CLUSTERING

What is Clustering

- **Cluster**: a collection of data objects
  - Similar to one another within the same cluster
  - Dissimilar to the objects in other clusters
- **Cluster analysis**
  - Grouping a set of data objects into clusters
- **Clustering** is unsupervised classification:
  - no predefined classes
- Typical applications
  - As a stand-alone tool to get insight into data distribution
  - As a preprocessing step for other algorithms

Examples of Clustering Applications

- **Marketing**: Help marketers discover distinct groups in their customer bases, and then use this knowledge to develop targeted marketing programs
- **Land use:** Identification of areas of similar land use in an earth observation database
- **Insurance**: Identifying groups of motor insurance policy holders with a high average claim cost
- **Urban planning**: Identifying groups of houses according to their house type, value, and geographical location
- **Seismology**: Observed earth quake epicenters should be clustered along continent faults

What Is a Good Clustering?

- A **good clustering method** will produce clusters with
  - **High intra-class similarity**: cohesive within clusters
  - **Low inter-class similarity**: distinctive between clusters
- Precise definition of clustering quality is difficult
  - Application-dependent
  - Ultimately subjective
- The **quality** of a clustering method depends on
  - The similarity measure used by the method
  - Its ability to discover some or all of the *hidden* patterns
- **Dissimilarity/Similarity metric**
  - Similarity is expressed in terms of a distance function: d(i, j)
  - The definitions of distance functions are usually rather different for nominal, binary, ordinal, and numeric attributes

What Is a Good Clustering?

How many clusters?

Six Clusters

Two Clusters

Four Clusters

Properties of Distance Metrics

- Properties of a distance metric d(i,j):
  - $d(i,j) \geq 0$
  - $d(i,i) = 0$
  - $d(i,j) = d(j,i)$
  - $d(i,j) \leq d(i,k) + d(k,j)$

Distance Metrics - Numeric Attributes

Minkowski Distance:

$$d(i,j) = \sqrt[q]{(|x_{i1} - x_{j1}|^q + |x_{i2} - x_{j2}|^q + \ldots + |x_{ip} - x_{jp}|^q)}$$

Manhattan Distance: q=1

$$d(i,j) = |x_{i1} - x_{j1}| + |x_{i2} - x_{j2}| + \ldots + |x_{ip} - x_{jp}|$$

Euclidean Distance: q=2

$$d(i,j) = \sqrt{(|x_{i1} - x_{j1}|^2 + |x_{i2} - x_{j2}|^2 + \ldots + |x_{ip} - x_{jp}|^2)}$$

Distance Metrics - Binary Attributes

Symmetric binary attributes: Both 0 and 1 are equally important.

Asymmetric binary attributes: 0 and 1 are not equally important.

A contingency table for binary attributes:

- For objects i and j

		Object *j*	
		1	0
Object *i*	1	*a*	*b*
	0	*c*	*d*

Distance measure for **symmetric binary attributes**:

$$d(i,j) = (b+c) / (a+b+c+d)$$

Distance measure for **asymmetric binary attributes**:

$$d(i,j) = (b+c) / (a+b+c)$$

Distance Metrics - Nominal & Ordinal Attributes

Nominal Attributes:

- Simple Matching:

  $d(i,j) = (p-m) / p$ where m is # of matching nominal attributes
  p is # of total nominal attributes

- Use a binary attribute for each value of a nominal attribute.

Ordinal Attributes:

  - Replace attribute values with numeric values 1,2,3,… and treat them as numerical values.

Distance Metrics

Combining Distance Metrics for Mixed Typed Attributes:

- Bring all attributes a common scale of interval [0.0,1.0]
- If our objects have p attributes of mixed type, the distance between two objects can be computed as follows:

$$\mathrm{d}(i,j) = \frac{\sum_{k=1}^{p} \delta_{ij}^{k}\, d_{ij}^{k}}{\sum_{k=1}^{p} \delta_{ij}^{k}}$$

- δ_{ij}^{k} is 0 if there is a missing attribute value for attribute k or k^{th} attribute is an asymmetrical binary attribute and object attribute values are 0.
- Otherwise δ_{ij}^{k} is 1
- d_{ij}^{k} can be computed as follows:
  - $d_{ij}^{k} = \frac{|x_{ik} - xjk|}{max_{k} - mink}$ if k^{th} attribute is a numerical attribute
  - d_{ij}^{k} is 0 or 1 if k^{th} attribute is a categorical attribute

Major Clustering Approaches

- **Partitioning:** Construct various partitions and then evaluate them by some criterion
  - A division data objects into non-overlapping subsets (clusters) such that each data object is in exactly one subset
- **Hierarchical:** Create a hierarchical decomposition of the set of objects using some criterion
  - A set of nested clusters organized as a hierarchical tree
- **Density-based**: Guided by connectivity and density functions
- **Graph-Based:** Adapt to the characteristics of the data set to find the natural clusters

Partitioning Algorithms

- **Partitioning method:** Construct a partition of a database ***D*** of ***n*** objects into a set of ***k*** clusters
- Given a *k*, find a partition of *k clusters* that optimizes the chosen partitioning criterion
  - Global optimal: exhaustively enumerate all partitions
  - Heuristic methods: ***k-means*** and ***k-medoids*** algorithms
  - ***k-means*** : Each cluster is represented by the center of the cluster
  - ***k-medoids*** or PAM (Partition around medoids): Each cluster is represented by one of the objects in the cluster

K-Means Clustering Algorithm

- **Input**
  - k: the number of clusters
  - D: a dataset containing *n* elements
- **Output**: a set of *k* clusters
- **Method**

  (1) arbitrarily choose *k* elements from *D* as the initial ***cluster mean values***

  (2) **repeat**

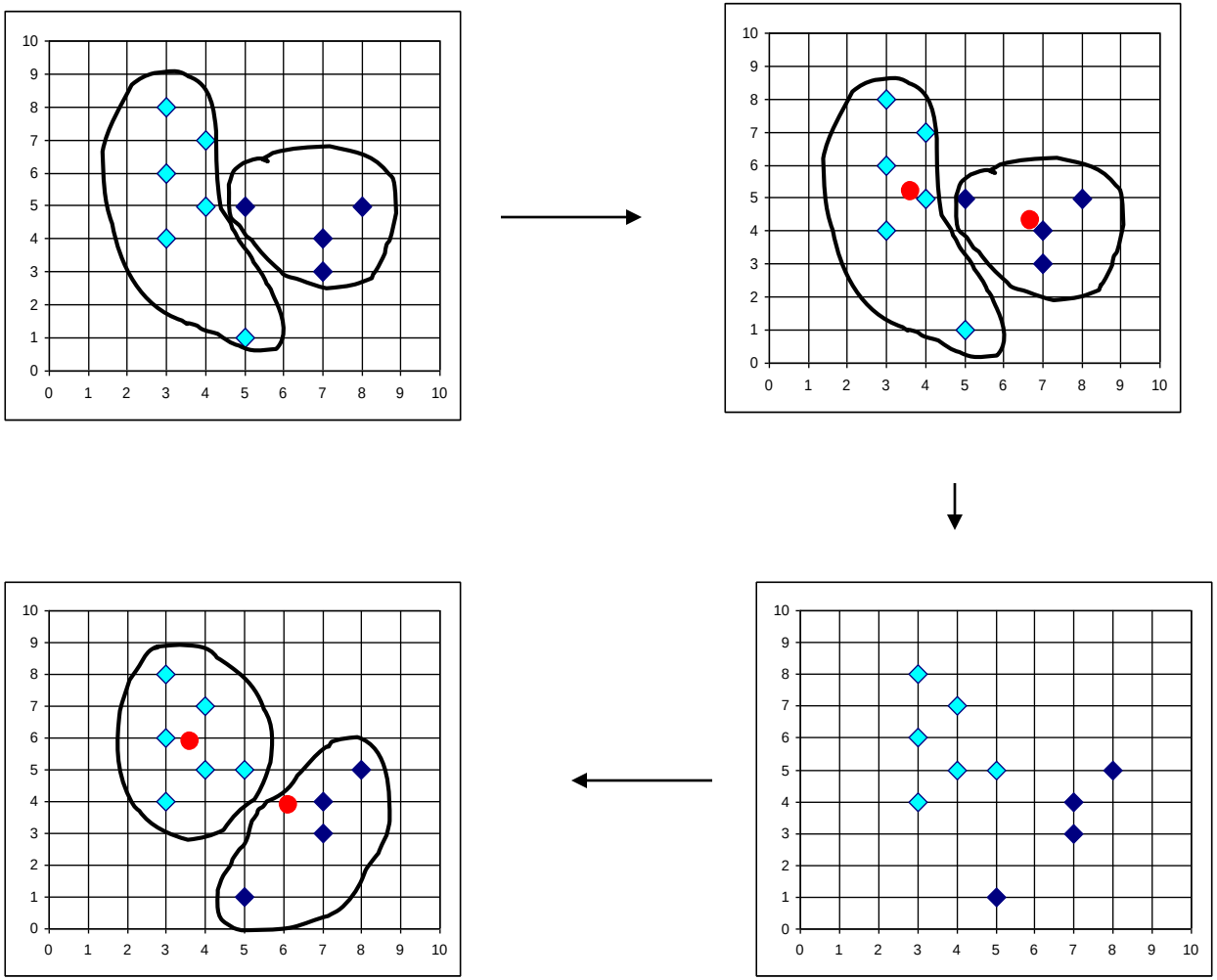
  (3) assign each element to the cluster whose cluster mean value is closest to the element

  (4) once all of the elements are assigned to clusters,

  calculate the *actual* new cluster means

  (5) **until** there is no change between the new and old cluster means

K-Means Clustering (Example)



K-Means Clustering (Example)

- Show two iterations of k-means clustering algorithm for following data points.
- Assume that k=2 and initial cluster means are (1,1) ; (2,3).
- Give cluster means and contents of clusters after each iteration.
- Use Manhattan distance metric.

Points
0,2
1,1
2,0
1,3
2,3
4,4
5,6

K-Means Clustering (Example)

Points	1,1	2,3	1,1	3,4
0,2	2	3	2	5
1,1	0	3	0	5
2,0	2	3	2	5
1,3	2	1	2	3
2,3	3	0	3	2
4,4	6	3	6	1
5,6	9	6	9	4

1st Iteration:	C1: 0,2 1,1 2,0	M1: 1,1
	C2: 1,3 2,3 4,4 5,6	M2:3,4
2nd Iteration:	C1: 0,2 1,1 2,0 1,3	M1: 1,6/4
	C2: 2,3 4,4 5,6	M2:11/3,13/3

Evaluating K-means Clusters

- Most common measure is Sum of Squared Error (SSE)
  - For each point, the error is the distance to the nearest cluster mean
  - To get SSE, we square these errors and sum them.

$$SSE = \sum_{i=1}^{K} \sum_{x \in C_i} dist^2(m_i, x)$$

  - x is a data point in cluster C_i and m_i is the *representative point* for cluster C_i and m_i corresponds to the center (mean) of the cluster

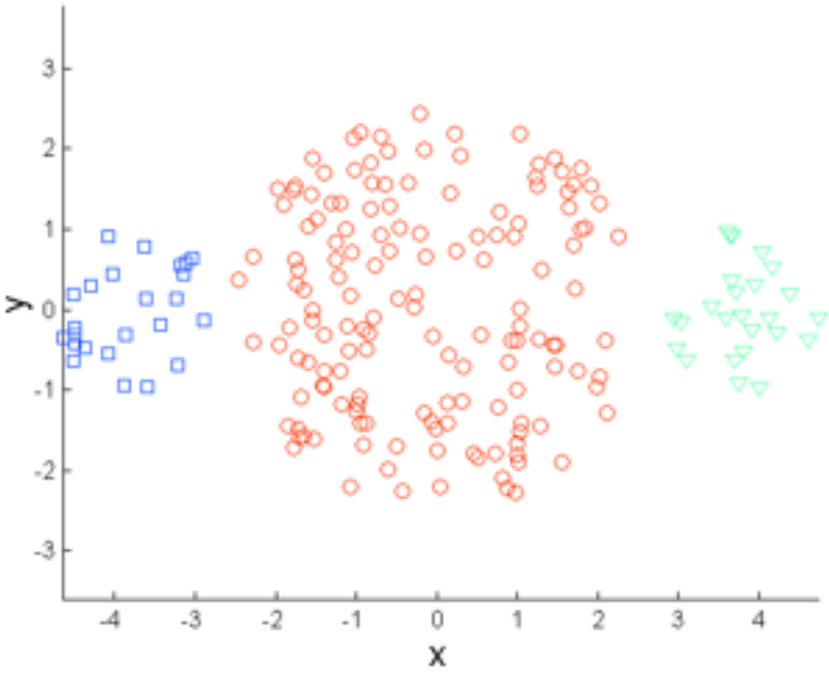
- The K-Means algorithm attempts to minimize the squared error for all elements in all clusters.

Comments on the K-Means Method

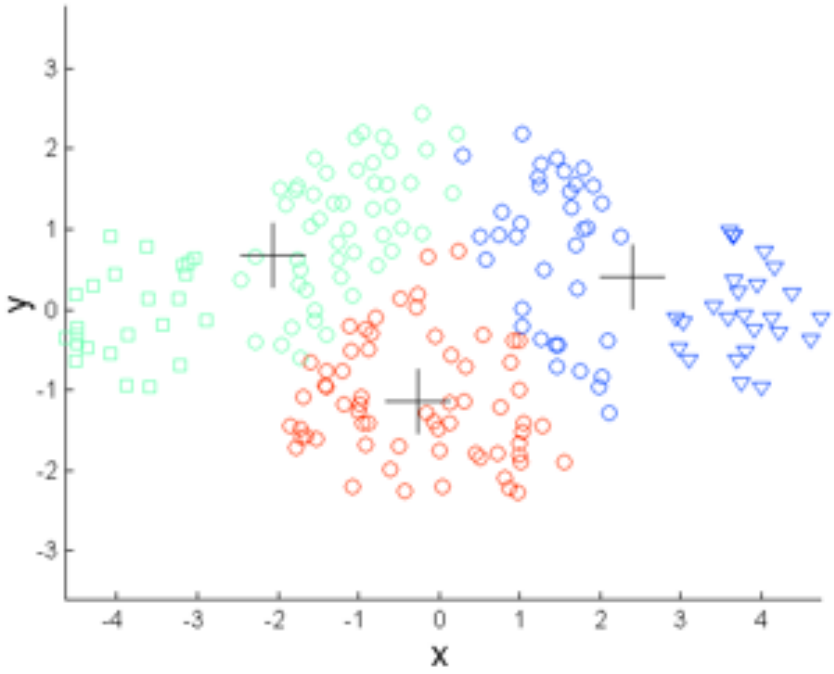
- **Strengths**
  - *Relatively efficient*: $O(tkn)$, where n is # objects, k is # clusters, and t is # iterations. Normally, $k, t << n$.
  - Often terminates at a *local optimum*. The *global optimum* may be found using techniques such as *simulated annealing* and *genetic algorithms*

- **Weaknesses**
  - Applicable only when *mean* is defined (what about categorical data?)
  - Need to specify *k*, the *number* of clusters, in advance
  - Trouble with noisy data and *outliers*
  - Not suitable to discover clusters with *non-convex shapes*
  - K-means has problems when clusters are of differing
    - Sizes
    - Densities
    - Non-globular shapes

Limitations of K-means: Differing Sizes

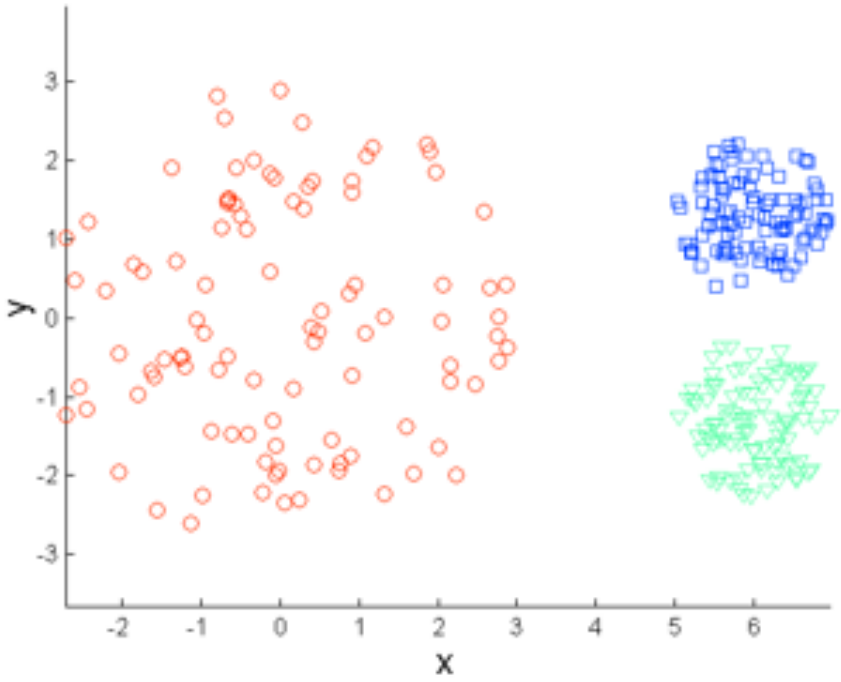


Original Points

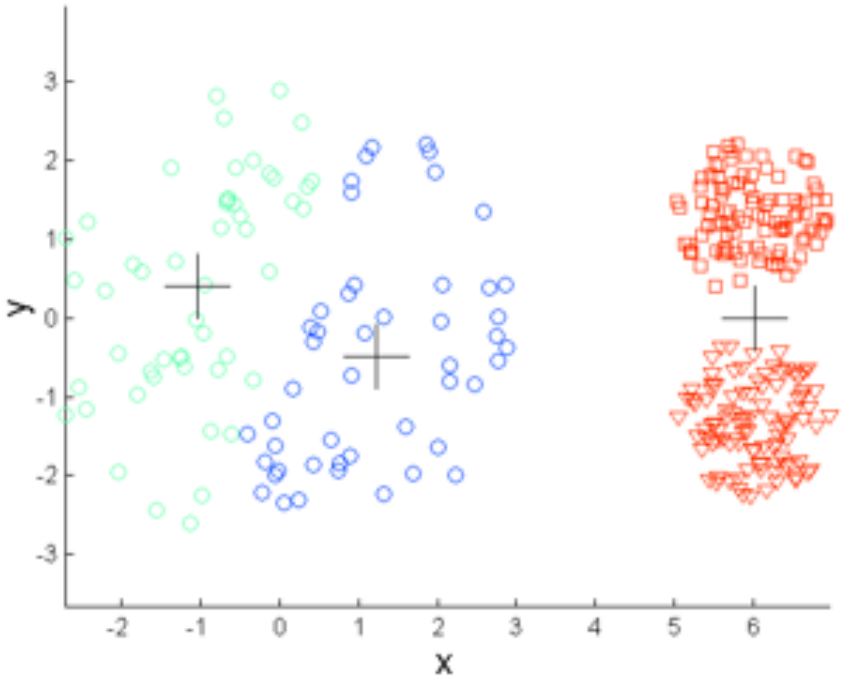


K-means (3 Clusters)

Limitations of K-means: Differing Density

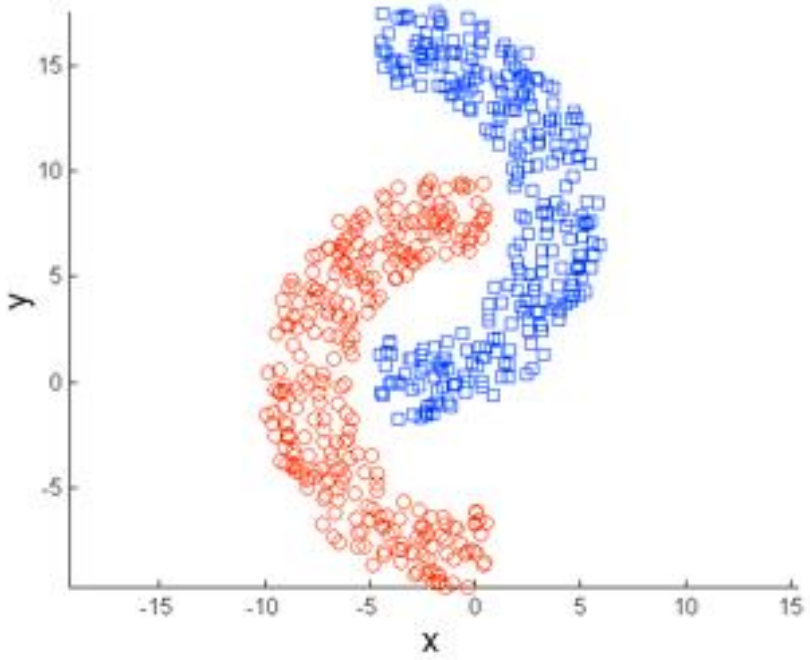


Original Points

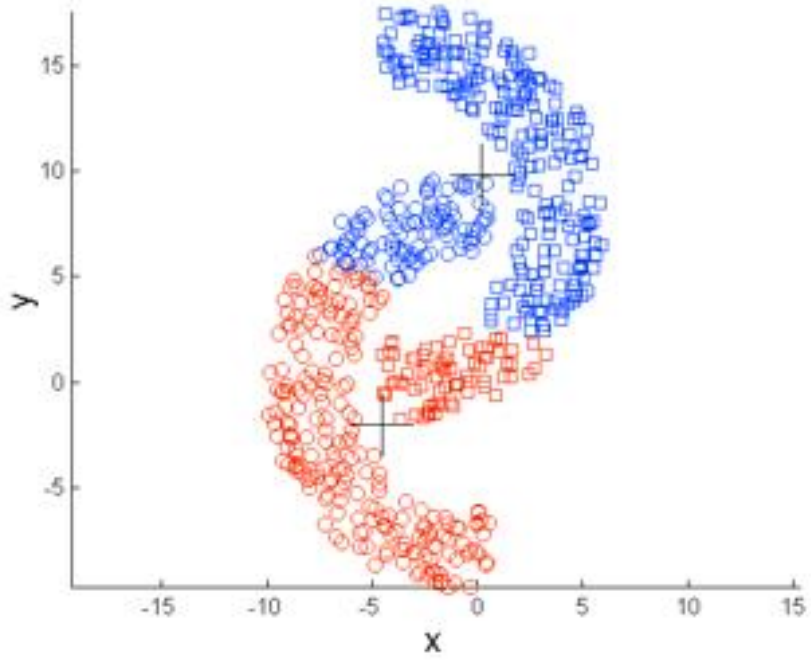


K-means (3 Clusters)

Limitations of K-means: Non-globular Shapes



Original Points



K-means (2 Clusters)

K-medoids **Clustering**

- ***K-means*** is appropriate when we can work with Euclidean distances
- Thus, *K*-means can work only with numerical, quantitative variable types
- Euclidean distances do not work well in at least two situations
  - Some variables are categorical
  - Outliers can be potential threats
- A general version of **K-means** algorithm called ***K-medoids*** can work with any distance measure
- ***K-medoids*** clustering is computationally more intensive

K-medoids Algorithm

- **Step 0**: Arbitrary choose K objects as initial medoids. Assign remaining objects to their nearest medoids to create initial K clusters
- **Step 1:** For a given cluster assignment C, find the observation in the cluster minimizing the total distance to other points in that cluster:

$$i_k^* = \arg\min_{\{i:C(i)=k\}} \sum_{C(j)=k} d(x_i, x_j).$$

- **Step 2**: Assign $m_k = x_{ik}$ where k is 1..K

- **Step 3**: Given a set of cluster centers $\{m_1, \ldots, m_K\}$, minimize the total error by assigning each observation to the closest current cluster center.

- Iterate steps 1 to 3

K-medoids Example

- Show two iterations of k-medoids clustering algorithm for following data points.
- Assume that k=2 and initial cluster means are (1,1) ; (2,3).
- Give cluster medoids after 1st iteration and contents of clusters after each iteration.
- Use Manhattan distance metric.

Points
0,2
1,1
2,0
1,3
2,3
5,5
6,6
7,7

K-medoids Example

Points	1,1	2,3	1,1	5,5
0,2	2	3	2	8
1,1	0	3	0	8
2,0	2	3	2	8
1,3	2	1	2	6
2,3	3	0	3	5
5,5	8	5	6	0
6,6	10	7	10	2
7,7	12	9	9	4

1st Iteration:

C1: 0,2 1,1 2,0

0,2: 0+2+4=6

1,1: 2+0+2=4

2,0: 4+2+0=6

M1: 1,1

C2: 1,3 2,3 5,5 6,6 7,7

1,3: 0+1+6+8+10=25

2,3: 1+0+5+7+9=22

5,5: 8+6+5+0+2+4=17

6,6 :8+7+2+0+2=19

7,7: 10+9+4+2+0=24

M2:5,5

2nd Iteration

C1: 0,2 1,1 2,0 1,3 2,3

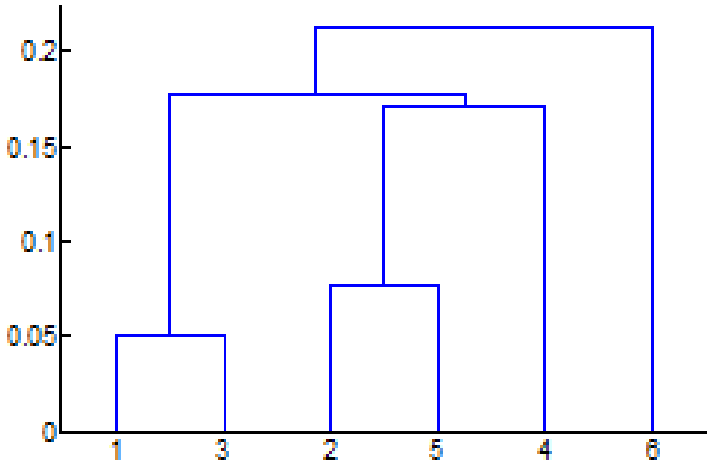
C2: 5,5 6,6 7,7

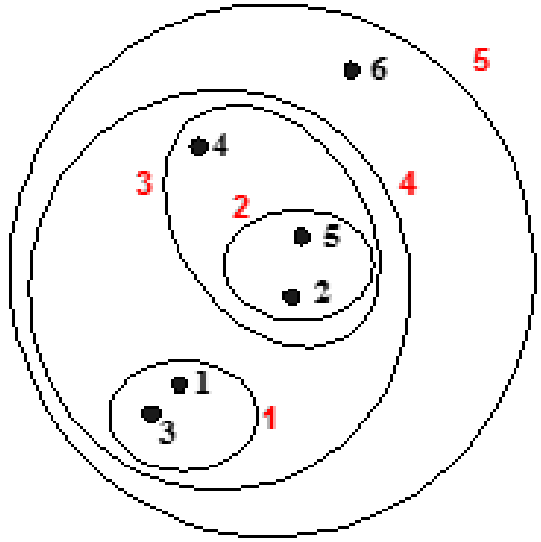
K-medoids Summary

- Generalized *K*-means
- Computationally much costlier that *K*-means
- Apply when dealing with categorical data
- Apply when data points are not available, but only pair-wise distances are available
- Converges to local minimum

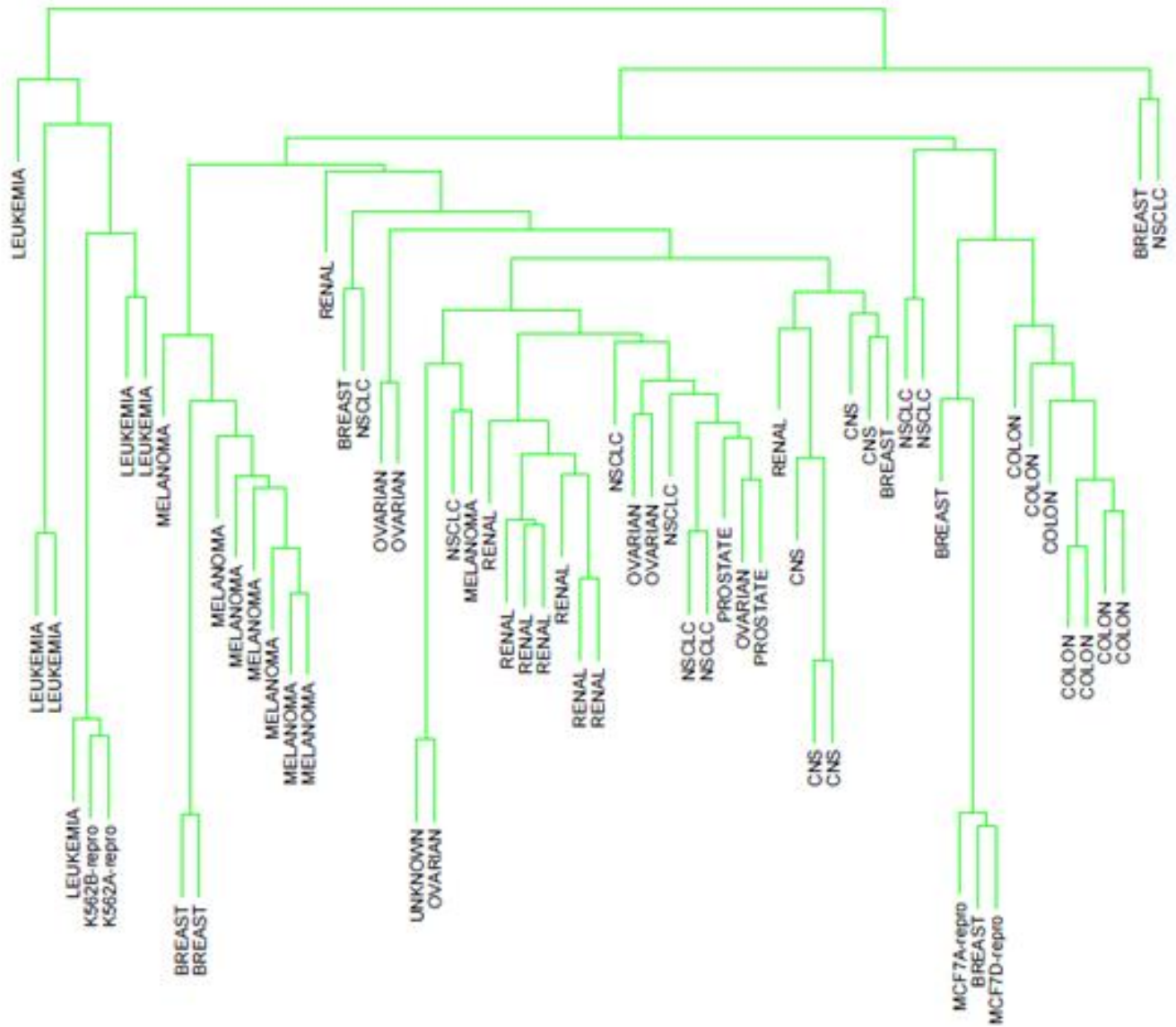
Hierarchical Clustering

- Produces a set of nested clusters organized as a hierarchical tree
- Can be visualized as a dendrogram
  - A tree like diagram that records the sequences of merges or splits
- Do not have to assume any particular number of clusters
  - Any desired number of clusters can be obtained by ‘cutting’ the dendogram at the proper level



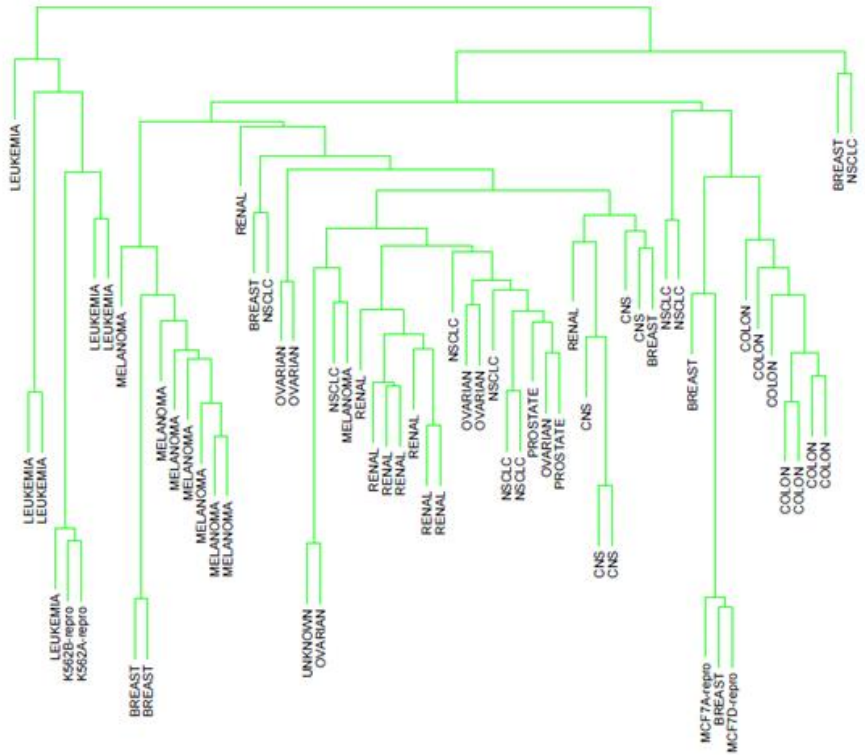


An Example Hierarchical Clustering



A *Dendrogram* Shows How the Clusters are Merged Hierarchically

- Decompose data objects into several levels of nested partitioning (tree of clusters), called a ***dendrogram***.
- A clustering of the data objects is obtained by cutting the ***dendrogram*** at the desired level. Then each connected component forms a cluster.



Hierarchical Clustering

- Two main types of hierarchical clustering

Agglomerative (Bottom-up):

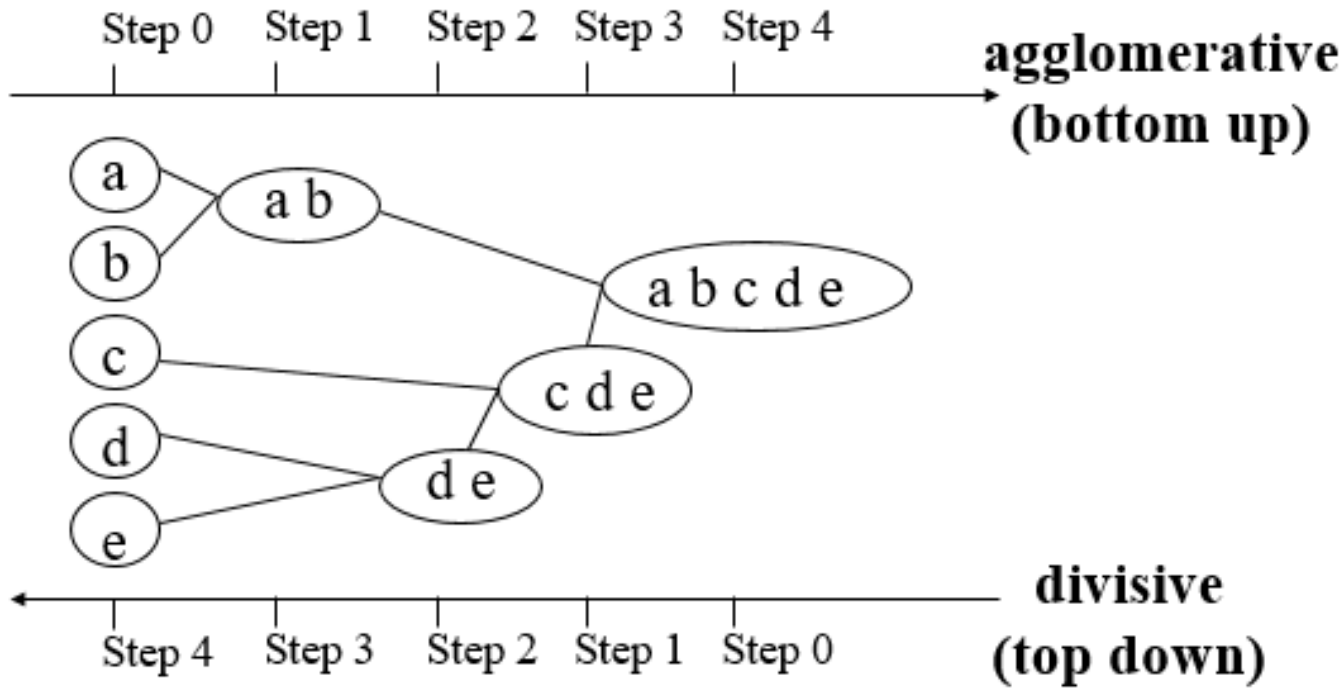
- Start with the points as individual clusters
- At each step, merge the closest pair of clusters until only one cluster (or k clusters) left

Divisive (Top-down):

- Start with one, all-inclusive cluster
- At each step, split a cluster until each cluster contains a point (or there are k clusters)

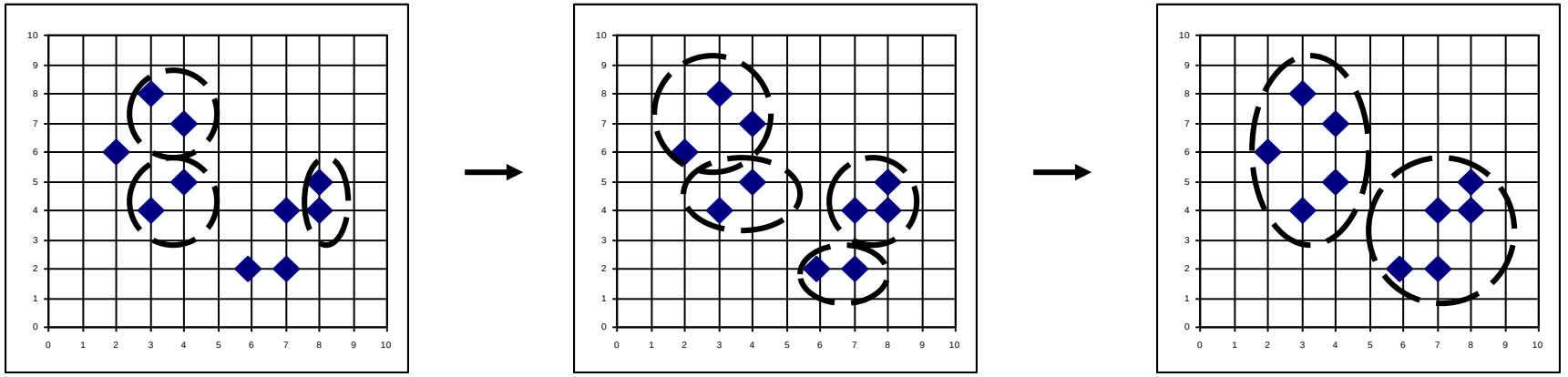
Hierarchical Clustering

- Use distance matrix as clustering criteria.
- This method does not require the number of clusters k as an input, but needs a termination condition



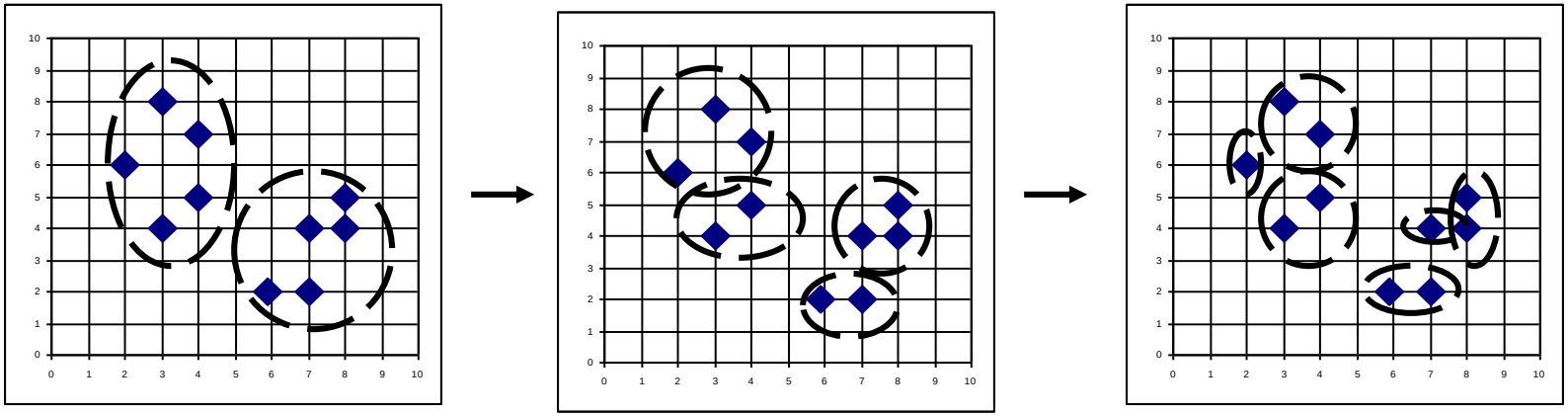
Agglomerative Nesting (Bottom Up)

- Produces tree of clusters (nodes)
- Initially: each object is a cluster (leaf)
- Recursively merges nodes that have the least dissimilarity
- Criteria: min distance, max distance, avg distance, center distance
- Eventually all nodes belong to the same cluster (root)



Divisive Analysis (Top Down)

- Inverse order of **Agglomerative**
- Start with root cluster containing all objects
- Recursively divide into subclusters
- Eventually each cluster contains a single object



Agglomerative (bottom-up) Clustering Algorithm

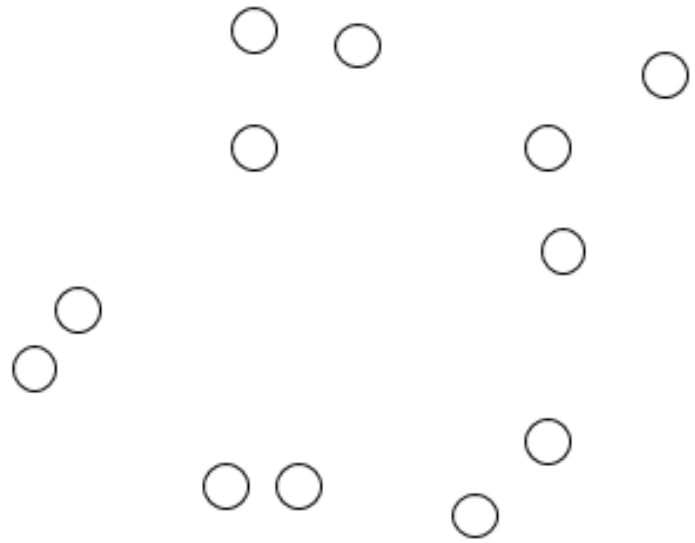
- More popular hierarchical clustering technique

- Basic algorithm is straightforward
  1. Compute the proximity matrix
  2. Let each data point be a cluster
  3. **Repeat**
  4. Merge the two closest clusters
  5. Update the proximity matrix
  6. **Until** only a single cluster remains

- ***Key operation is the computation of the proximity of two clusters***
  - Different approaches to defining the distance between clusters distinguish the different algorithms

Starting Situation

- Start with clusters of individual points and a proximity matrix

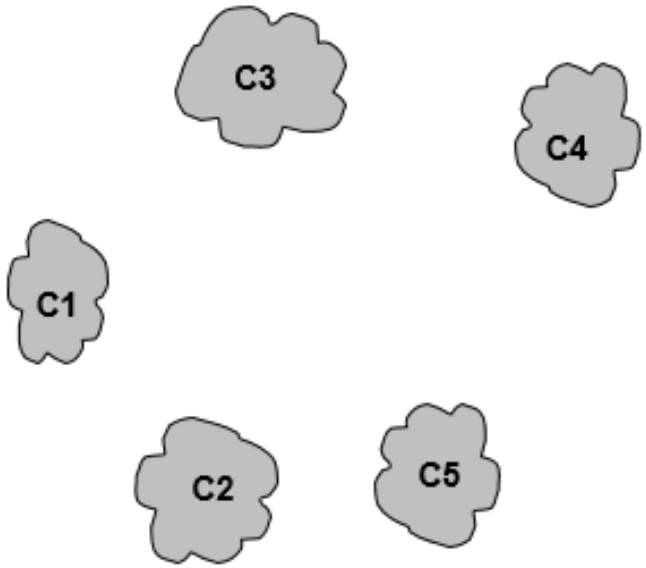


	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Proximity Matrix

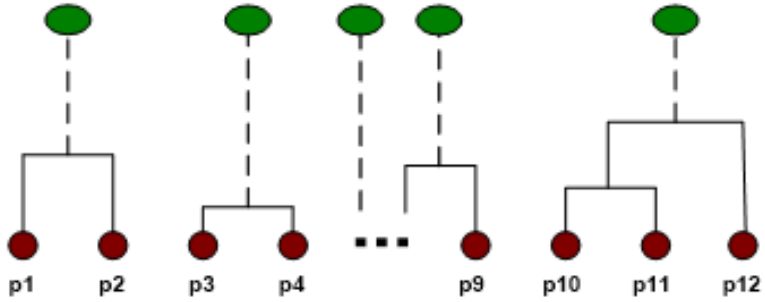
Intermediate Situation

- After some merging steps, we have some clusters.



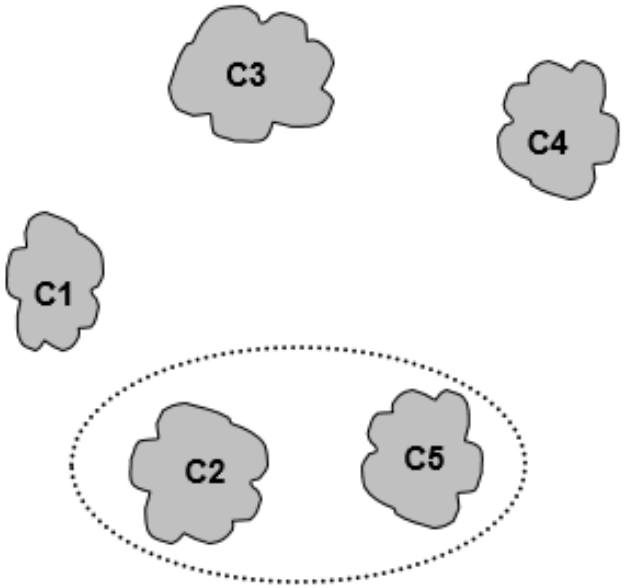
	C1	C2	C3	C4	C5
C1					
C2					
C3					
C4					
C5					

Proximity Matrix



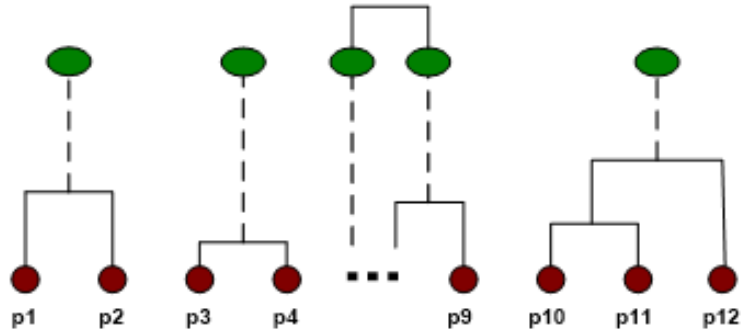
Intermediate Situation

- We want to merge the two closest clusters (C2 and C5) and update the proximity matrix.



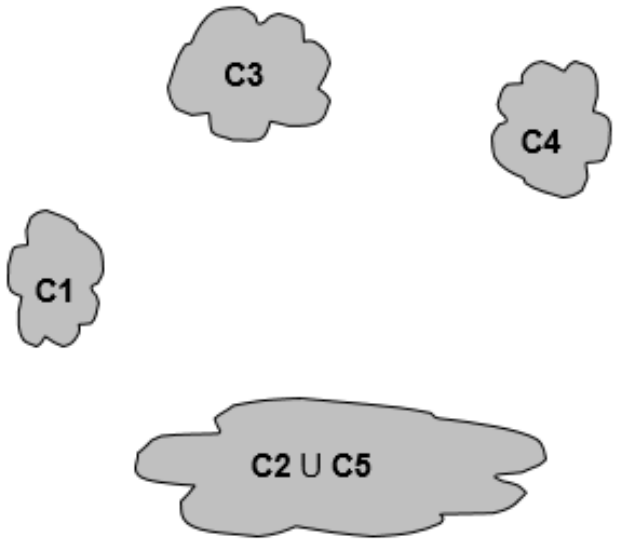
	C1	C2	C3	C4	C5
C1					
C2					
C3					
C4					
C5					

Proximity Matrix



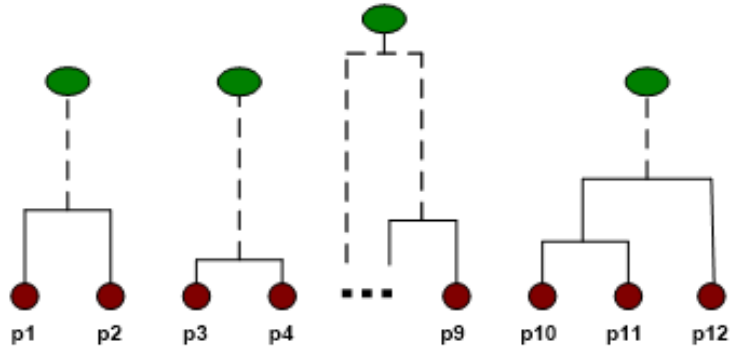
After Merging

- The question is “How do we update the proximity matrix?”

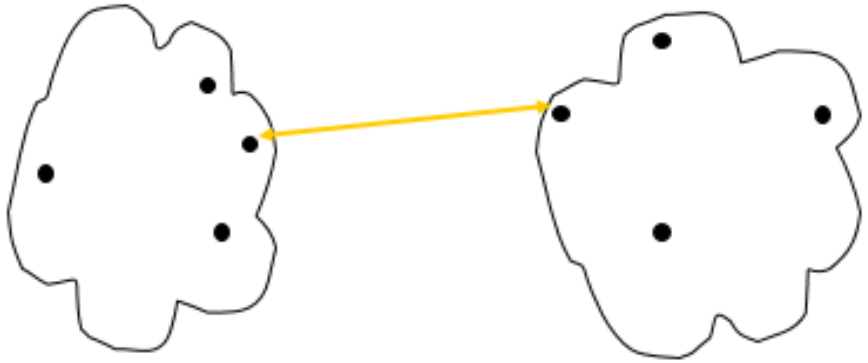


	C1	C2 U C5	C3	C4
C1		?		
C2 U C5	?	?	?	?
C3		?		
C4		?		

Proximity Matrix



How to Define Inter-Cluster Similarity

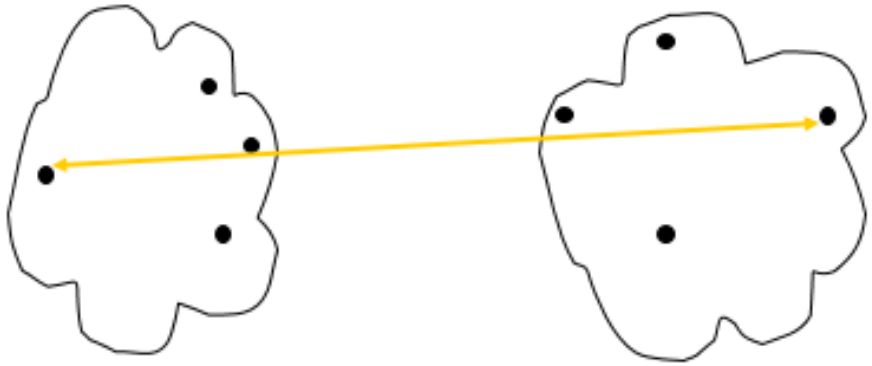


	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Proximity Matrix

- **MIN: Single linkage (nearest neighbor)**
- MAX
- Group Average
- Distance Between Centroids

How to Define Inter-Cluster Similarity

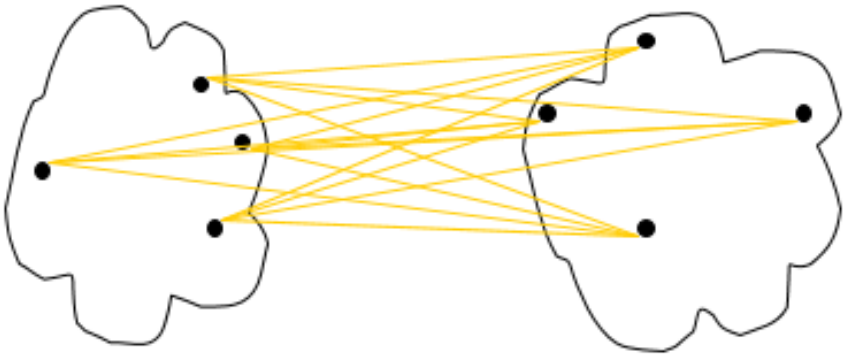


	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Proximity Matrix

- MIN
- **MAX: Complete linkage (furthest neighbor)**
- Group Average
- Distance Between Centroids

How to Define Inter-Cluster Similarity

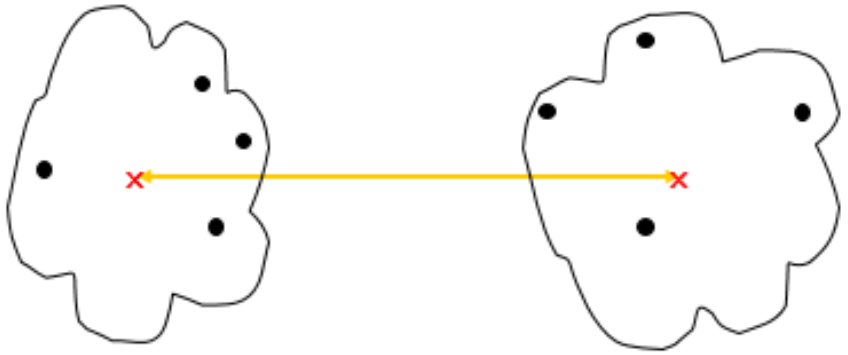


	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Proximity Matrix

- MIN
- MAX
- **Group Average**
- Distance Between Centroids

How to Define Inter-Cluster Similarity



	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Proximity Matrix

- MIN
- MAX
- Group Average
- **Distance Between Centroids**

Linkage Functions

- We know how to measure distance between two objects, but defining distance between an object and a cluster, or defining distance between two clusters is not obvious.
  - **Single linkage (nearest neighbor):** the distance between two clusters is determined by the distance of the two closest objects (nearest neighbors) in the different clusters.

$$d_{SL}(G,H) = \min_{\substack{i \in G \\ j \in H}} d_{ij}$$

  - **Complete linkage (furthest neighbor):** the distances between clusters are determined by the greatest distance between any two objects in the different clusters (i.e., by the "furthest neighbors").

$$d_{CL}(G,H) = \max_{\substack{i \in G \\ j \in H}} d_{ij}$$

  - **Group average linkage:** the distance between two clusters is calculated as the average distance between all pairs of objects in the two different clusters.

$$d_{GA}(G,H) = \frac{1}{N_G N_H} \sum_{i \in G} \sum_{j \in H} d_{ij}$$

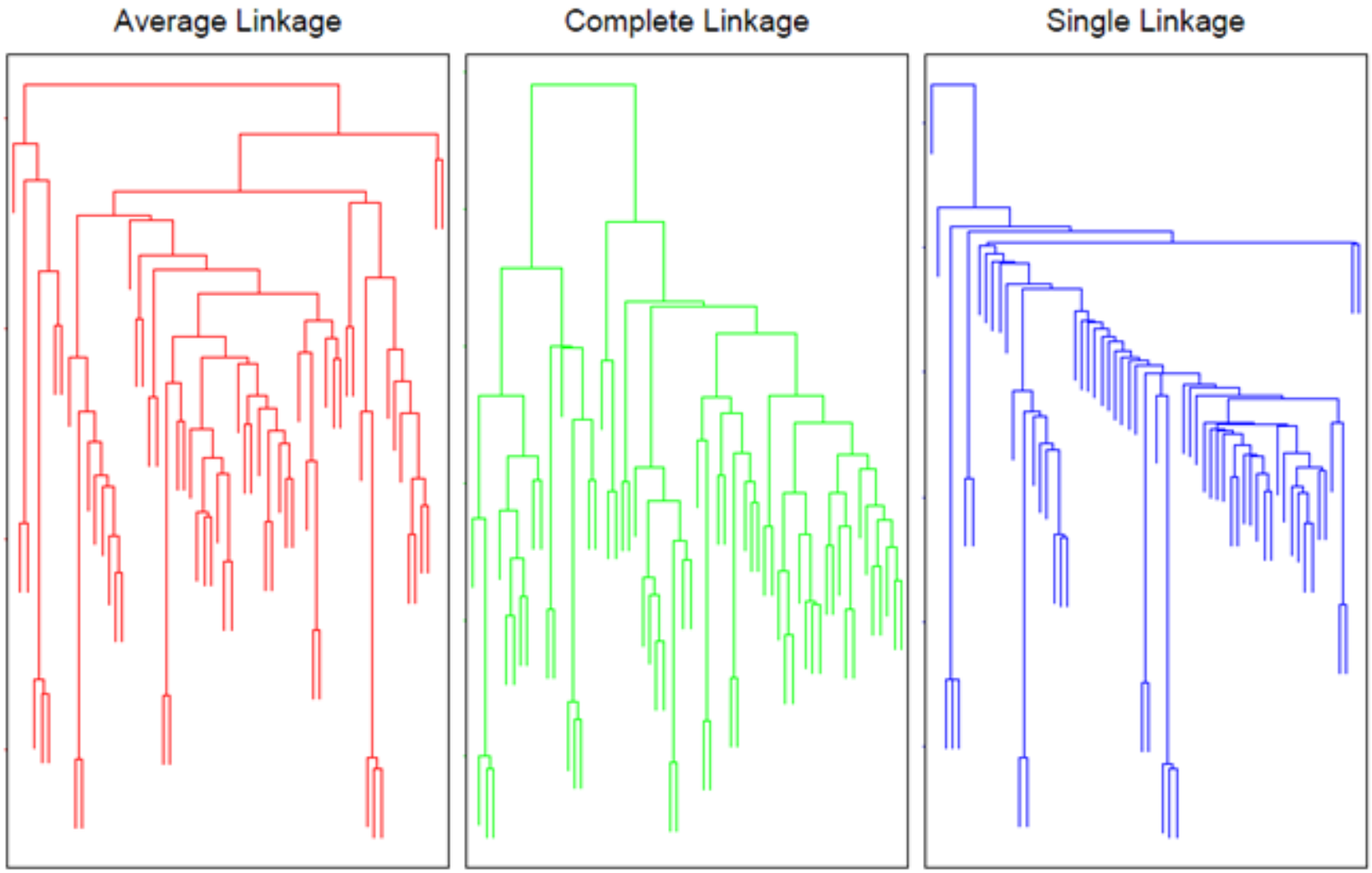
Linkage Functions

- SL considers only a single pair of data points; if this pair is close enough then action is taken. So, SL can form a “chain” by combining relatively far apart data points.
- SL often violates the compactness property of a cluster. SL can produce clusters with large diameters (D_G).

$$D_G = \max_{i \in G,\, j \in G} d_{ij}$$

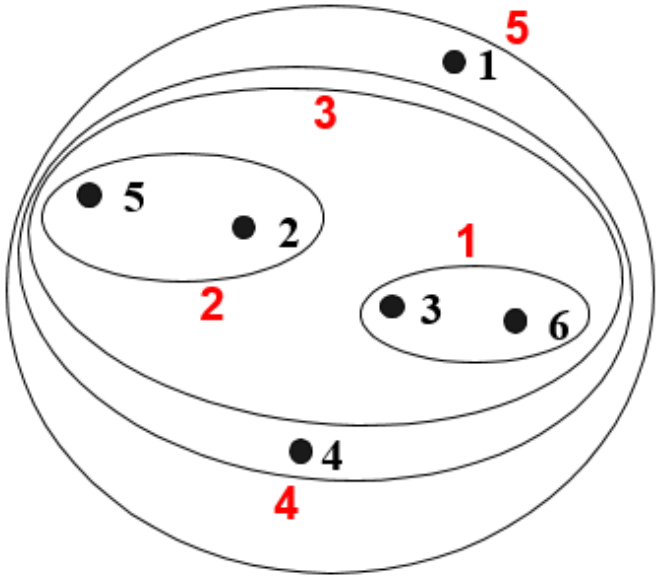
- CL is just the opposite of SL; it produces many clusters with small diameters.
- CL can violate “closeness” property- two close data points may be assigned to different clusters.
- GA is a compromise between SL and CL

Linkage Functions

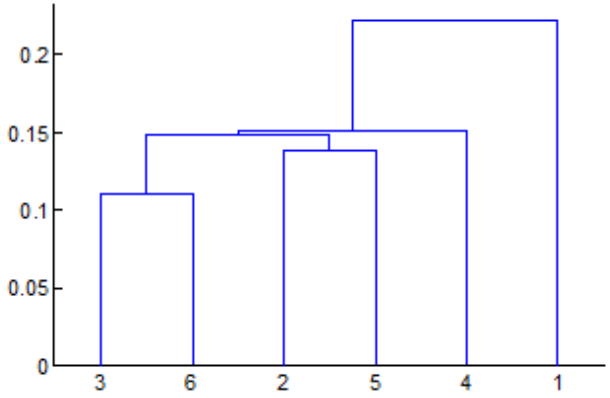


Cluster Similarity: MIN (Single Linkage)

- Similarity of two clusters is based on the two most similar (closest) points in the different clusters
- Determined by one pair of points, i.e., by one link in the proximity graph



Nested Clusters



Dendrogram

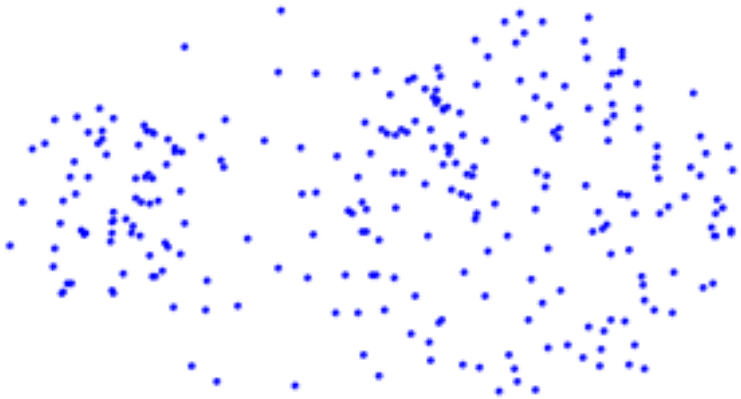
Strength of MIN (Single Linkage)

Original Points

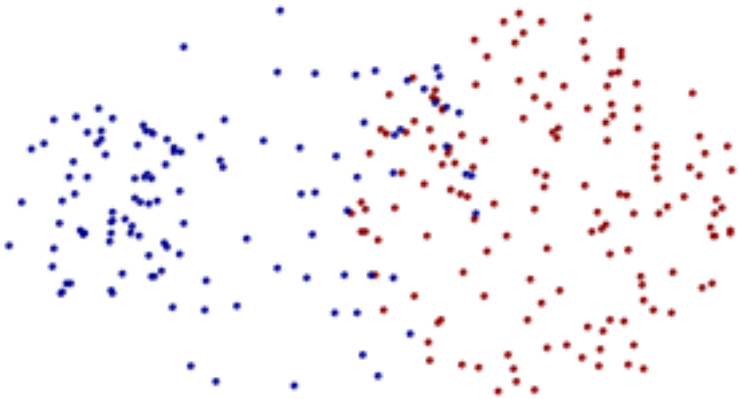
Two Clusters

- Can handle non-elliptical shapes

Limitations of MIN (Single Linkage)



Original Points

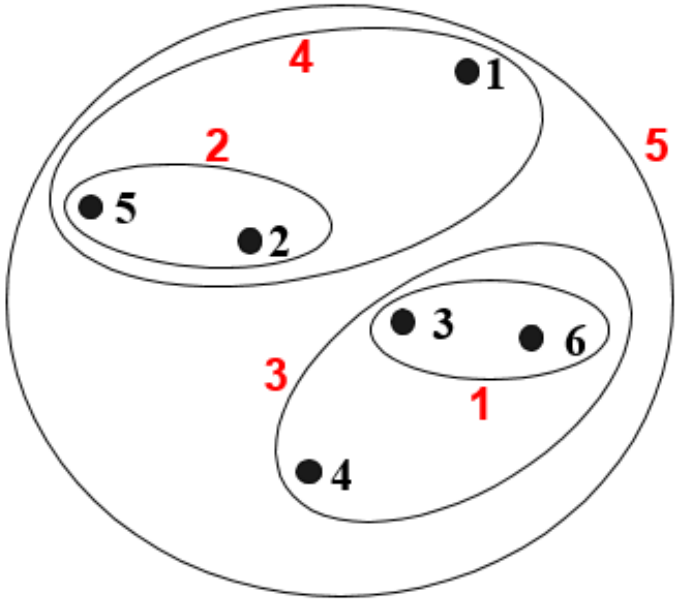


Two Clusters

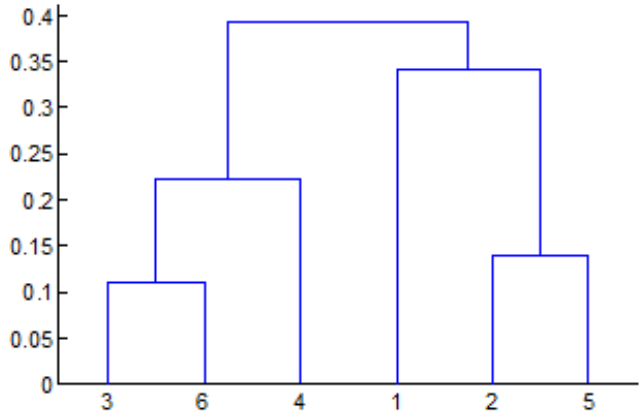
- Sensitive to noise and outliers

Cluster Similarity: MAX (Complete Linkage)

- Similarity of two clusters is based on the two least similar (most distant) points in the different clusters
  - Determined by all pairs of points in the two clusters

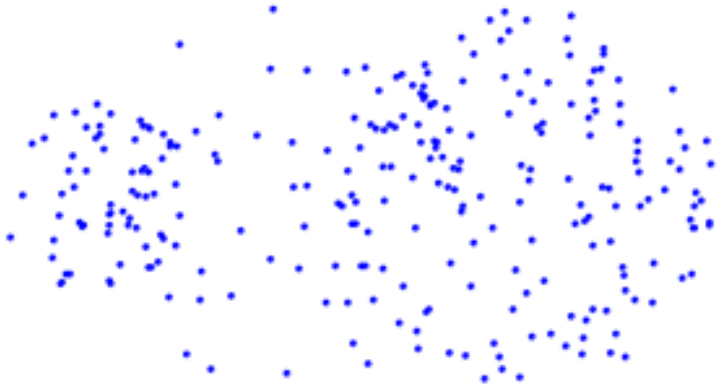


Nested Clusters

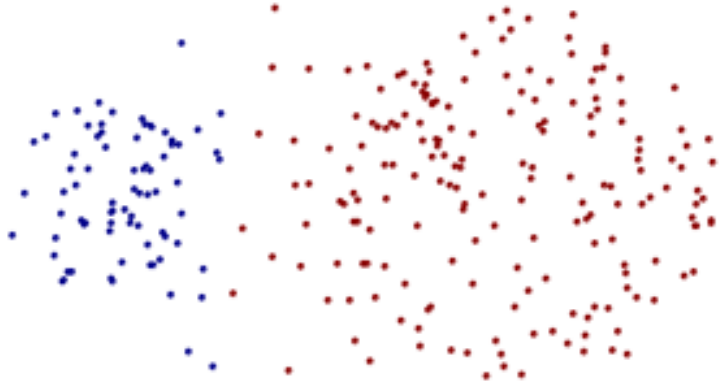


Dendrogram

Strength of MAX (Complete Linkage)



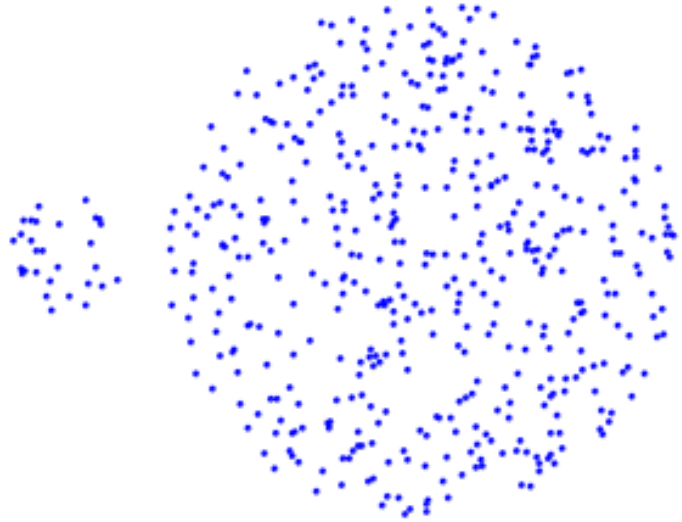
Original Points



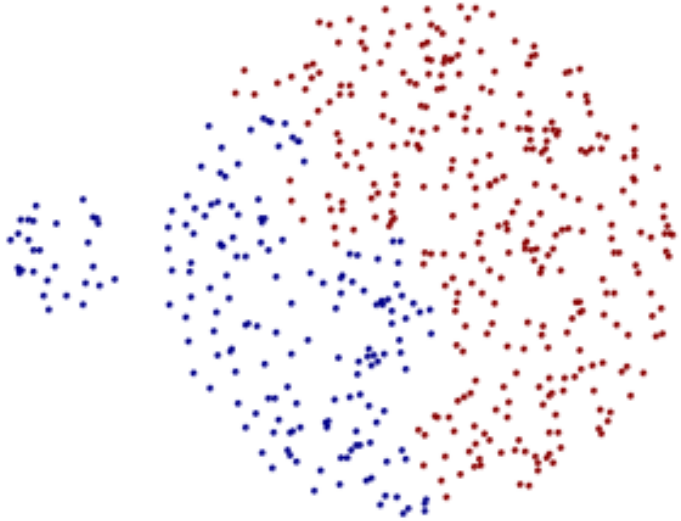
Two Clusters

- Less susceptible to noise and outliers

Limitations of MAX (Complete Linkage)



Original Points



Two Clusters

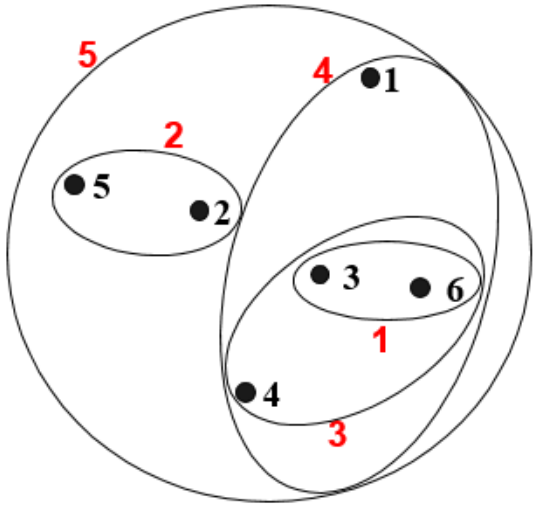
- Tends to break large clusters
- Biased towards globular clusters

Cluster Similarity: Group Average

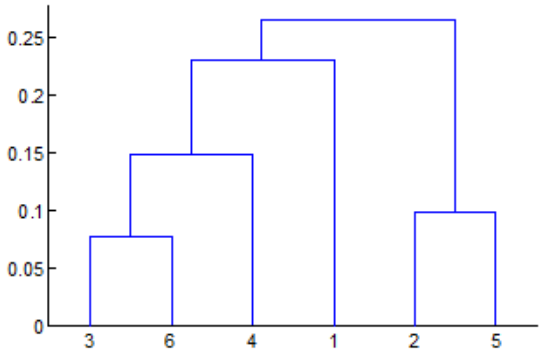
- Proximity of two clusters is the average of pairwise proximity between points in the two clusters.

$$\text{proximity}(\text{Cluster}_i, \text{Cluster}_j) = \frac{\sum_{\substack{p_i \in \text{Cluster}_i \\ p_j \in \text{Cluster}_j}} \text{proximity}(p_i, p_j)}{|\text{Cluster}_i| * |\text{Cluster}_j|}$$

- Need to use average connectivity for scalability since total proximity favors large clusters



Nested Clusters



Dendrogram

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

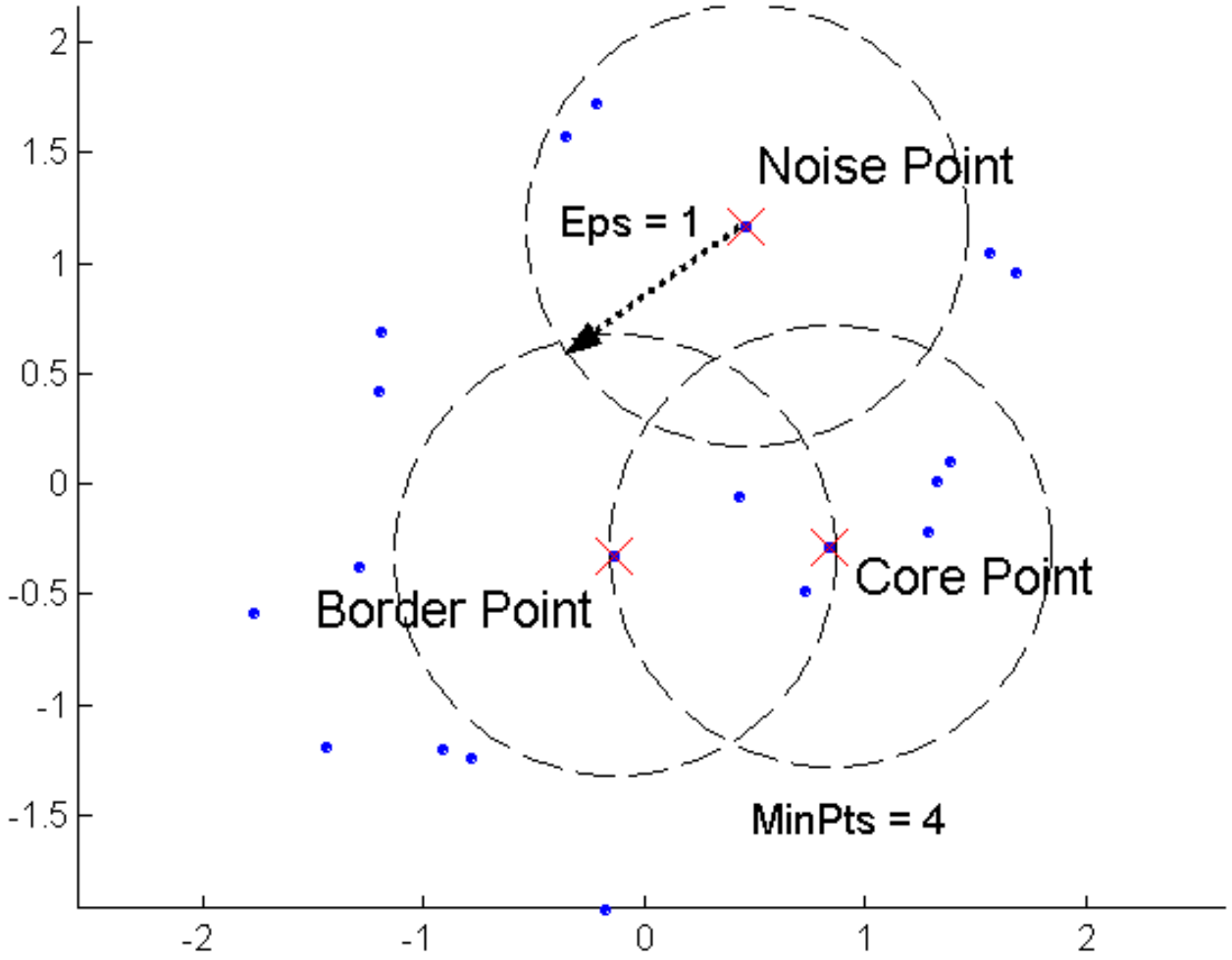
DBSCAN
a density-based algorithm

DBSCAN is a density-based algorithm.

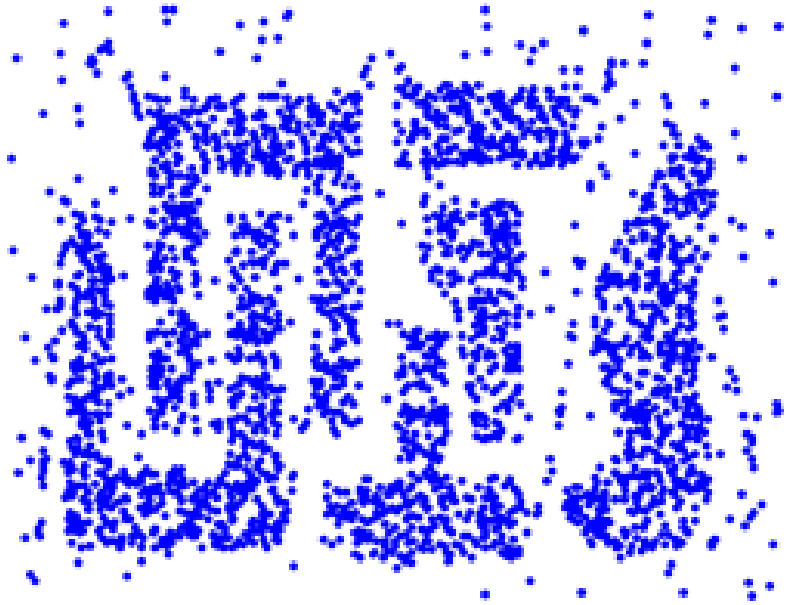
- DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is a density based clustering algorithm.
- The algorithm grows regions with sufficiently high density into clusters and discovers clusters of arbitrary shape in spatial databases with noise.
- It defines a cluster as a maximal set of *density-connected* points.

- **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 neighborhood of a core point
  - A noise point is any point that is not a core point or a border point.

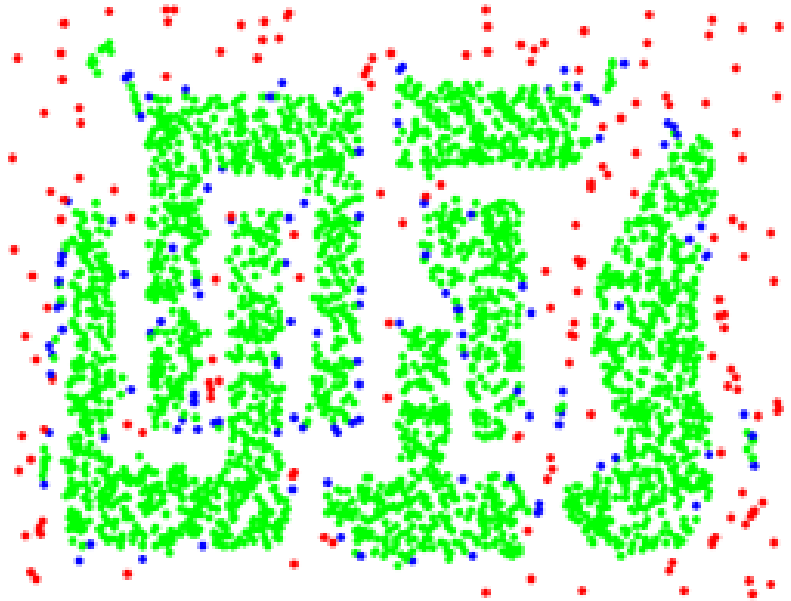
DBSCAN:
Core, Border, and Noise Points



DBSCAN:
Core, Border, and Noise Points



Original Points



Point types: **core**, **border** and **noise**

Eps = 10, MinPts = 4

DBSCAN algorithm

1: Label all points as core, border, or noise points.

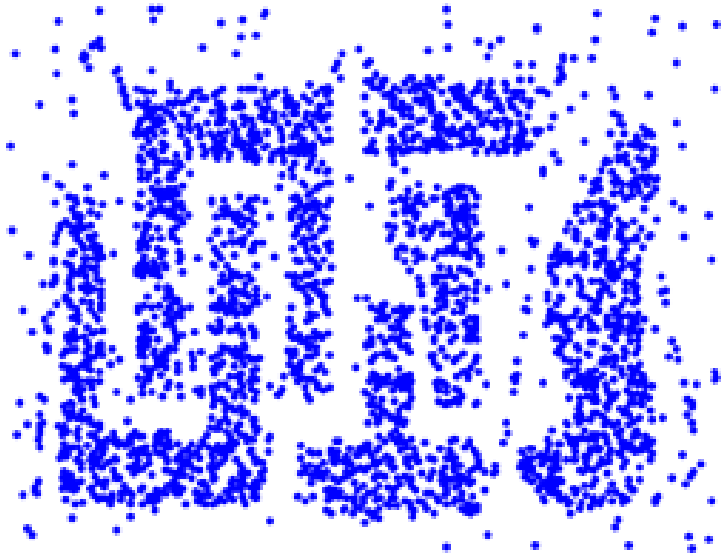
2: Eliminate noise points,

3: Put an edge between all core points that are within Eps of each other.

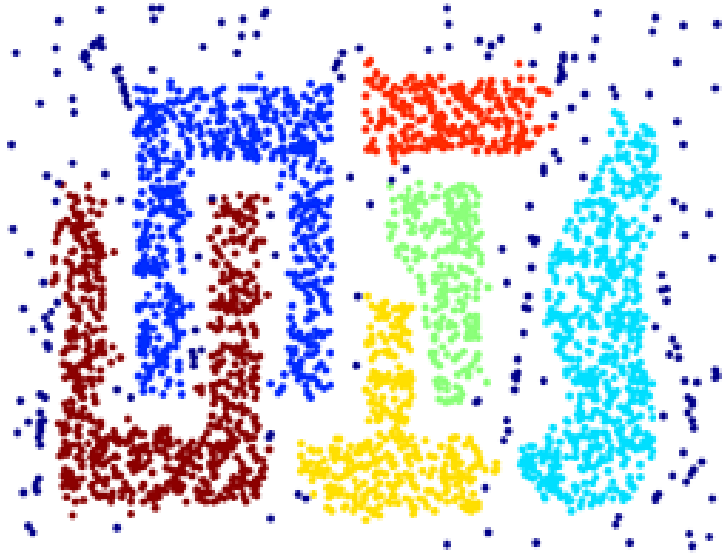
4: Make each group of connected core points into a separate cluster.

5: Assign each border point to one of the clusters of its associated core points.

When DBSCAN Works Well



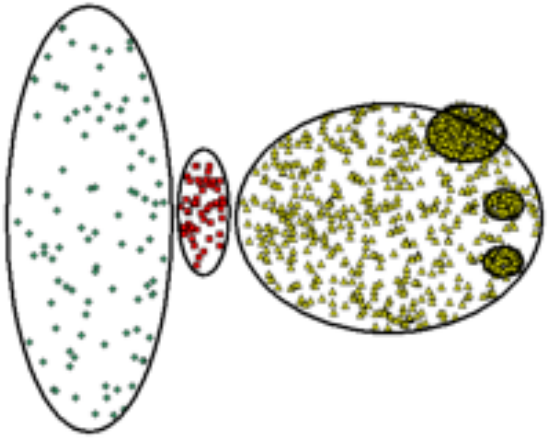
Original Points



Clusters

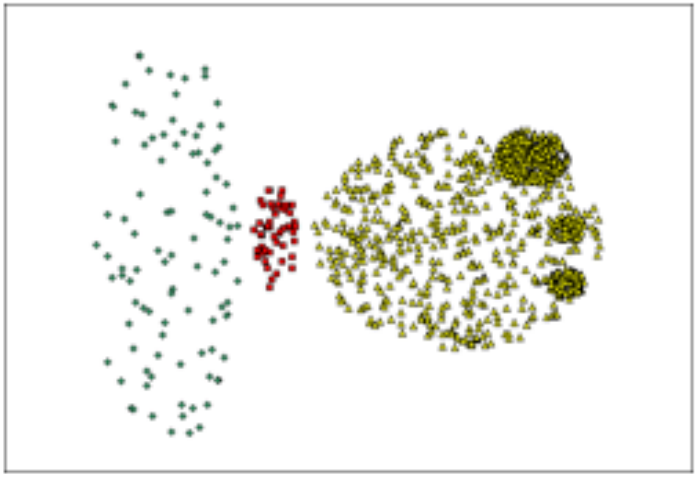
- Resistant to Noise
- Can handle clusters of different shapes and sizes

When DBSCAN Does NOT Work Well

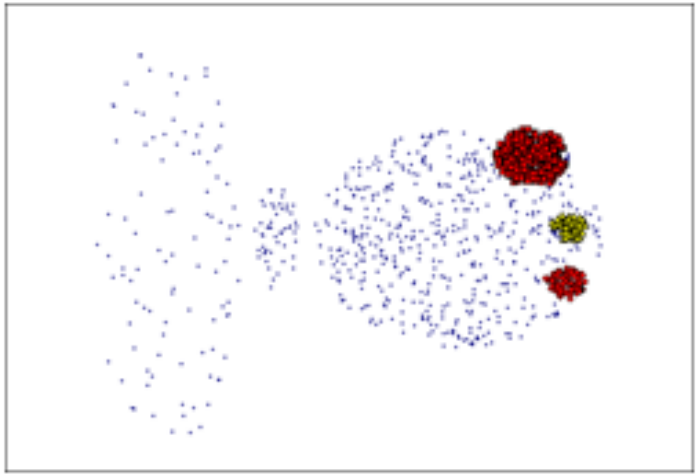


Original Points

- Varying densities
- High-dimensional data



(MinPts=4, Eps=9.75).

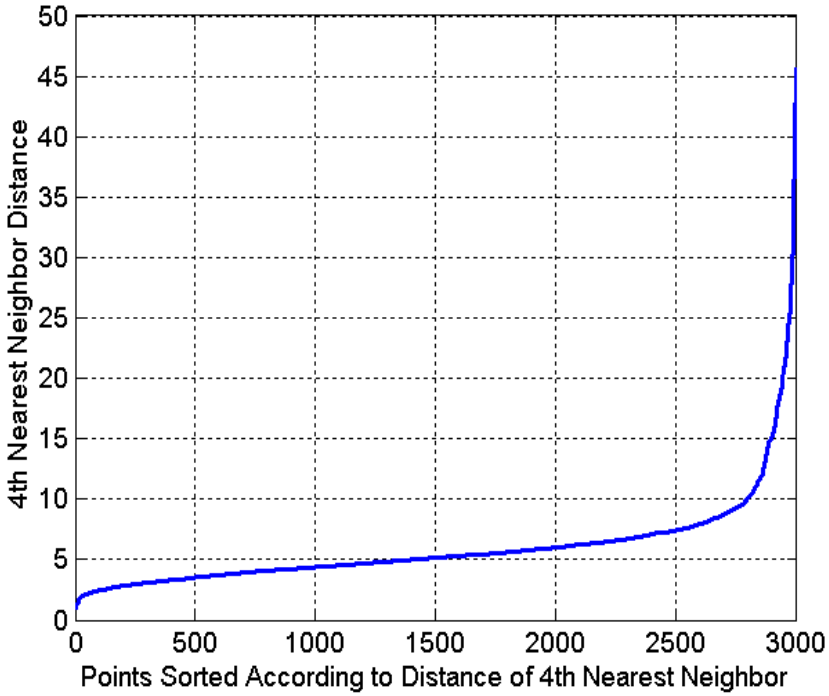


(MinPts=4, Eps=9.92)

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

- A sharp change at the value of k-dist that corresponds to a suitable value of ***Eps***.
- Select this distance as ***Eps*** parameter and take the value of **k** as the ***MinPts*** parameter.



DBSCAN: Example

Data Points: (1,1) (1,2) (2,1) (2,2) (3,2) (4,1) (5,1)

MinPts: 3 Eps:1.5

Determine types of points as core, border, noise. Use Euclidean distance.

Which points are in the cluster?

	(1,1)	(1,2)	(2,1)	(2,2)	(3,2)	(4,1)	(5,1)
(1,1)							
(1,2)							
(2,1)							
(2,2)							
(3,2)							
(4,1)							
(5,1)							

DBSCAN: Example

Data Points: Core: (1,1) (1,2) (2,1) (2,2) (3,2)

Border: (4,1) Noise(5,1)

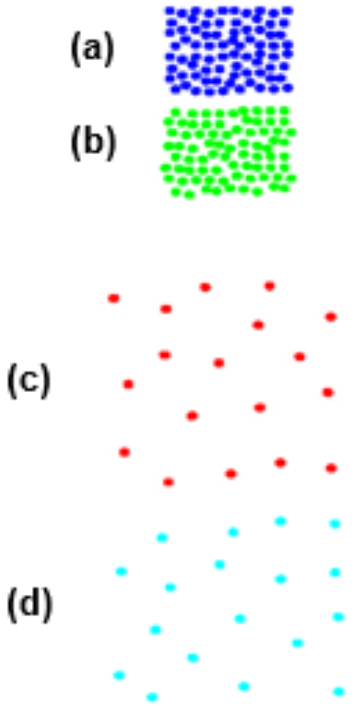
MinPts: 3 Eps:1.5

Determine types of points as core, border, noise. Use Euclidean distance.

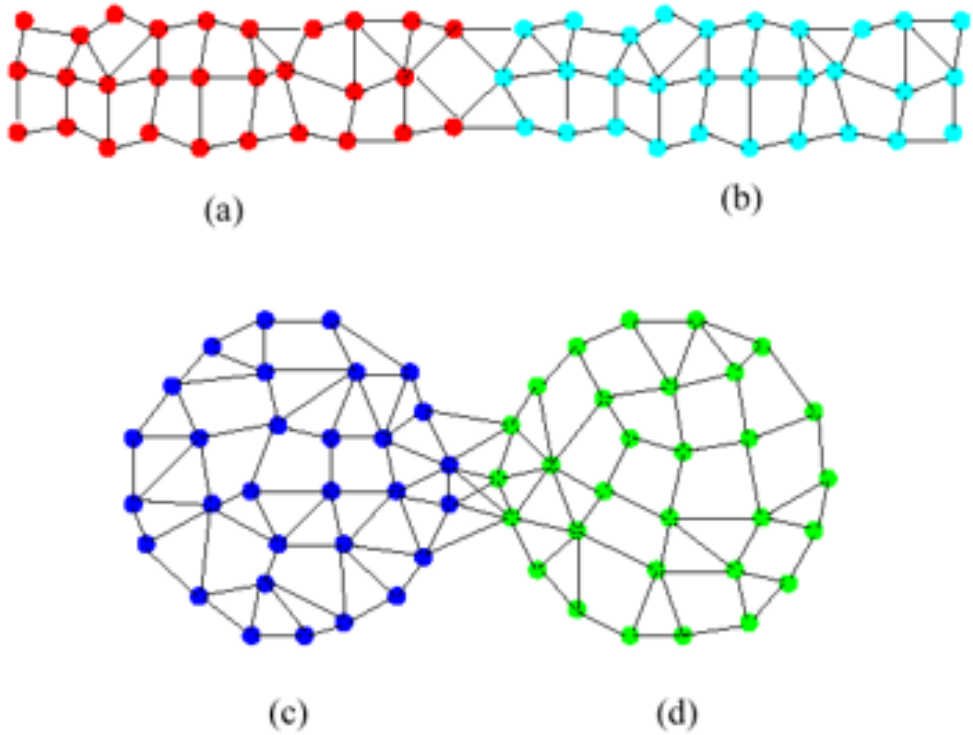
Which points are in the cluster?

	(1,1)	**(1,2)**	**(2,1)**	**(2,2)**	**(3,2)**	**(4,1)**	**(5,1)**
(1,1)	0	1	1	Sq(2)	Sq(5)	3	4
(1,2)	1	0	Sq(2)	1	2	Sq(10)	Sq(17)
(2,1)	1	Sq(2)	0	1	Sq(2)	2	4
(2,2)	Sq(2)	1	1	0	1	Sq(5)	Sq(10)
(3,2)	Sq(5)	2	Sq(2)	1	0	Sq(2)	Sq(5)
(4,1)	3	Sq(10)	2	Sq(5)	Sq(2)	0	1
(5,1)	4	Sq(17)	4	Sq(10)	Sq(5)	1	0

Limitations of Current Merging Schemes



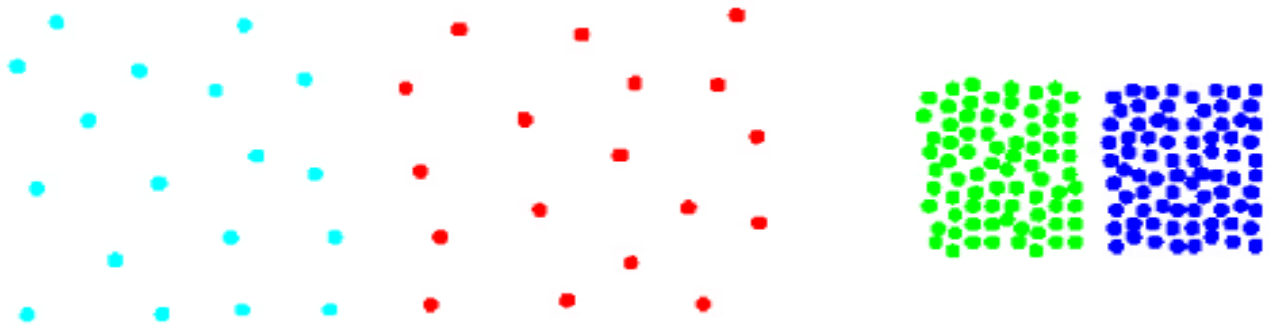
Closeness schemes
will merge (a) and (b)



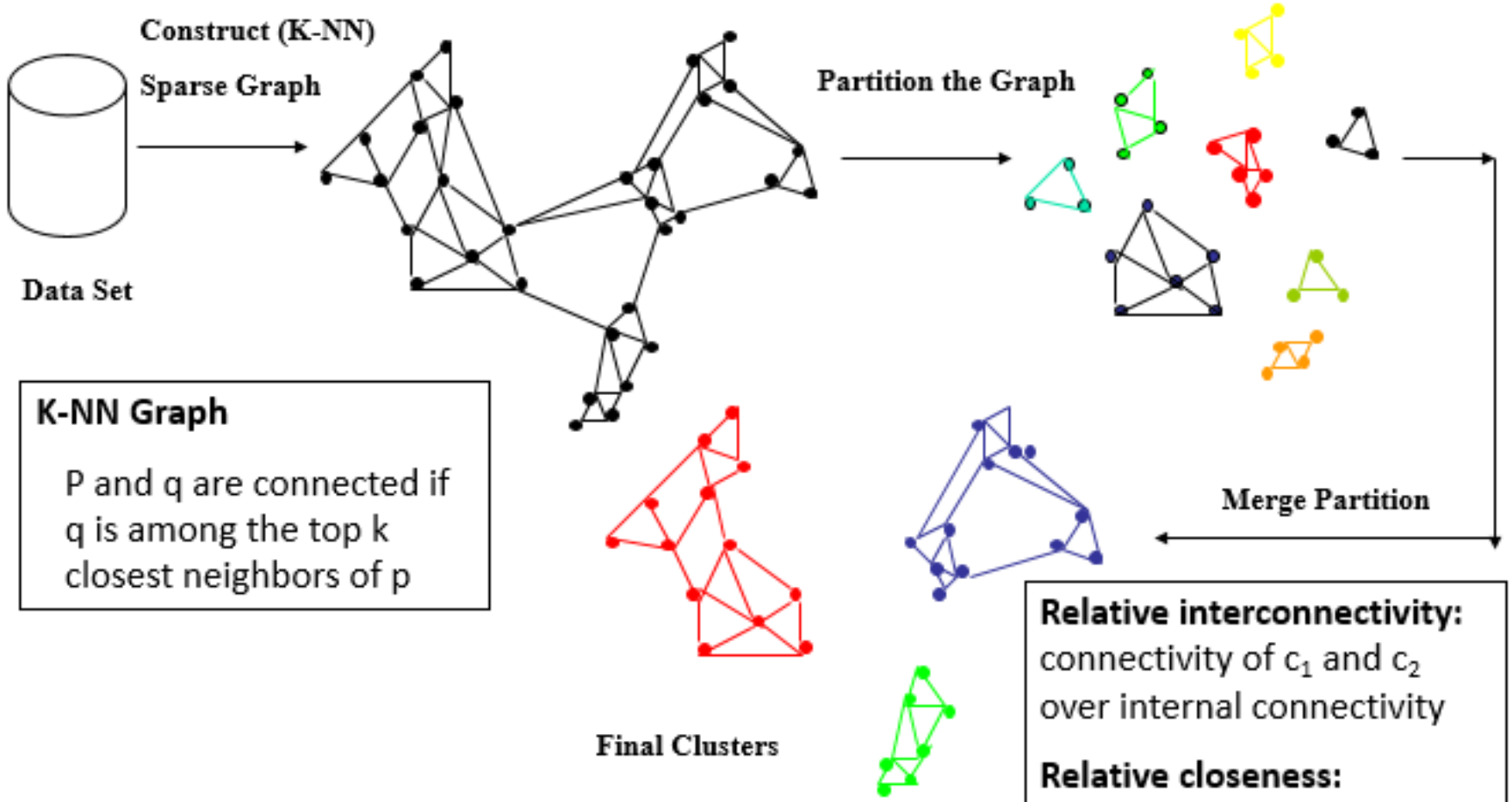
Average connectivity schemes
will merge (c) and (d)

Graph-Based Clustering
Chameleon: Clustering Using Dynamic Modeling

- Adapt to the characteristics of the data set to find the natural clusters
- Use a dynamic model to measure the similarity between clusters
  - Main property is 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*
- Chameleon: Hierarchical Clustering with Dynamic Modeling



Overall Framework of CHAMELEON



K-NN Graph

P and q are connected if q is among the top k closest neighbors of p

Relative interconnectivity: connectivity of c_1 and c_2 over internal connectivity

Relative closeness: closeness of c_1 and c_2 over internal closeness

Chameleon: Deciding Which Clusters to Merge

The **relative interconnectivity, *RI*(*Ci*, *Cj*),** between two clusters, *Ci* and *Cj*, is defined as the absolute interconnectivity between *Ci* and *Cj*, normalized with respect to the internal interconnectivity of the two clusters, *Ci* and *Cj*.

$$RI(C_i, C_j) = \frac{|EC_{\{C_i, C_j\}}|}{\frac{1}{2}(|EC_{C_i}| + |EC_{C_j}|)}$$

- where $EC_{\{Ci, Cj\}}$ is the edge cut (edges to be cut to separate clusters) for a cluster containing both *Ci* and *Cj*. Similarly, EC_{Ci} (or EC_{Cj}) is the minimum sum of the cut edges that partition *Ci* (or *Cj*) into two roughly equal parts.

Chameleon: Deciding Which Clusters to Merge

The relative closeness, *RC(Ci, Cj)*, between a pair of clusters, *Ci* and *Cj*, is the absolute closeness between *Ci* and *Cj*, normalized with respect to the internal closeness of the two clusters, *Ci* and *Cj*.

$$RC(C_i, C_j) = \frac{\overline{S}_{EC_{\{C_i, C_j\}}}}{\frac{|C_i|}{|C_i|+|C_j|}\overline{S}_{EC_{C_i}} + \frac{|C_j|}{|C_i|+|C_j|}\overline{S}_{EC_{C_j}}}$$

where $\overline{S}_{EC_{\{C_i,C_j\}}}$ is the average weight of the edges that connect vertices in C_i to vertices in C_j, and $\overline{S}_{EC_{C_i}}$ (or $\overline{S}_{EC_{C_j}}$) is the average weight of the edges that belong to the min-cut bisector of cluster C_i (or C_j).

Cluster Evaluation
Measuring Clustering Quality

- 3 kinds of measures: **External**, **internal** and **relative**

- **External**: supervised, employ criteria not inherent to the dataset
  - Compare a clustering against prior or expert-specified knowledge (i.e., the ground truth) using certain clustering quality measure
- **Internal**: unsupervised, criteria derived from data itself
  - Evaluate the goodness of a clustering by considering how well the clusters are separated, and how compact the clusters are.
- **Relative**: directly compare different clustering results, usually those obtained via different parameter settings for the same algorithm

Internal Measures: Cohesion and Separation

- Cluster Cohesion: Measures how closely related are objects in a cluster
- Cluster Separation: Measure how distinct or well-separated a cluster is from other clusters

- Example: Squared Error
  - **Cohesion** is measured by the within cluster sum of squares (SSE)

$$SSE = \sum_{i} \sum_{x \in C_i} (x - m_i)^2$$

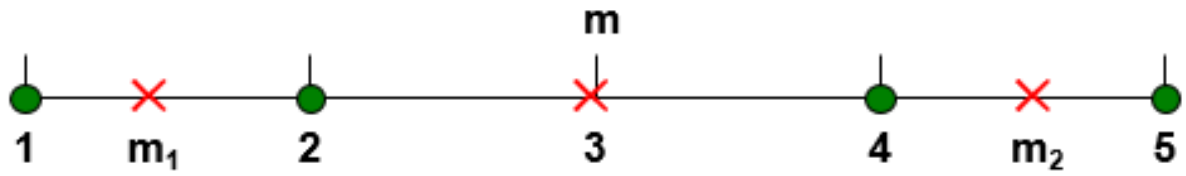
  - **Separation** is measured by the between cluster sum of squares

$$BSS = \sum_{i} |C_i| (m - m_i)^2$$

    - Where $|C_i|$ is the size of cluster i

Internal Measures: Cohesion and Separation

- Example: SSE
  - BSS + SSE = constant



K=1 cluster:

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

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

$$Total = 10 + 0 = 10$$

K=2 clusters:

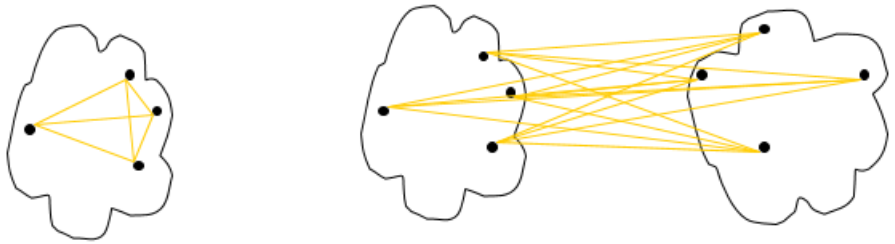
$$SSE = (1-1.5)^2 + (2-1.5)^2 + (4-4.5)^2 + (5-4.5)^2 = 1$$

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

$$Total = 1 + 9 = 10$$

Internal Measures: Cohesion and Separation

- A proximity graph based approach can also be used for cohesion and separation.
  - Cluster cohesion is the sum of the weight of all links within a cluster.
  - Cluster separation is the sum of the weights between nodes in the cluster and nodes outside the cluster.



cohesion separation

$$cohesion(C_i) = \sum_{\substack{\mathbf{x} \in C_i \\ \mathbf{y} \in C_i}} proximity(\mathbf{x}, \mathbf{y})$$

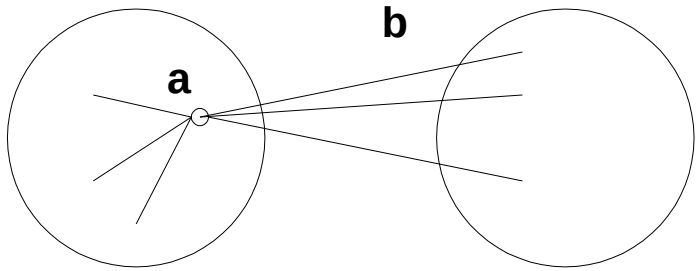
$$separation(C_i, C_j) = \sum_{\substack{\mathbf{x} \in C_i \\ \mathbf{y} \in C_j}} proximity(\mathbf{x}, \mathbf{y})$$

Internal Measures: 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



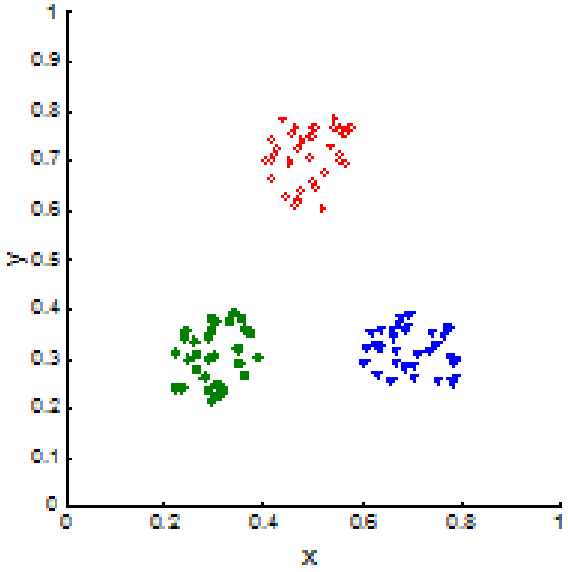
Measuring Cluster Validity Via Correlation

- 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

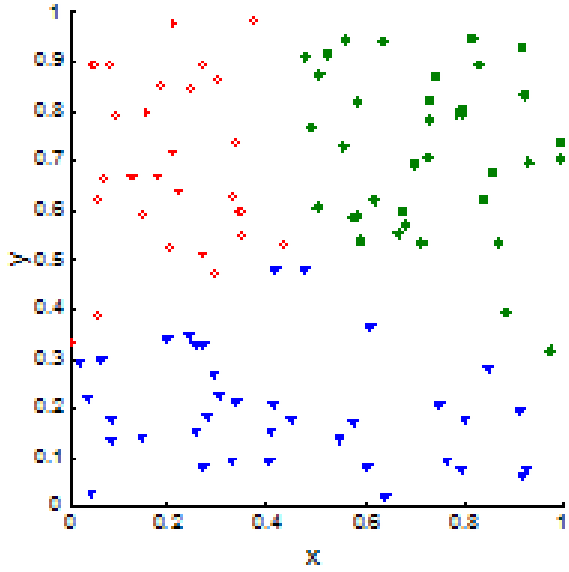
- Compute the correlation between the two matrices
- High correlation indicates that points that belong to the same cluster are close to each other.

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

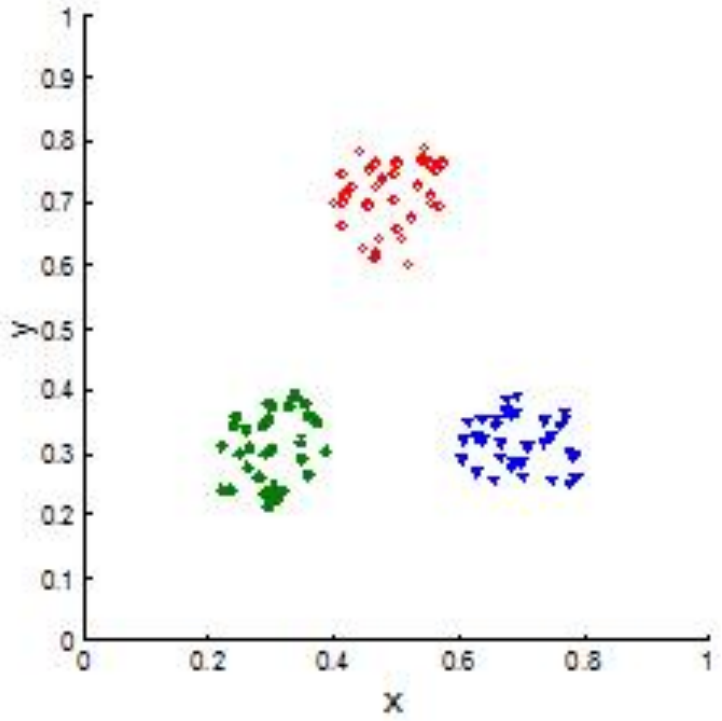
Using Similarity Matrix for Cluster Validation

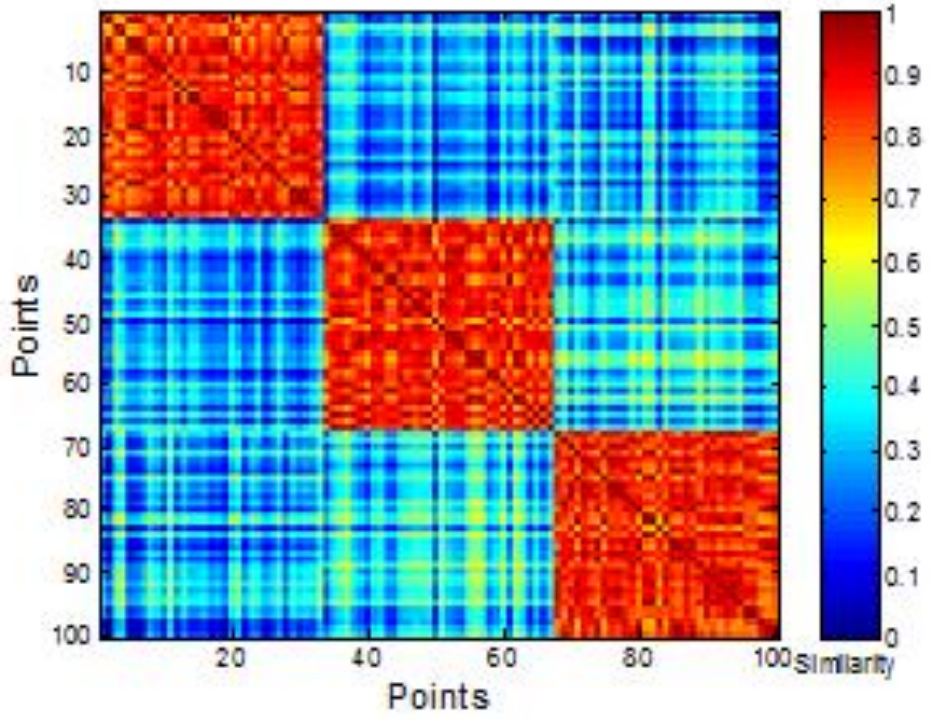
- The **ideal similarity matrix** is constructed by creating a matrix that has one row and one column for each data point; and assigning a 1 to an entry if the associated pair of points belongs to the same cluster. All other entries are 0.
- **Actual similarity matrix** is created by filling entries with similarities between points as follows:
  - Similarity = 1 – ((d - mind) / (maxd - mind))

- We sort the rows and columns of the similarity matrix so that all objects belonging to the same class are together, then an ideal similarity matrix has a **block diagonal** structure.

Using Similarity Matrix for Cluster Validation

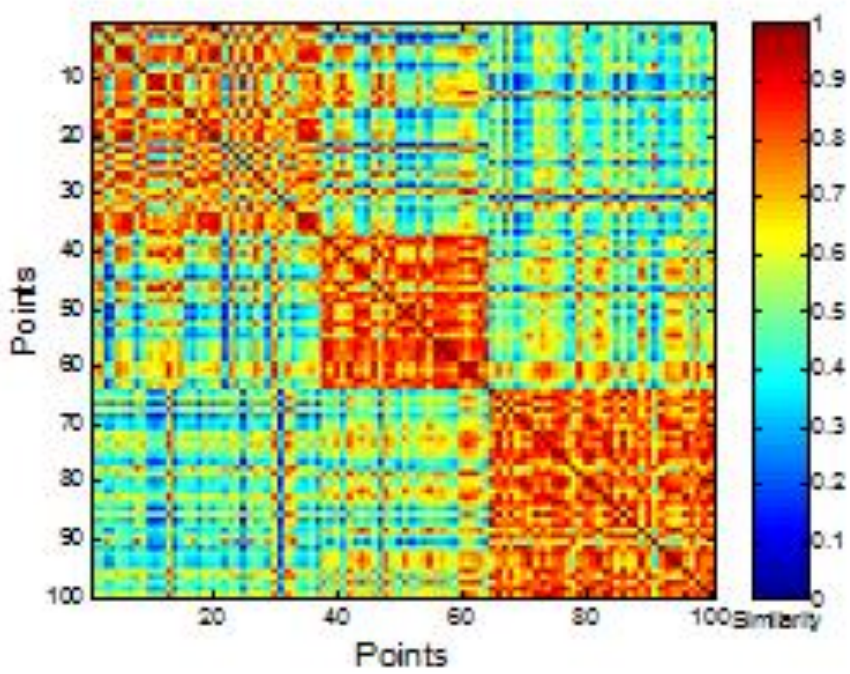
- Order the similarity matrix with respect to cluster labels and inspect visually.

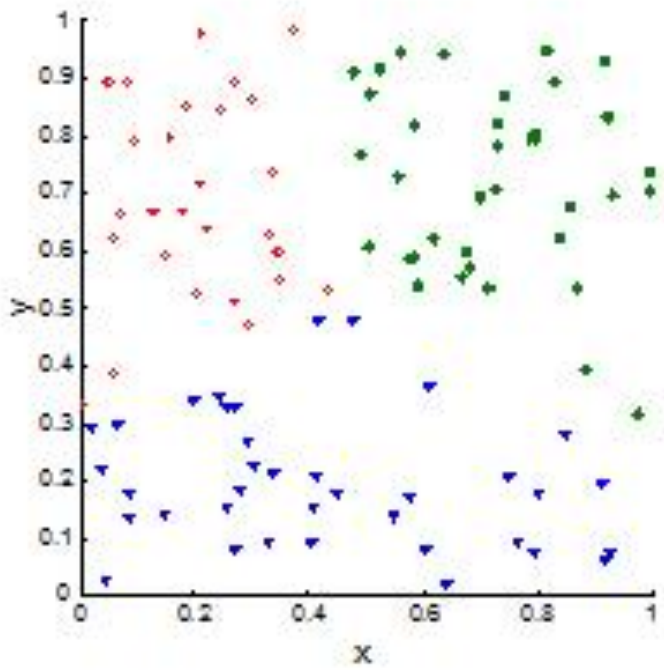




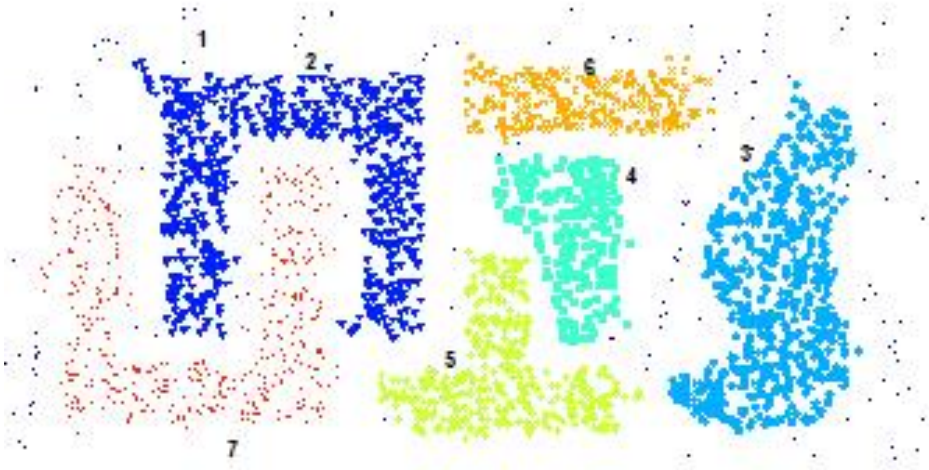
Using Similarity Matrix for Cluster Validation

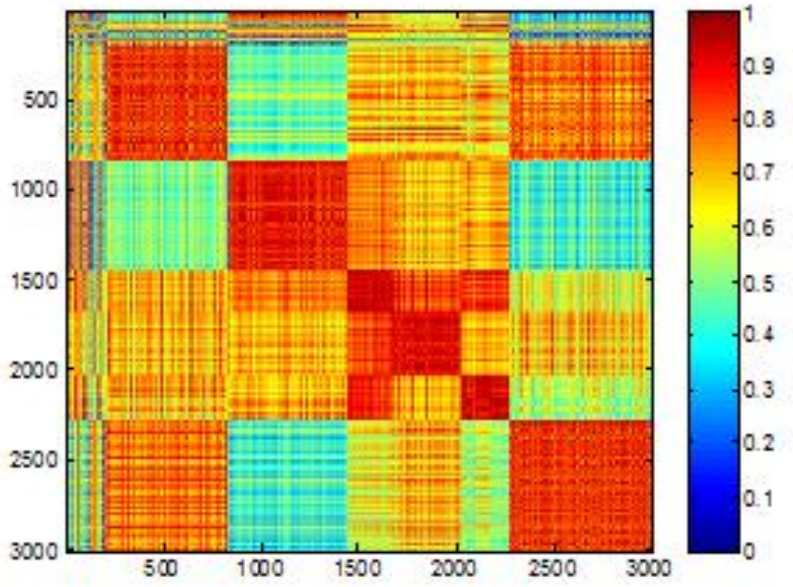
- Clusters in random data are not so crisp





Using Similarity Matrix for Cluster Validation

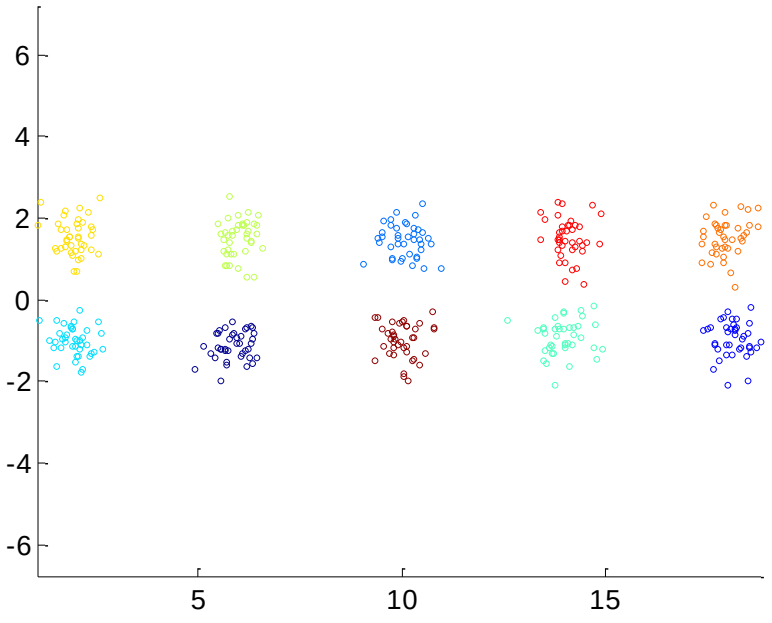


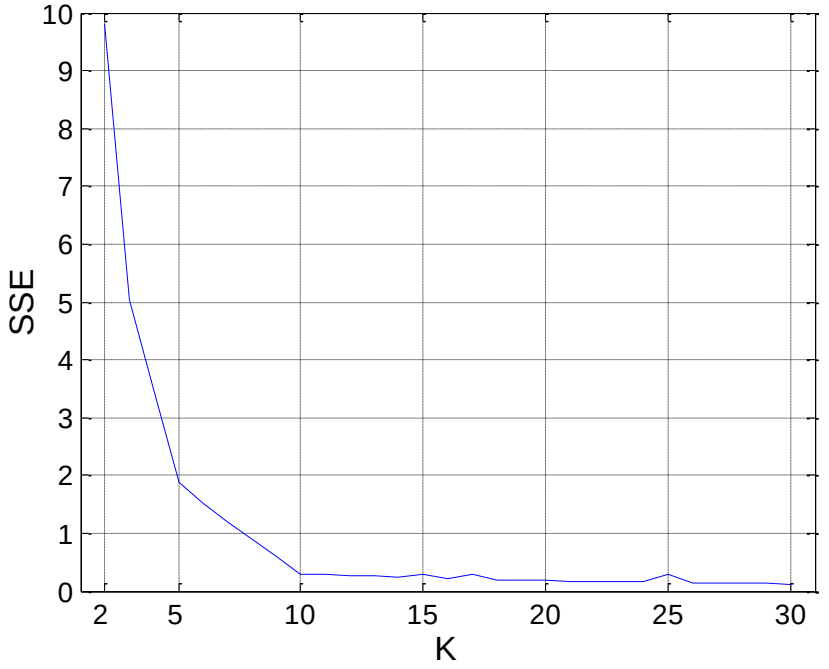


DBSCAN

Determining the Correct Number of Clusters

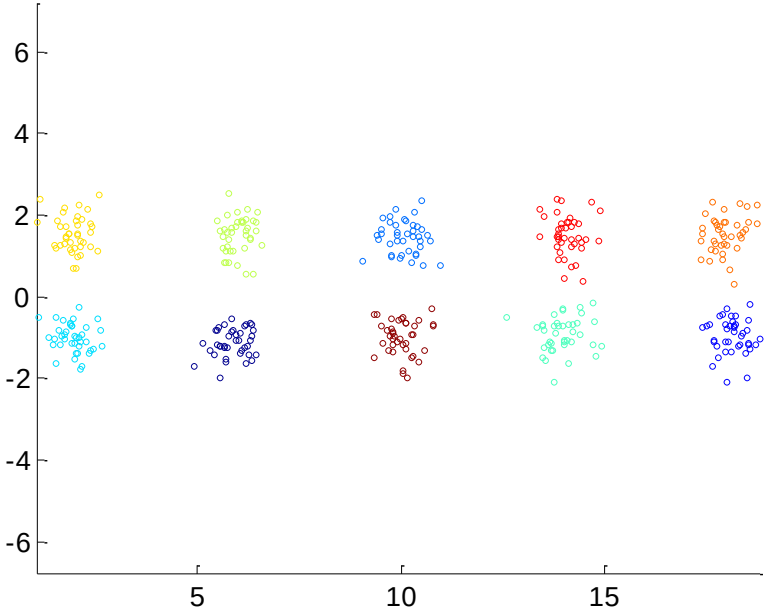
- SSE is good for comparing two clusterings or two clusters (average SSE).
- It can also be used to estimate the number of clusters

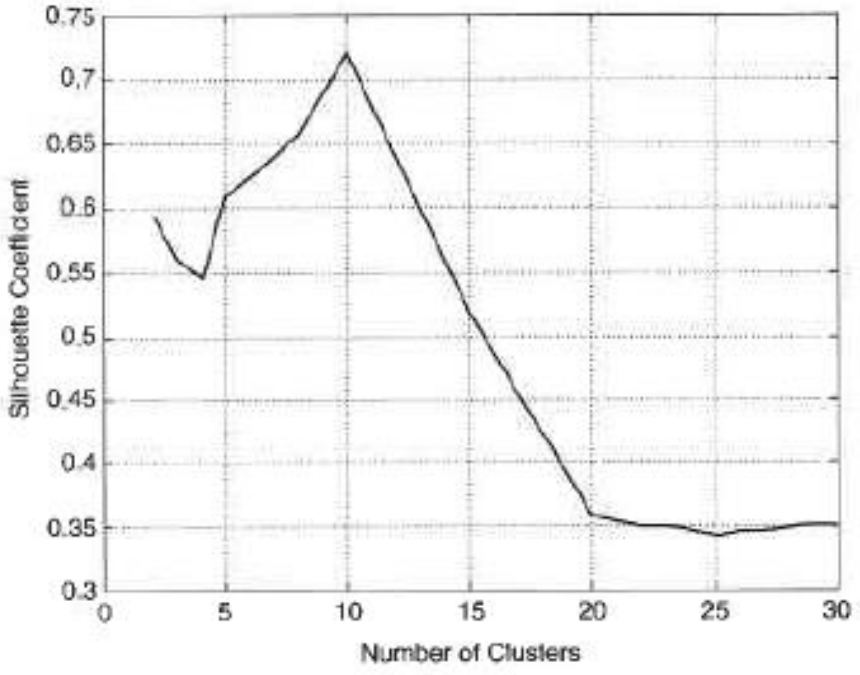




Determining the Correct Number of Clusters

- Various unsupervised cluster evaluation measures can be used to approximately determine the correct or natural number of clusters.
  - Silhouette Coefficient





Clustering Tendency

- Does a data set have clusters?

Hopkins Statistic:

- Generate p points that are randomly distributed across the data space and
- also sample p actual data points.
- find the distance to the nearest neighbor in the original data set.
  - Let the u_i be the nearest neighbor distances of the artificially generated points,
  - while the w_i are the nearest neighbor distances of the sample of points from the original data set.
- The Hopkins statistic H is $$H = \frac{\sum_{i=1}^{p} w_i}{\sum_{i=1}^{p} u_i + \sum_{i=1}^{p} w_i}$$
- If the randomly generated points and the sample of data points have roughly the same nearest neighbor distances, then H will be near 0.5.
- Values of H near 0 and 1 indicate, respectively, data that is highly clustered and data that is regularly distributed in the data space.

External Measures of Cluster Validity

Entropy: The degree to which each cluster consists of objects of a single class. 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 i belongs to class j as $p_{ij} = m_{ij}/m_i$, where m_i is the number of objects in cluster i and m_{ij} is the number of objects of class j in cluster i. Using this class distribution, the entropy of each cluster i is calculated using the standard formula, $e_i = -\sum_{j=1}^{L} p_{ij} \log_2 p_{ij}$, where 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., $e = \sum_{i=1}^{K} \frac{m_i}{m} e_i$, where K is the number of clusters and m is the total number of data points.