

ICCS342/413 - Lecture 16

# Cluster Analysis

*Sunsern Cheamanunkul*

Adapted from Cluster Analysis: Basic Concepts by Tan, Steinbach, Kumar

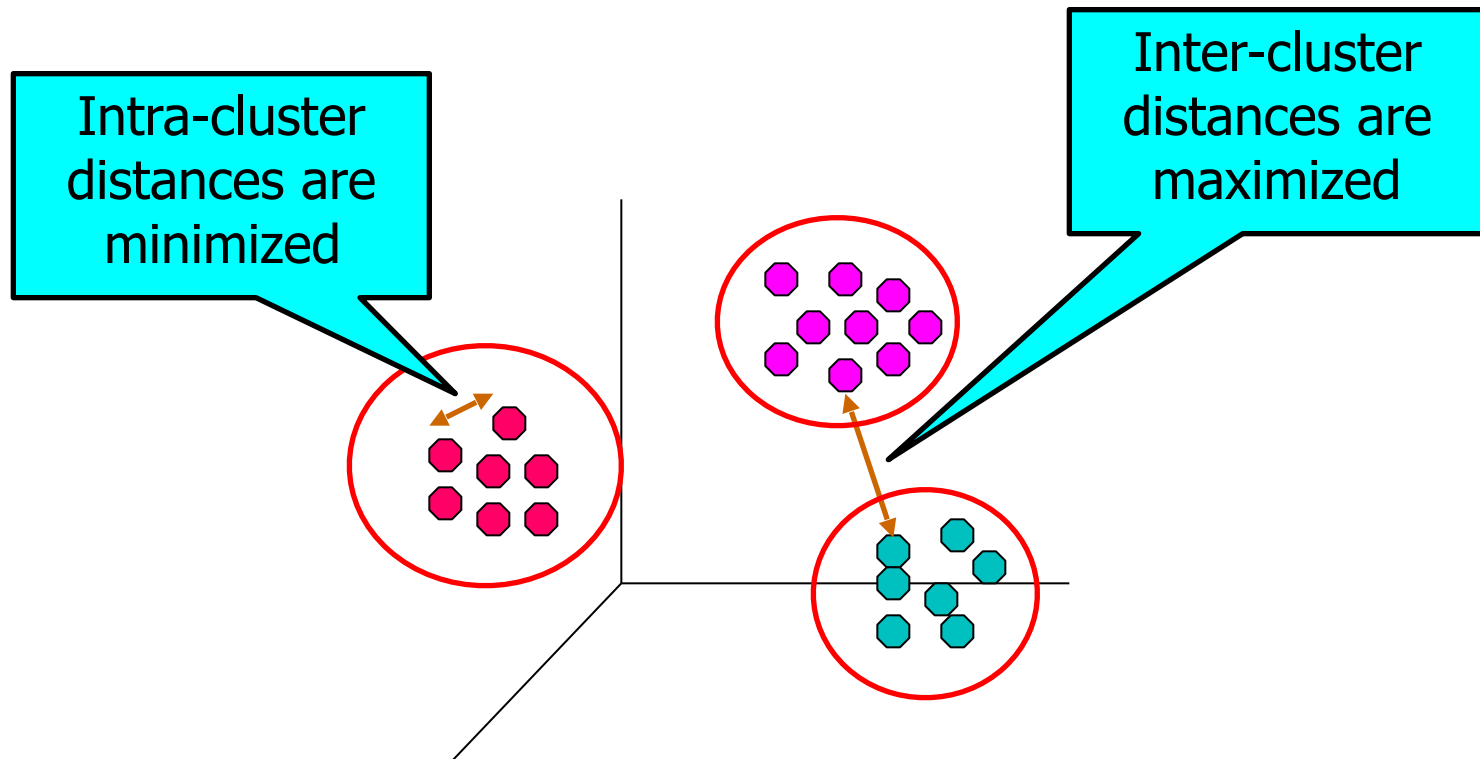


Mahidol University  
International College

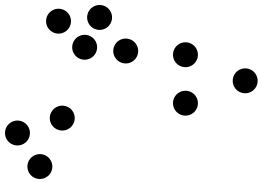


# What is Cluster Analysis?

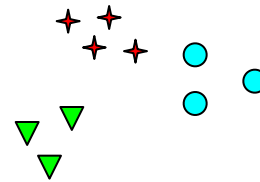
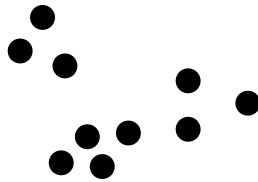
- Finding groups of objects such that the objects in a group will be similar (or related) to one another and different from (or unrelated to) the objects in other groups



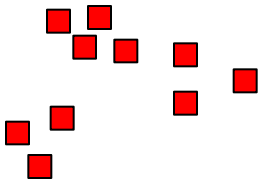
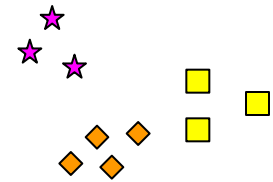
# Notion of a Cluster



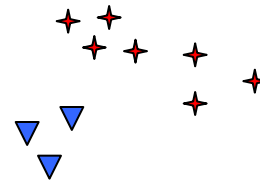
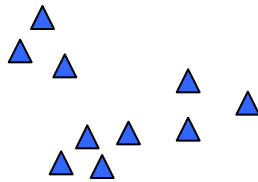
How many clusters?



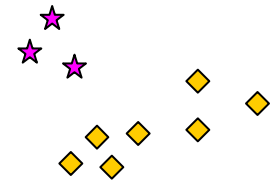
Six Clusters



Two Clusters



Four Clusters



# Problem Statement

- Given a set of points, with a notion of distance between points, group the points into some number of **clusters**, so that members of a cluster are in some sense as close to each other as possible.

# Distance Measures

- Each clustering problem is based on some kind of “distance” between points.
- Two major classes of distance measure:
  - Euclidean
  - Non-Euclidean

# Euclidean Vs. Non-Euclidean

- A **Euclidean space** has some number of real-valued dimensions and “dense” points.
  - There is a notion of “average” of two points.
  - A **Euclidean distance** is based on the locations of points in such a space.
- A **Non-Euclidean distance** is based on properties of points, but not their “location” in a space.

# Axiom of a Distance Measure

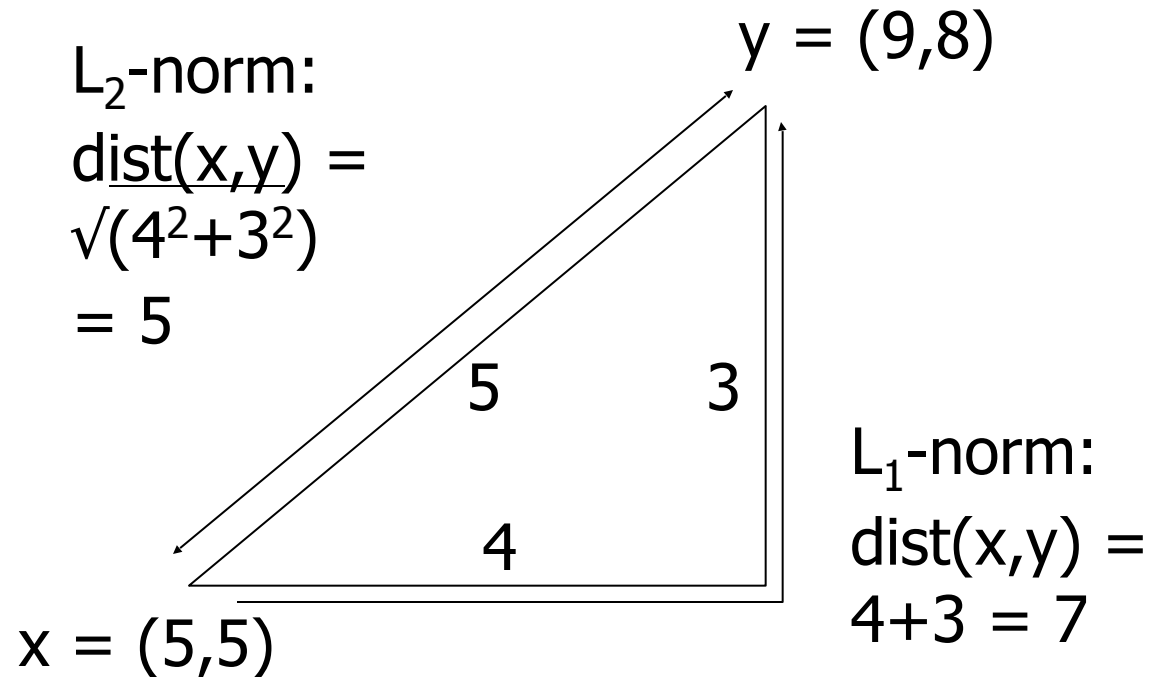
- $d$  is a **distance measure** if it is a function from pairs of points to reals such that:
  - $d(x,y) \geq 0$ .
  - $d(x,y) = 0$  iff  $x = y$ .
  - $d(x,y) = d(y,x)$ .
  - $d(x,y) \leq d(x,z) + d(z,y)$  (**triangle inequality**).

# Euclidean Distances

- $L_2$  norm :  $d(x,y)$  = square root of the sum of the squares of the differences between  $x$  and  $y$  in each dimension.
  - The most common notion of “distance.”
- $L_1$  norm : sum of the differences in each dimension.
  - Manhattan distance = distance if you had to travel along coordinates only.



# Examples of Euclidean Distances



# Another Euclidean Distance

- $L_\infty$  *norm* :  $d(x,y)$  = the maximum of the differences between  $x$  and  $y$  in any dimension.
- Note: the maximum is the limit as  $n$  goes to  $\infty$  of what you get by taking the  $n$  th power of the differences, summing and taking the  $n$  th root.

# Non-Euclidean Distances

- **Jaccard distance** for sets = 1 minus ratio of sizes of intersection and union.
- **Cosine distance** = angle between vectors from the origin to the points in question.
- **Edit distance** = number of inserts and deletes to change one string into another.

# Jaccard Distance

- Example:  $p_1 = 10111$ ;  $p_2 = 10011$ .
  - Size of intersection = 3; size of union = 4,  
Jaccard measure (not distance) =  $3/4$ .
- Need to make a distance function satisfying triangle inequality and other laws.
- $d(x,y) = 1 - (\text{Jaccard measure})$  works.

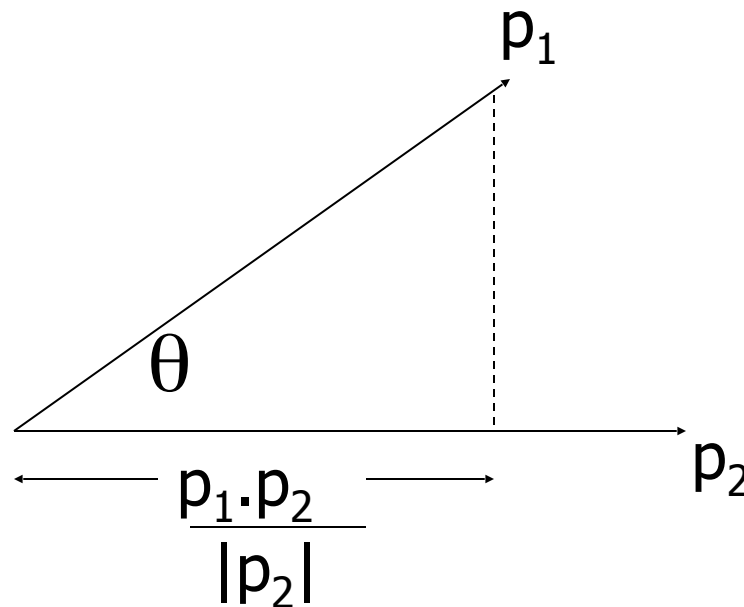
# Jaccard Distance

- $d(x,x) = 0$  because  $x \cap x = x \cup x$ .
- $d(x,y) = d(y,x)$  because union and intersection are symmetric.
- $d(x,y) \geq 0$  because  $|x \cap y| \leq |x \cup y|$ .
- $d(x,y) \leq d(x,z) + d(z,y)$  trickier --- next slide.

# Cosine Distance

- Think of a point as a vector from the origin  $(0,0,\dots,0)$  to its location.
- Two points' vectors make an angle, whose cosine is the normalized dot-product of the vectors:  $p_1 \cdot p_2 / |p_2| |p_1|$ .
  - Example  $p_1 = (00111)$ ;  $p_2 = (10011)$ .
  - $p_1 \cdot p_2 = 2$ ;  $|p_1| = |p_2| = \sqrt{3}$ .
  - $\cos(\theta) = 2/3$ ;  $\theta$  is about 48 degrees.

# Cosine Distance

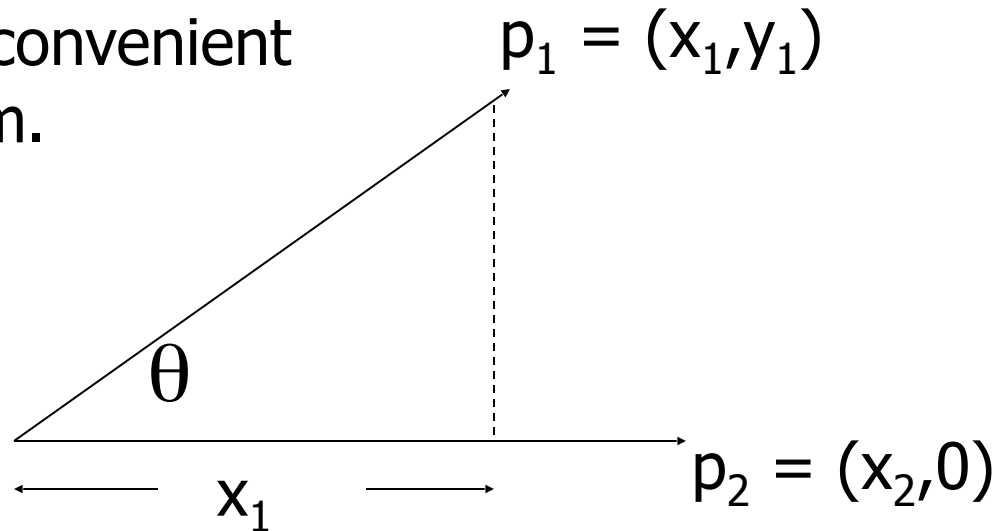


$$\text{dist}(p_1, p_2) = \theta = \arccos(p_1 \cdot p_2 / |p_2| |p_1|)$$

# Cosine Distance

Dot product is invariant under rotation, so pick convenient coordinate system.

$$p_1 \cdot p_2 = x_1 * x_2.$$
$$|p_2| = x_2.$$



$$x_1 = p_1 \cdot p_2 / |p_2|$$



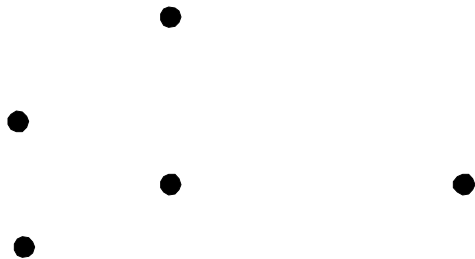
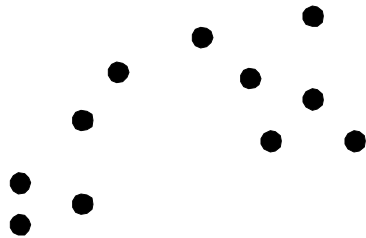
# Cosine Distance

- $d(x,x) = 0$  because  $\arccos(1) = 0$ .
- $d(x,y) = d(y,x)$  by symmetry.
- $d(x,y) \geq 0$  because angles are chosen to be in the range 0 to 180 degrees.
- **Triangle inequality**: physical reasoning. If I rotate an angle from  $x$  to  $z$  and then from  $z$  to  $y$ , I can't rotate less than from  $x$  to  $y$ .

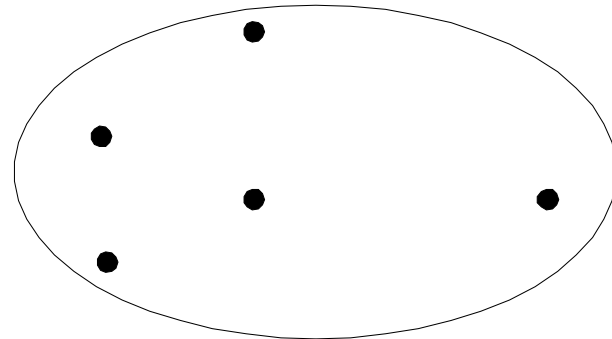
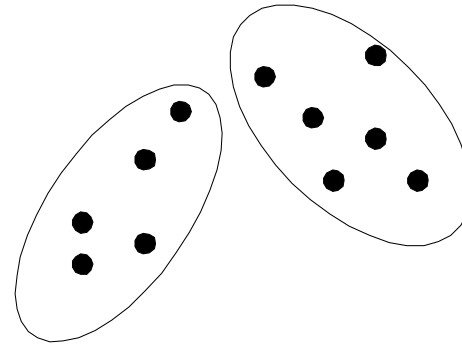
# Types of Cluster Analysis

- **Hierarchical:**
  - Initially, each point in cluster by itself.
  - Repeatedly combine the two “closest” clusters into one.
- **Partitioning:**
  - Maintain a set of clusters.
  - Place points into “closest” cluster.

# Partitioning Clustering



**Original Points**



**A Partitional Clustering**

# Partitioning Clustering

- a typical clustering analysis approach via **iteratively** partitioning training data set to learn a partition of the given data space
- learning a partition on a data set to produce several non-empty clusters (usually, the number of clusters given in advance)
- in principle, optimal partition achieved via **minimizing the sum of squared distance to its “representative object” in each cluster**

$$E = \sum_{k=1}^K \sum_{\mathbf{x} \in C_k} d^2(\mathbf{x}, \mathbf{m}_k)$$

# Partitioning Clustering

- Given a  $K$ , find a partition of  $K$  clusters to optimize the chosen partitioning criterion (cost function)
  - global optimum: exhaustively search all partitions
- The **K-means** algorithm: a heuristic method
  - K-means algorithm (MacQueen'67): each cluster is represented by the center of the cluster and the algorithm converges to stable centroids of clusters.
  - K-means algorithm is the simplest partitioning method for clustering analysis and widely used in data mining applications.

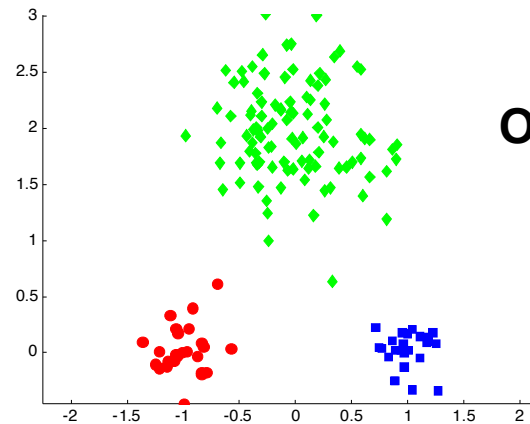
# K-means Algorithm

- Given the cluster number  $K$ , the K-means algorithm is carried out in three steps after initialization:

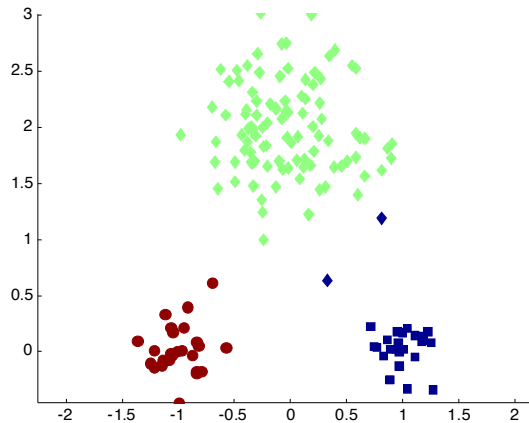
Initialization: set seed points (randomly)

- 1) Assign each object to the cluster of the nearest seed point measured with a specific distance metric
- 2) Compute seed points as the centroids of the clusters of the current partition (the centroid is the center, i.e., **mean point**, of the cluster)
- 3) Go back to Step 1, stop when no more new assignment (i.e., membership in each cluster no longer changes)

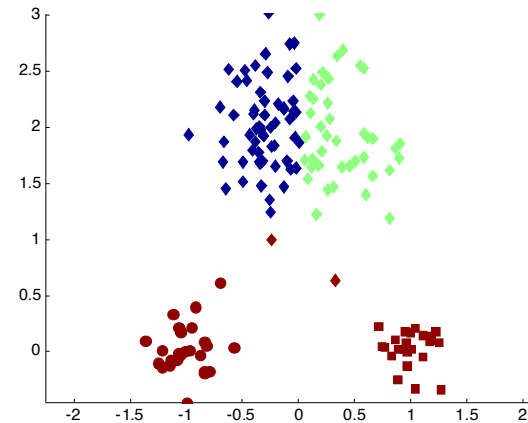
# K-means clustering



**Original Points**

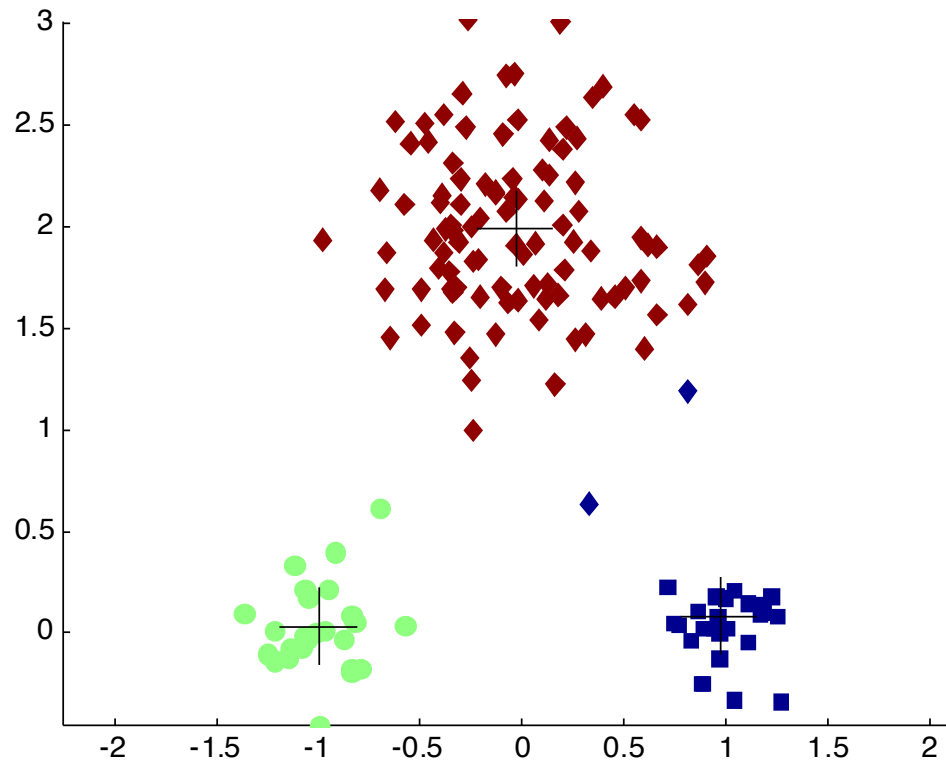


**Optimal Clustering**



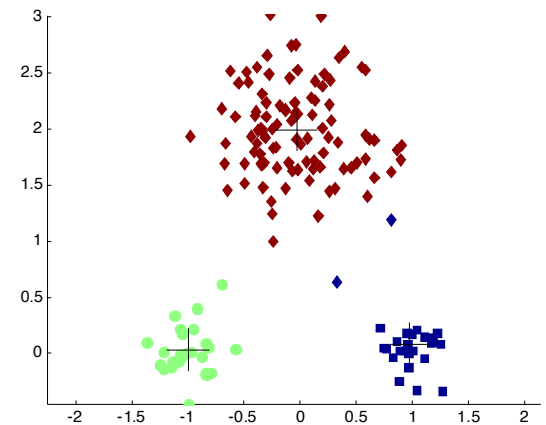
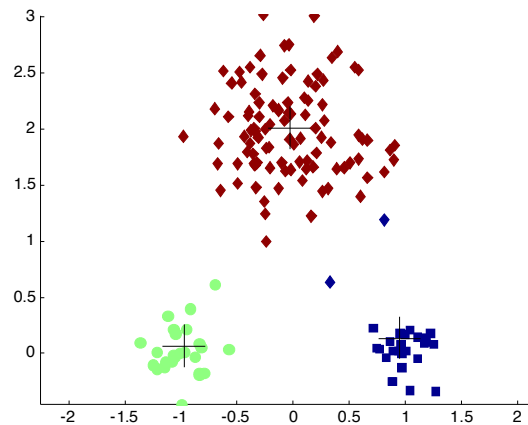
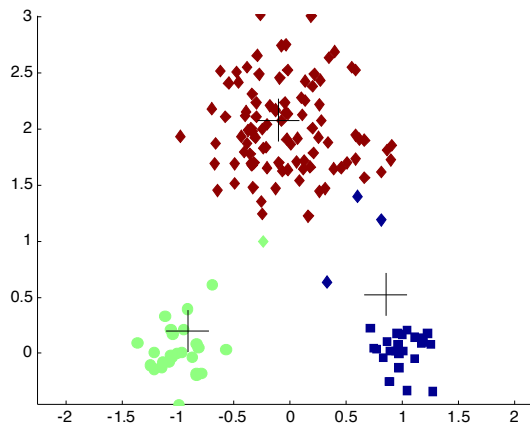
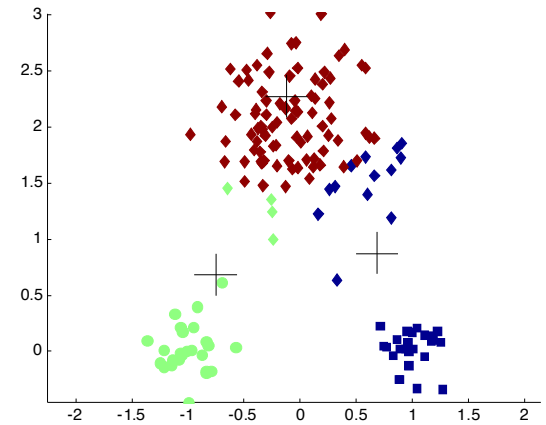
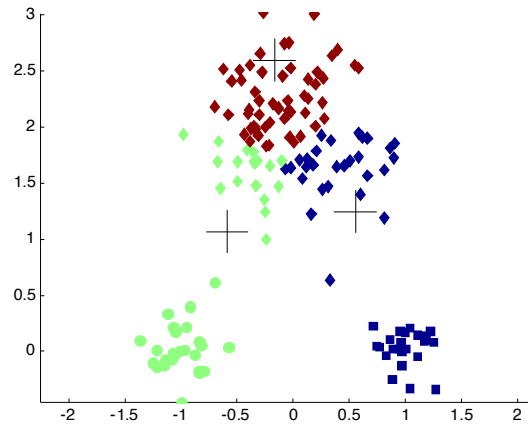
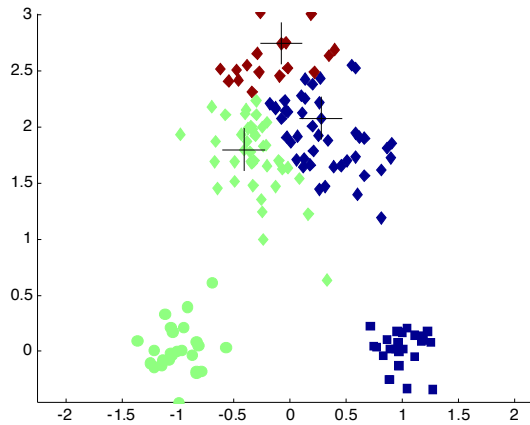
**Sub-optimal Clustering**

# Initial Centroids





# Initial Centroids



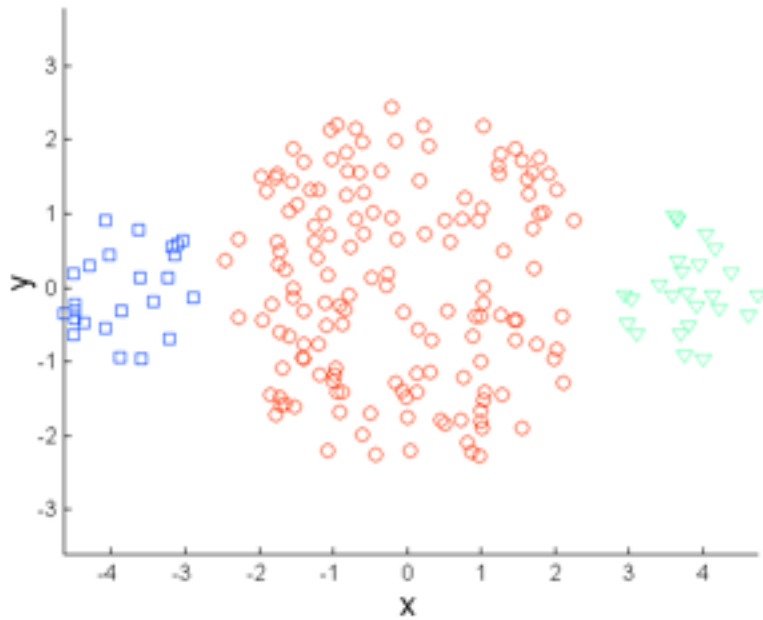
# Evaluating K-means Clusters

- 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.

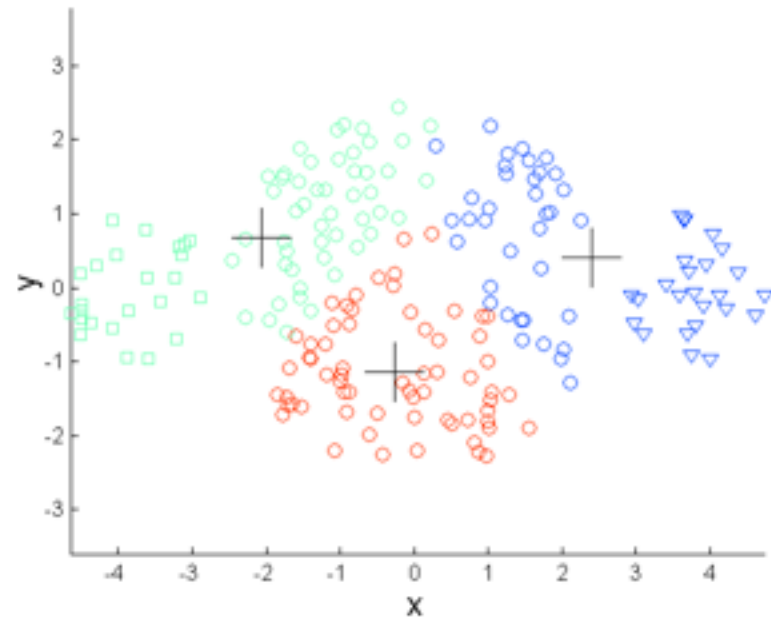
$$SSE = \sum_{i=1}^K \sum_{x \in C_i} dist^2(m_i, x)$$

- $x$  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 clusters, 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  $K$

# Limitations of K-means

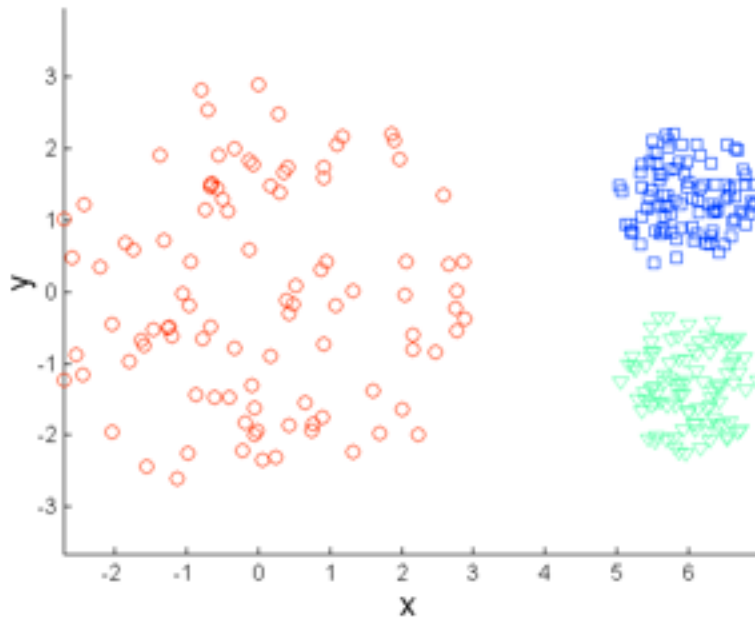


**Original Points**

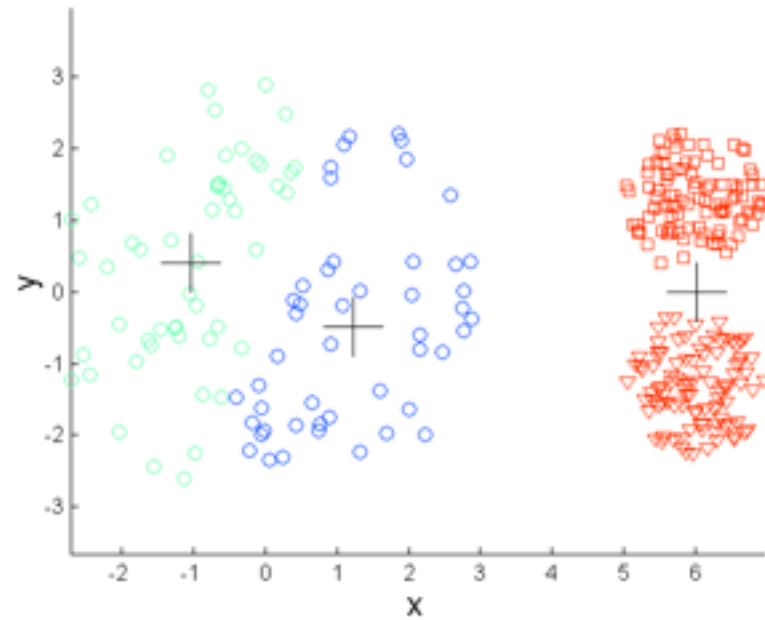


**K-means (3 Clusters)**

# Limitations of K-means

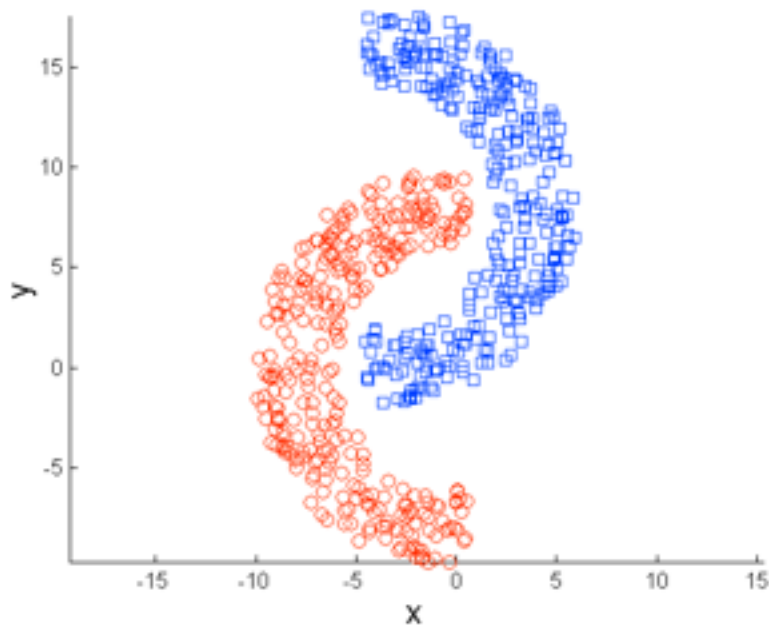


**Original Points**

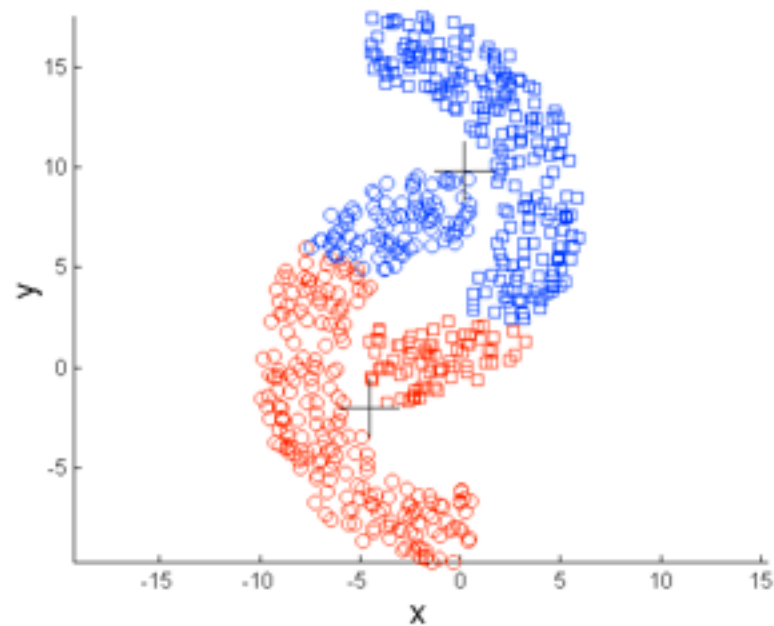


**K-means (3 Clusters)**

# Limitations of K-means

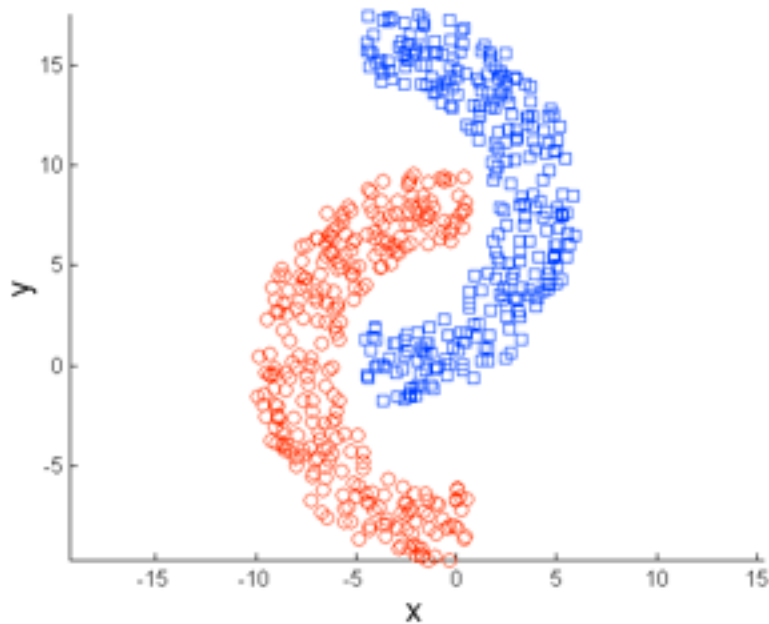


**Original Points**

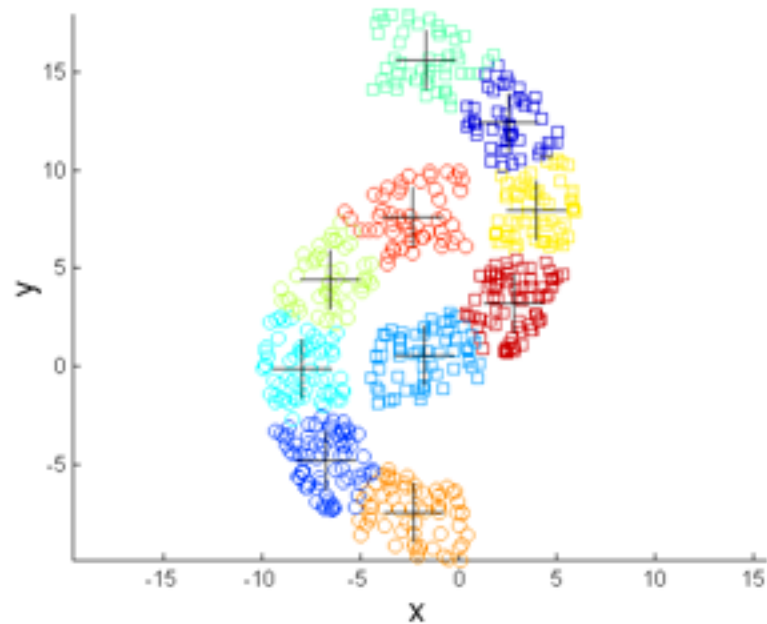


**K-means (2 Clusters)**

# Overcoming K-means Limitations



**Original Points**



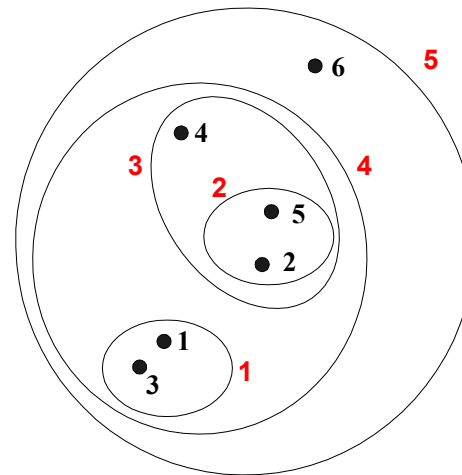
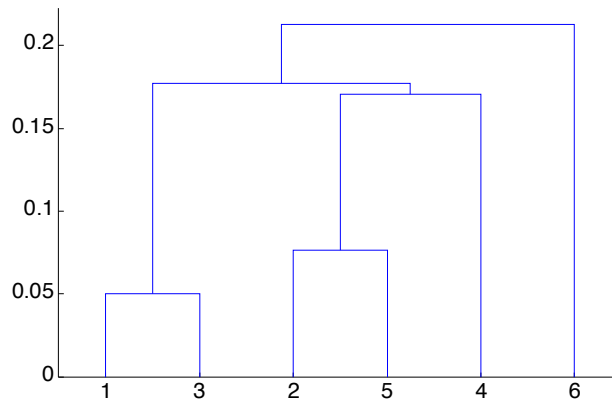
**K-means Clusters**

# K-means Summary

- **K-means** algorithm is a simple yet popular method for clustering analysis
- Its performance is determined by initialization and appropriate distance measure
- There are several **variants** of K-means to overcome its weaknesses
  - K-Medoids: resistance to noise and/or outliers
  - K-Modes: extension to categorical data clustering analysis
  - CLARA: extension to deal with large data sets
  - Mixture models (EM algorithm): handling uncertainty of clusters

# 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

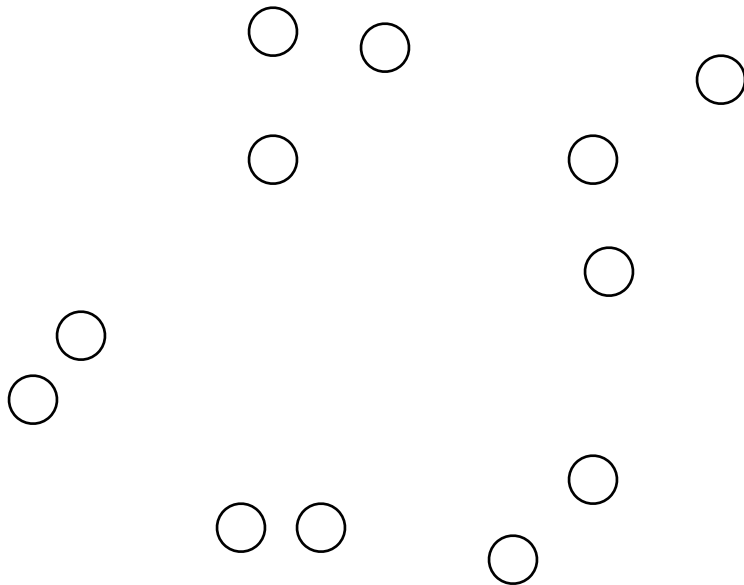
- Two main types of hierarchical clustering
  - Agglomerative:
    - Start with the points as individual clusters
    - At each step, merge the closest pair of clusters until only one cluster (or  $k$  clusters) left
  - Divisive:
    - Start with one, all-inclusive cluster
    - At each step, split 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

- More popular hierarchical clustering technique
- Basic algorithm is straightforward
  - Compute the proximity matrix
  - Let each data point be a cluster
  - **Repeat**
    - Merge the two closest clusters
    - Update the proximity matrix
  - **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

# Example

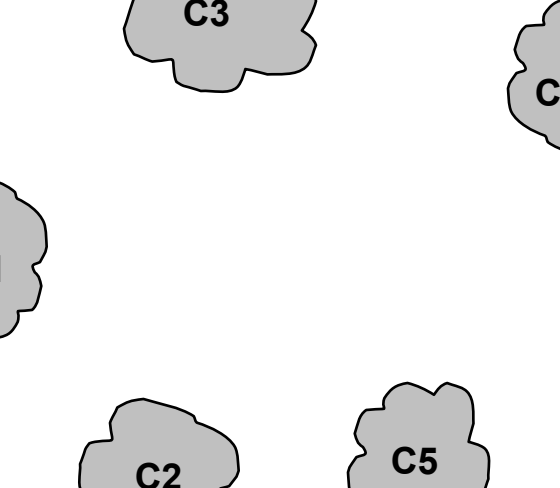
- Start with clusters of individual points and a proximity matrix



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

**Proximity Matrix**



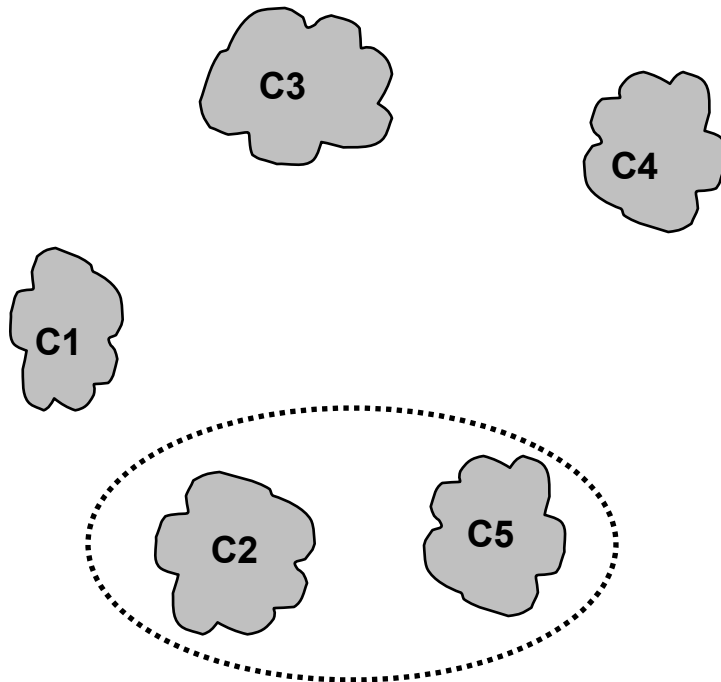
- 
- A diagram showing five clusters of cells, each represented by a grey, irregularly shaped blob with a black outline. The clusters are labeled C1, C2, C3, C4, and C5 in bold black text. C1 is on the left, C2 is at the bottom left, C3 is at the top center, C4 is at the top right, and C5 is at the bottom right.

## Proximity Matrix



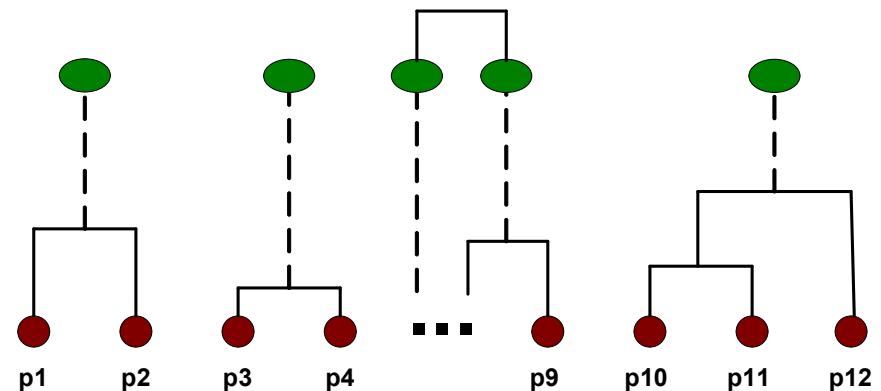
# Example

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



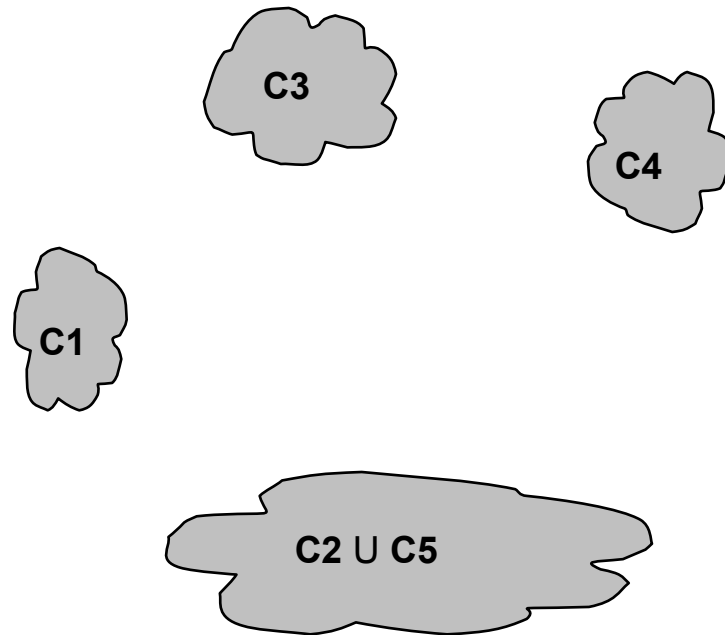
	c1	c2	c3	c4	c5
C1					
C2					
C3					
C4					
C5					

**Proximity Matrix**



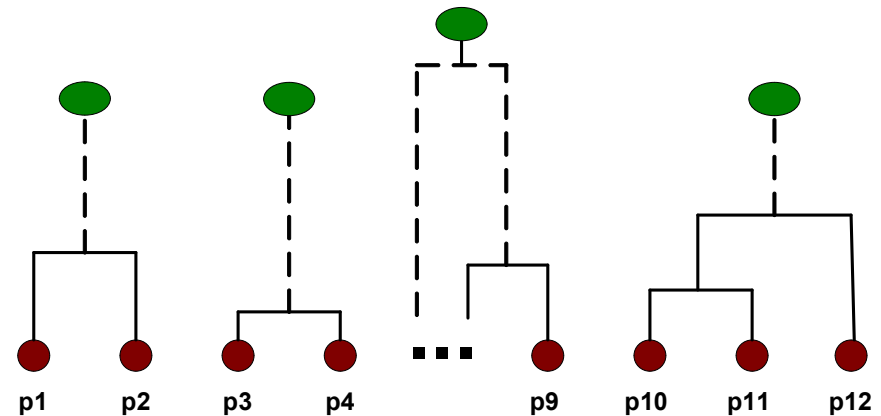
# Example

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

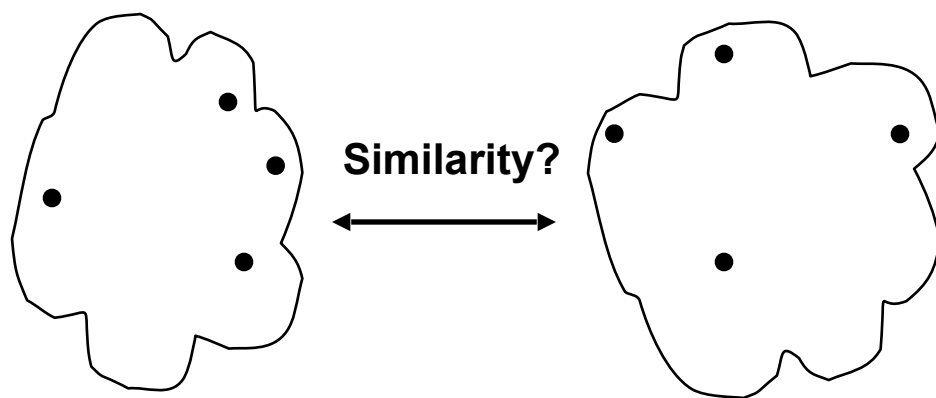


	C1	$C2 \cup C5$	C3	C4
C1		?		
$C2 \cup C5$	?	?	?	?
C3		?		
C4		?		

**Proximity Matrix**



# How to Define Inter-Cluster Similarity



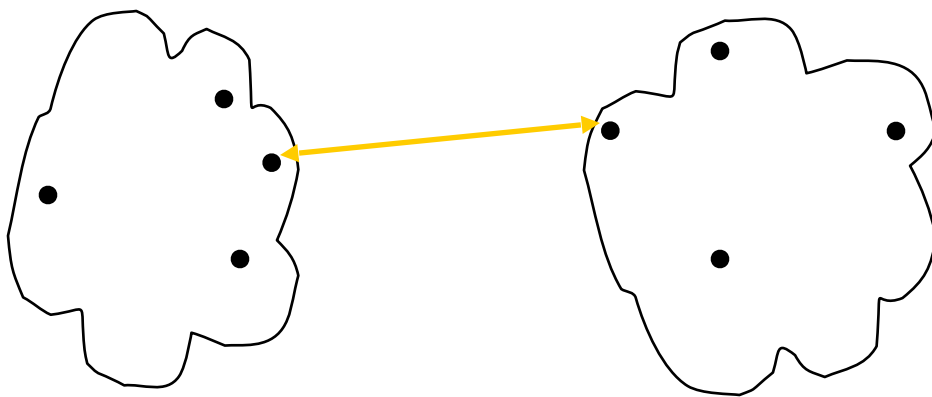
- MIN
- MAX
- Group Average
- Distance Between Centroids
- Other methods driven by an objective function
  - Ward's Method uses squared error

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

**Proximity Matrix**



# How to Define Inter-Cluster Similarity

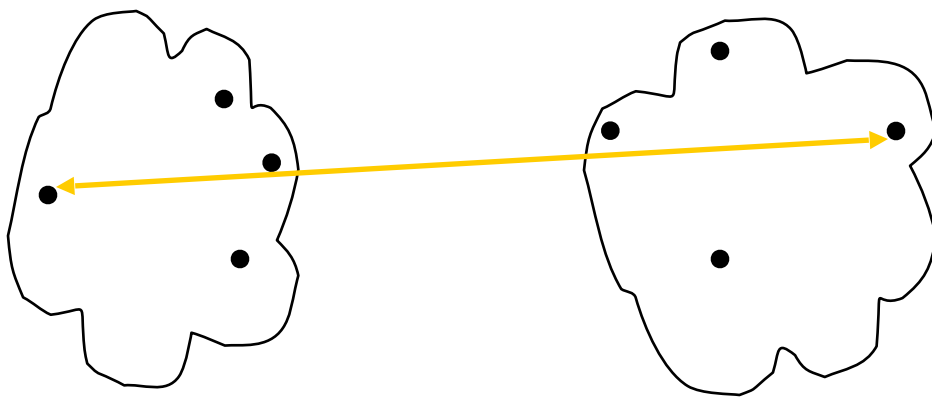


- **MIN**
- **MAX**
- Group Average
- Distance Between Centroids
- Other methods driven by an objective function
  - Ward's Method uses squared error

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

**Proximity Matrix**

# How to Define Inter-Cluster Similarity

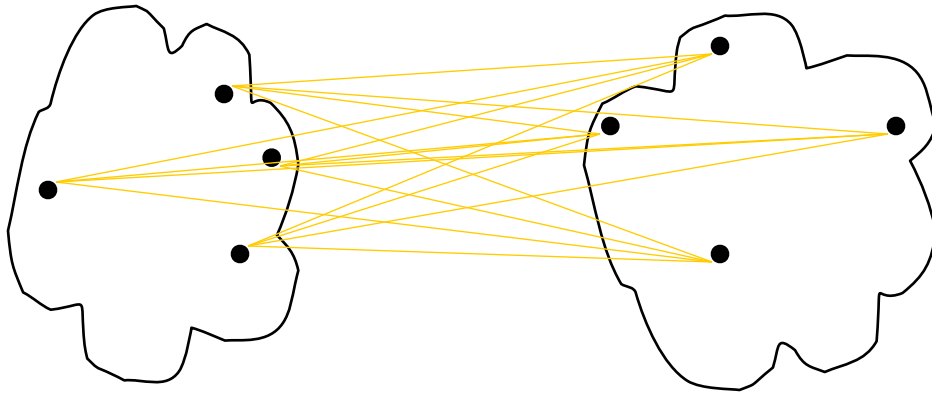


- MIN
- **MAX**
- Group Average
- Distance Between Centroids
- Other methods driven by an objective function
  - Ward's Method uses squared error

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

**Proximity Matrix**

# How to Define Inter-Cluster Similarity

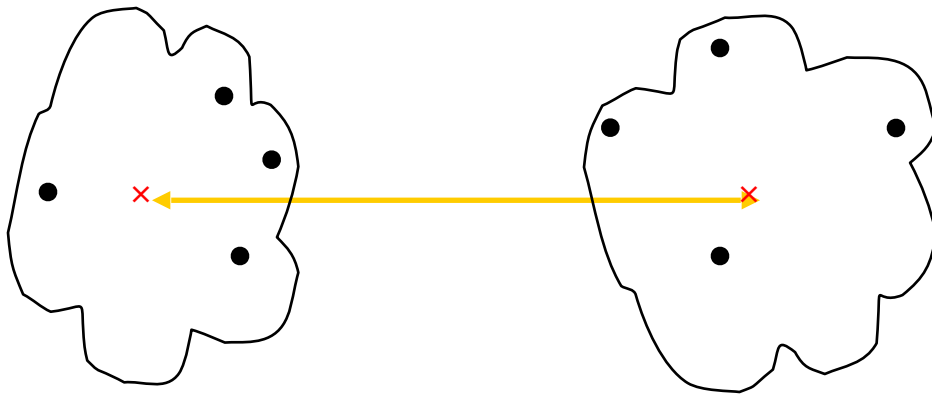


- MIN
- MAX
- **Group Average**
- Distance Between Centroids
- Other methods driven by an objective function
  - Ward's Method uses squared error

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

**Proximity Matrix**

# How to Define Inter-Cluster Similarity



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

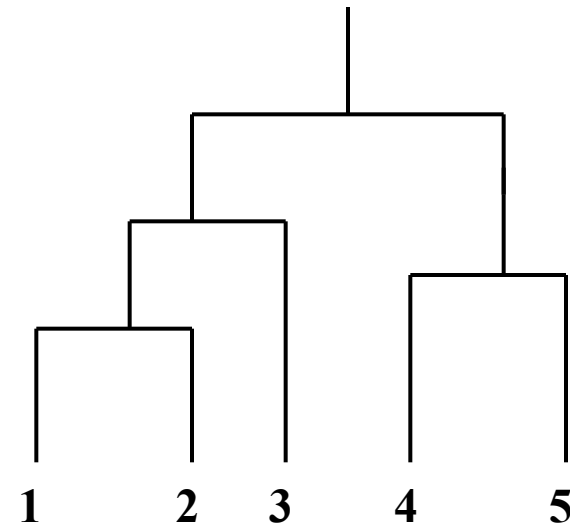
	p1	p2	p3	p4	p5	...
p1						
p2						
p3						
p4						
p5						
.						

**Proximity Matrix**

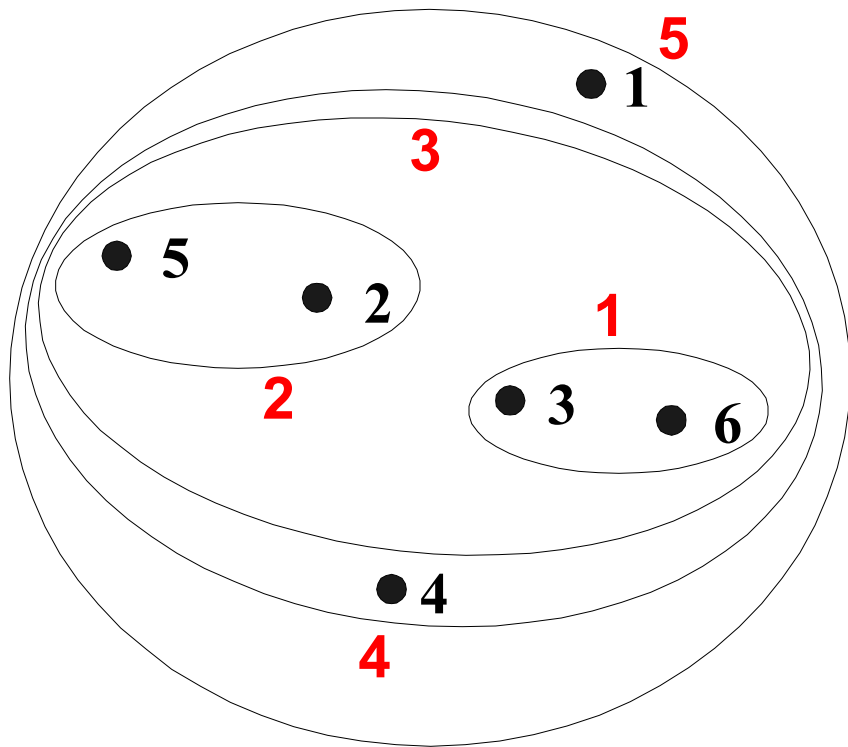
# Cluster Similarity: MIN or Single Link

- 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.

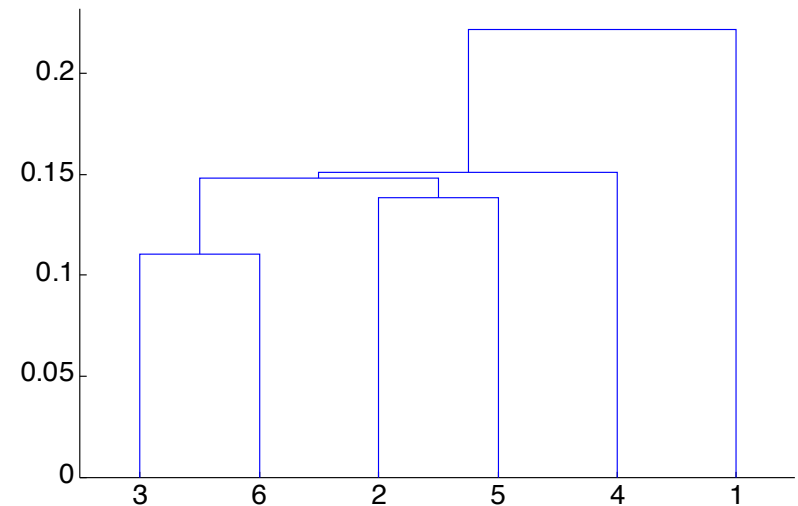
	I1	I2	I3	I4	I5
I1	1.00	0.90	0.10	0.65	0.20
I2	0.90	1.00	0.70	0.60	0.50
I3	0.10	0.70	1.00	0.40	0.30
I4	0.65	0.60	0.40	1.00	0.80
I5	0.20	0.50	0.30	0.80	1.00



# Hierarchical Clustering: MIN

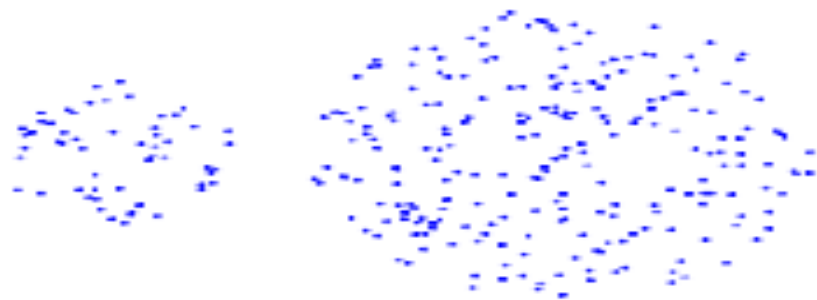


**Nested Clusters**

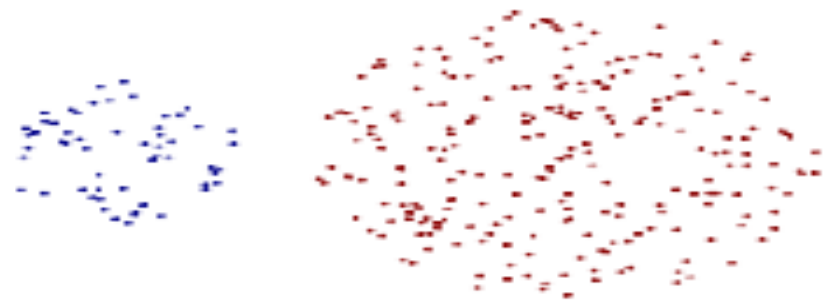


**Dendrogram**

# Strength of MIN



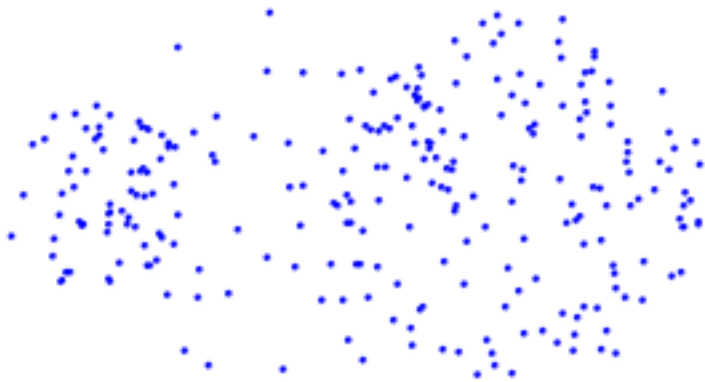
**Original Points**



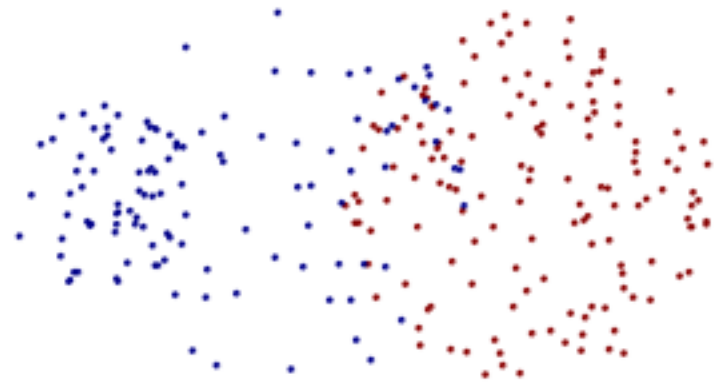
**Two Clusters**

- Can handle non-elliptical shapes

# Limitations of MIN



**Original Points**



**Two Clusters**

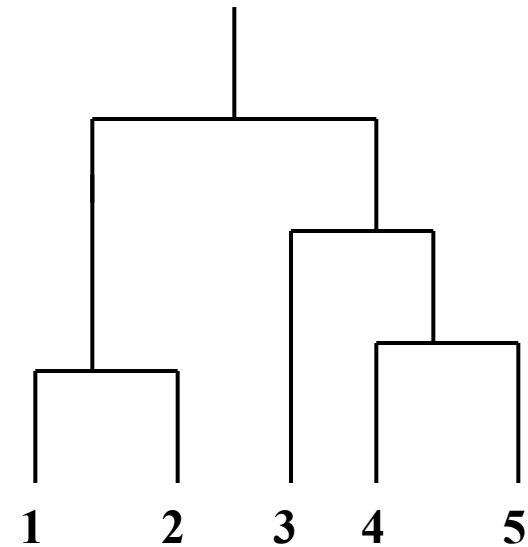
- **Sensitive to noise and outliers**



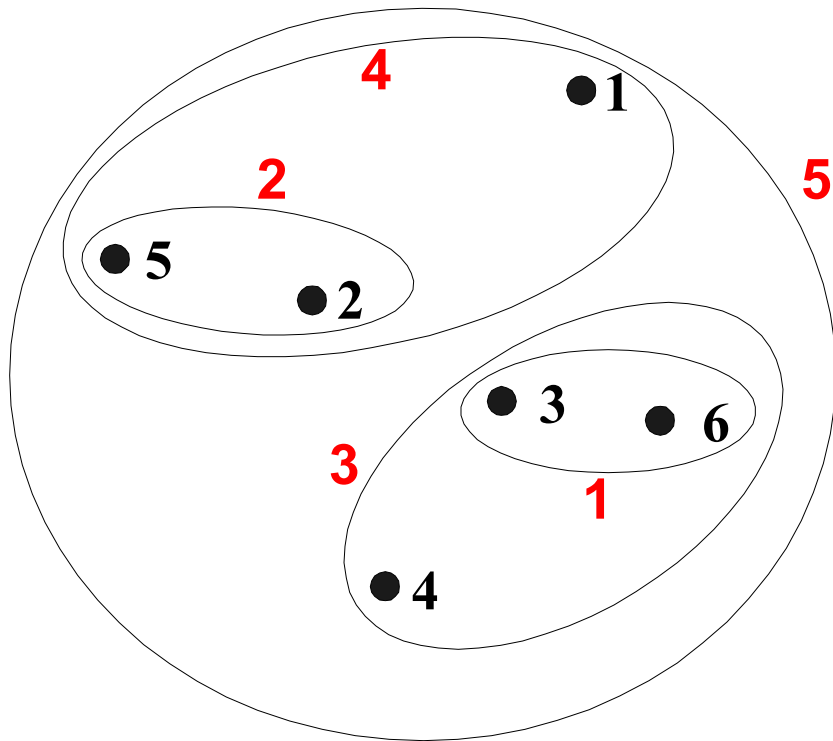
# Cluster Similarity: MAX or Complete Linkage

- 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

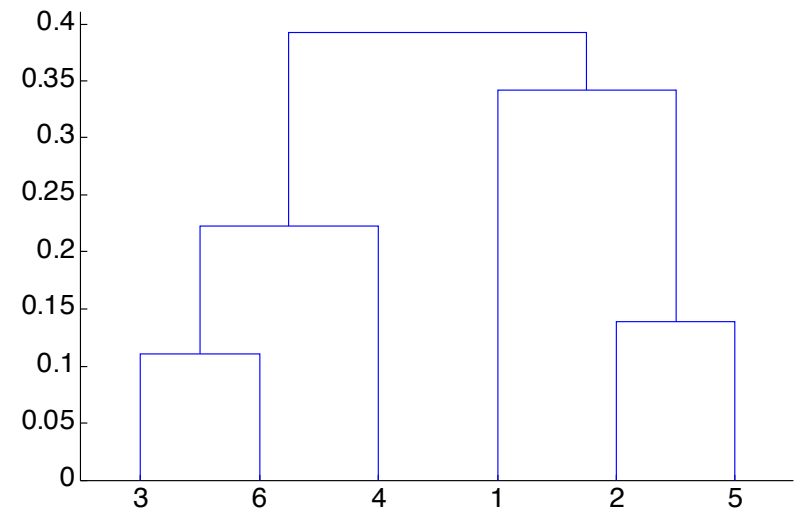
	I1	I2	I3	I4	I5
I1	1.00	0.90	0.10	0.65	0.20
I2	0.90	1.00	0.70	0.60	0.50
I3	0.10	0.70	1.00	0.40	0.30
I4	0.65	0.60	0.40	1.00	0.80
I5	0.20	0.50	0.30	0.80	1.00



# Hierarchical Clustering: MAX

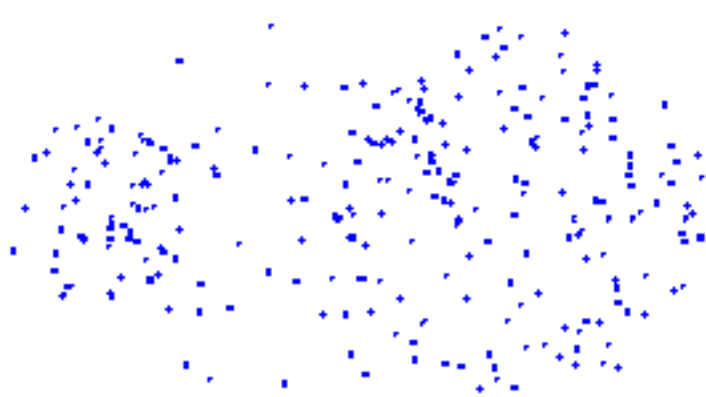


**Nested Clusters**

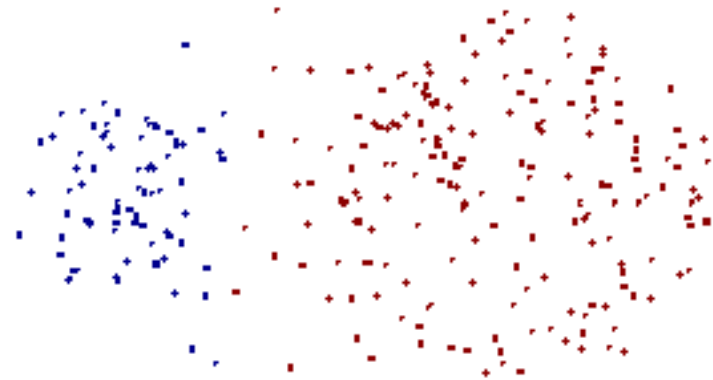


**Dendrogram**

# Strength of MAX



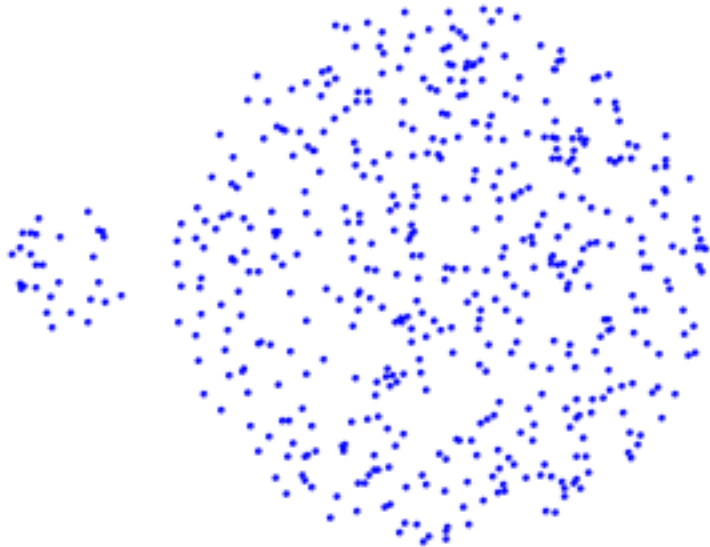
**Original Points**



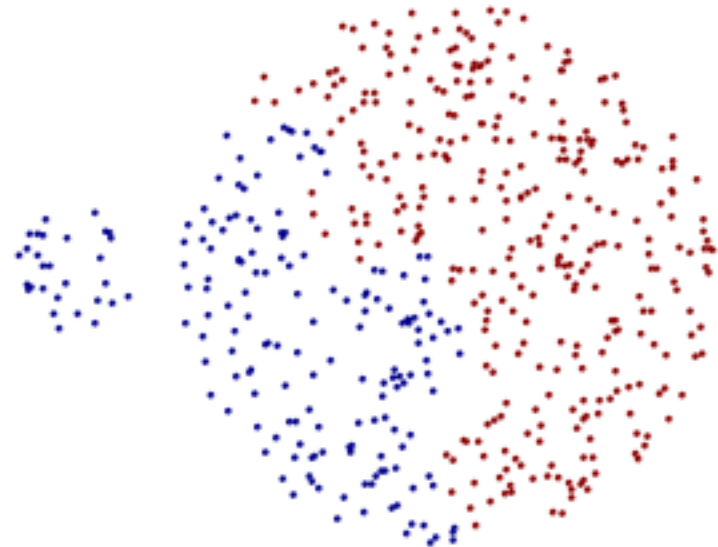
**Two Clusters**

- **Less susceptible to noise and outliers**

# Limitations of MAX



**Original Points**



**Two Clusters**

- Tends to break large clusters
- Biased towards globular clusters

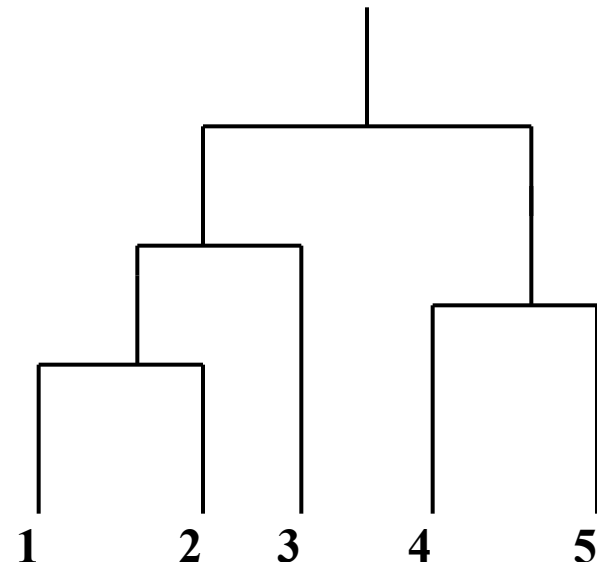
# Cluster Similarity: Group Average

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

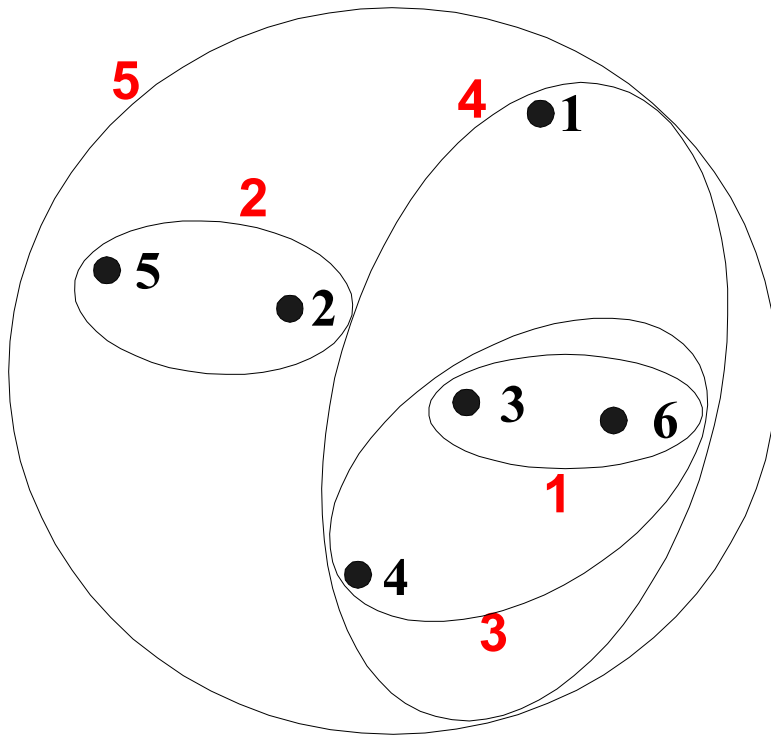
$$\text{proximity}(\text{Cluster}_i, \text{Cluster}_j) = \frac{\sum_{\substack{p_i \in \text{Cluster}_i \\ p_j \in \text{Cluster}_j}} \text{proximity}(p_i, p_j)}{|\text{Cluster}_i| * |\text{Cluster}_j|}$$

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

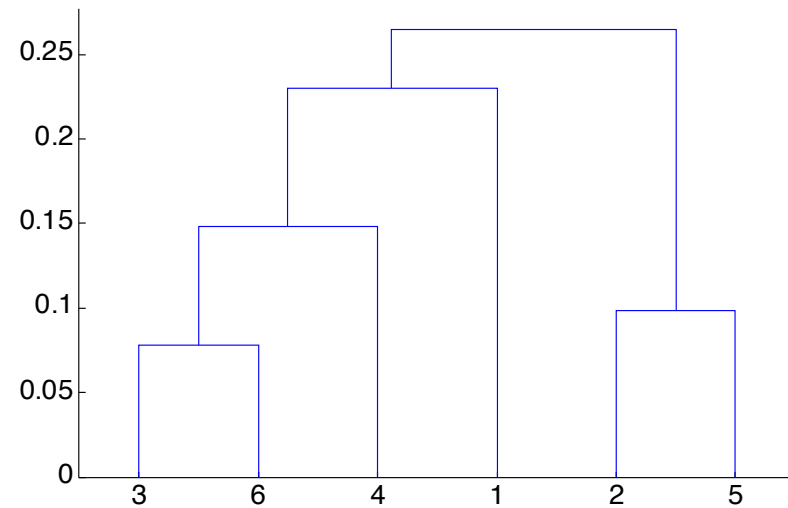
	I1	I2	I3	I4	I5
I1	1.00	0.90	0.10	0.65	0.20
I2	0.90	1.00	0.70	0.60	0.50
I3	0.10	0.70	1.00	0.40	0.30
I4	0.65	0.60	0.40	1.00	0.80
I5	0.20	0.50	0.30	0.80	1.00



# Hierarchical Clustering: Group Average



**Nested Clusters**



**Dendrogram**

# Hierarchical Clustering: Group Average

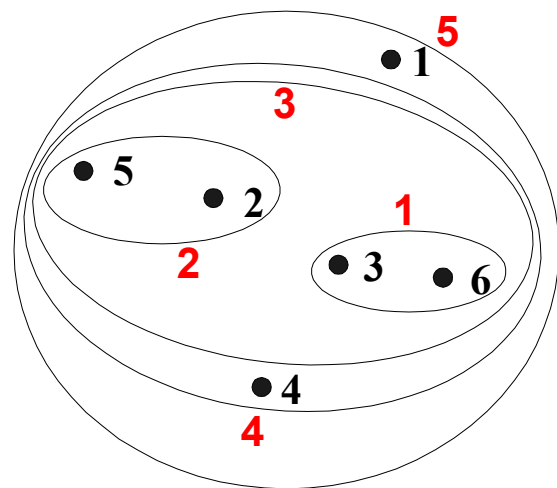
- Compromise between Single and Complete Link
- Strengths
  - Less susceptible to noise and outliers
- Limitations
  - Biased towards globular clusters

# Cluster Similarity: Ward's Method

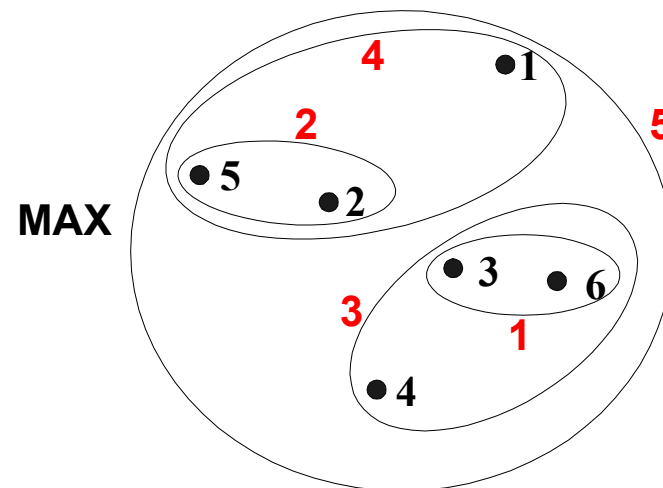
- Similarity of two clusters is based on the increase in squared error when two clusters are merged
  - Similar to group average if distance between points is distance squared
- Less susceptible to noise and outliers
- Biased towards globular clusters
- Hierarchical analogue of K-means
  - Can be used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



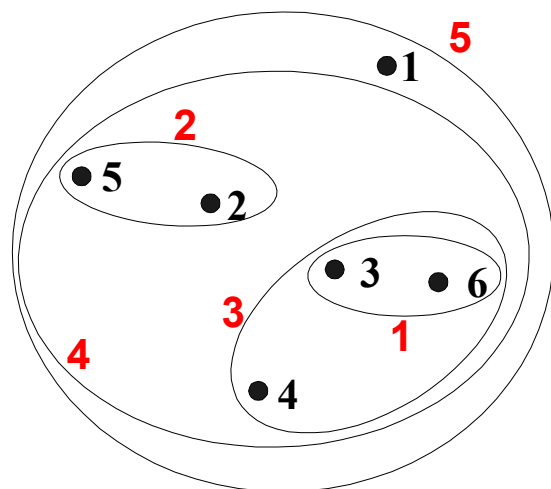
# Hierarchical Clustering: Comparison



MIN

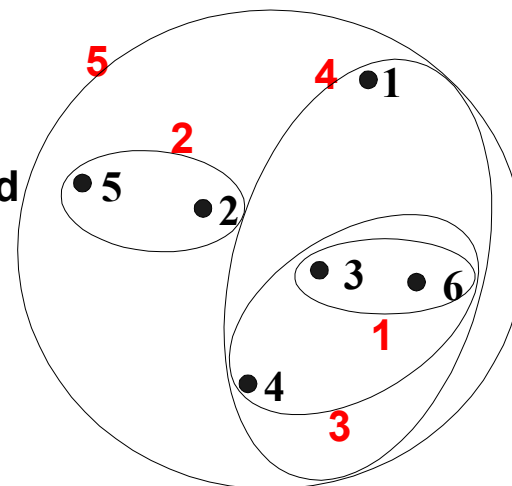


MAX



Group Average

Ward's Method



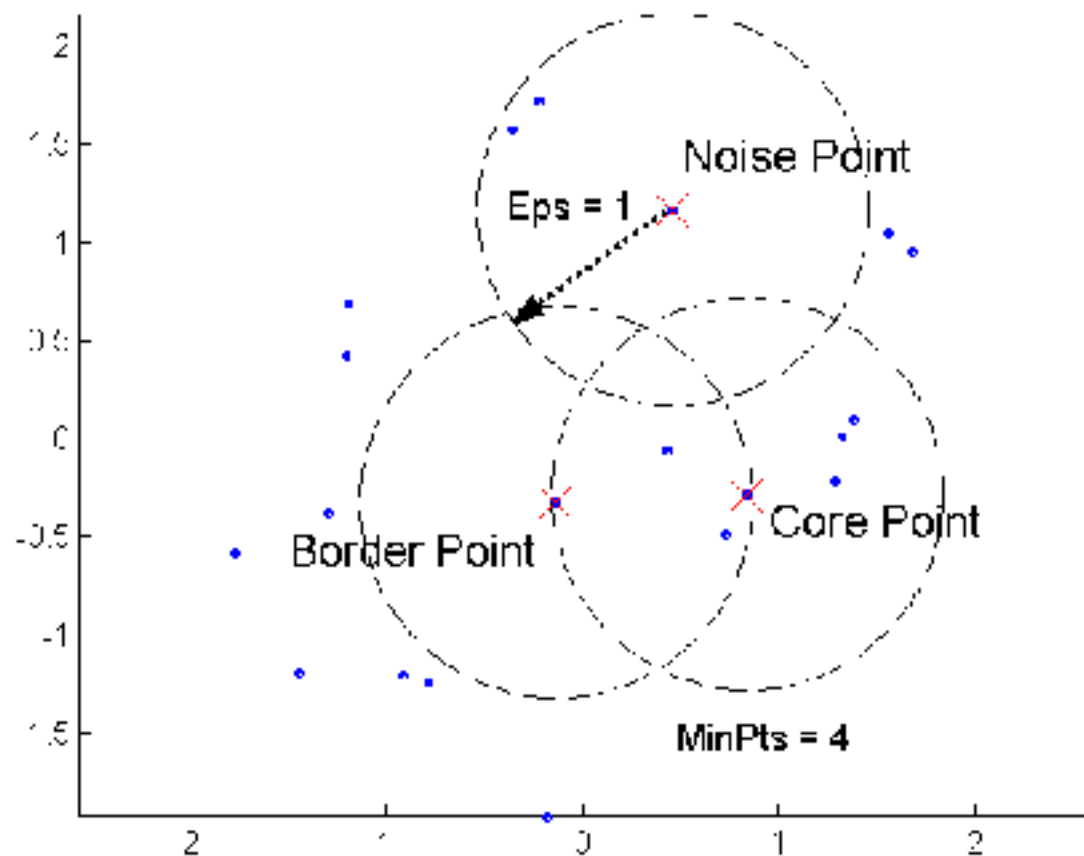
## **Hierarchical Clustering: Problems and Limitations**

- 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

# DBSCAN

- DBSCAN is a density-based algorithm.
  - Density = number of points within a specified radius (Eps)
  - A point is a **core point** if it has more than a specified number of points (MinPts) within Eps
    - These are points that are at the interior of a cluster
  - A **border point** has fewer than MinPts within Eps, but is in the 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 Algorithm

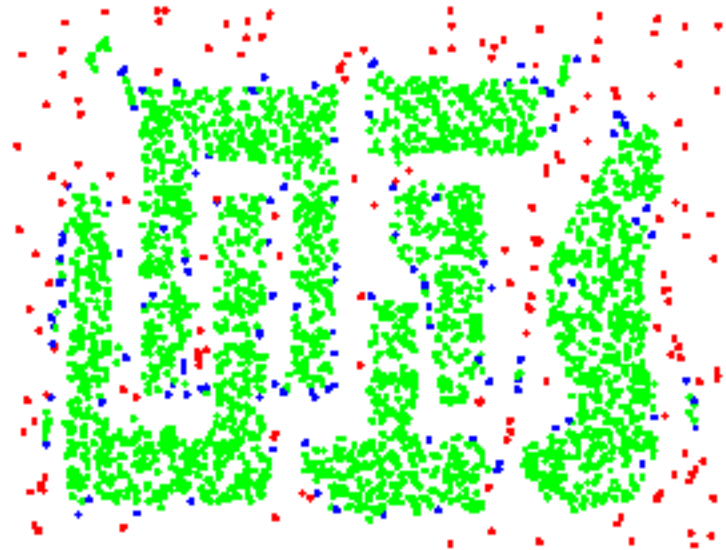
- Eliminate noise points
- Perform clustering on the remaining points

```
current_cluster_label  $\leftarrow$  1
for all core points do
    if the core point has no cluster label then
        current_cluster_label  $\leftarrow$  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



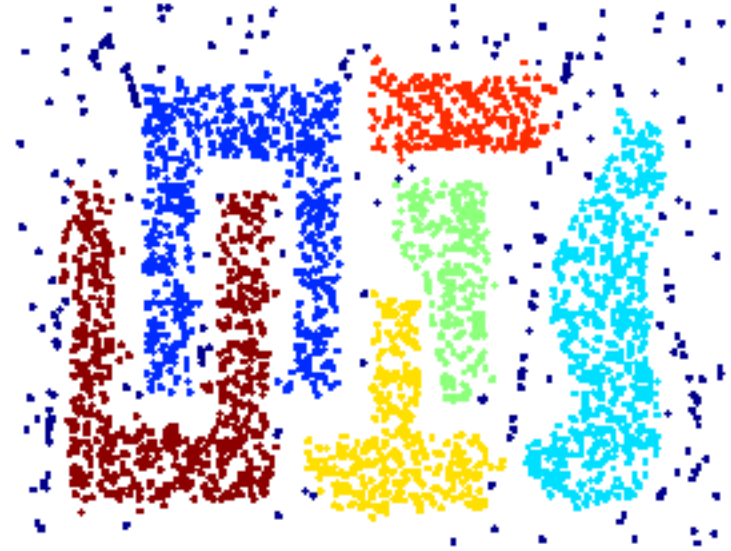
Point types: **core**,  
**border** and **noise**

Eps = 10, MinPts = 4

# When DBSCAN Works Well



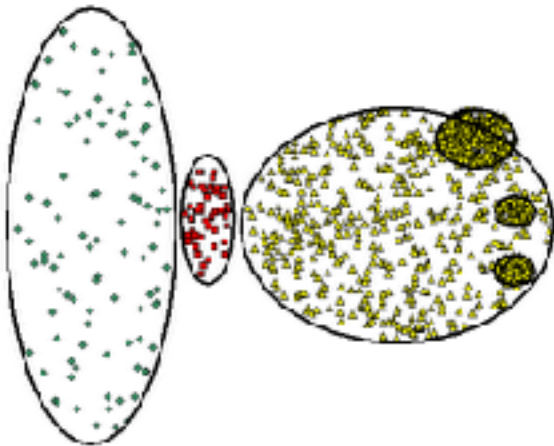
**Original Points**



**Clusters**

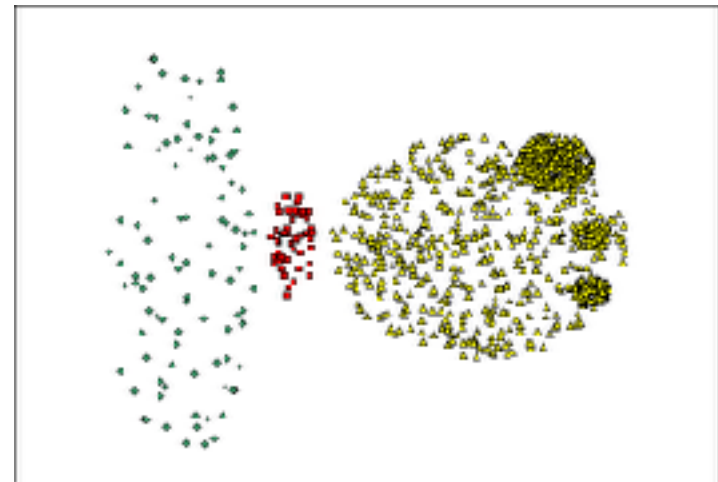
- **Resistant to Noise**
- **Can handle clusters of different shapes and sizes**

# When DBSCAN Does NOT Work Well

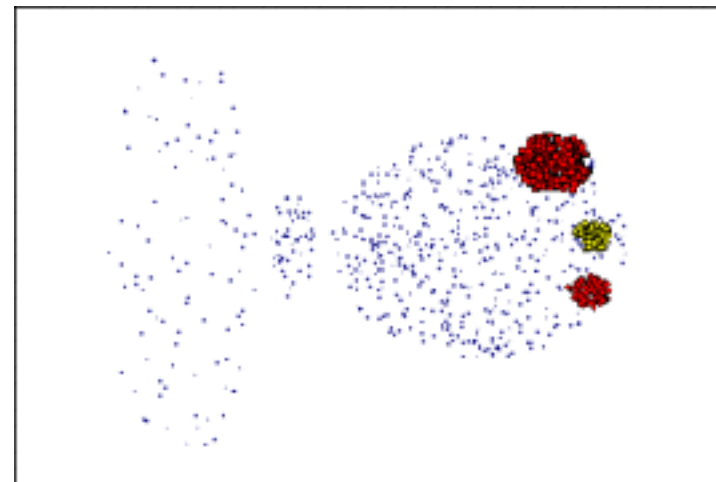


**Original Points**

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



(MinPts=4, Eps=9.75).

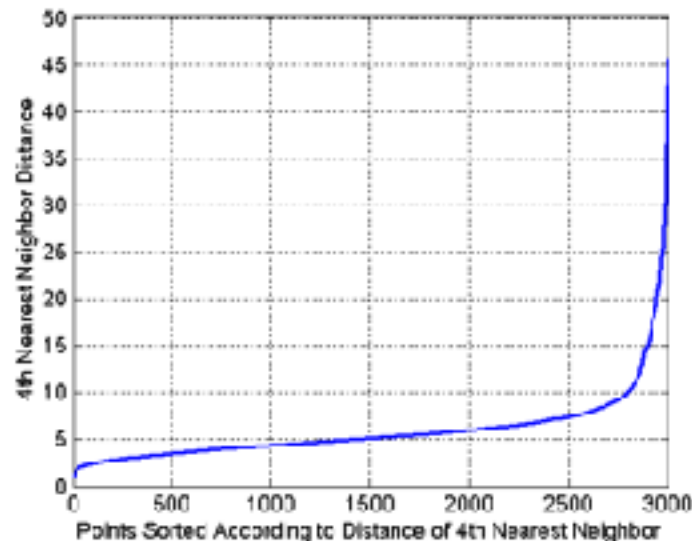


(MinPts=4, Eps=9.92)



# DBSCAN: Determining EPS and MinPts

- Idea is that for points in a cluster, their  $k^{\text{th}}$  nearest neighbors are at roughly the same distance
- Noise points have the  $k^{\text{th}}$  nearest neighbor at farther distance
- So, plot sorted distance of every point to its  $k^{\text{th}}$  nearest neighbor



# The Curse of Dimensionality

- 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.
  - Assuming random points within a bounding box, e.g., values between 0 and 1 in each dimension.