Introduction to Data Mining

- INFO411 -

An Introduction to

Clustering, Visualization, and Cluster Analysis

Presented by
Wanqing Li

Overview

1. Definition and Motivation
2. Clustering
   a) Types of clusters and clustering methods
   b) K-means
   c) Self-Organizing Maps
   d) Hierarchical Clustering
   e) Density based clustering
   f) Cluster Validation
3. Conclusions

What is Cluster Analysis?

- Aims to subdivide samples into groups (clusters) so that
  - Samples in the same cluster share similarities.
  - Samples in different clusters are not similar.
- A clustering algorithm uses the features of a set of samples; it does not use a target value -> unsupervised algorithm.

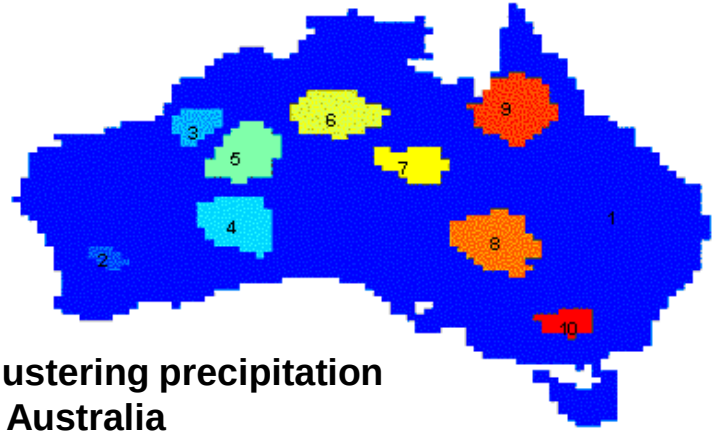
Application examples:

- Grouping together molecules with similar medical properties.
- Grouping together the users of a site on the basis of their behavior.
- Grouping a set of customers on the basis of their records.

Applications of Cluster Analysis

- **Understanding**
  - Group related documents for browsing, group genes and proteins that have similar functionality, or group stocks with similar price fluctuations
- **Summarization**
  - Reduce the size of large data sets
- **Visualization**
  - Reduce dimensionality of data.

	Discovered Clusters	*Industry Group*
1	Applied-Matl-DOWN,Bay-Network-Down,3-COM-DOWN, Cabletron-Sys-DOWN,CISCO-DOWN,HP-DOWN, DSC-Comm-DOWN,INTEL-DOWN,LSI-Logic-DOWN, Micron-Tech-DOWN,Texas-Inst-Down,Tellabs-Inc-Down, Natl-Semiconduct-DOWN,Oracl-DOWN,SGI-DOWN, Sun-DOWN	Technology1-DOWN
2	Apple-Comp-DOWN,Autodesk-DOWN,DEC-DOWN, ADV-Micro-Device-DOWN,Andrew-Corp-DOWN, Computer-Assoc-DOWN,Circuit-City-DOWN, Compaq-DOWN, EMC-Corp-DOWN, Gen-Inst-DOWN, Motorola-DOWN,Microsoft-DOWN,Scientific-Atl-DOWN	Technology2-DOWN
3	Fannie-Mae-DOWN,Fed-Home-Loan-DOWN, MBNA-Corp-DOWN,Morgan-Stanley-DOWN	Financial-DOWN
4	Baker-Hughes-UP,Dresser-Inds-UP,Halliburton-HLD-UP, Louisiana-Land-UP,Phillips-Petro-UP,Unocal-UP, Schlumberger-UP	Oil-UP



Clustering precipitation in Australia

What is not Cluster Analysis?

- Supervised classification
  - Have (and the use of) class label information
- Simple segmentation
  - Dividing students into different registration groups alphabetically, by last name.
- Results of a query
  - Groupings are a result of an external specification
- Graph partitioning
  - Some mutual relevance and synergy, but areas are not identical

Clustering is a hard problem!

Why is it hard?

- Clustering in two dimensions looks easy.
- Clustering small amounts of data looks easy.
- And in most cases, looks are **not** deceiving.
- But many applications involve not 2, but 10 or 10,000 dimensions
  - **High-dimensional spaces look different:** Almost all pairs of points are at about the same distance!
  - How many clusters do we want vs how many clusters do we have?
- Notion of similarity
  - i.e. is a goose more similar to a chicken than to a pigeon?

Clustering is a hard problem!

1. Notion of a Cluster can be Ambiguous

How many clusters?

Six Clusters

Two Clusters

Four Clusters

Clustering is a hard problem!

2. Identifying clusters in high dimensional data.

Clustering is a hard problem!

3. Notion of similarity:

Example: Say we wish to automatically cluster music into categories so that we can match categories to preferences of a user.

- The dimension of music is tens of millions.
- What makes one song similar to another song; what makes songs dissimilar?
- But what are categories really?

Clustering is a hard problem!

Another example: Cluster text documents

- Represent a document by a vector $(x_1, x_2, \ldots, x_k)$, where $x_i = 1$ iff the i^{th} word (in some order) appears in the document
  - It actually doesn't matter if k is infinite; i.e., we don't limit the set of words.

- Assumption: Documents with similar sets of words may be about the same topic.
  - Is this assumption valid?
  - Word vector is sparse. How to compute a similarity value between documents?

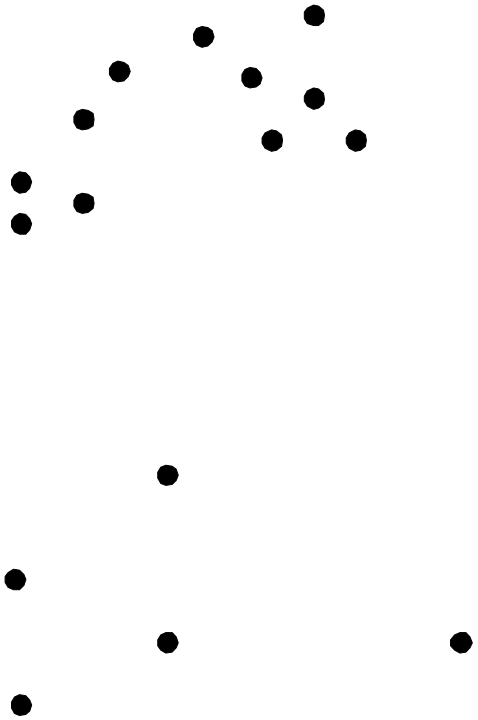
Measuring similarity

- Common similarity measures:
  - **Sets as vectors:** Measure similarity by the **cosine distance**
  - **Sets as sets:** Measure similarity by the **Jaccard distance**
  - **Sets as points:** Measure similarity by **Euclidean distance**
- But there are many other:
  - Hamming distance, Manhatten distance, L1 norm, Pearson Linear Correlation,…
- Knowing what to use can be an art.

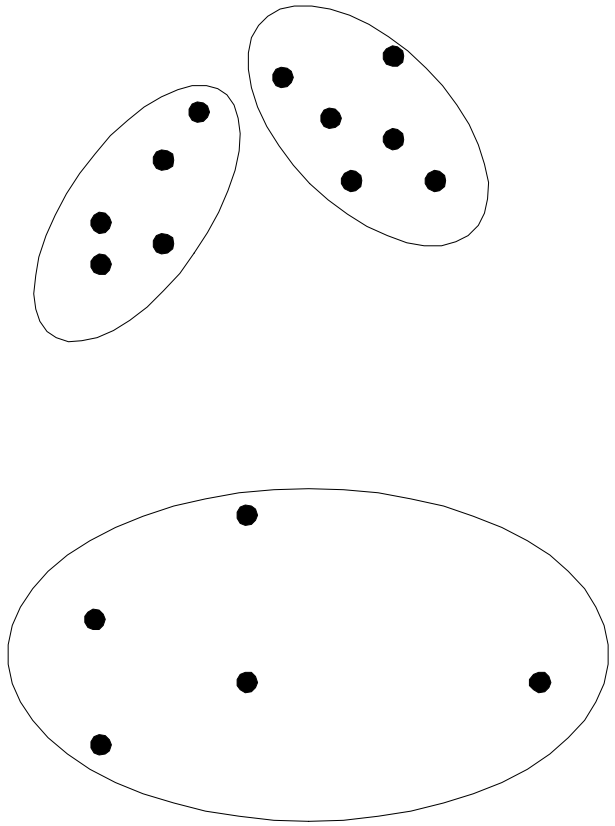
Types of Clustering

- Important distinction between hierarchical and partitional sets of clusters
- Partitional Clustering
  - A division data objects into non-overlapping subsets (clusters) such that each data object is in exactly one subset
- Hierarchical clustering
  - A set of nested clusters organized as a hierarchical tree

Partitional Clustering

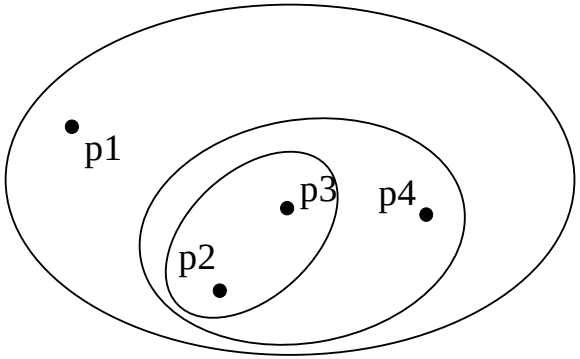


Original Points

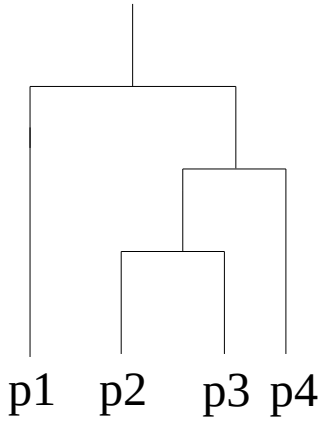


A Partitional Clustering

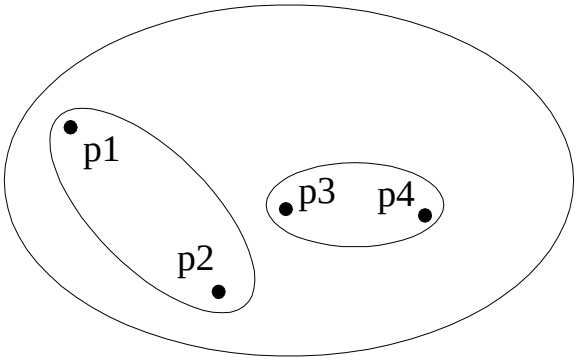
Hierarchical Clustering



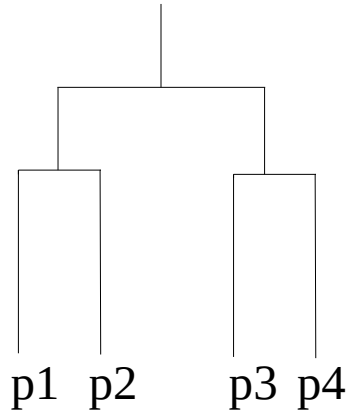
Traditional Hierarchical Clustering



Traditional Dendrogram



Non-traditional Hierarchical Clustering



Non-traditional Dendrogram

Which is better?

No general answer although:

- Partitional clustering is often better when clusters have nice, convex shapes.
- Hierarchical can be better when the shape of clusters are weird.

- Problem: We do not normally know apriori the shape of the clusters.

Other Distinctions Between Sets of Clusters

- Exclusive versus non-exclusive
  - In non-exclusive clustering, points may belong to multiple clusters.
  - Can represent multiple classes or ‘border’ points
- Fuzzy versus non-fuzzy
  - In fuzzy clustering, a point belongs to every cluster with some weight between 0 and 1
  - Weights must sum to 1
  - Probabilistic clustering has similar characteristics
- Partial versus complete
  - In some cases, we only want to cluster some of the data
- Heterogeneous versus homogeneous
  - Cluster of widely different sizes, shapes, and densities

Fuzzy v.s. Probabilistic

Fuzzy Sets and Probability

A Cautionary Tale

- Quite different meanings
- Example - bottles of liquid:

Fuzzy Bottle

0.7 Drinkable

Probabilistic Bottle

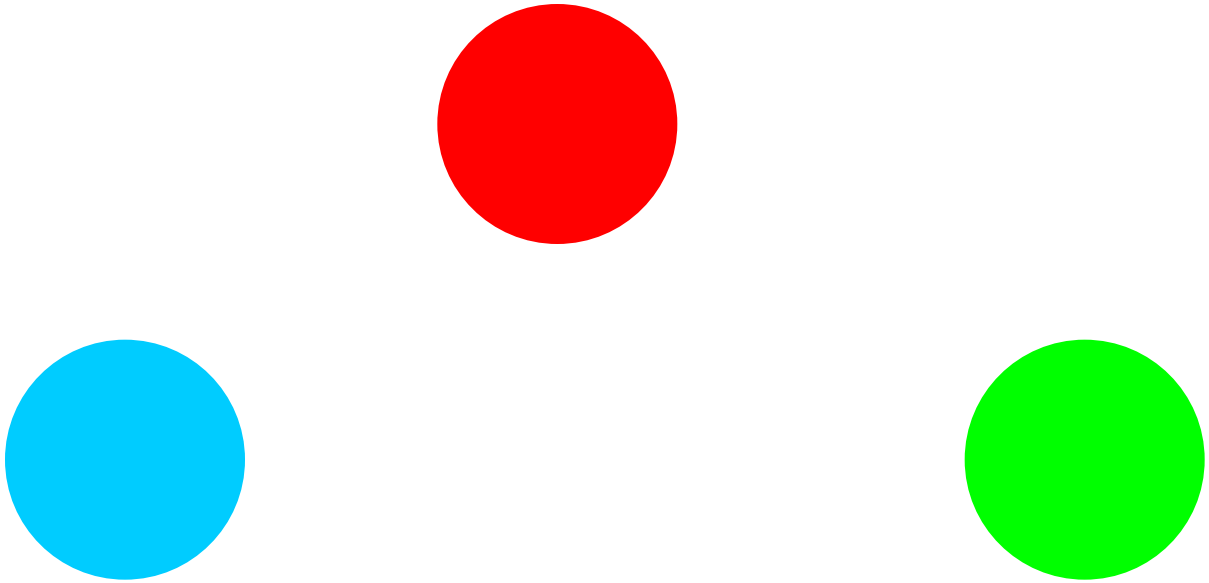
0.7 Drinkable

Types of Clusters

- Well-separated clusters
- Center-based clusters
- Contiguous clusters
- Density-based clusters
- Property or Conceptual
- Described by an Objective Function

Types of Clusters: Well-Separated

- Well-Separated Clusters:
  - A cluster is a set of points such that any point in a cluster is closer (or more similar) to every other point in the cluster than to any point not in the cluster.



3 well-separated clusters

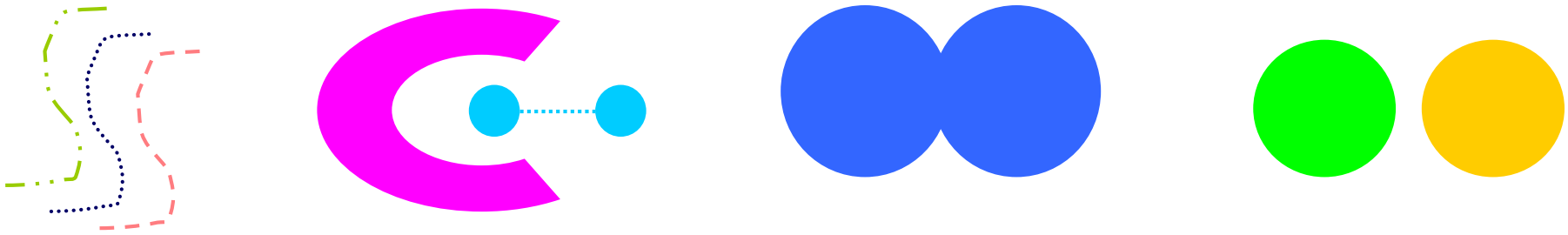
Types of Clusters: Center-Based

- Center-based
  - A cluster is a set of objects such that an object in a cluster is closer (more similar) to the “center” of a cluster, than to the center of any other cluster
  - The center of a cluster is often a **centroid**, the average of all the points in the cluster, or a **medoid**, the most “representative” point of a cluster. In general, those points are called **prototypes**.

4 center-based clusters

Types of Clusters: Contiguity-Based

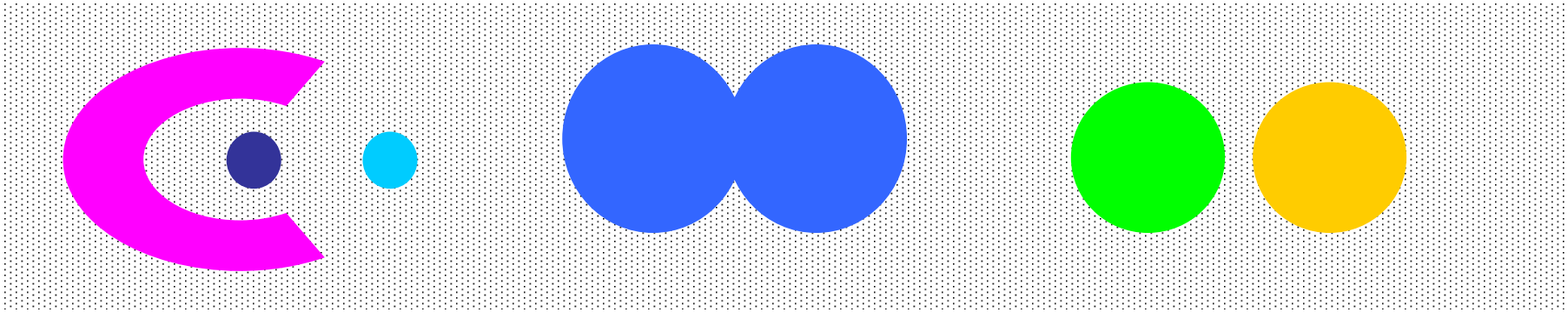
- Contiguous Cluster (Nearest neighbor or Transitive)
  - A cluster is a set of points such that a point in a cluster is closer (or more similar) to one or more other points in the cluster than to any point not in the cluster.



8 contiguous clusters

Types of Clusters: Density-Based

- Density-based
  - A cluster is a dense region of points, which is separated by low-density regions, from other regions of high density.
  - Used when the clusters are irregular or intertwined, and when noise and outliers are present. This is typical for clusters which are not well separated.

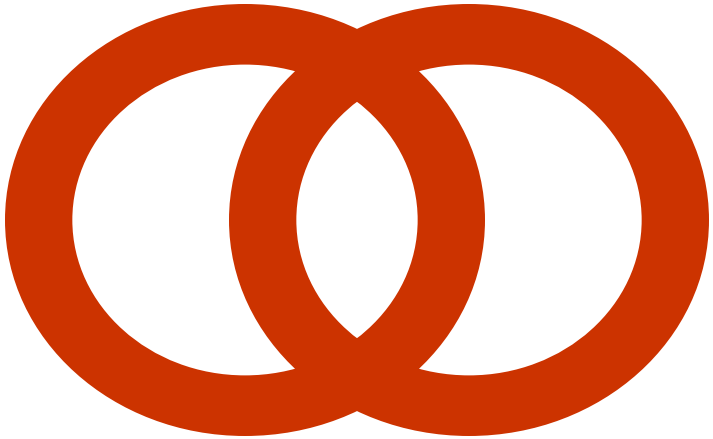


6 density-based clusters

Types of Clusters: Conceptual Clusters

- Shared Property or Conceptual Clusters
  - Finds clusters that share some common property or represent a particular concept.

.



2 Overlapping Circles

Characteristics of the Input Data Are Important

- Type of proximity or density measure
  - This is a derived measure, but central to clustering
- Sparseness
  - Dictates type of similarity
  - Adds to efficiency
- Attribute type
  - Dictates type of similarity
- Type of Data
  - Dictates type of similarity
  - Other characteristics, e.g., autocorrelation
- Dimensionality
- Noise and Outliers
- Type of Distribution

Clustering Algorithms

- K-means and its variants
  - Fast and simple algorithm.
- Self Organizing Maps
  - Topology preserving mapping.
  - Linear computational complexity.
  - Can be seen as an extension to K-means
- Hierarchical clustering.
- Density-based clustering.
- BFR (**B**radley-**F**ayyad-**R**eina), CURE (**C**lustering **U**sing **RE**presentatives), …and many more

- Some of these will be covered in this lecture.

K-means Clustering

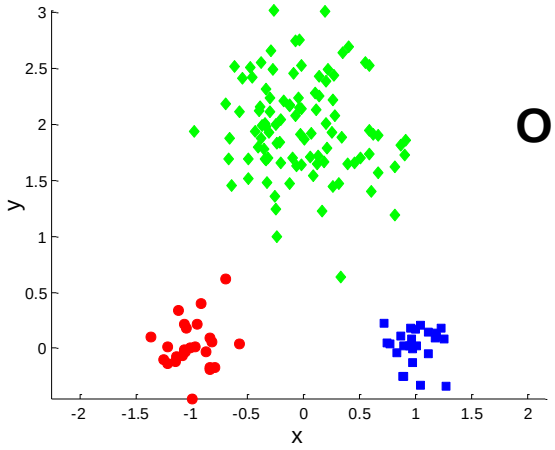
- Is a machine learning algorithm
- Partitional clustering approach
- Each cluster is associated with a **centroid** (also called prototype,codebook)
- Each point is assigned to the cluster with the closest centroid
- Number of clusters, K, must be specified
- The basic algorithm is very simple

1: Select K points as the initial centroids.
2: **repeat**
3: Form K clusters by assigning all points to the closest centroid.
4: Recompute the centroid of each cluster.
5: **until** The centroids don't change

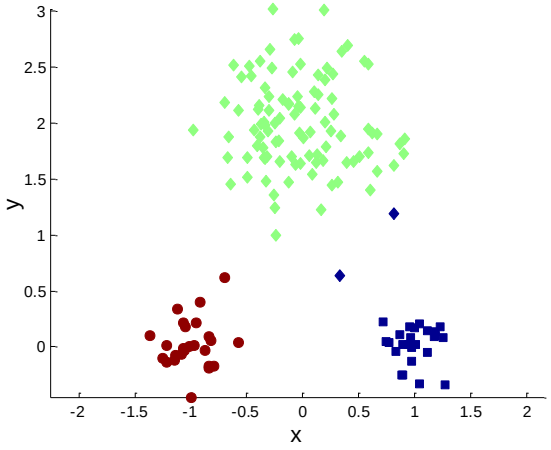
K-means Clustering – Details

- Initial centroids are often chosen randomly.
  - Clusters produced vary from one run to another.
- The centroid is (typically) the mean of the points in the cluster.
- ‘Closeness’ is measured by Euclidean distance, cosine similarity, correlation, etc.
- K-means will converge for common similarity measures mentioned above.
- Most of the convergence happens in the first few iterations.
  - Often the stopping condition is changed to ‘Until relatively few points change clusters’
- Complexity is O(n * K * I * d)
  - n = number of data points, K = number of clusters, I = number of iterations, d = number of attributes

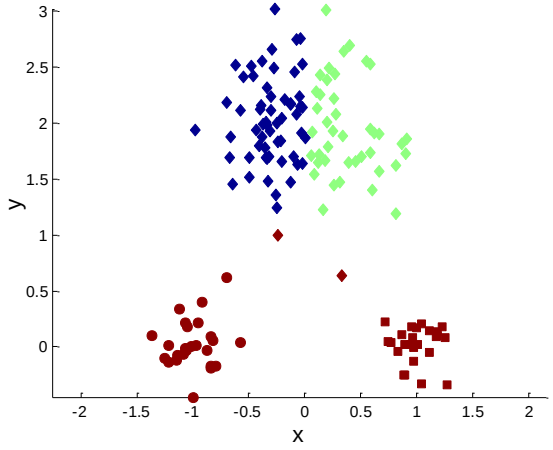
Two different K-means Clusterings



Original Points

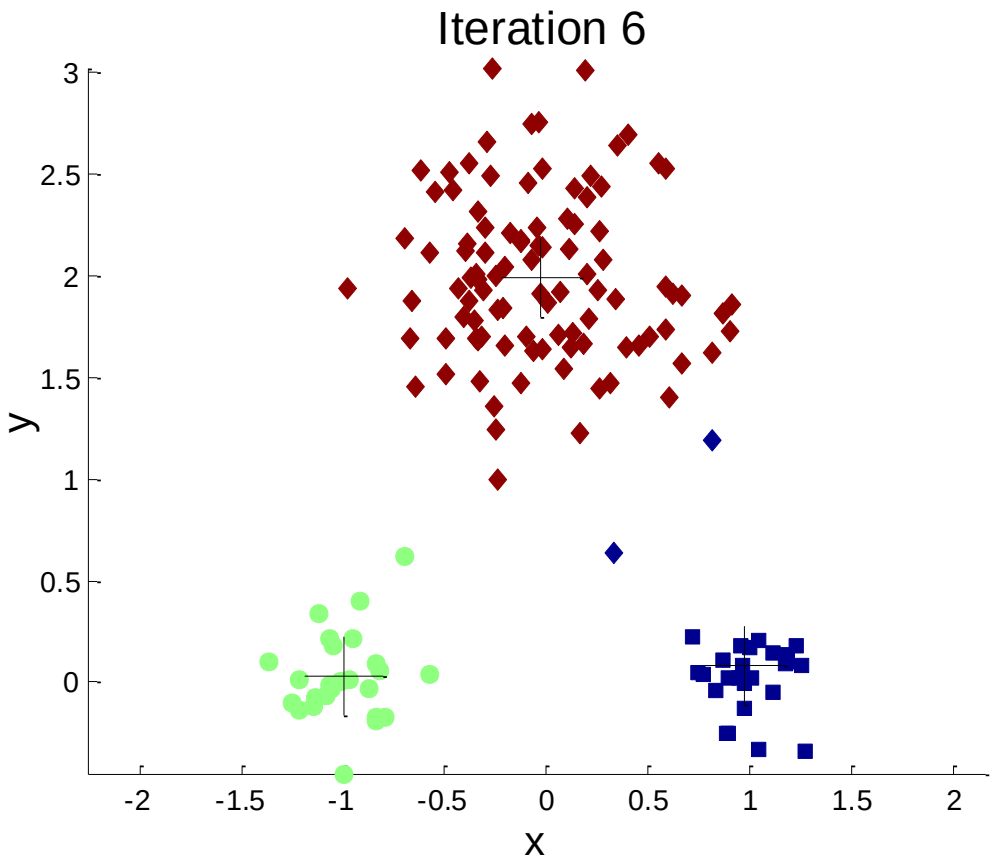


Optimal Clustering

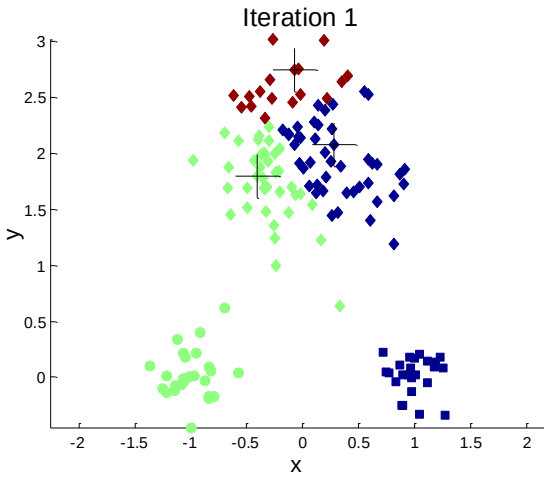


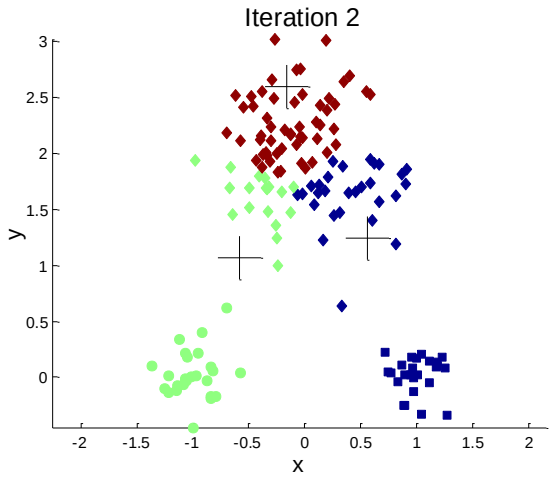
Sub-optimal Clustering

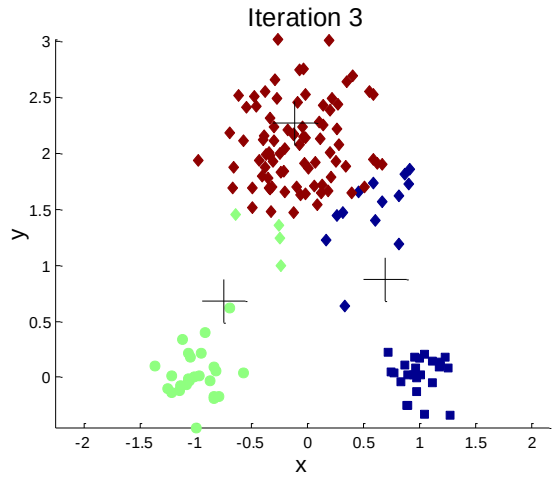
Importance of Choosing Initial Centroids

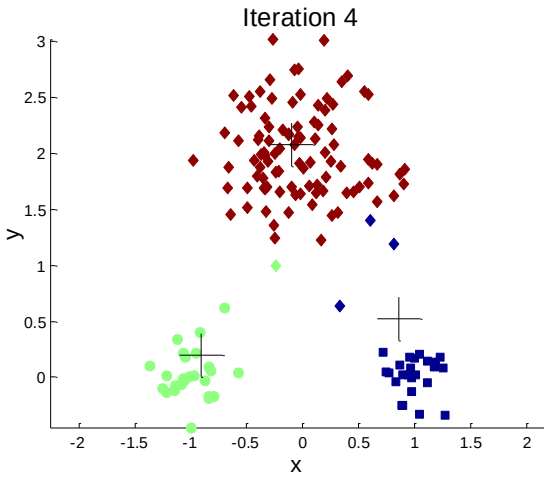


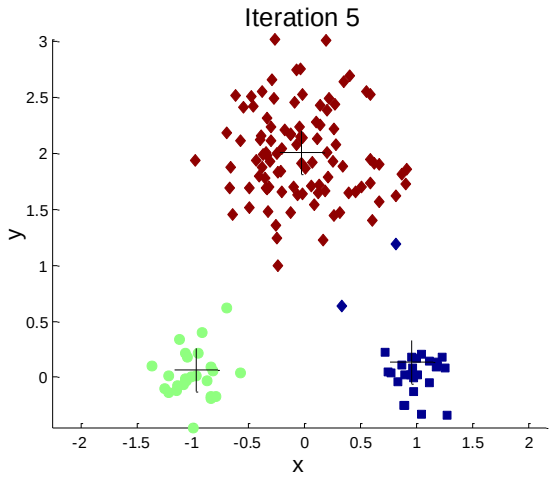
Importance of Choosing Initial Centroids

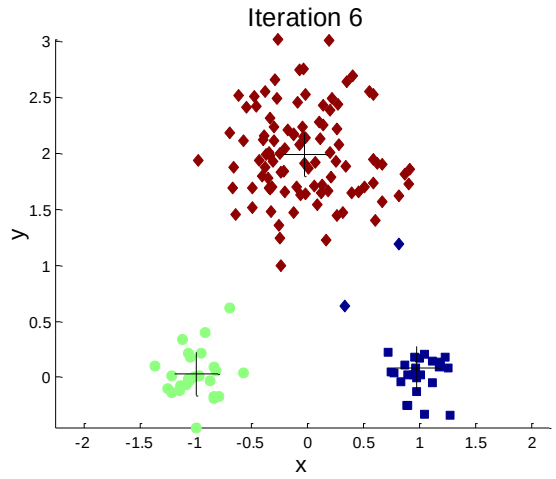




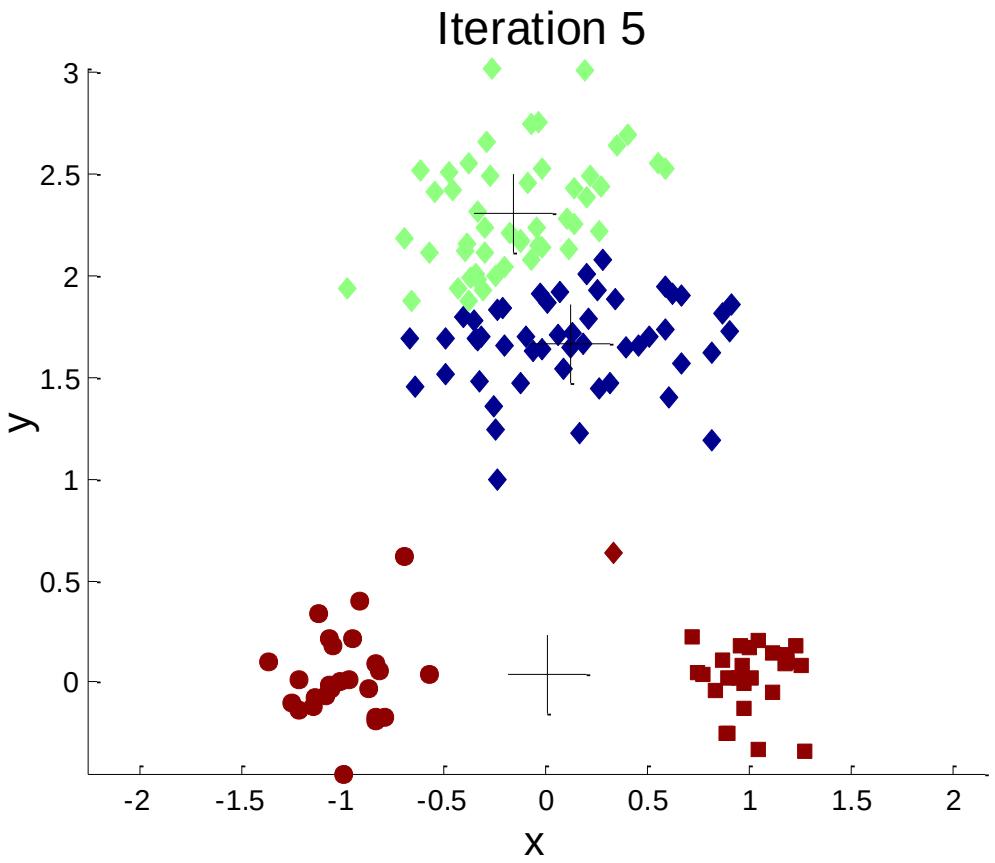




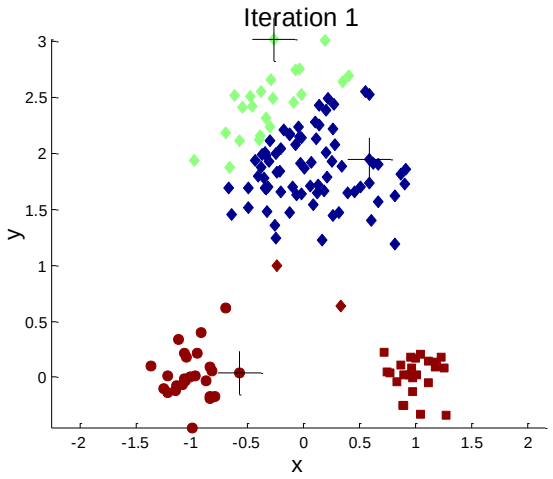


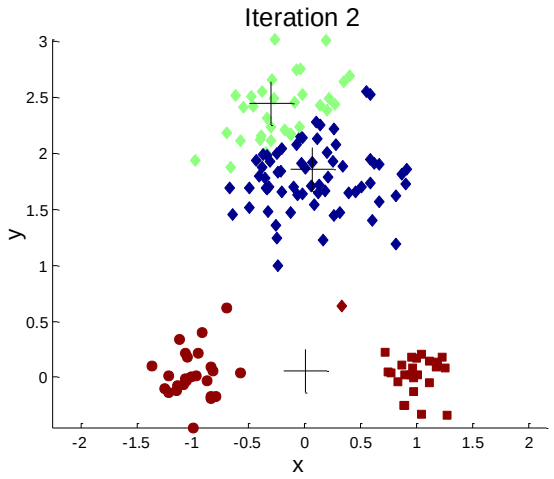


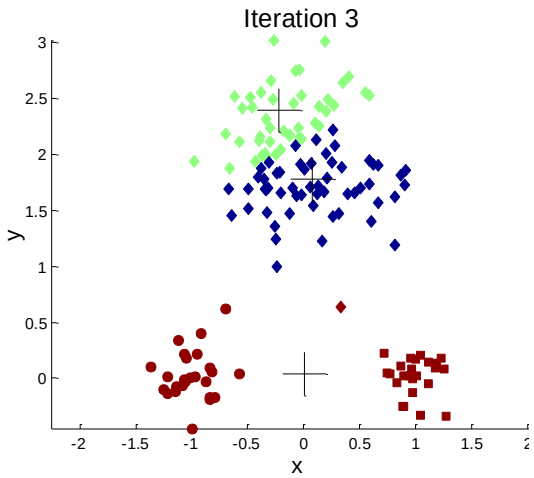
Importance of Choosing Initial Centroids …

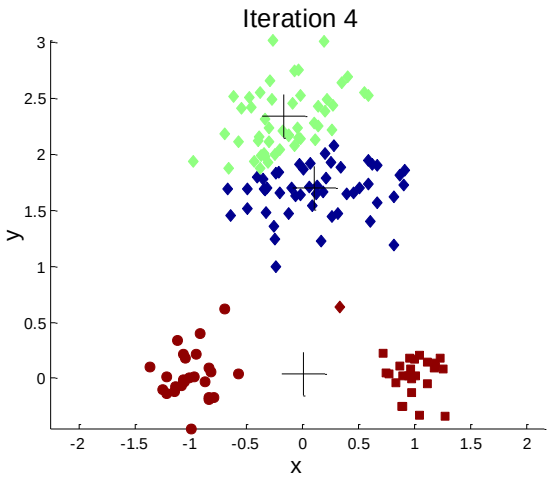


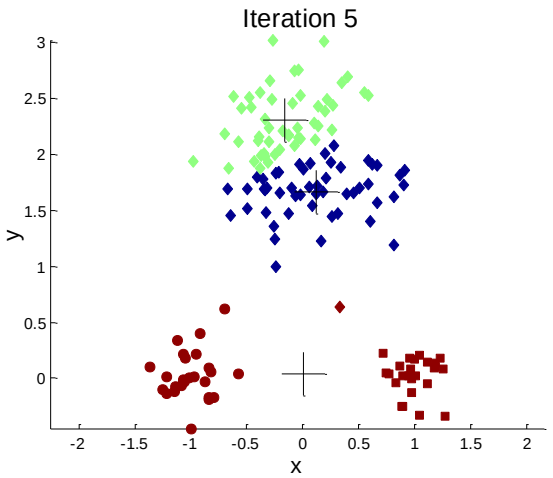
Importance of Choosing Initial Centroids …











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
  - 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
    - can show that m_i corresponds to the center (mean) of the cluster
  - Given two clusterings, we can choose the one with the smallest error.
  - One easy way to reduce SSE is to increase K, the number of clusters
    - A good clustering with smaller K can have a lower SSE than a poor clustering with higher K

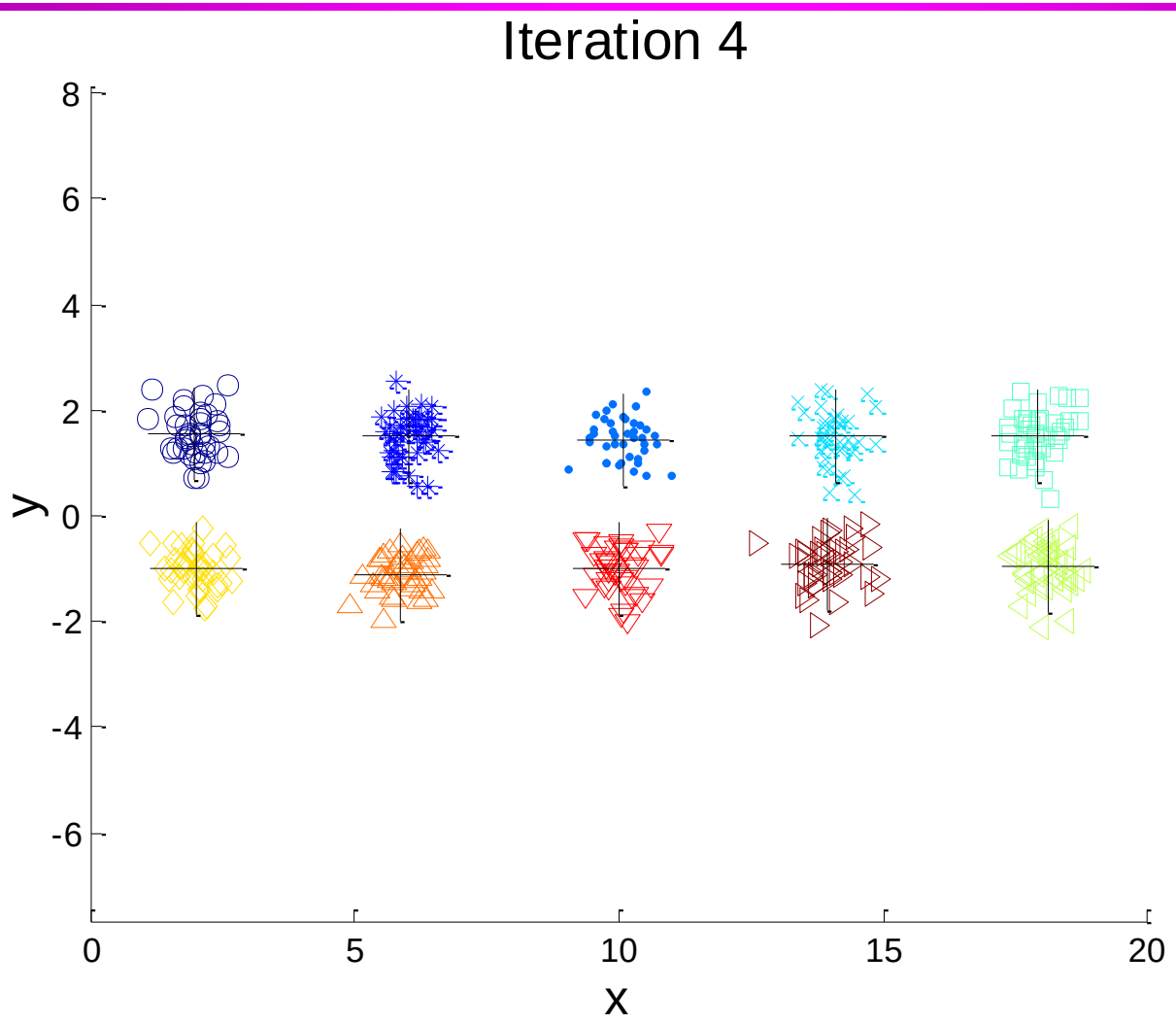
Problems with Selecting Initial Points

- If there are K 'real' clusters then the chance of selecting one centroid from each cluster is small.
  - Chance is relatively small when K is large
  - If clusters are the same size, n, then

$$P = \frac{\text{number of ways to select one centroid from each cluster}}{\text{number of ways to select } K \text{ centroids}} = \frac{K!n^K}{(Kn)^K} = \frac{K!}{K^K}$$

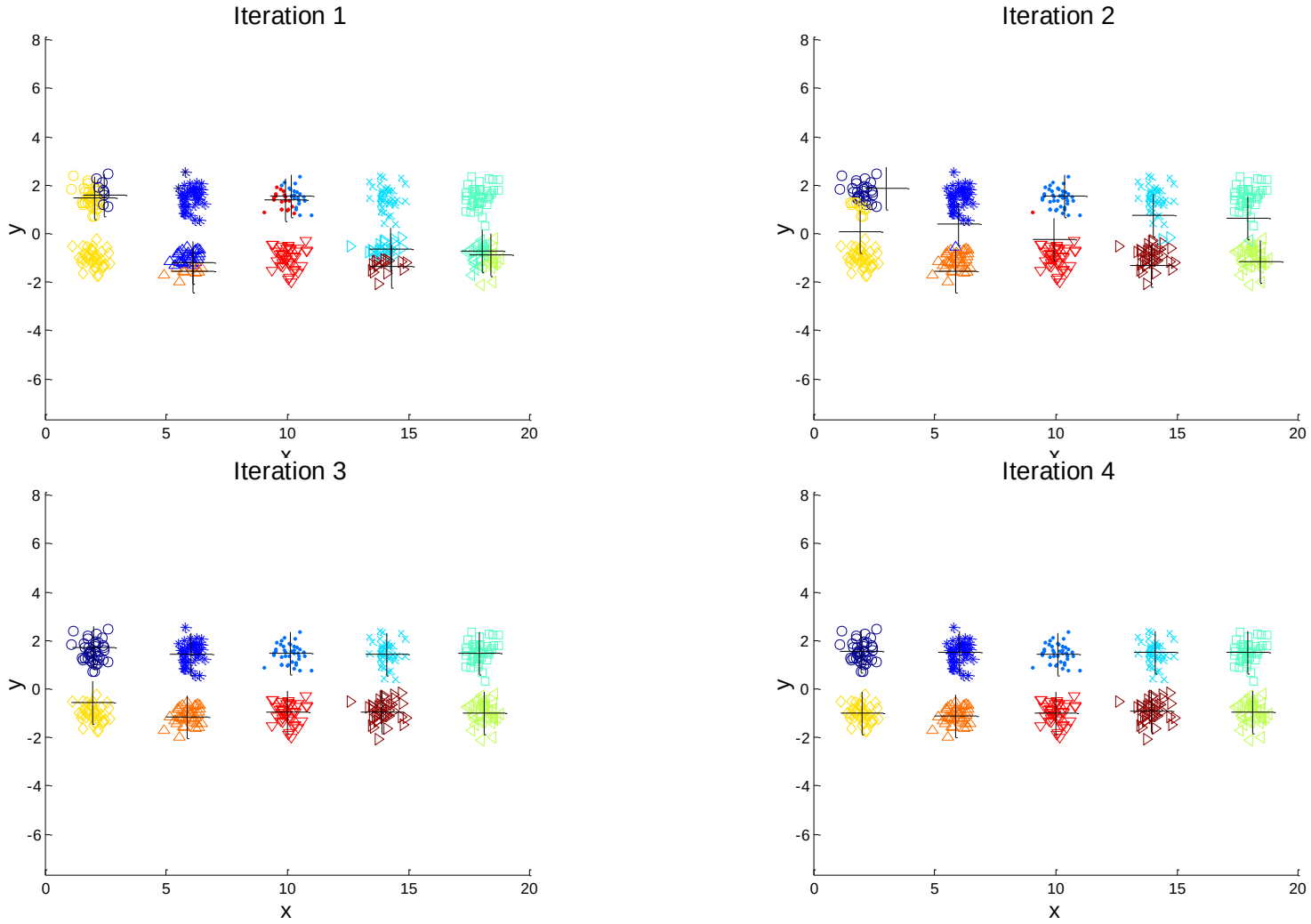
  - For example, if K = 10, then probability = $10!/10^{10} = 0.00036$
  - Sometimes the initial centroids will readjust themselves in 'right' way, and sometimes they don't
  - Consider an example of five pairs of clusters

10 Clusters Example



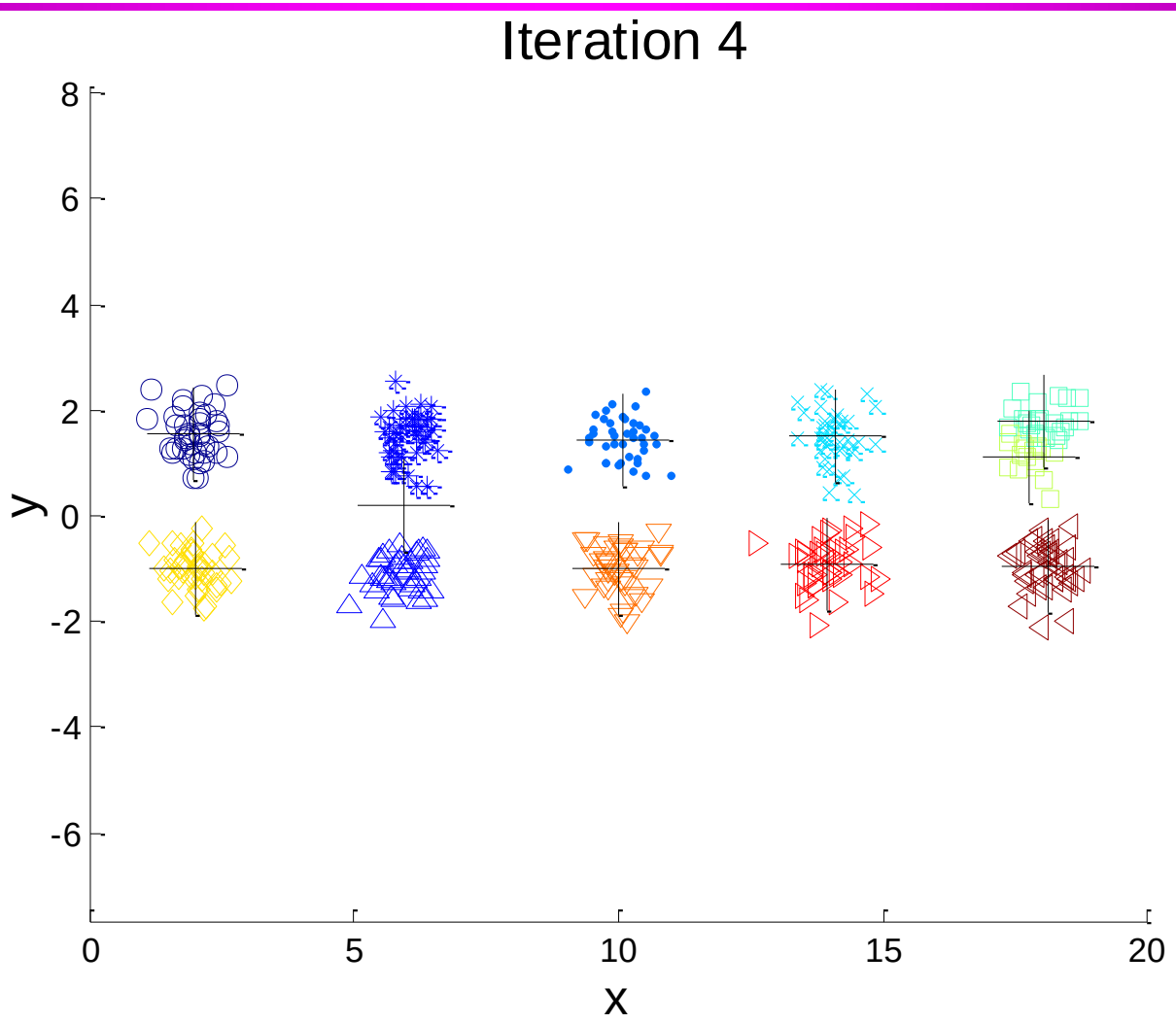
Starting with two initial centroids in one cluster of each pair of clusters

10 Clusters Example



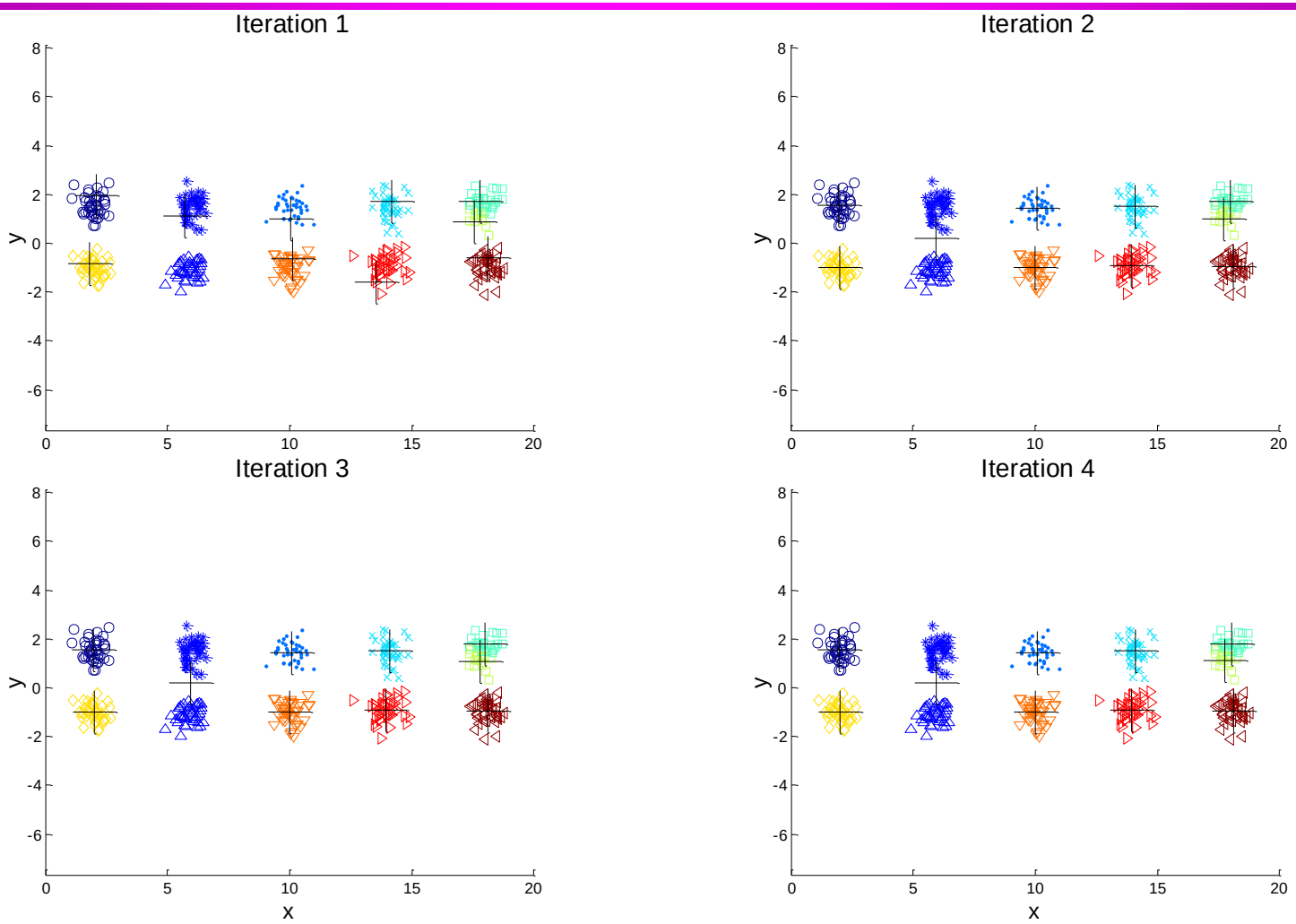
Starting with two initial centroids in one cluster of each pair of clusters

10 Clusters Example



Starting with some pairs of clusters having three initial centroids, while other have only one.

10 Clusters Example



Starting with some pairs of clusters having three initial centroids, while other have only one.

Solutions to Initial Centroids Problem

- Multiple runs
  - Helps, but probability is not on your side
- Sample and use hierarchical clustering to determine initial centroids
- Select more than k initial centroids and then select among these initial centroids
  - Select most widely separated
- Postprocessing
- Bisecting K-means
  - Not as susceptible to initialization issues

Handling Empty Clusters

- Basic K-means algorithm can yield empty clusters

- Several strategies to avoid empty clusters. Reassign the affected prototype by:
  - Choose the point that contributes most to SSE
  - Choose a point from the cluster with the highest SSE

  If there are several empty clusters, the above can be repeated several times.

Updating Centers Incrementally

- In the basic K-means algorithm, centroids are updated after all points are assigned to a centroid

- An alternative is to update the centroids after each assignment (incremental approach)
  - Each assignment updates zero or two centroids
  - More expensive
  - Introduces an order dependency
  - Never get an empty cluster
  - Can use “weights” to change the impact

Pre-processing and Post-processing

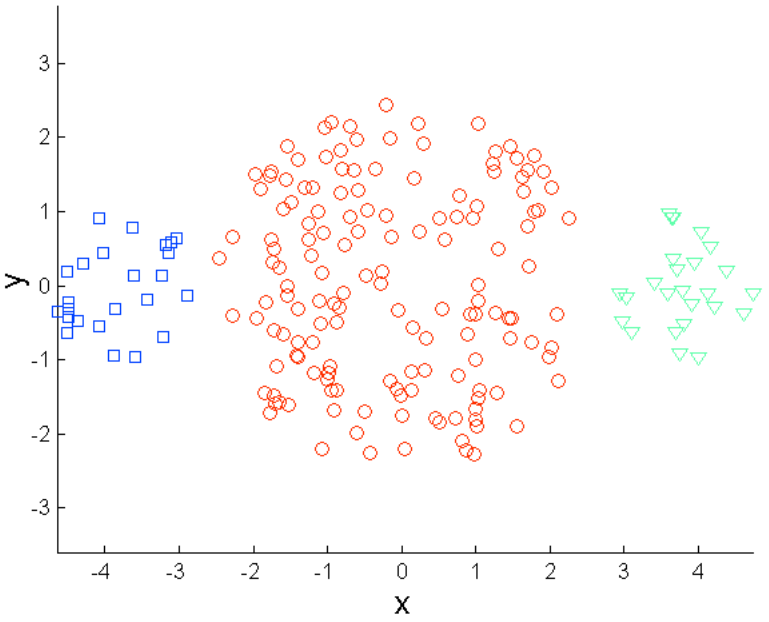
- Pre-processing
  - Normalize the data
  - Eliminate outliers
- Post-processing
  - Eliminate small clusters that may represent outliers
  - Split ‘loose’ clusters, i.e., clusters with relatively high SSE
  - Merge clusters that are ‘close’ and that have relatively low SSE
  - Can use these steps during the clustering process.

Limitations of K-means

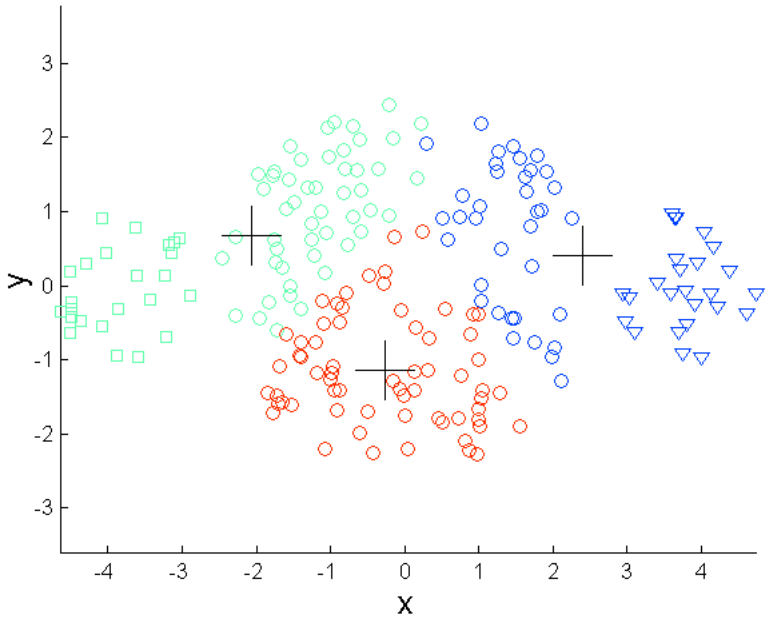
- K-means has problems when clusters are of differing
  - Sizes
  - Densities
  - Non-globular shapes

- K-means has problems when the data contains outliers.

Limitations of K-means: Differing Sizes

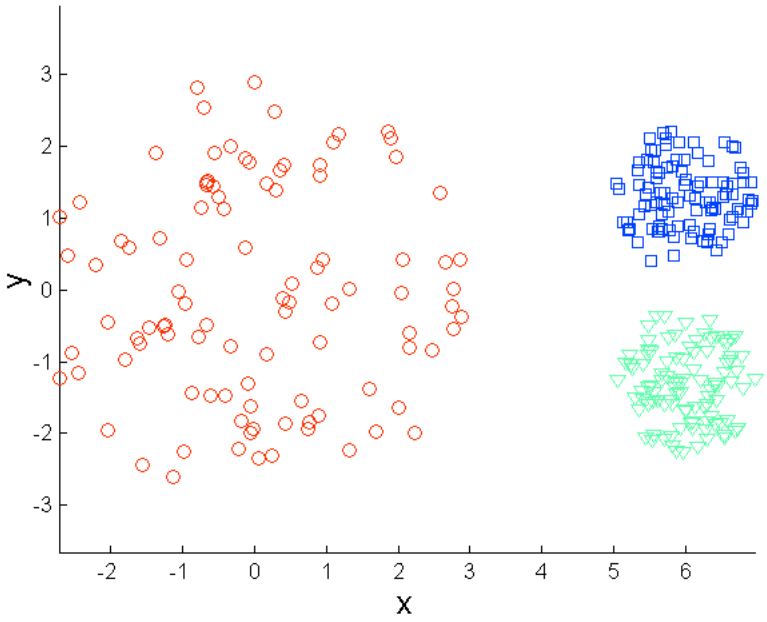


Original Points

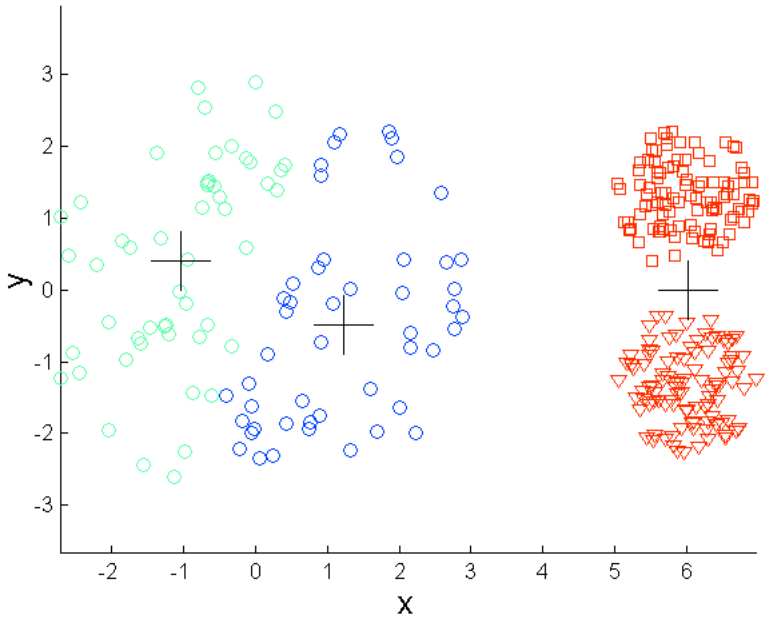


K-means (3 Clusters)

Limitations of K-means: Differing Density

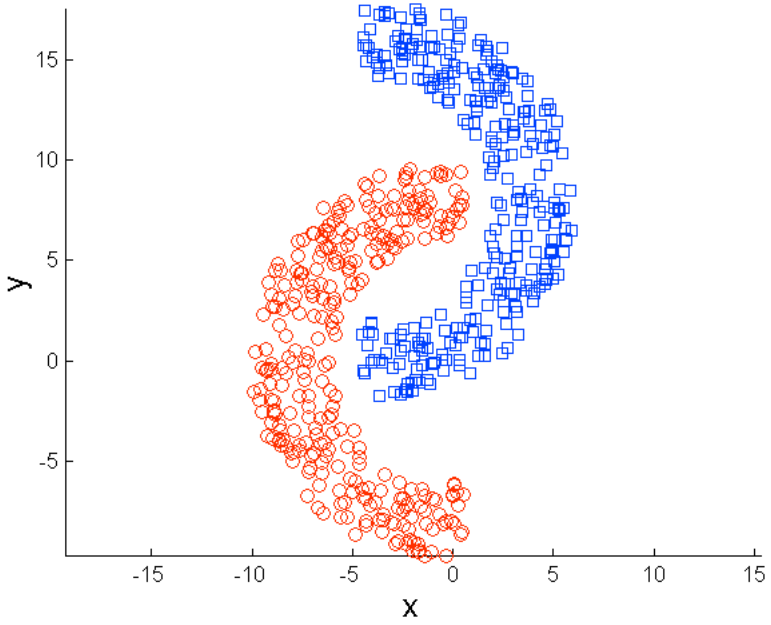


Original Points

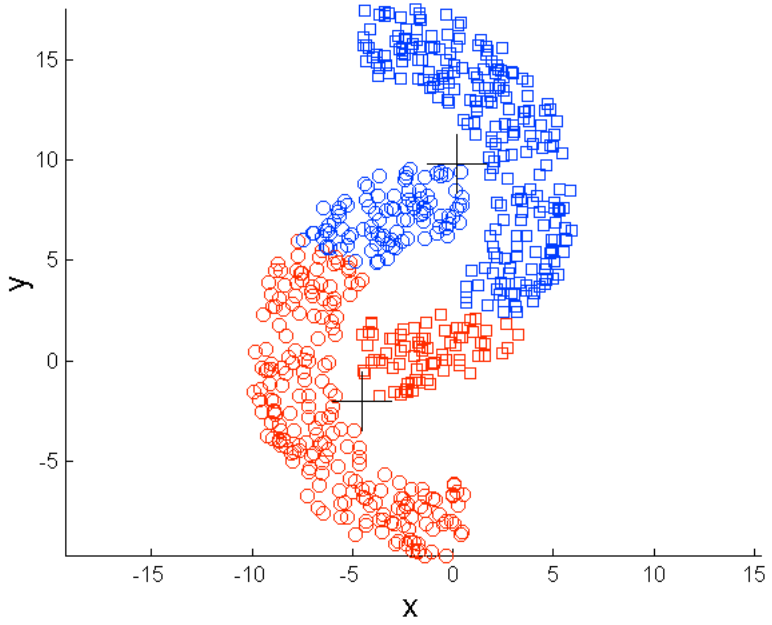


K-means (3 Clusters)

Limitations of K-means: Non-globular Shapes

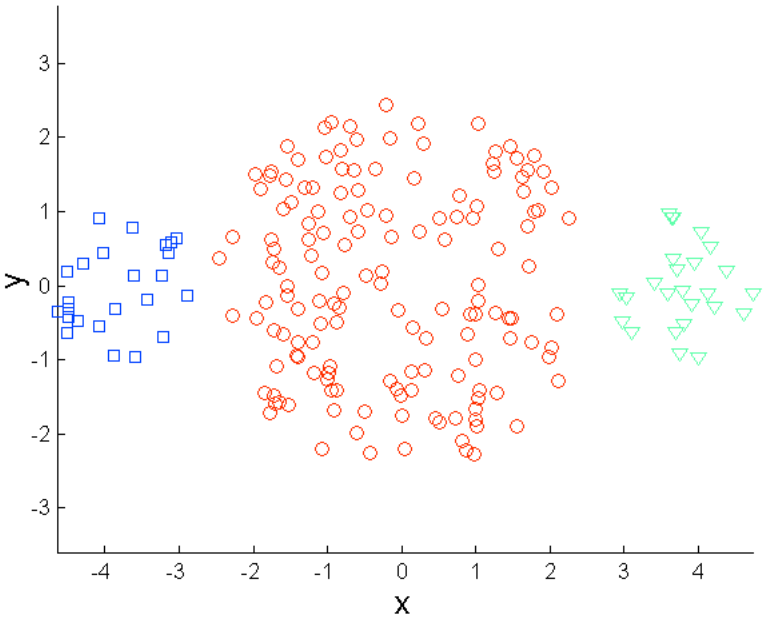


Original Points

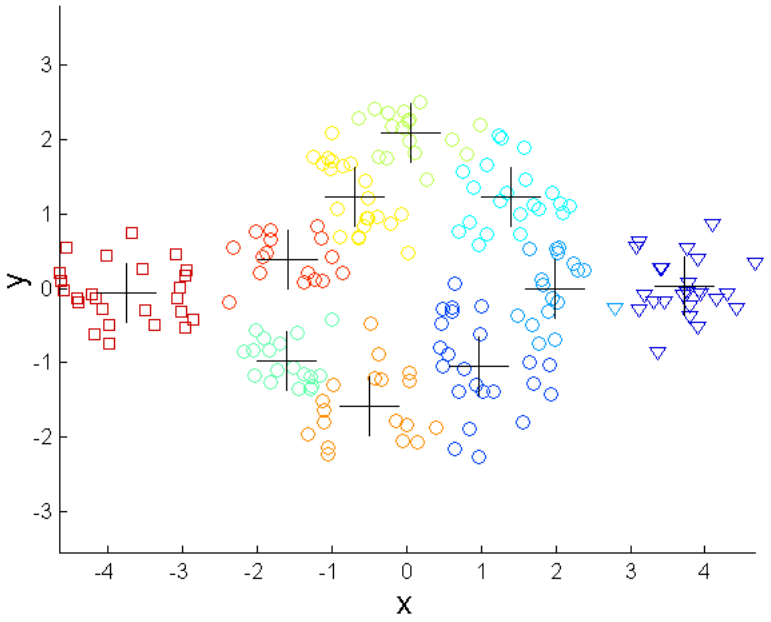


K-means (2 Clusters)

Overcoming K-means Limitations



Original Points

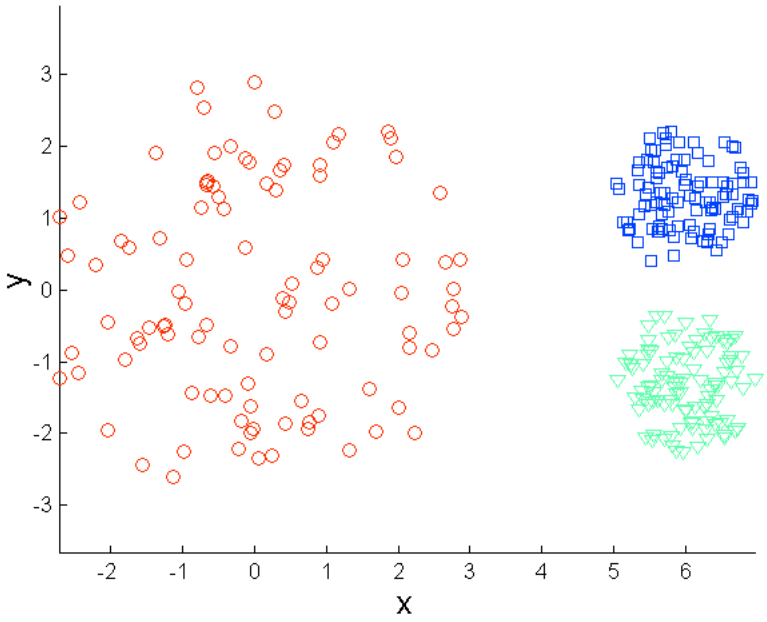


K-means Clusters

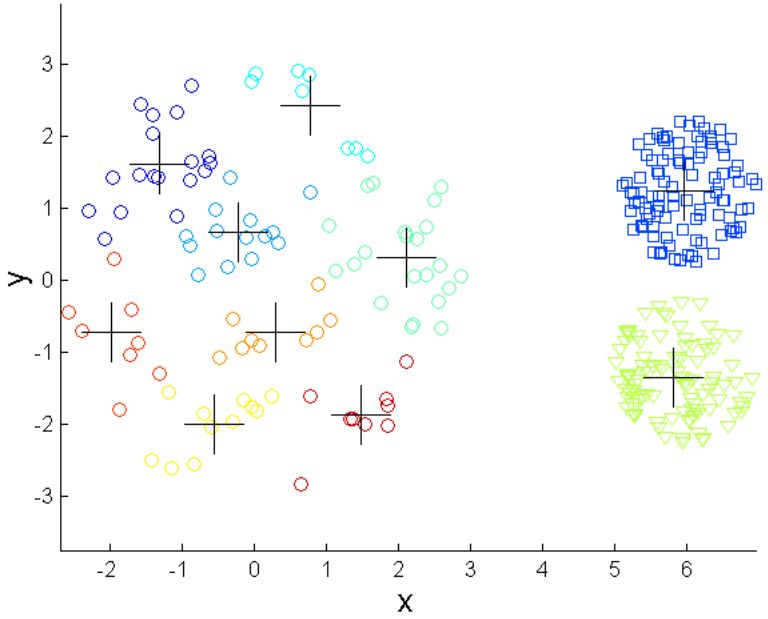
One solution is to use many clusters.

Find parts of clusters, but need to put together.

Overcoming K-means Limitations

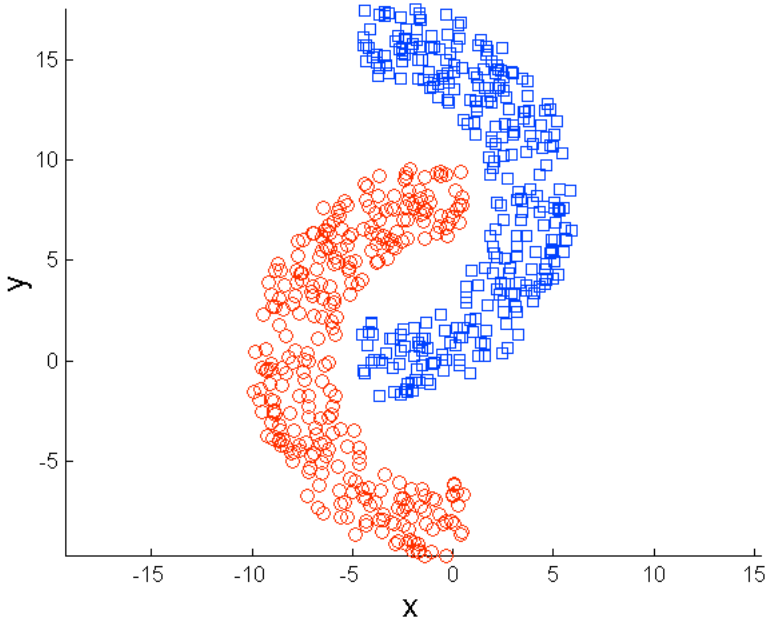


Original Points

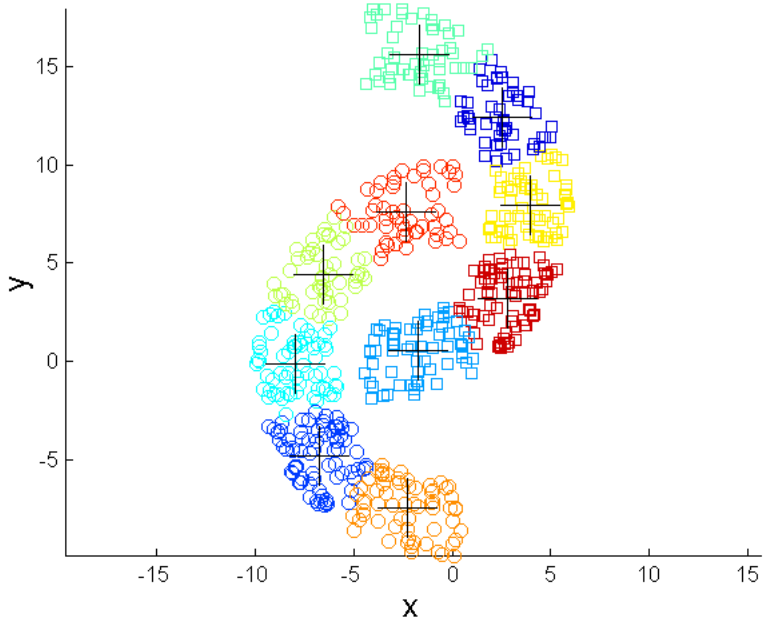


K-means Clusters

Overcoming K-means Limitations



Original Points



K-means Clusters

Self-Organizing Maps

The Self-Organizing Map (SOM):

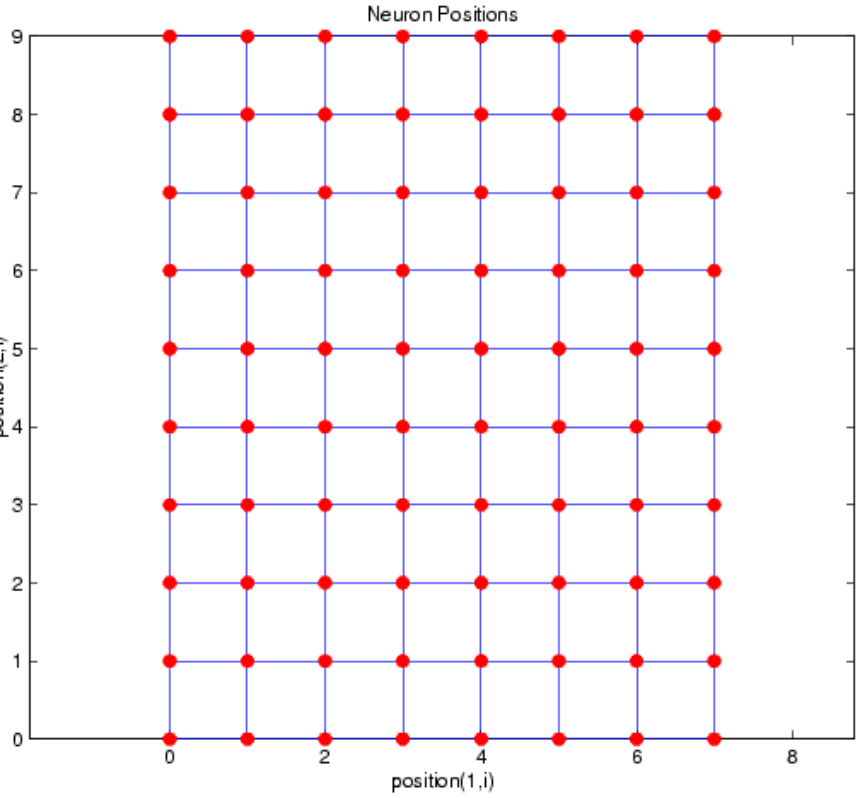
- Developed by T.Kohonen in ‘86
- A very fameous unsupervised machine learning method.
- Perform a topology preserving feature mapping.
- A neural network popularly used for cluster analysis, dimension reduction, and **visualization** (of high dimensional data).
- The SOM algorithm is neurobiologically inspired, incorporating all the mechanisms that are basic to self-organization: competition, cooperation, and self-amplification.
- The Kohonen’s SOM algorithm is very simple to implement, yet mathematically it is very difficult to analyze its properties in a general setting.

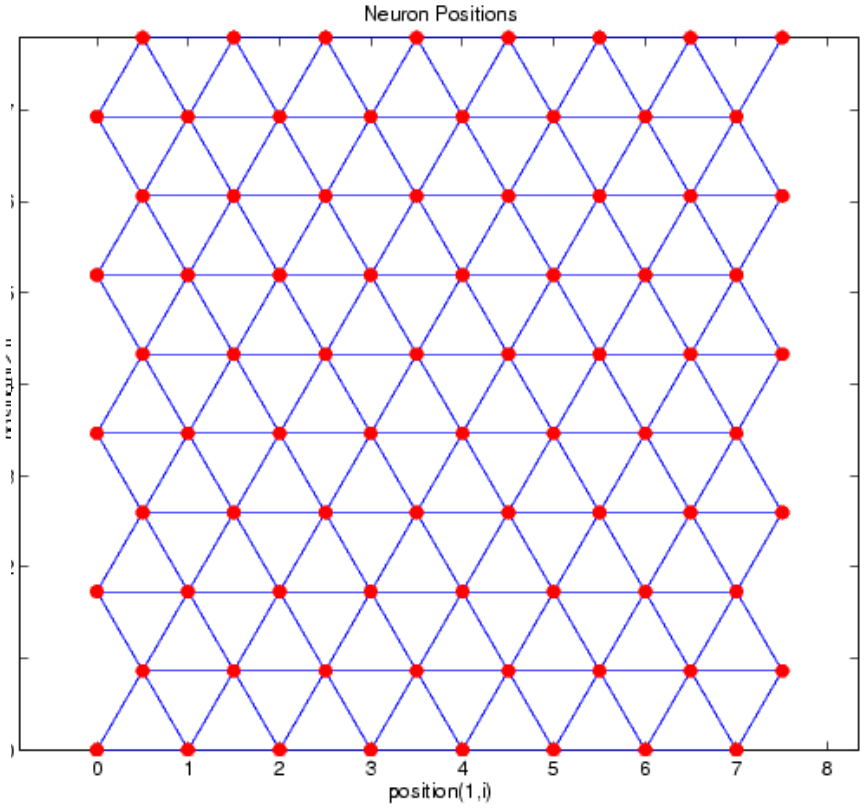
Self-Organizing Maps

- A SOM consists of a number of **neurons**.
  - The neurons are organized on a regular n-dimensional grid. Very common is n=2.
- We also specify a topology:
  - Topology defines the geometric relationships between the neurons. Common relationships are rectangular or hexagonal.
- Associated with each neuron is a codebook vector.
  - Codebooks are of the same dimension as the input data.
- A learning algorithm learns a mapping from the high dimensional input space of the data points onto the codebooks in the 2D grid
  - Such that similar samples are mapped in close proximity; dissimilar samples far from each other.

Self-Organizing Maps

Example: A 2D SOM of size 8x10 neurons using a rectangular topology (left), and hexagonal topology (right).





Training Self-Organizing Maps

The codebook vectors are initialized using random values from within the (input) feature data space.

The training algorithm of the self-organizing map consists of two steps:

1. Competitive step
2. Cooperative step

These two steps are repeated for a number of iterations.

Self-Organizing Maps

Competitive step:

Given codebooks $\boldsymbol{w}_j = [w_{j1}, w_{j2}, \ldots, w_{jm}]^T$, where $j = 1,2,\ldots,l$, l is the total number of neurons in the network, and m is the dimension of the input data, T is the transpose operator.

- Select one input vector from a data point $\boldsymbol{x} = [x_1, x_2, \ldots, x_m]^T$ where m is the dimension of $\boldsymbol{x}$.
- Find the best matching codebook

$$i^* = \arg\min_j \|\boldsymbol{x} - \boldsymbol{w}_j\| \quad ^*$$

, where i^* is the index of the best matching codebook, and is said to be the winner for $\boldsymbol{x}$.

* The Euclidean distance meassure is used here. Other distance meassures can be used.

Self-Organizing Maps

Cooperative step: The winner and all of its neighbours are updated:

$$\Delta w_{ij} = \eta \Lambda(i, i^*)(x_j - w_{ij})$$

, where

$$\Lambda(i, i^*) = \exp(-|r_i - r_{i^*}|^2 / 2\sigma^2)$$

,η is a learning rate, and σ is a neighborhood radius. η is a positive float value smaller than 1, whereas σ must always be larger than 1. r_i is the geographic location (the coordinates on the map) of the i-th codebook vector.

SOM example (1)

TABLE 9.3 Animal Names and Their Attributes

Animal		Dove	Hen	Duck	Goose	Owl	Hawk	Eagle	Fox	Dog	Wolf	Cat	Tiger	Lion	Horse	Zebra	Cow
is	small	1	1	1	1	1	1	0	0	0	0	1	0	0	0	0	0
	medium	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0
	big	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1
has	2 legs	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
	4 legs	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
	hair	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
	hooves	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
	mane	0	0	0	0	0	0	0	0	0	1	0	0	1	1	1	0
	feathers	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
likes to	hunt	0	0	0	0	1	1	1	1	0	1	1	1	1	0	0	0
	run	0	0	0	0	0	0	0	0	1	1	0	1	1	1	1	0
	fly	1	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0
	swim	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0

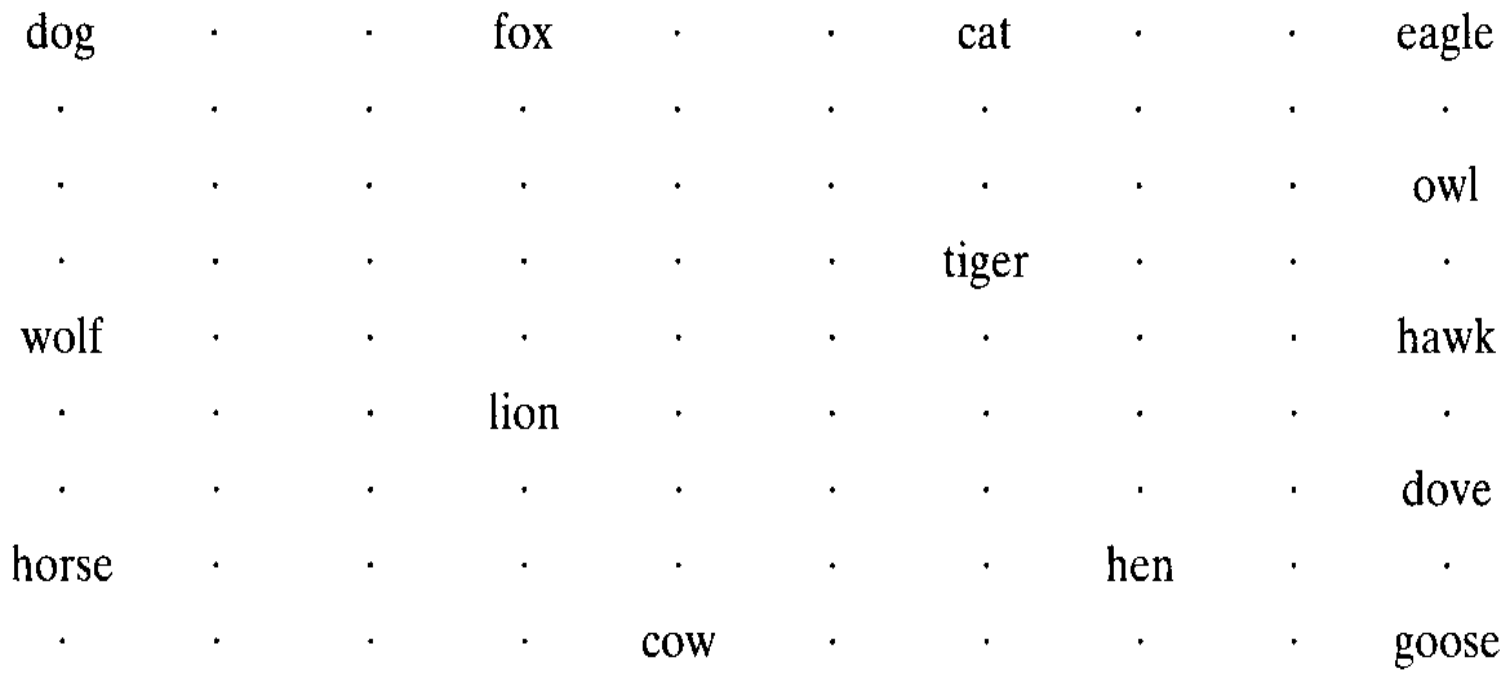
SOM example (2)

Thus, the input domain consists of 16 animals which are described by a 13-dimensional binary feature vector.

$\Rightarrow$ The codebook vectors will be of the same dimension.

- Assume we are training a SOM of size 10 x 10 whose neurons have a rectangular relationship, then the result may look as follows:

SOM example (3)



Feature map: Find for each input sample the best codebook then label the codebook by the sample.

Illustration from Kohonen's book

SOM example (4)

dog	dog	fox	fox	fox	cat	cat	cat	eagle	eagle
dog	dog	fox	fox	fox	cat	cat	cat	eagle	eagle
wolf	wolf	wolf	fox	cat	tiger	tiger	tiger	owl	owl
wolf	wolf	lion	lion	lion	tiger	tiger	tiger	hawk	hawk
wolf	wolf	lion	lion	lion	tiger	tiger	tiger	hawk	hawk
wolf	wolf	lion	lion	lion	owl	dove	hawk	dove	dove
horse	horse	lion	lion	lion	dove	hen	hen	dove	dove
horse	horse	zebra	cow	cow	cow	hen	hen	dove	dove
zebra	zebra	zebra	cow	cow	cow	hen	hen	duck	goose
zebra	zebra	zebra	cow	cow	cow	duck	duck	duck	goose

Semantic map: Find for each codebook the best matching input sample, label codebook accordingly.

Illustration from Kohonen's book

Self-Organizing Maps

- SOMs are a massive parallel systems which can (once trained) map data in $\log_2(m)$ time!
- Hence, SOMs are very popular in many data mining exercises.
- High dimensional input data is mapped onto a 2-dimensional grid (dimension reduction)
- Since SOMs perform a topology preserving mapping, and hence, SOMs are also a useful tool for knowledge discovery tasks.
- Note: SOMs do not identify or extract clusters!
  - Apply k-means, DBScan, or other clustering algorithm to the mappings of the SOM in order to extract clusters.

Self-Organizing Maps

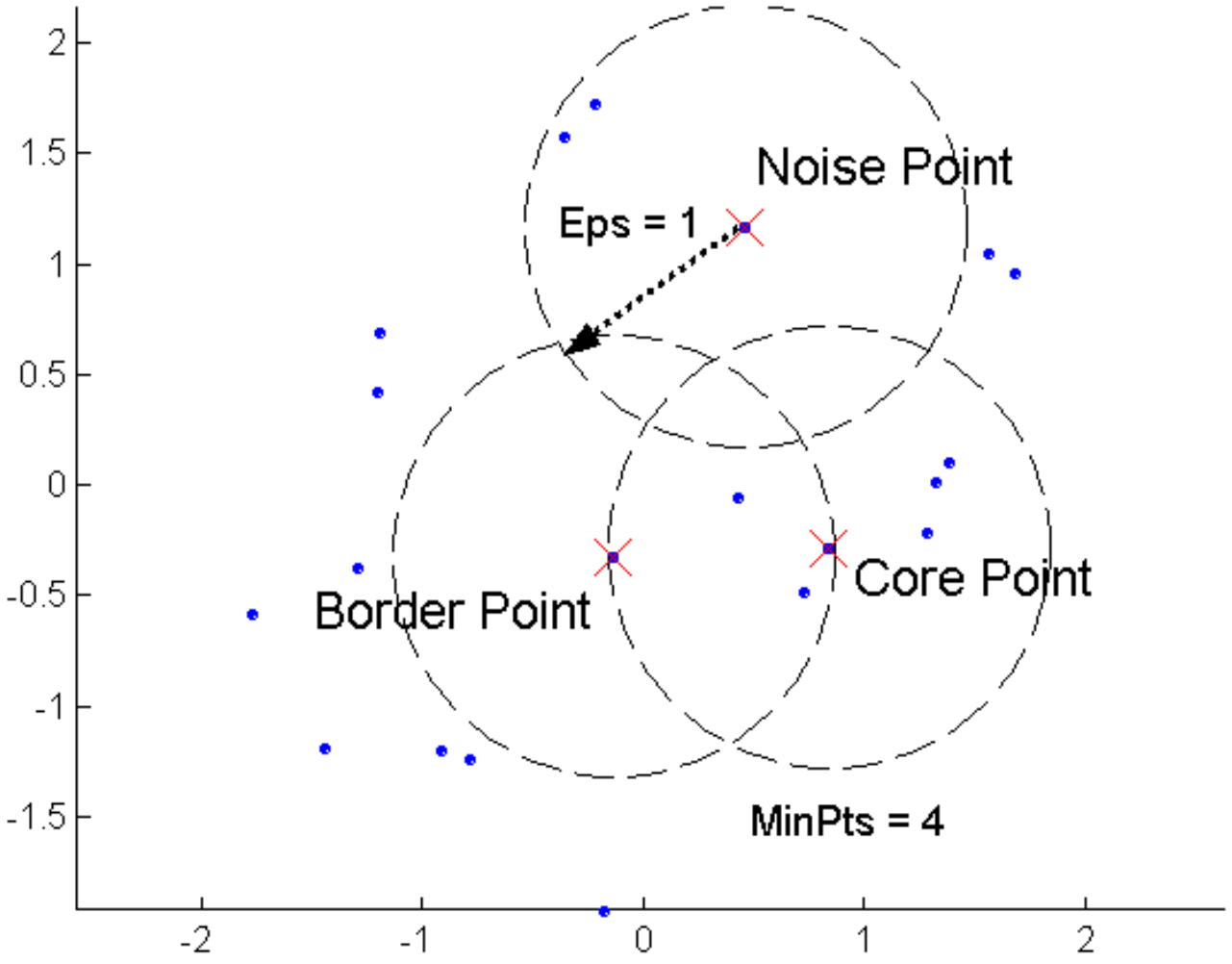
We have seen that SOMs map data which are similar to each other in the input space to nearby areas in the display space.

- Thus, the goal of the SOM (and in fact of all clustering methods) is to group together “similar” data – but what does this mean?
- No single answer – it depends on what we want to find or emphasize in the data; this is one reason why clustering is an “art”
- The similarity measure is often more important than the clustering algorithm used – don’t overlook this choice!

DBSCAN clustering algorithm

- 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 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



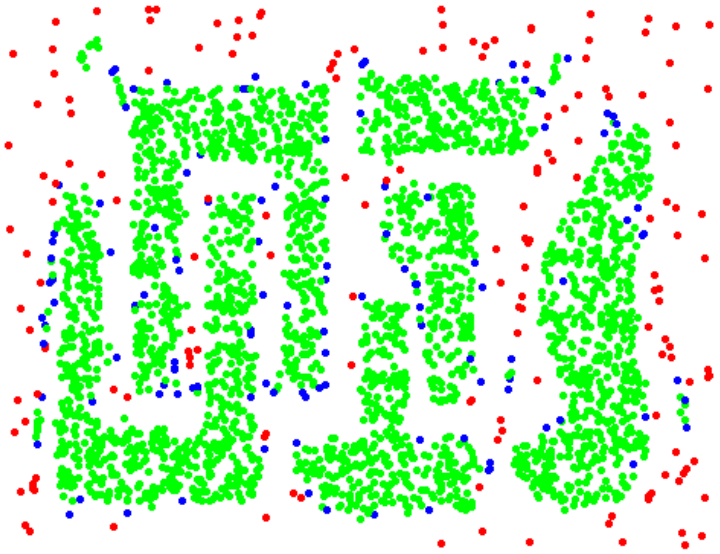
DBSCAN Clustering Algorithm

- Eliminate noise points
- Perform clustering on the remaining points

```
current_cluster_label ← 1
for all core points do
  if the core point has no cluster label then
    current_cluster_label ← current_cluster_label + 1
    Label the current core point with cluster label current_cluster_label
  end if
  for all points in the Eps-neighborhood, except i^th the point itself do
    if the point does not have a cluster label then
      Label the point with cluster label current_cluster_label
    end if
  end for
end for
```

DBSCAN: Core, Border and Noise Points

Original Points

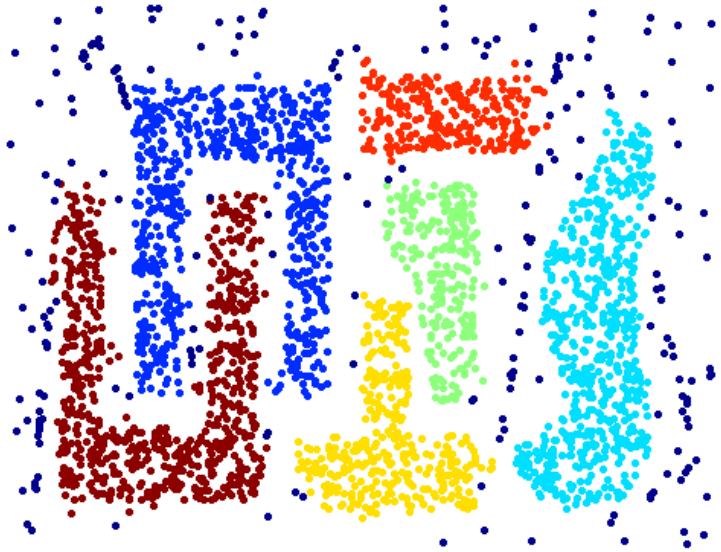


DBScan Point types:
core, border and
noise

Eps = 10, MinPts = 4

When DBSCAN Works Well

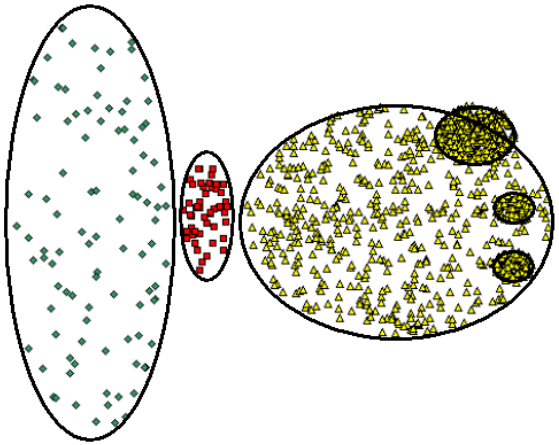
Original Points



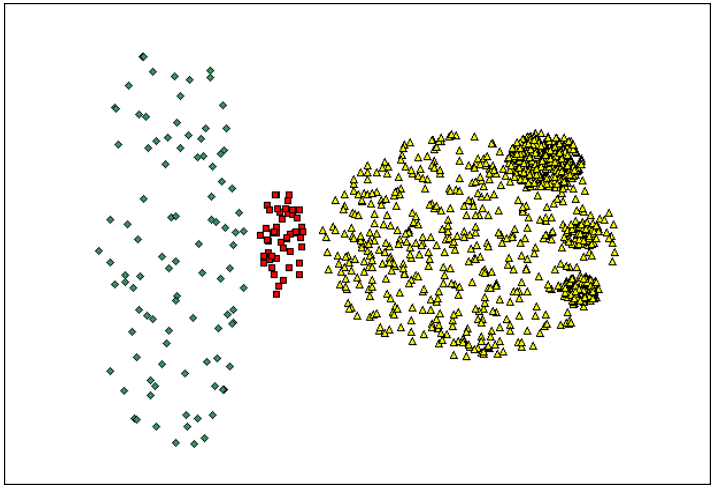
DBScan Clusters

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

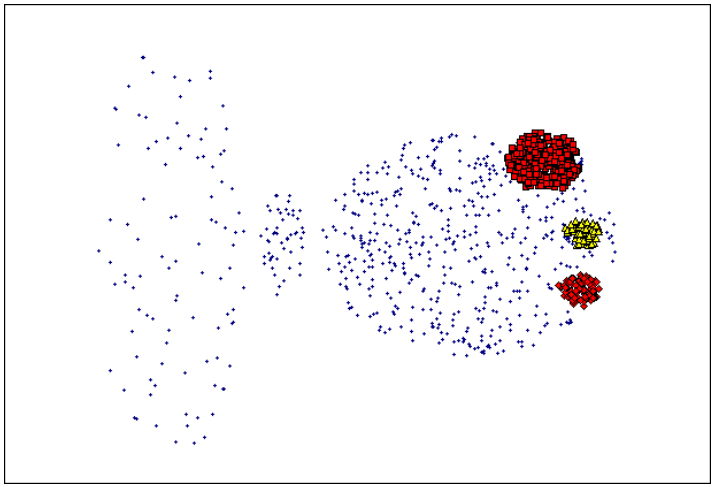
When DBSCAN Does NOT Work Well



Original Points



(MinPts=4, Eps=9.75).

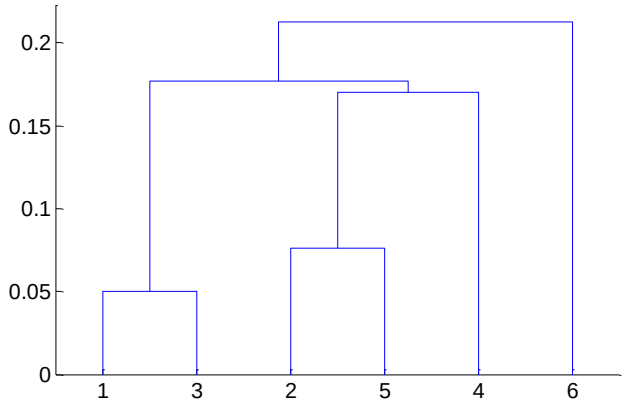


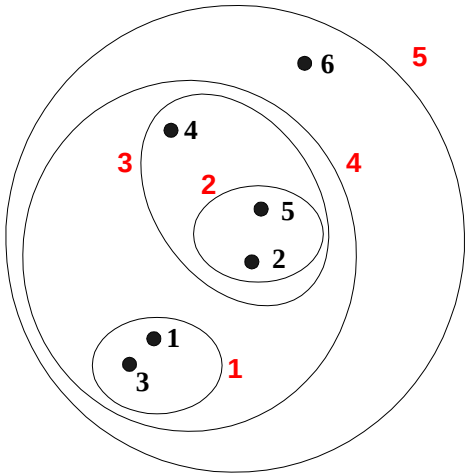
(MinPts=4, Eps=9.92)

- **Varying densities**
- **sparse and high-dimensional data**

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





Strengths of Hierarchical Clustering

- Do not have to assume any particular number of clusters
  - Any desired number of clusters can be obtained by ‘cutting’ the dendrogram at the proper level

- They may correspond to meaningful taxonomies
  - Example in biological sciences (e.g., animal kingdom, phylogeny reconstruction, …)

Hierarchical Clustering

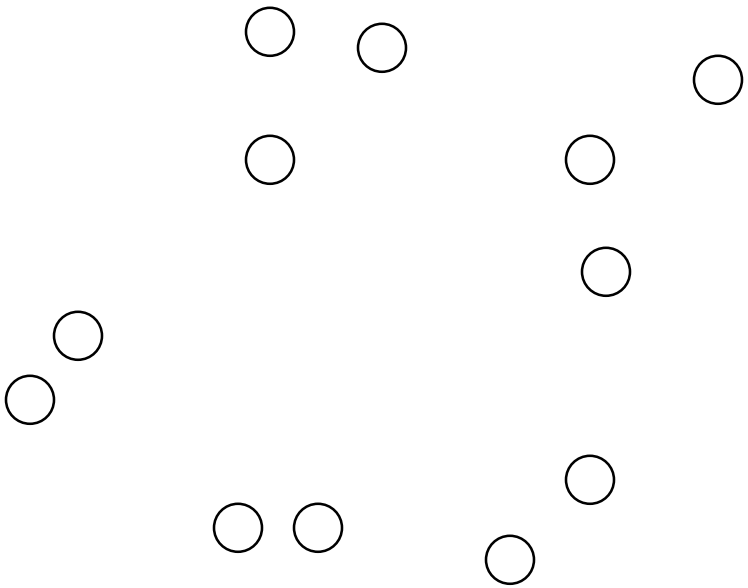
- Two main types of hierarchical clustering
  - Agglomerative:
    - Start with the points as individual clusters
    - At each step, merge the closest pair of clusters until only one cluster (or k clusters) left
  - Divisive:
    - Start with one, all-inclusive cluster
    - At each step, split a cluster until each cluster contains a point (or there are k clusters)
- Traditional hierarchical algorithms use a similarity or distance matrix
  - Merge or split one cluster at a time

Agglomerative 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

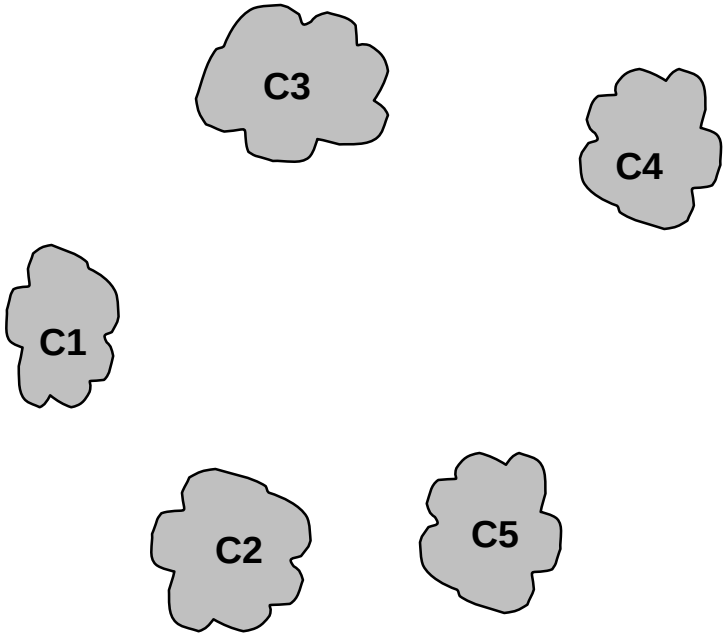


	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Proximity Matrix

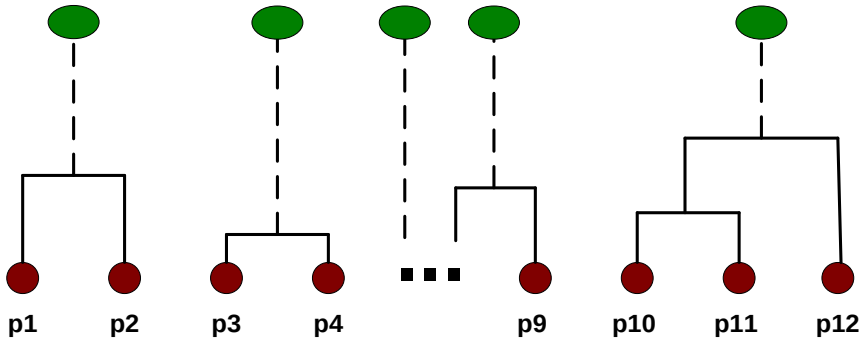
Intermediate Situation

- After some merging steps, we have some clusters



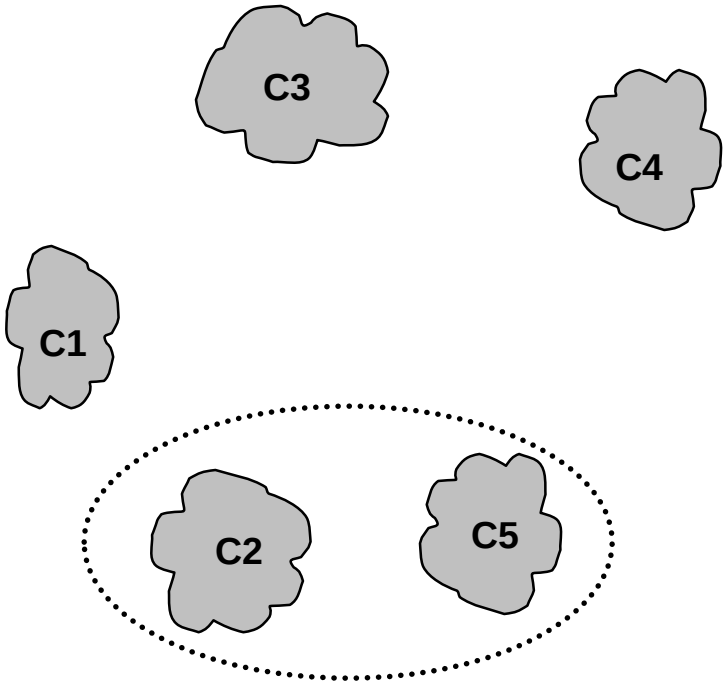
	C1	C2	C3	C4	C5
C1					
C2					
C3					
C4					
C5					

Proximity Matrix



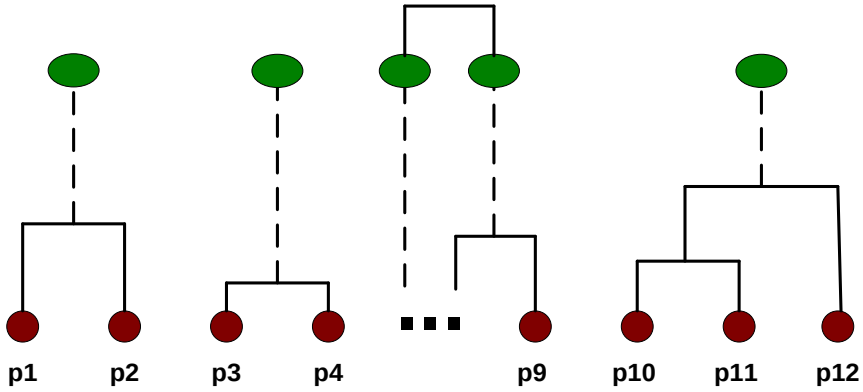
Intermediate Situation

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



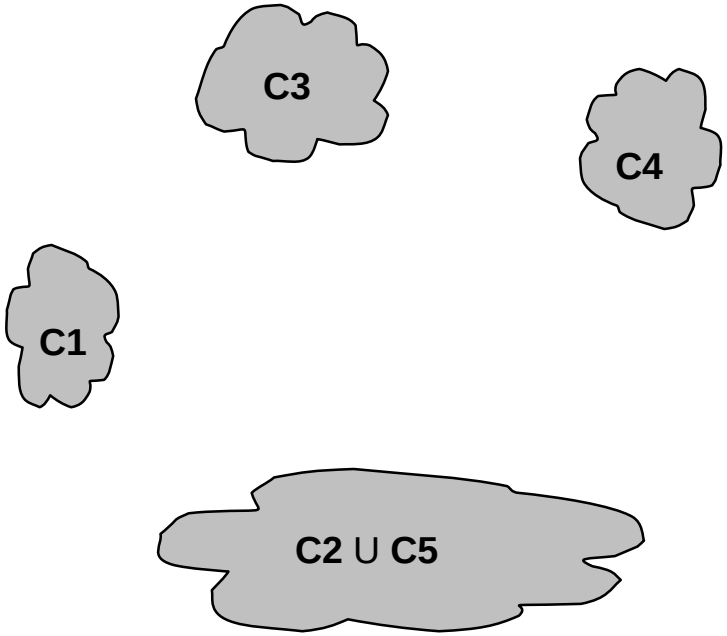
	C1	C2	C3	C4	C5
C1					
C2					
C3					
C4					
C5					

Proximity Matrix



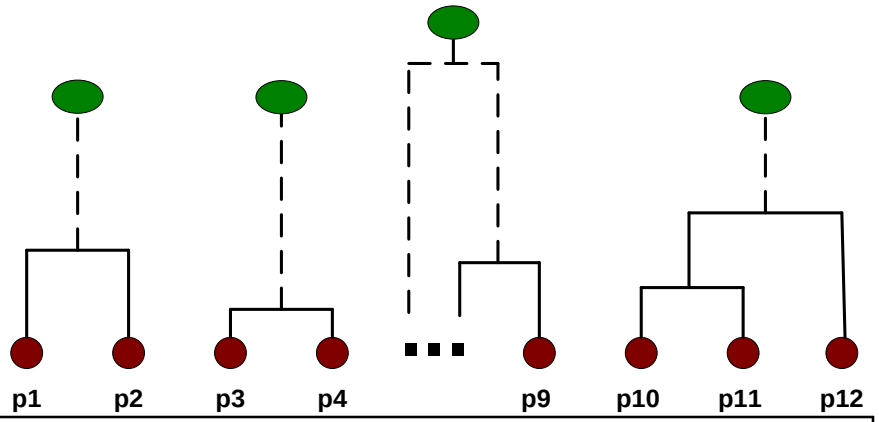
After Merging

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

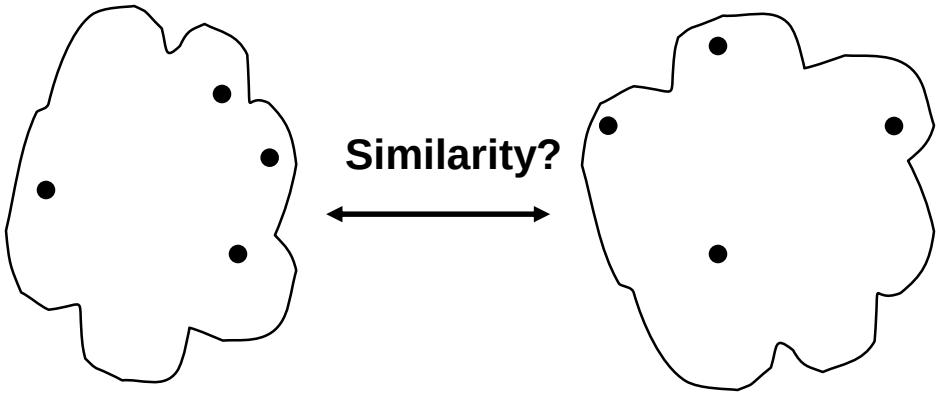


	C1	C2 U C5	C3	C4
C1		?		
C2 U C5	?	?	?	?
C3		?		
C4		?		

Proximity Matrix



How to Define Inter-Cluster Similarity

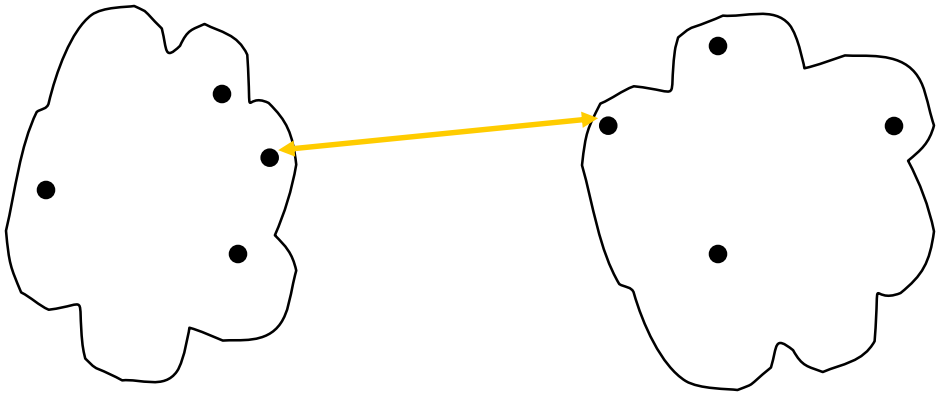


	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						

Proximity Matrix

- MIN
- MAX
- Group Average
- Distance Between Centroids
- Other methods driven by an objective function
  - Ward's Method uses squared error

How to Define Inter-Cluster Similarity

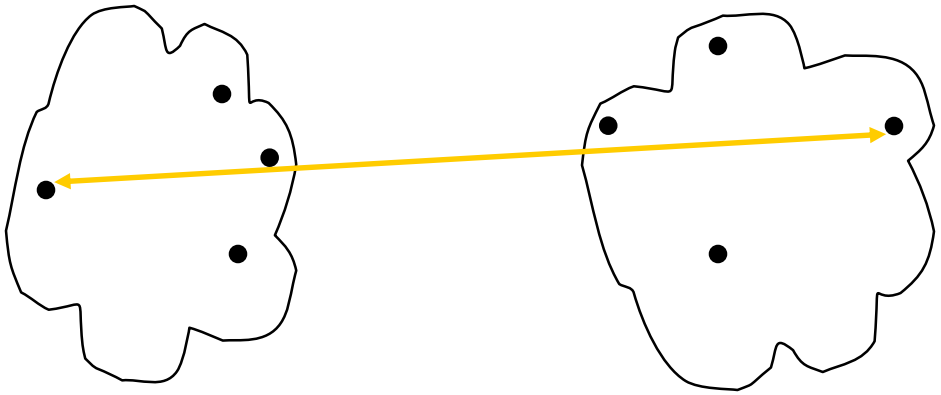


	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Proximity Matrix

- MIN
- MAX
- Group Average
- Distance Between Centroids
- Other methods driven by an objective function
  - Ward's Method uses squared error

How to Define Inter-Cluster Similarity

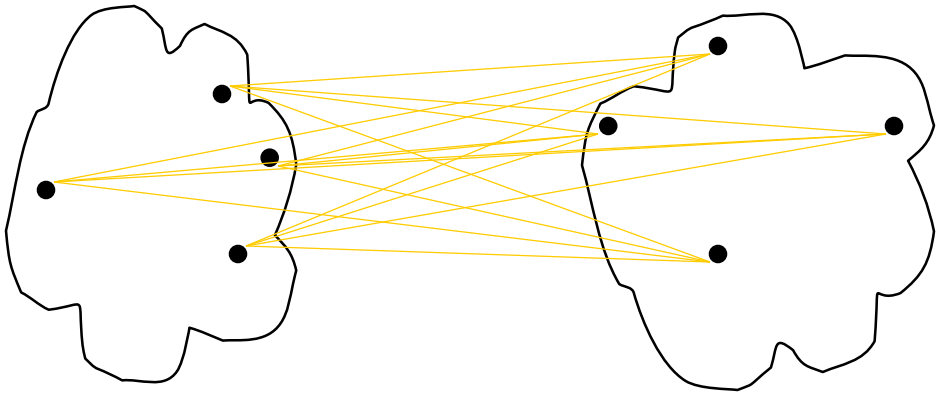


	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Proximity Matrix

- MIN
- MAX
- Group Average
- Distance Between Centroids
- Other methods driven by an objective function
  - Ward's Method uses squared error

How to Define Inter-Cluster Similarity

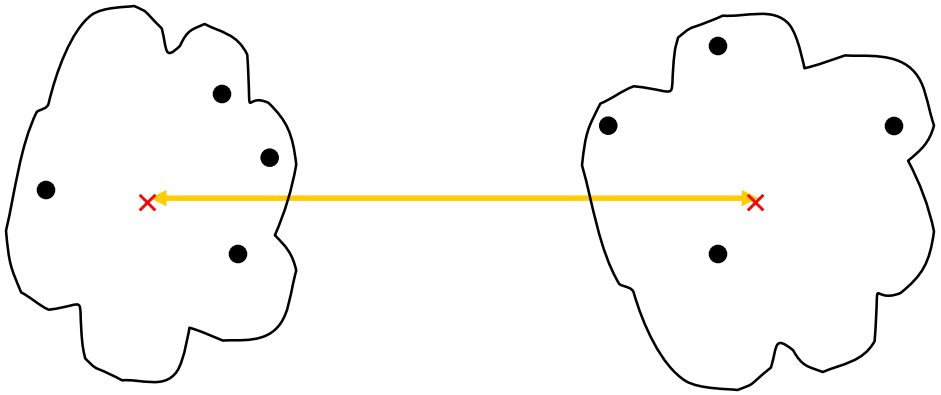


	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Proximity Matrix

- MIN
- MAX
- Group Average
- Distance Between Centroids
- Other methods driven by an objective function
  - Ward's Method uses squared error

How to Define Inter-Cluster Similarity

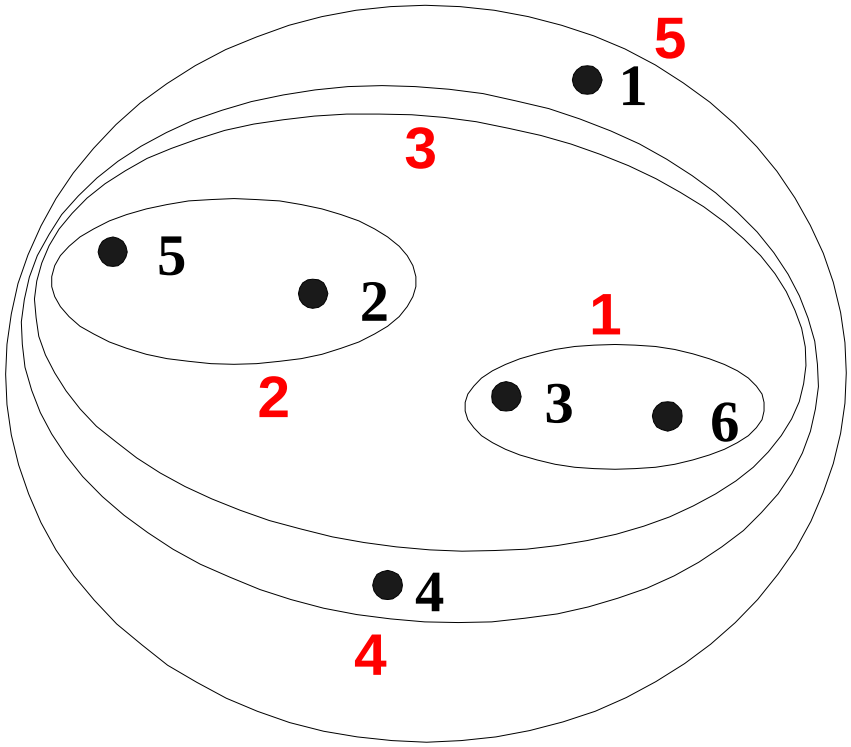


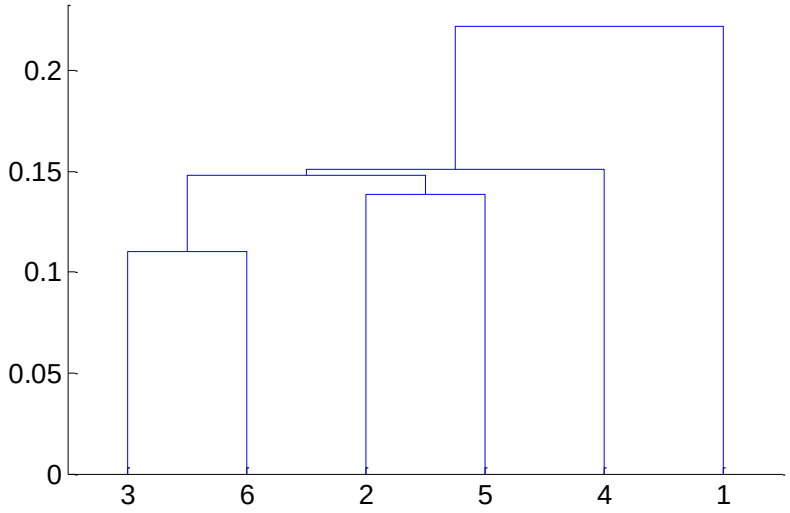
	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Proximity Matrix

- MIN
- MAX
- Group Average
- Distance Between Centroids
- Other methods driven by an objective function
  - Ward's Method uses squared error

Hierarchical Clustering: MIN

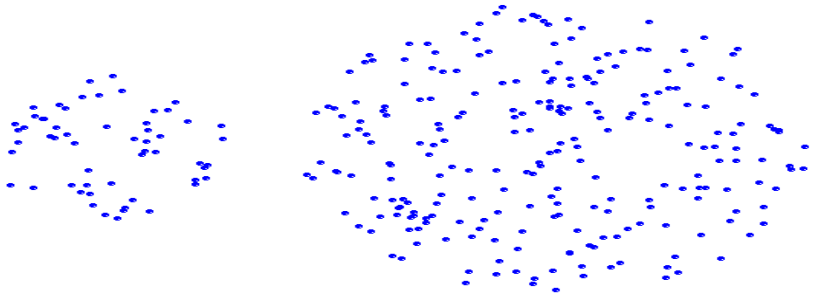




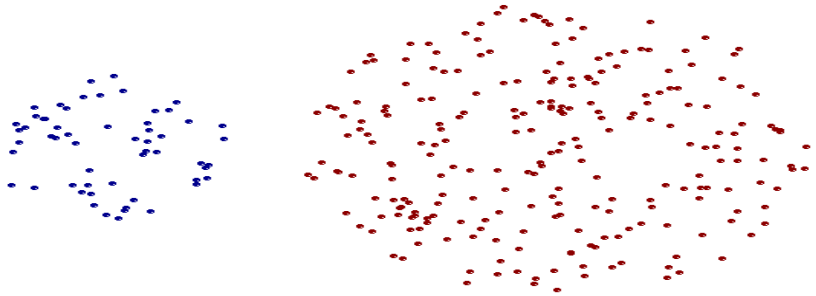
Nested Clusters

Dendrogram

Strength of MIN



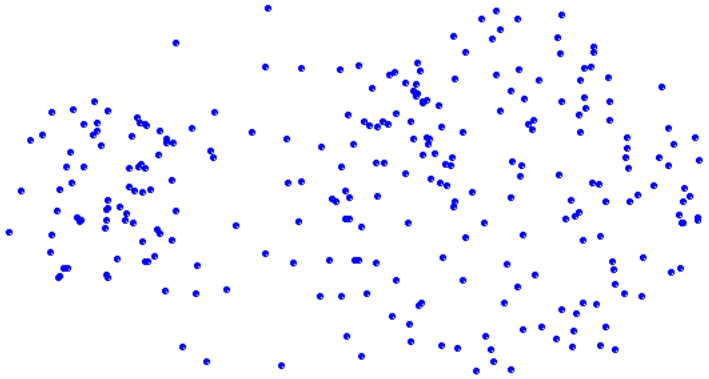
Original Points



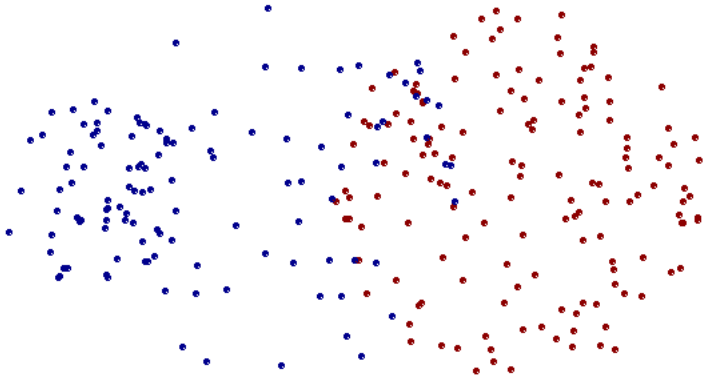
Two Clusters

- Can handle non-elliptical shapes

Limitations of MIN



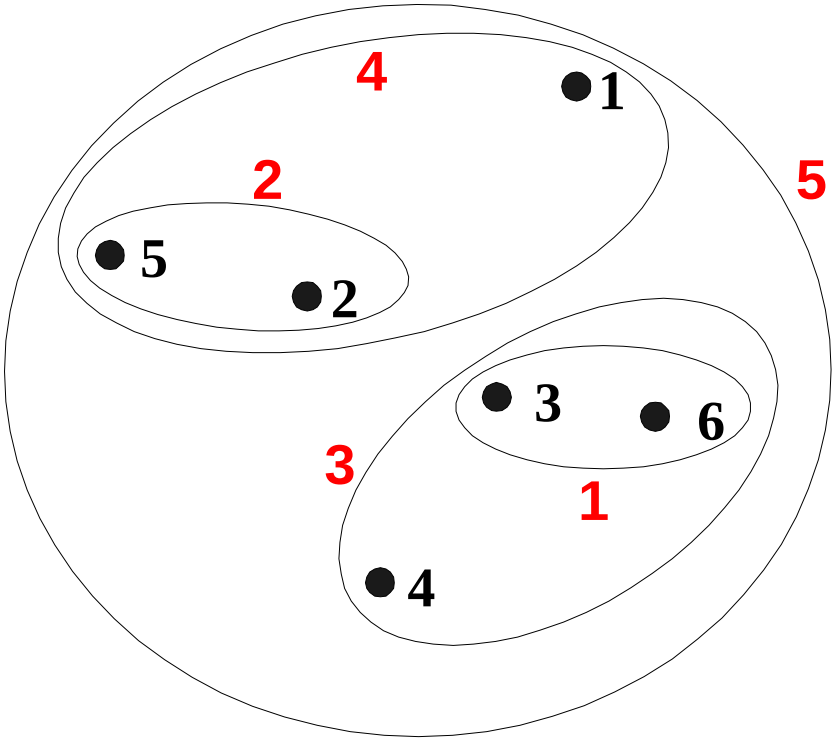
Original Points



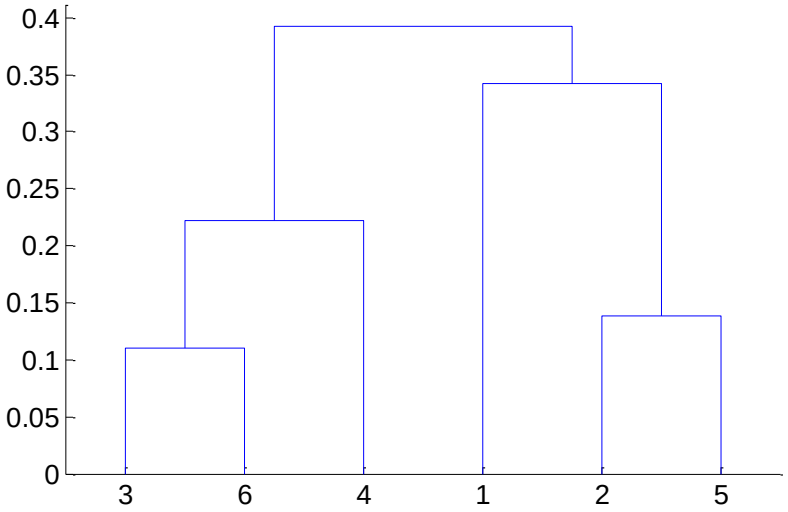
Two Clusters

- Sensitive to noise and outliers

Hierarchical Clustering: MAX

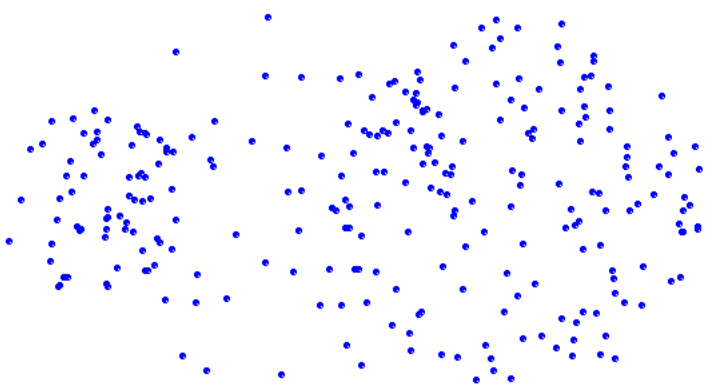


Nested Clusters

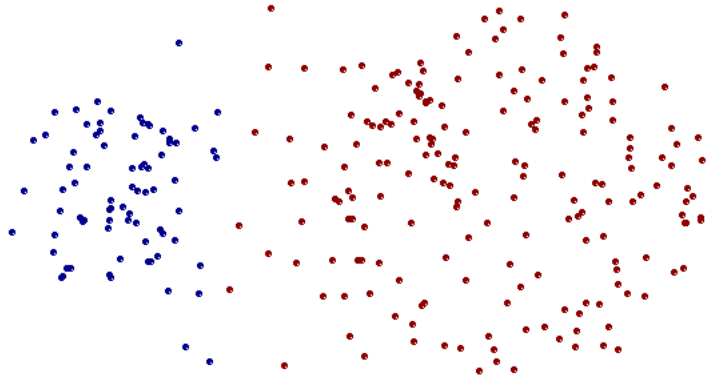


Dendrogram

Strength of MAX



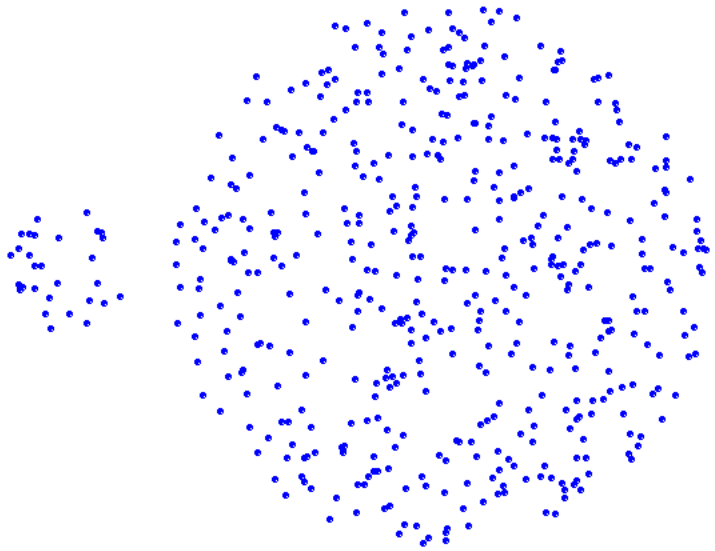
Original Points



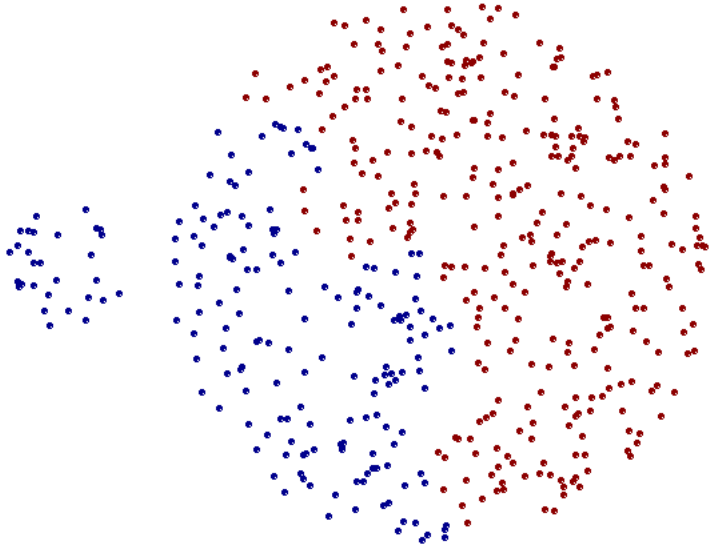
Two Clusters

- Less susceptible to noise and outliers

Limitations of MAX



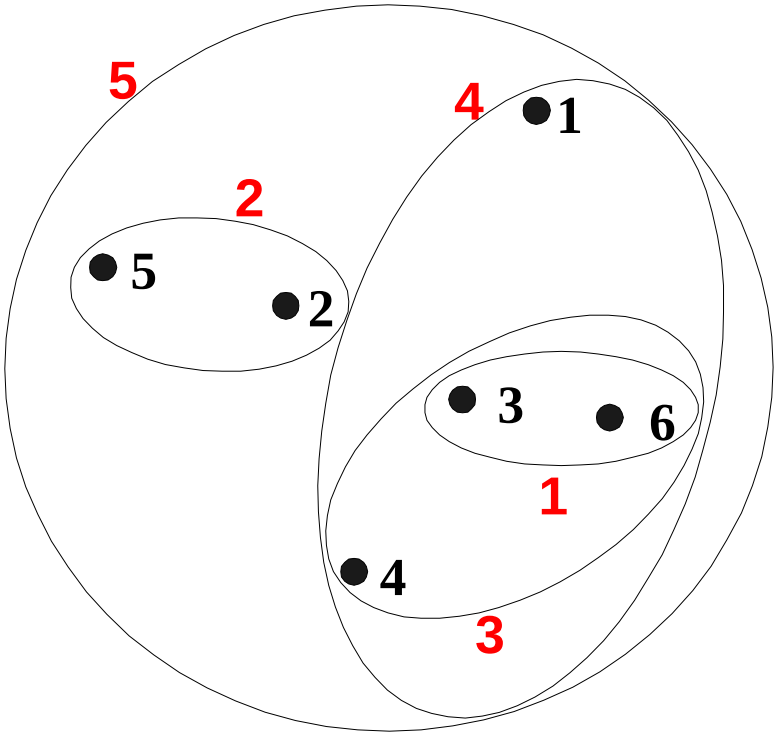
Original Points



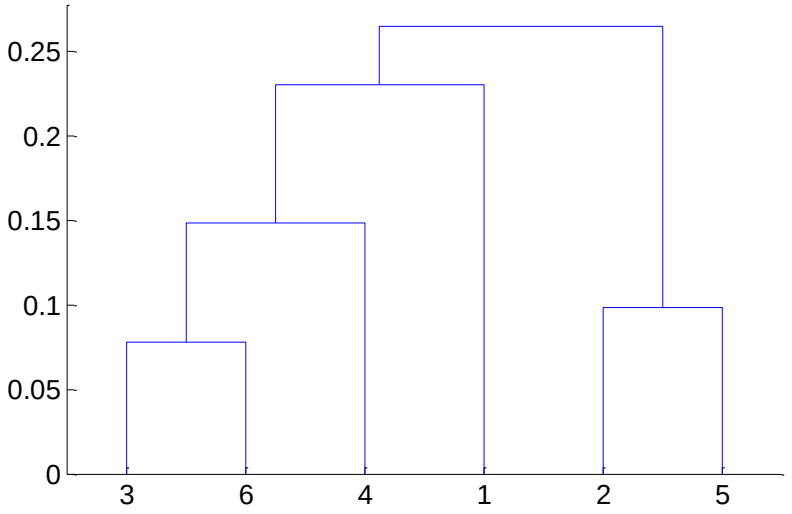
Two Clusters

- Tends to break large clusters
- Biased towards globular clusters

Hierarchical Clustering: Group Average



Nested Clusters



Dendrogram

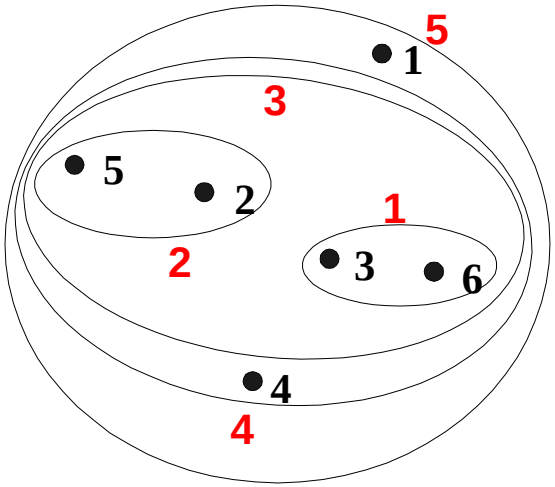
Hierarchical Clustering: Group Average

- Compromise between Single and Complete Link
- Strengths
  - Less susceptible to noise and outliers
- Limitations
  - Biased towards globular clusters

Cluster Similarity: Ward's Method

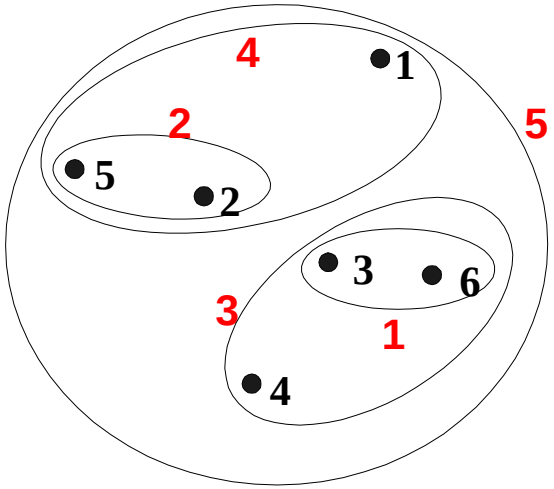
- Similarity of two clusters is based on the increase in squared error when two clusters are merged
  - Similar to group average if distance between points is distance squared
- Less susceptible to noise and outliers
- Biased towards globular clusters
- Hierarchical analogue of K-means
  - Can be used to initialize K-means

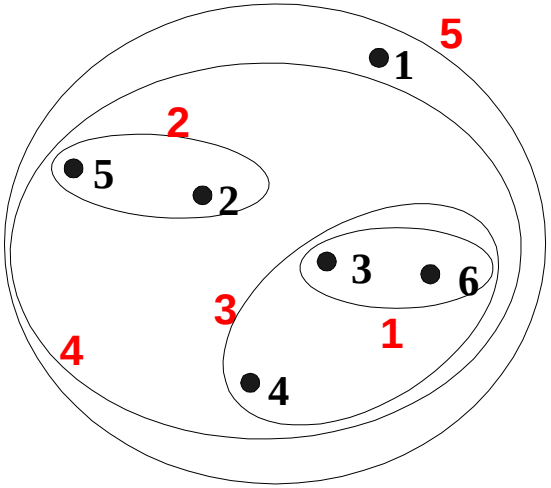
Hierarchical Clustering: Comparison



MIN

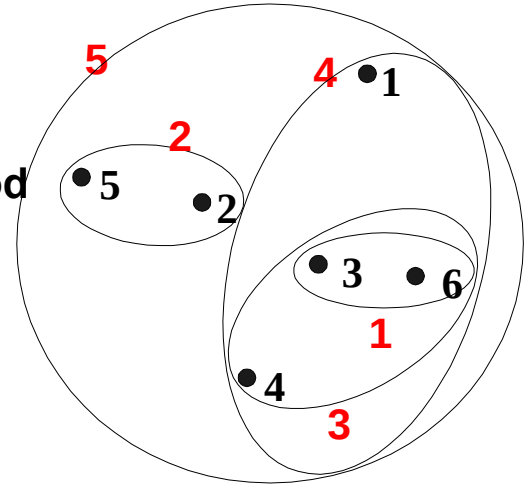
MAX





Group Average

Ward's Method



Hierarchical Clustering: Time and Space requirements

- $O(N^2)$ space since it uses the proximity matrix.
  - N is the number of points.

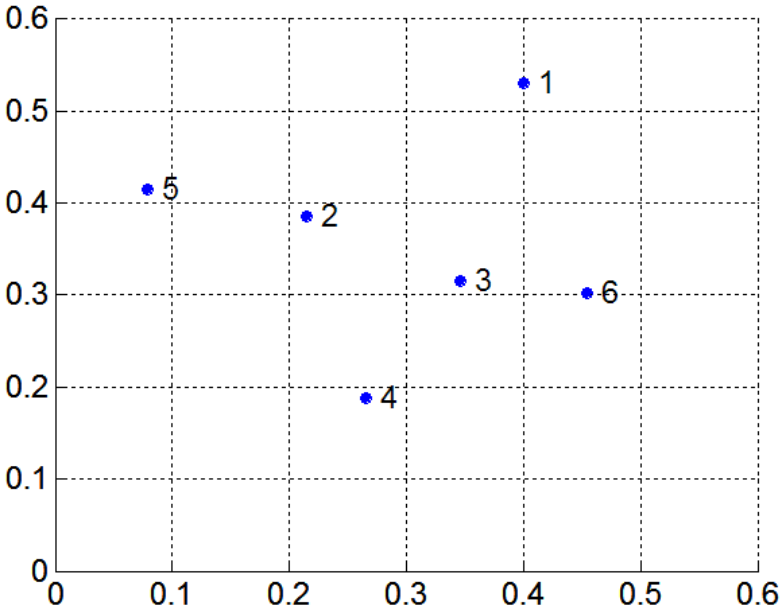
- $O(N^3)$ time in many cases
  - There are N steps and at each step the size, N^2, proximity matrix must be updated and searched
  - Complexity can be reduced to $O(N^2 \log(N))$ time for some approaches

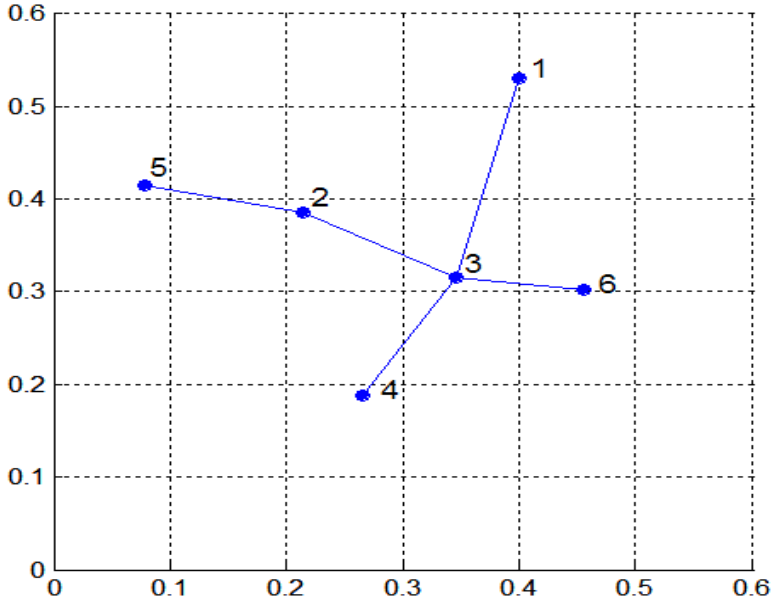
Hierarchical Clustering: Problems and Limitations

- Once a decision is made to combine two clusters, it cannot be undone
- No objective function is directly minimized
- Different schemes have problems with one or more of the following:
  - Sensitivity to noise and outliers
  - Difficulty handling different sized clusters and convex shapes
  - Breaking large clusters

MST: Divisive Hierarchical Clustering

- Build MST (Minimum Spanning Tree)
  - Start with a tree that consists of any point
  - In successive steps, look for the closest pair of points (p, q) such that one point (p) is in the current tree but the other (q) is not
  - Add q to the tree and put an edge between p and q





MST: Divisive Hierarchical Clustering

- Use MST for constructing hierarchy of clusters

Algorithm 7.5 MST Divisive Hierarchical Clustering Algorithm

1: Compute a minimum spanning tree for the proximity graph.
2: **repeat**
3: Create a new cluster by breaking the link corresponding to the largest distance (smallest similarity).
4: **until** Only singleton clusters remain

Cluster Validity

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

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.

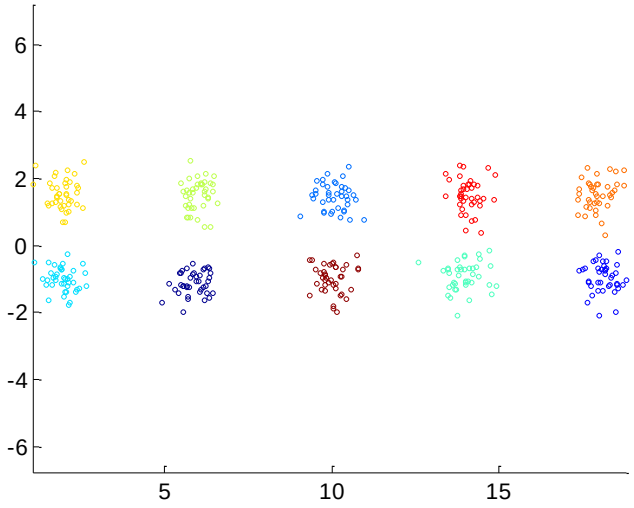
For 2, 3, and 4, we can further distinguish whether we want to evaluate the entire clustering or just individual clusters.

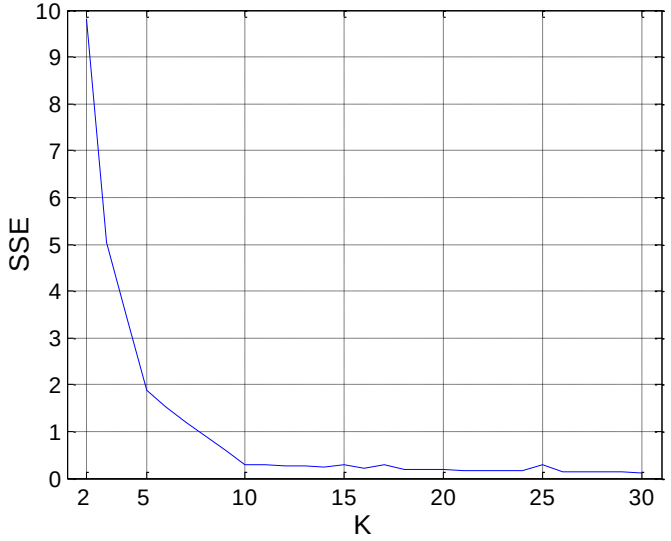
Measures of Cluster Validity

- 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.
    - Entropy
  - 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
- Sometimes these are referred to as criteria instead of indices
  - However, sometimes criterion is the general strategy and index is the numerical measure that implements the criterion.

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





Conclusions

- Unsupervised machine learning methods can be used to cluster or project large dimensional data very efficiently. However, the performance evaluation is subjective.
- Supervised machine learning methods are used whenever ground truth information exists, and when a noise insensitive model is required.
- There are many more clustering techniques which are not covered by these slides. For example, methods which can cluster sequential data (i.e. as in speech recognition, video analysis, etc.), or graph structured data (such as in molecular chemistry, document processing, etc.)

Readings

- Pang-Ning Tan, Micheale Steinbach, Vipin Kumar, "Introduction to Data Mining", Addison Wesley, 2006, ISBN 0-321-32136-7 **(Chapters 8 & 9)**
- A. B. M. Shawkat Ali, Saleh A. Wasimi, "Data Mining:Methods and Techniques", Thomson, 2007, ISBN 978-0-17-013676-1 **(Chapter 8)**
- Ian H. Witten, Eibe Frank "Data Mining Practical Machine Learning Tools and Techniques", Elsevier inc., 2005, ISBN 0-12-088407-0 **(Chapter 6.6)**
- Jiawei Han, Micheline Kamber, "Data Mining Concepts and Techniques", Morgan Kaufmann publishers, 2006, ISBN 978-1-55860-901-3 **(Chapter 7)**
- Margaret Dunham, "Data Mining Introductory and Advanced Topics, Pearson Education Inc., 2003, ISBN0-13-088892-3 **(Chapter 5)**

- [Graham Williams , "Data Mining with Rattle and R: the art of excavating data for knowledge discovery", Springer Verlag, 2011, ISBN 9781441998903.