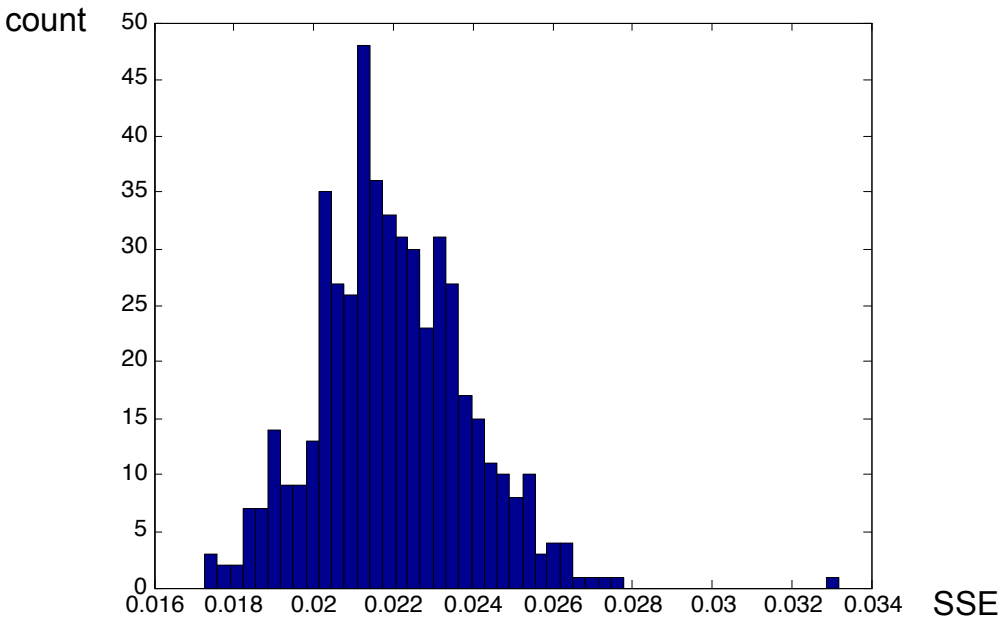
Framework for Cluster Validity

- Need a framework to interpret any measure.
  - For example, if our measure of evaluation has the value 10, is that good, fair, or poor?
- Statistics provide a framework for cluster validity
  - The more “atypical” a clustering result is, the more likely it represents valid structure in the data
  - Can compare the values of an index that result from random data or clusterings to those of a clustering result.
    - If the value of the index is unlikely, then the cluster results are valid
  - These approaches are more complicated and harder to understand.
- For comparing the results of two different sets of cluster analyses, a framework is less necessary.
  - However, there is the question of whether the difference between two index values is significant

Statistical Framework for SSE

- Example
  - Compare SSE of 0.005 for our nice cluster triple against cluster triples in random data
  - Histogram shows SSE of three clusters in 500 sets of random data points of size 100 distributed over the range 0.2 – 0.8 for x and y values

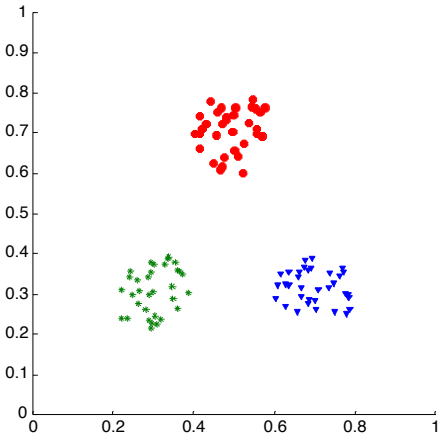


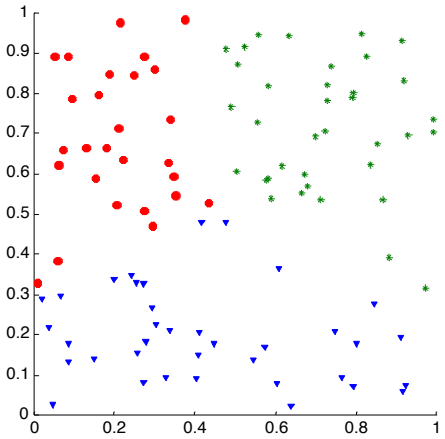


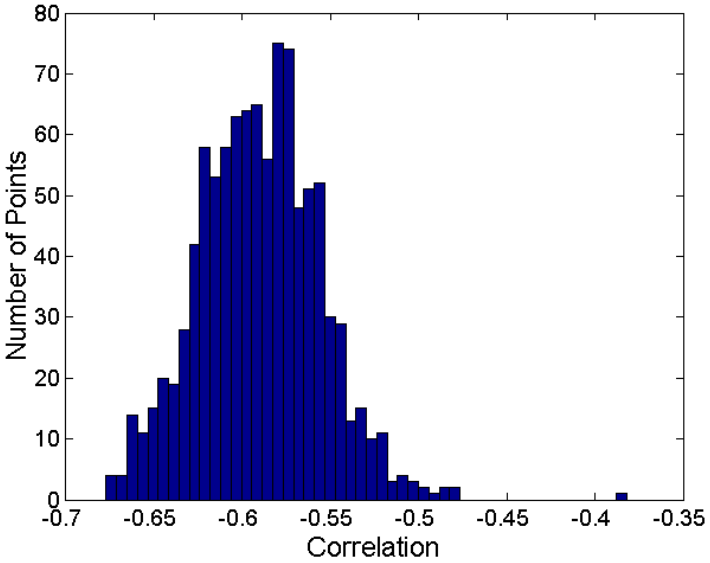
less than 1% probability that such a SSE value occurs by chance

Statistical Framework for Correlation

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







Corr = -0.9235

Corr = -0.5810

External Measures for Cluster Validation: E.g., 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

best

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 Validation

"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."

From *Algorithms for Clustering Data* by Jain and Dubes