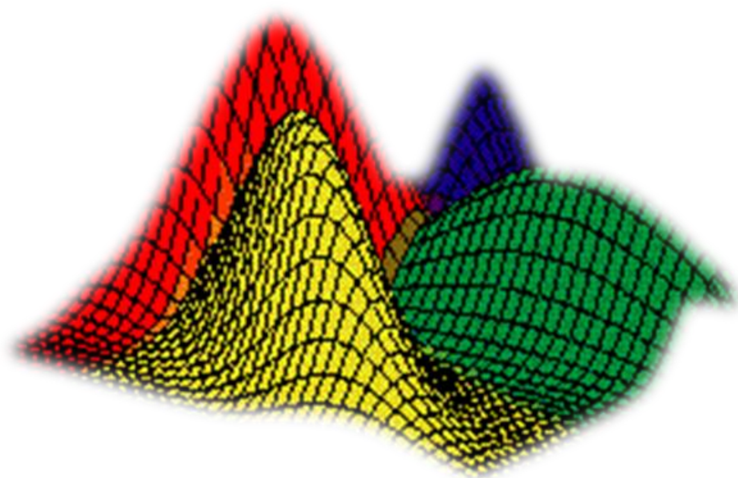


CSE 473: Pattern Recognition

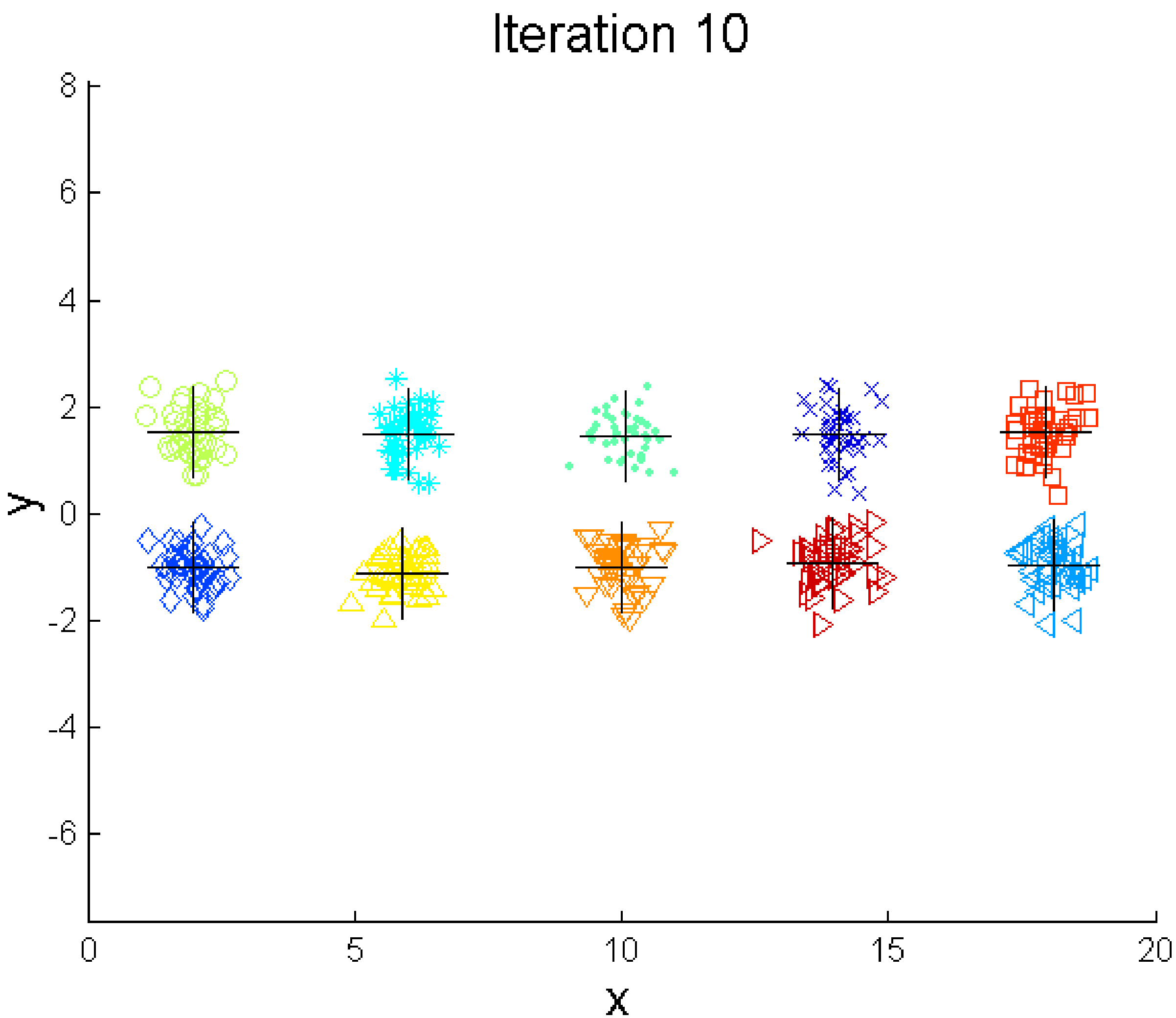
Unsupervised Learning:
Clustering

Bisecting *k*-means

- Bisecting *k*-means algorithm
  - Variant of *k*-means that can produce a partitional or a hierarchical clustering

```
Initialize the list of clusters to contain the cluster containing all points.
repeat
   Select a cluster from the list of clusters
   for i = 1 to number_of_iterations do
      Bisect the selected cluster using basic K-means
   end for
   Add the two clusters from the bisection with the lowest SSE to the list of clusters.
until Until the list of clusters contains K clusters
```

Bisecting *k*-means Example



Bisecting *k*-means as an initialization for a global *k*-means run

- bisecting *k*-means
  - bisects individual clusters
  - finds local minima of SSE
  - usually does not provide optimal clustering

Bisecting *k*-means as an initialization for a global *k*-means run

- bisecting *k*-means
  - bisects individual clusters
  - finds local minima of SSE
  - usually does not provide optimal clustering

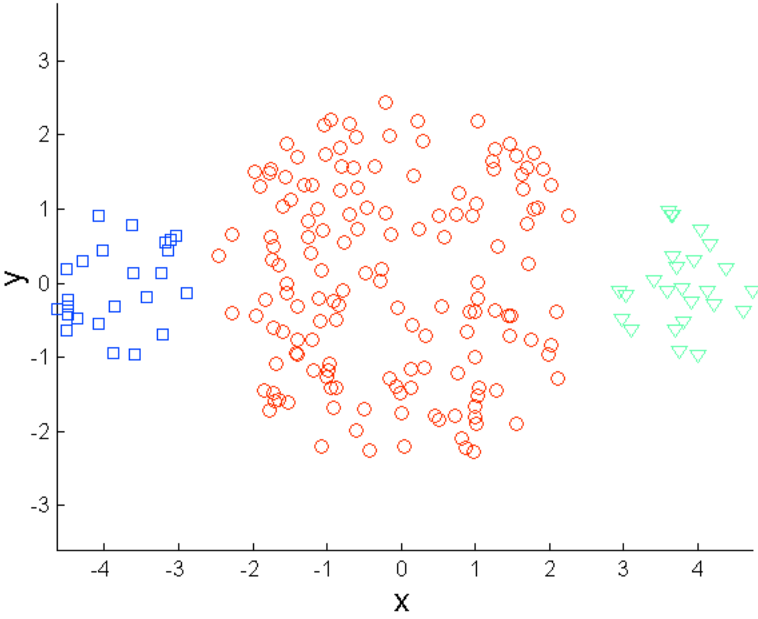
- final centroids from bisecting *k*-means can be used as initial centroids of a global *k*-means run

Limitations of *k*-means

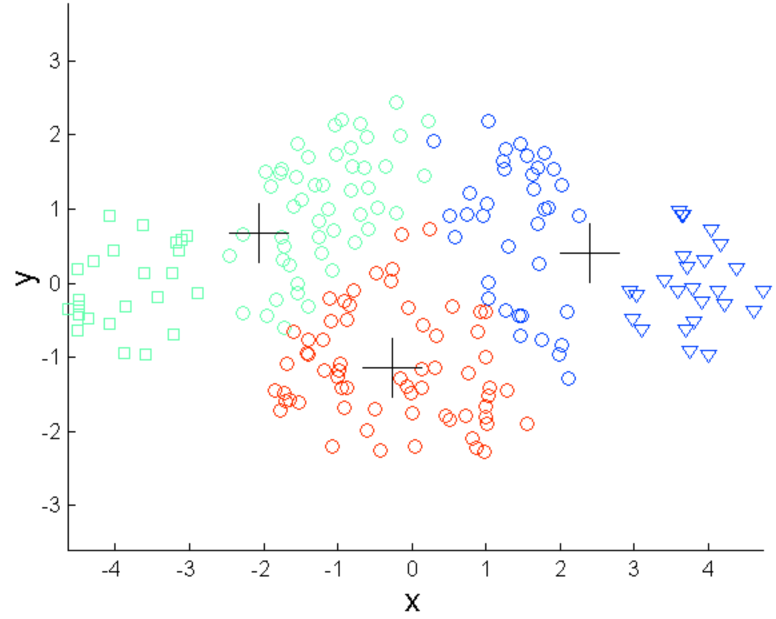
- *k*-means has problems when clusters are of differing
  - Sizes
  - Densities
  - Non-globular shapes

- *k*-means has problems when the data contains outliers.

Limitations of *k*-means: Differing Sizes

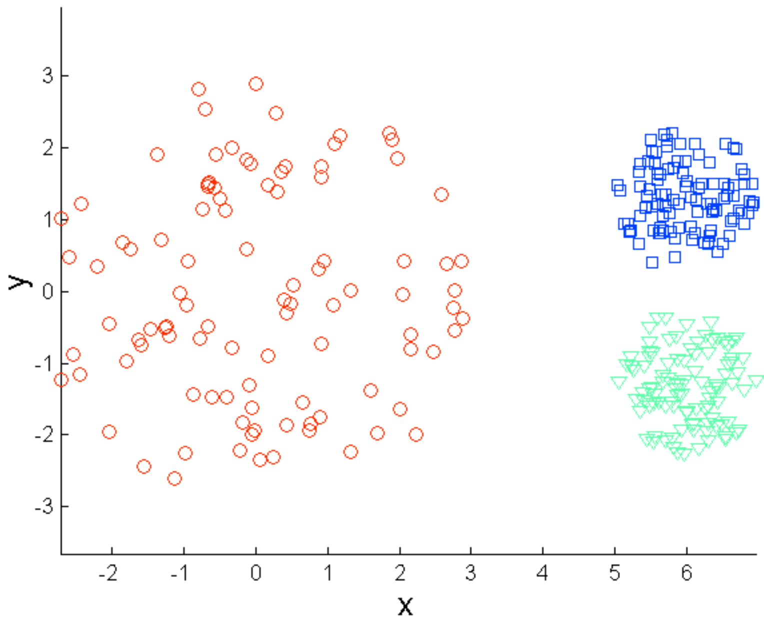


Original Points

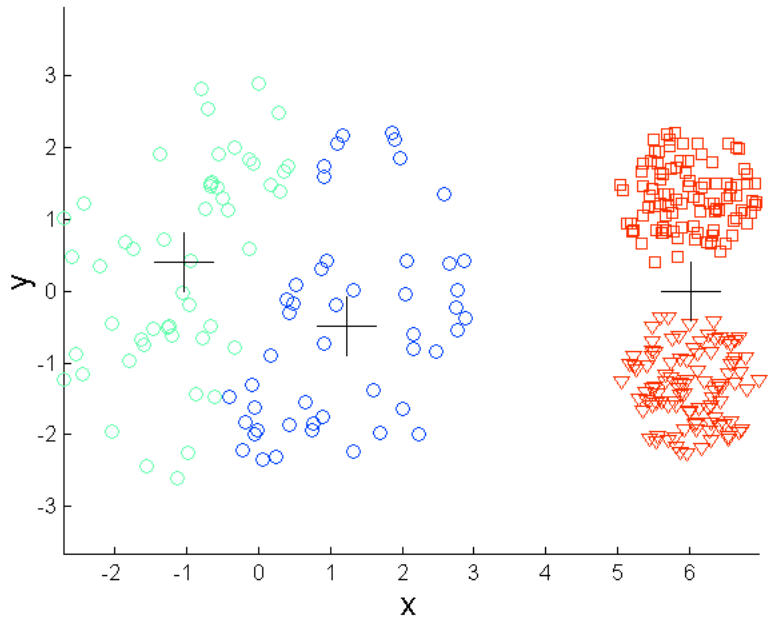


k-means (3 Clusters)

Limitations of *k*-means: Differing Density

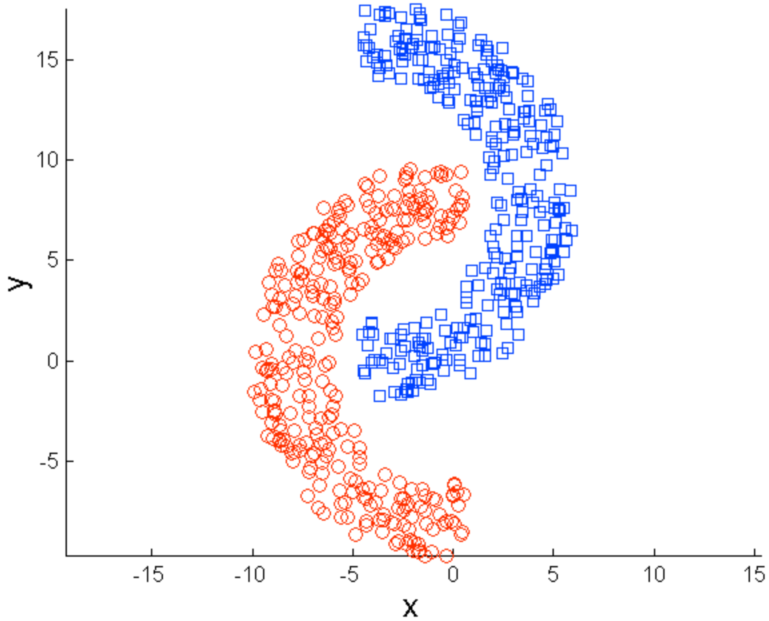


Original Points

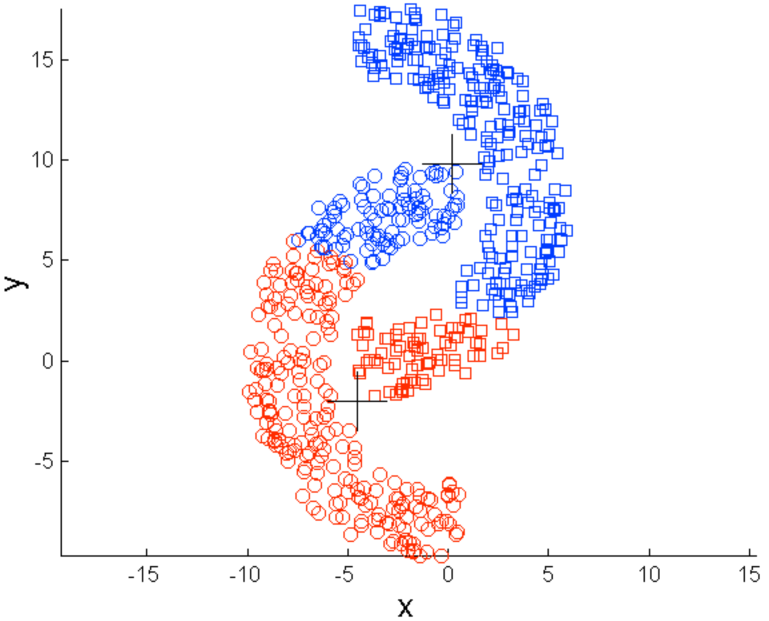


k-means (3 Clusters)

Limitations of *k*-means: Non-globular Shapes

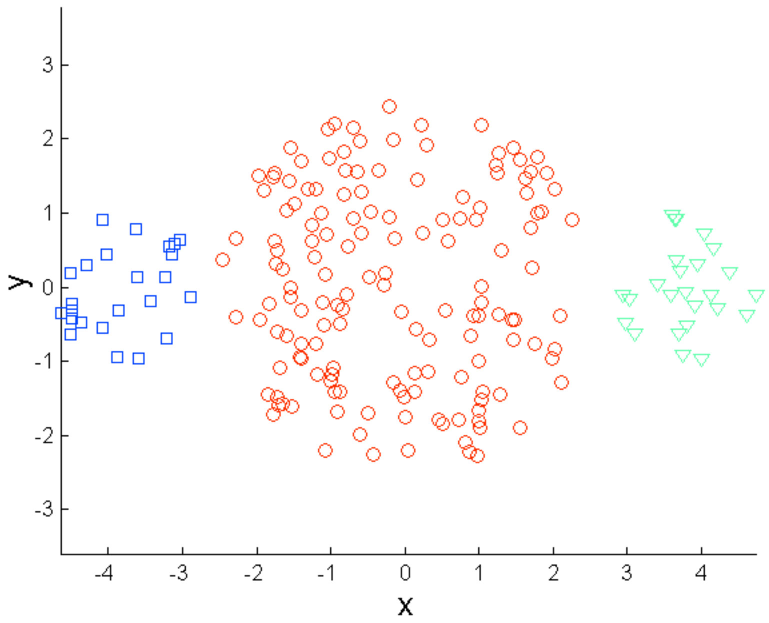


Original Points

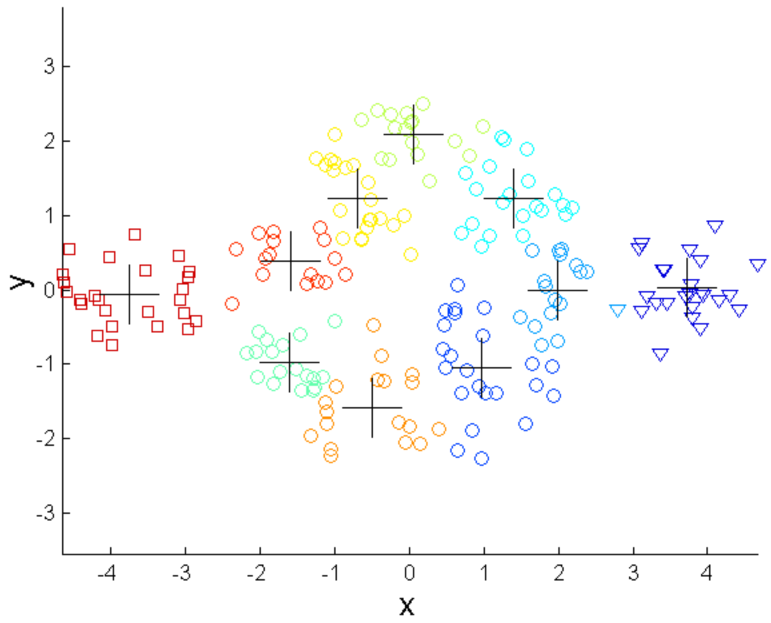


k-means (2 Clusters)

Overcoming *k*-means Limitations



Original Points

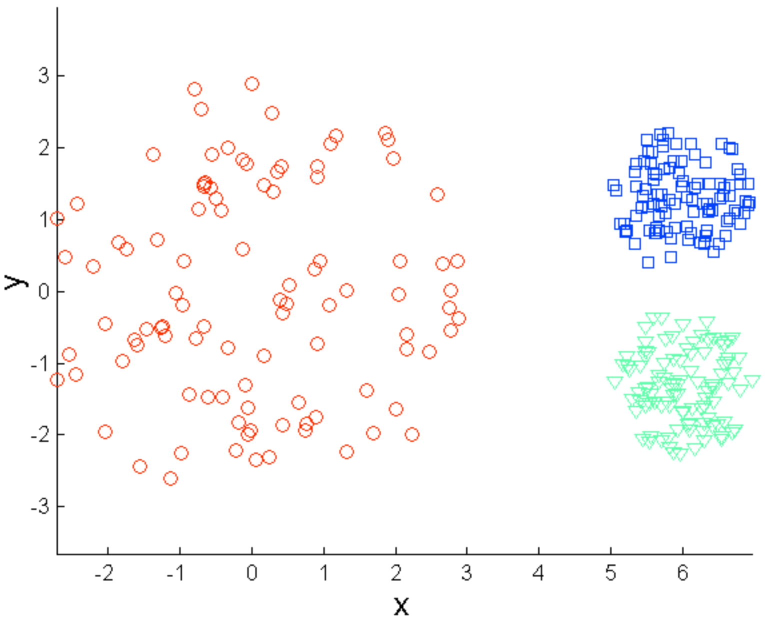


k-means Clusters

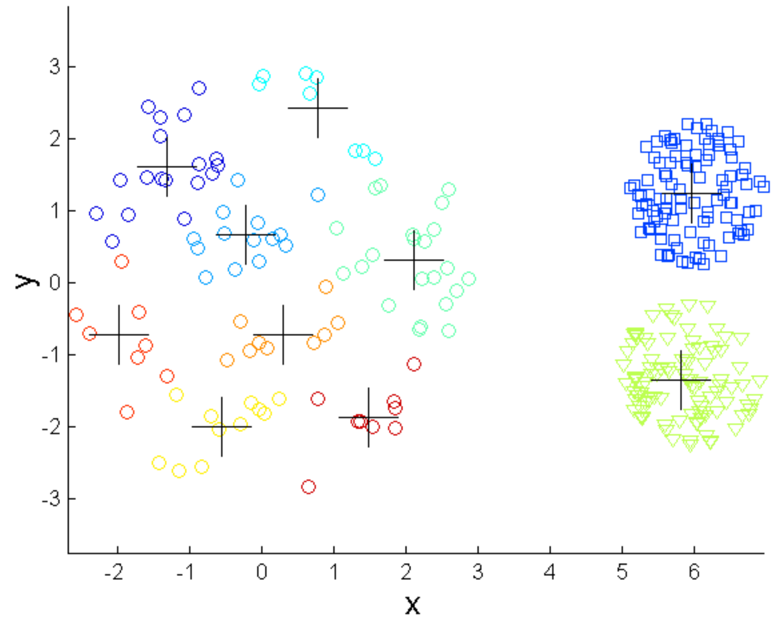
One solution is to use many clusters.

Find parts of clusters, but need to put together.

Overcoming *k*-means Limitations

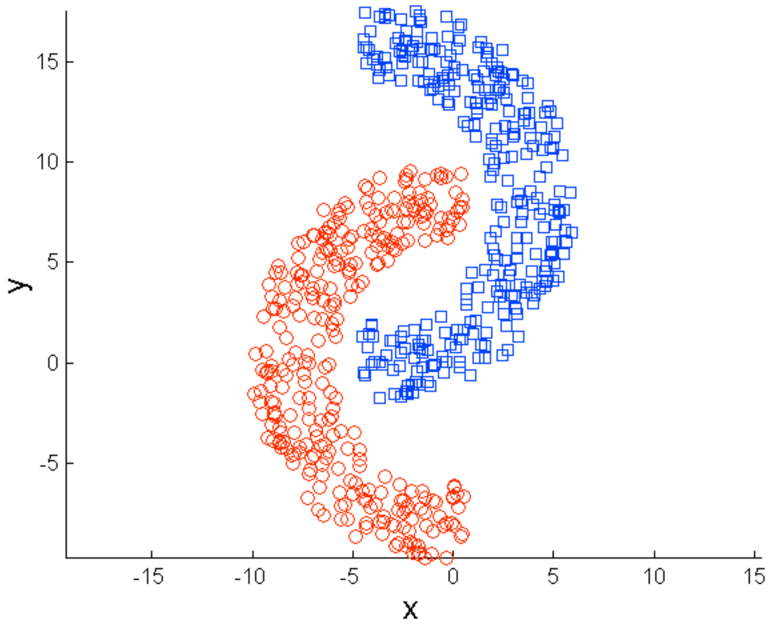


Original Points

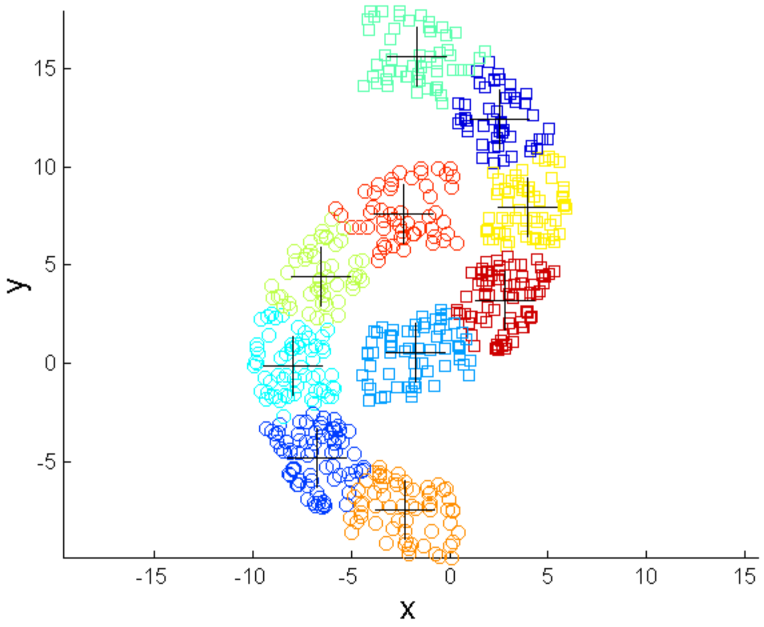


k-means Clusters

Overcoming *k*-means Limitations



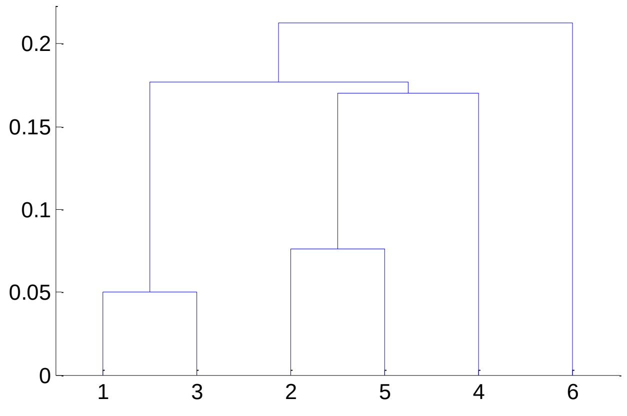
Original Points

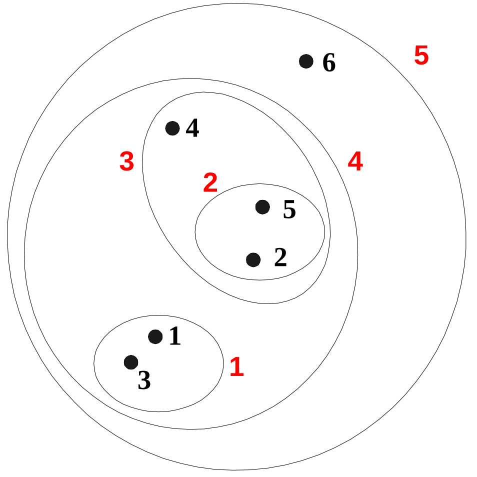


k-means Clusters

Hierarchical Clustering

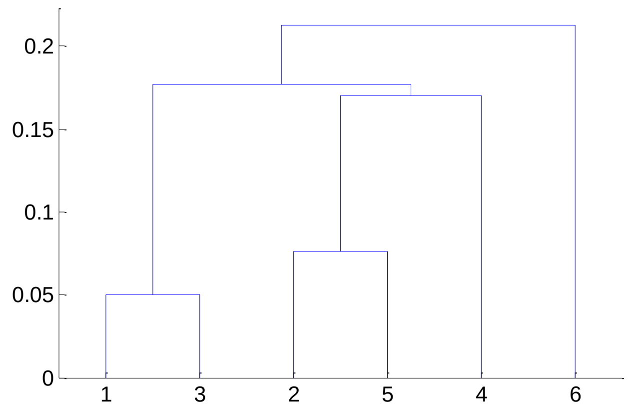
- Produces a set of nested clusters organized as a hierarchical tree
- Can be visualized as a dendrogram
  - A tree like diagram that records the sequences of merges or splits





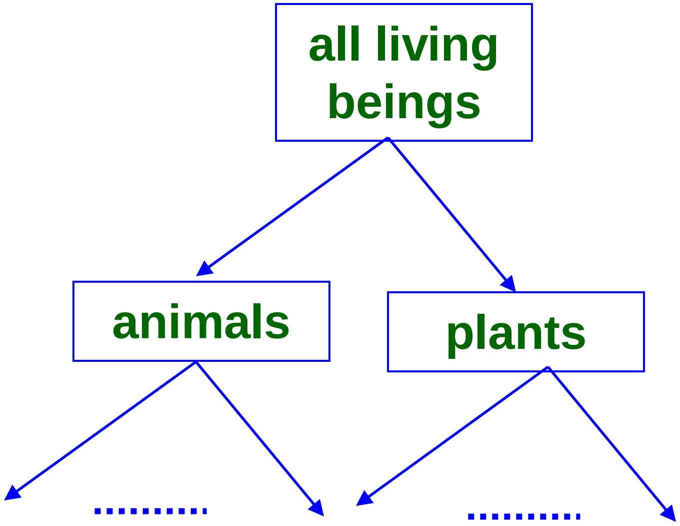
Strengths of Hierarchical Clustering

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



Strengths of Hierarchical Clustering

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



Hierarchical Clustering

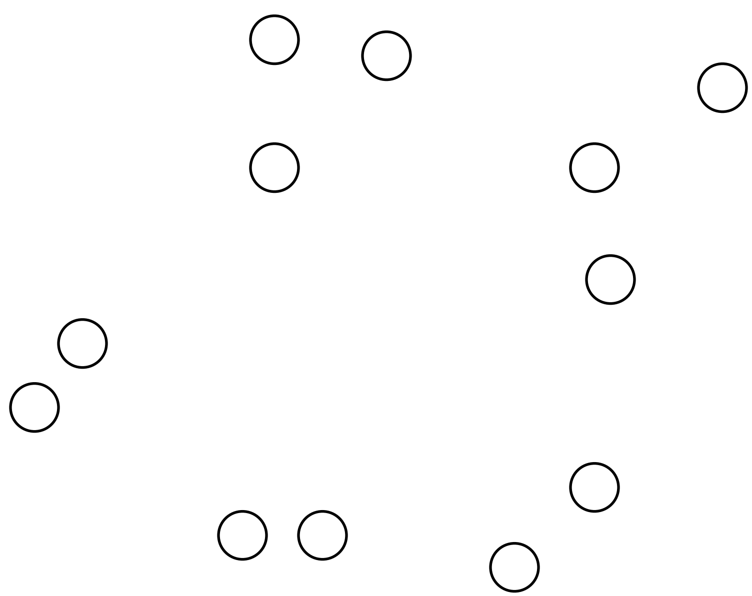
- Two main types of hierarchical clustering
  - Agglomerative:
    - Start with the points as individual clusters
    - At each step, merge the closest pair of clusters until only one cluster (or k clusters) left
  - Divisive:
    - Start with one, all-inclusive cluster
    - At each step, split a cluster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

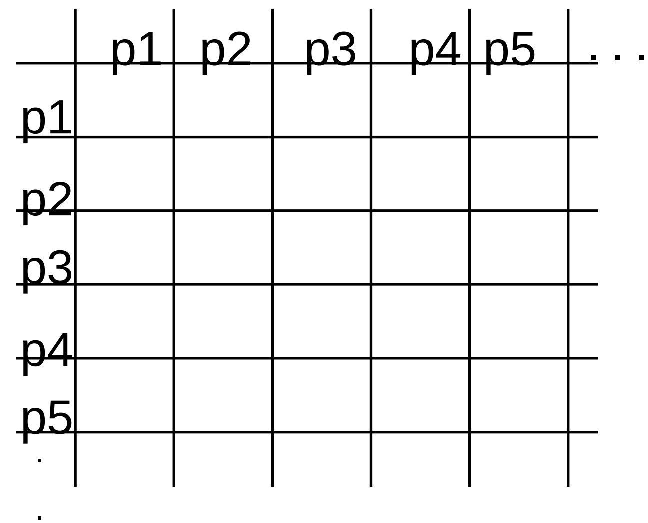
Agglomerative Clustering Algorithm

- 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 bet^{n} two clusters
  - Different approaches to defining the distance between clusters distinguish the different algorithms

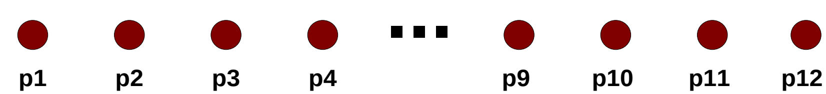
Starting Situation

- Start with clusters of individual points and a proximity matrix



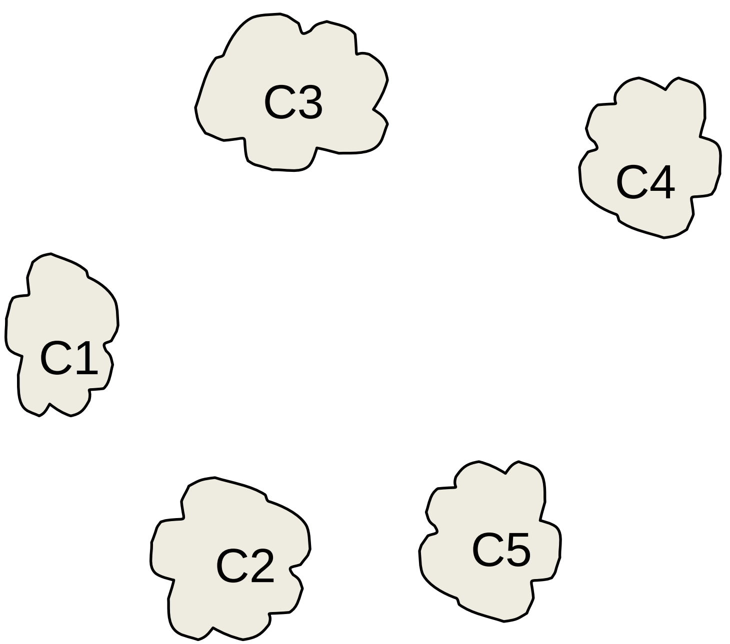


Proximity Matrix



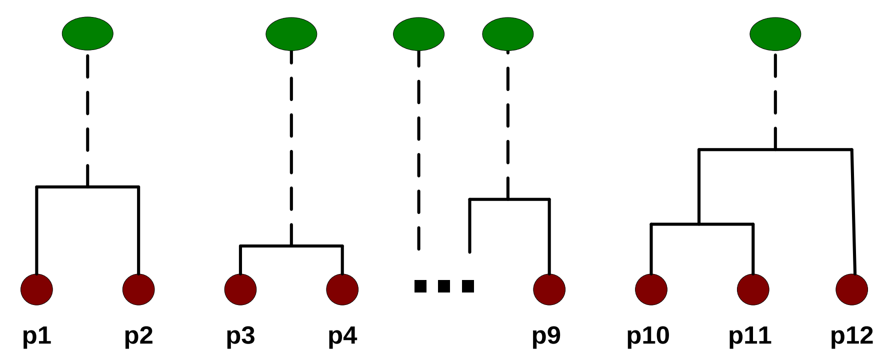
Intermediate Situation

- After some merging steps, we have some clusters



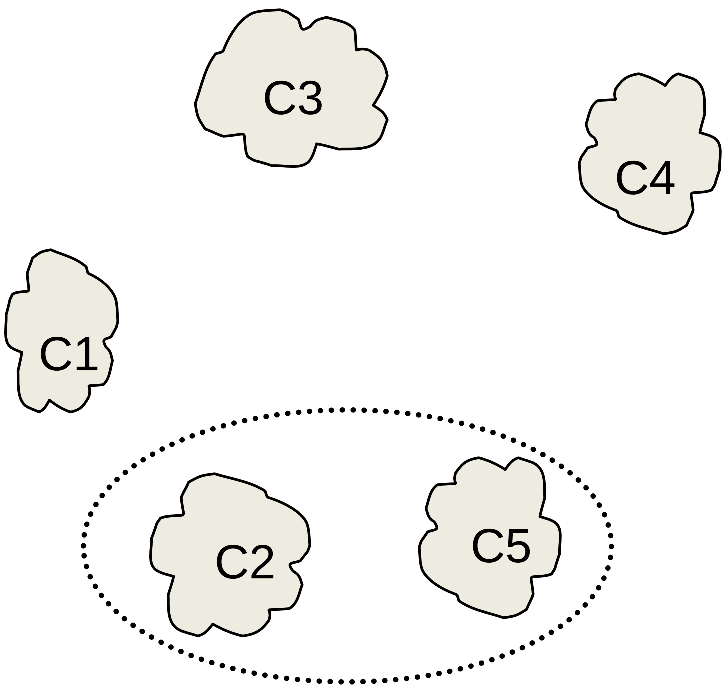
	C1	C2	C3	C4	C5
C1					
C2					
C3					
C4					
C5					

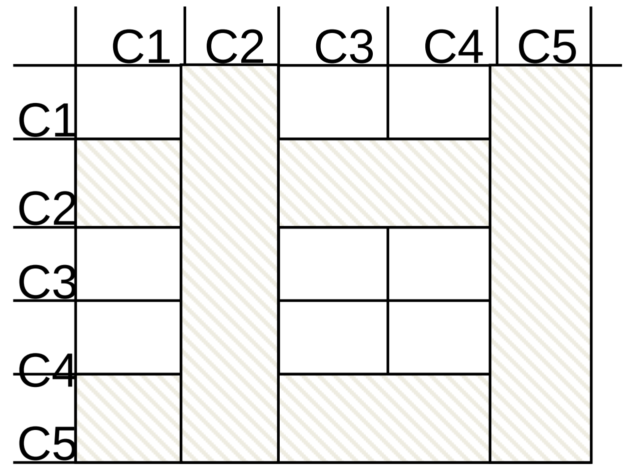
Proximity Matrix



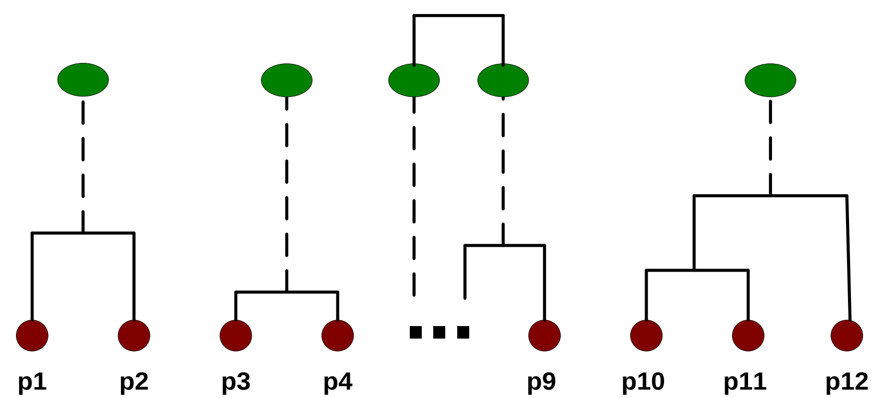
Intermediate Situation

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



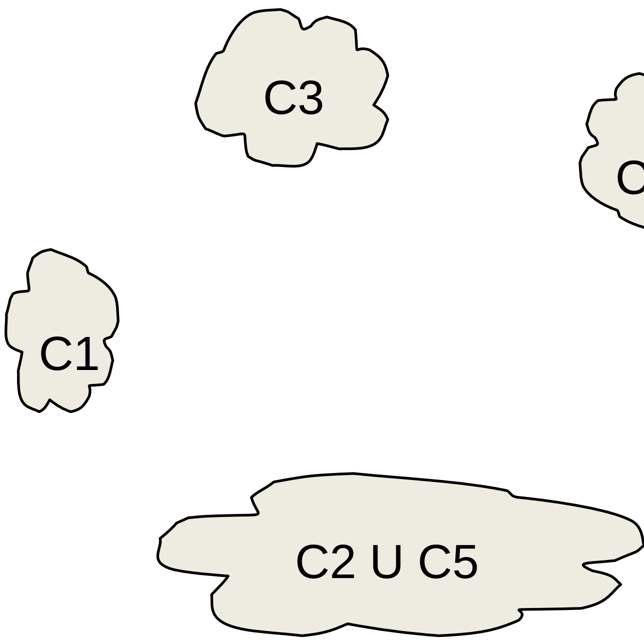


Proximity Matrix



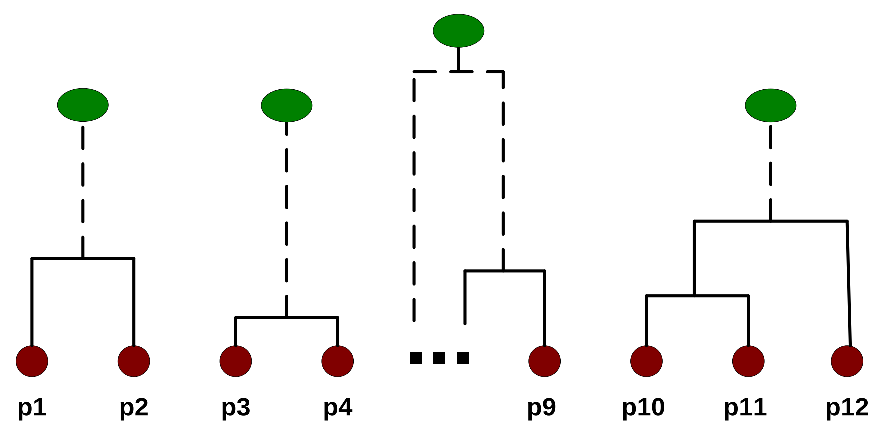
After Merging

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

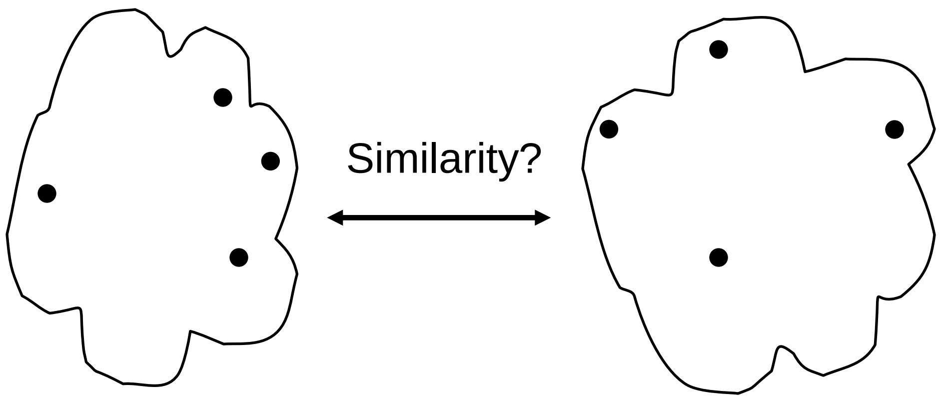


	C1	C2 U C5	C3	C4
C1		?		
C2 U C5	?	?	?	?
C3		?		
C4		?		

Proximity Matrix



How to Define Inter-Cluster Similarity

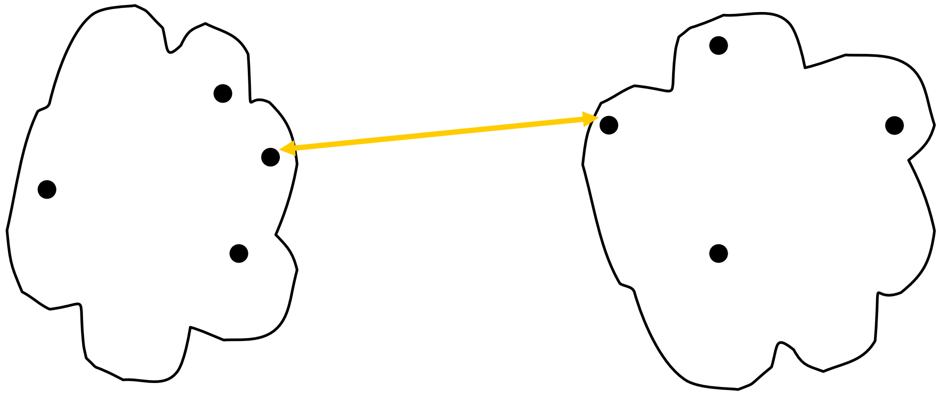


	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

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

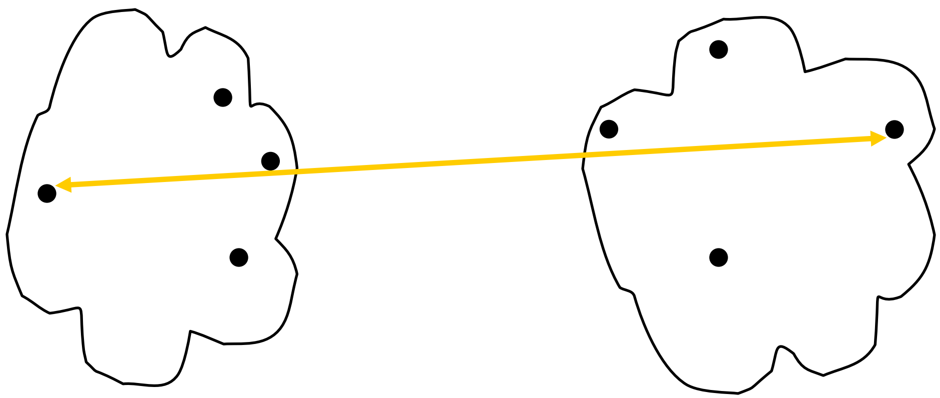


	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

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

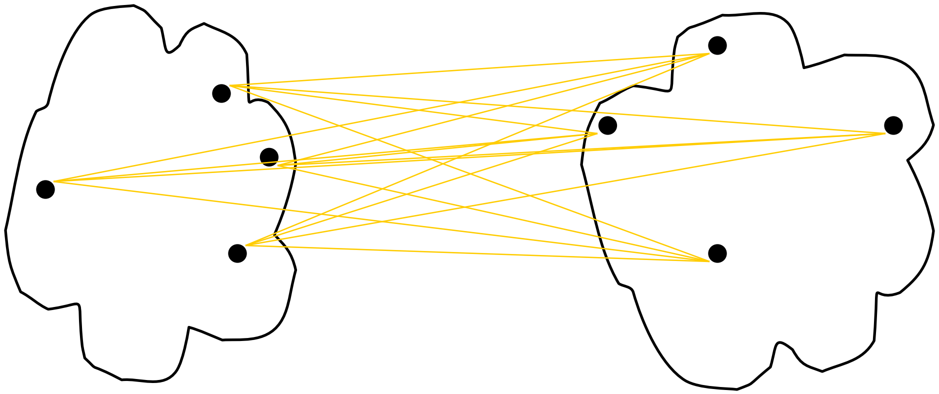


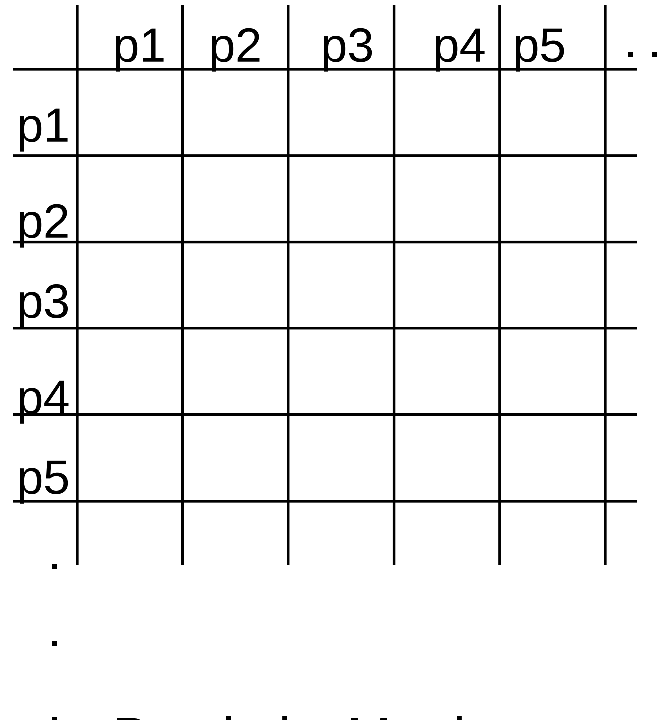
	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

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

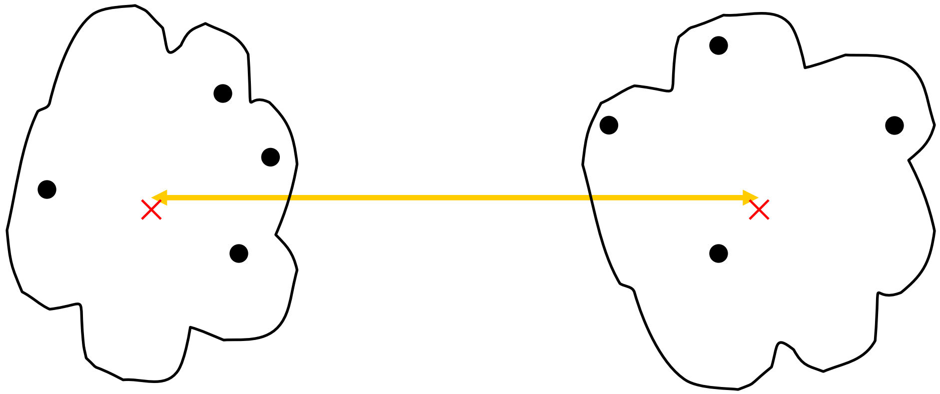




Proximity Matrix

- 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						
.						
.						
.						

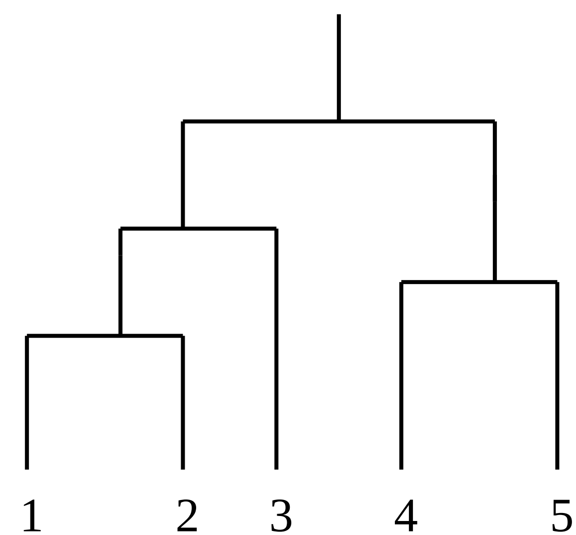
Proximity Matrix

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

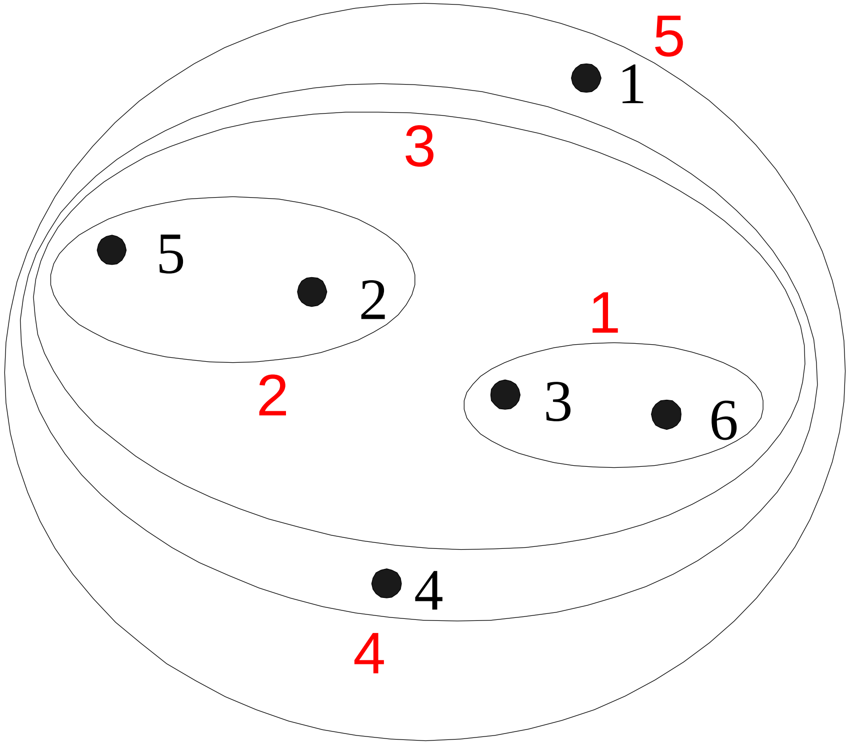
Cluster Similarity: MIN or Single Link

- Similarity of two clusters is based on the two most similar (closest) points in the different clusters

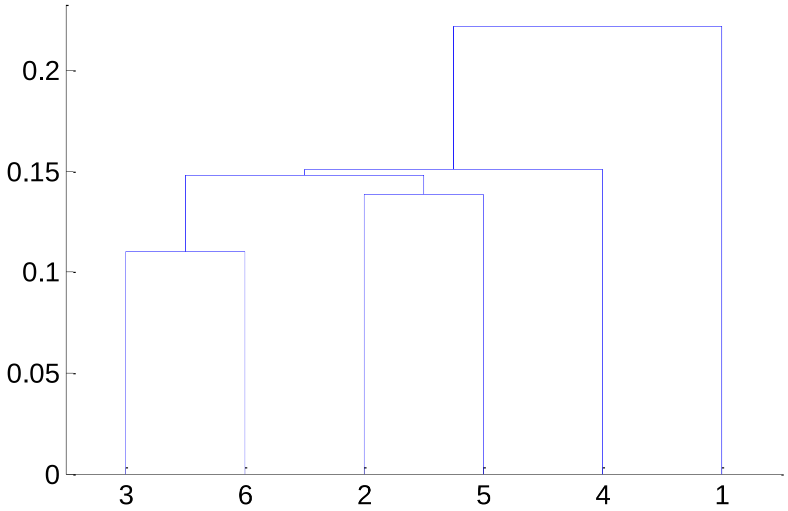
	I1	I2	I3	I4	I5
I1	1.00	0.90	0.10	0.65	0.20
I2	0.90	1.00	0.70	0.60	0.50
I3	0.10	0.70	1.00	0.40	0.30
I4	0.65	0.60	0.40	1.00	0.80
I5	0.20	0.50	0.30	0.80	1.00



Hierarchical Clustering: MIN



Nested Clusters

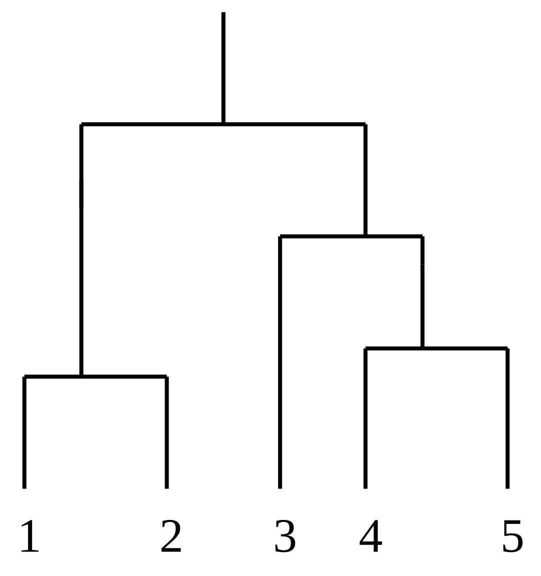


Dendrogram

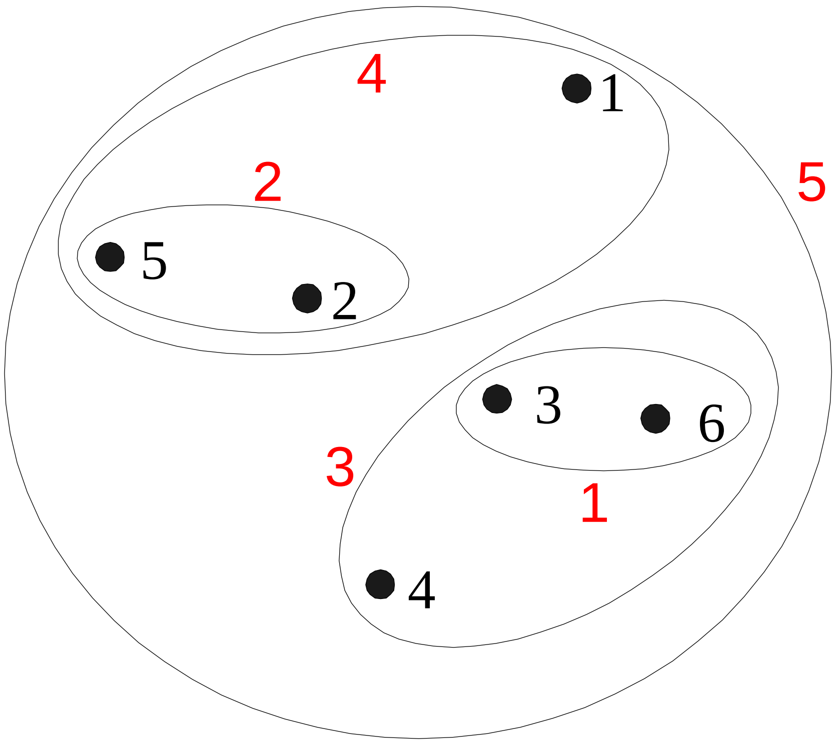
Cluster Similarity: MAX or Complete Linkage

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

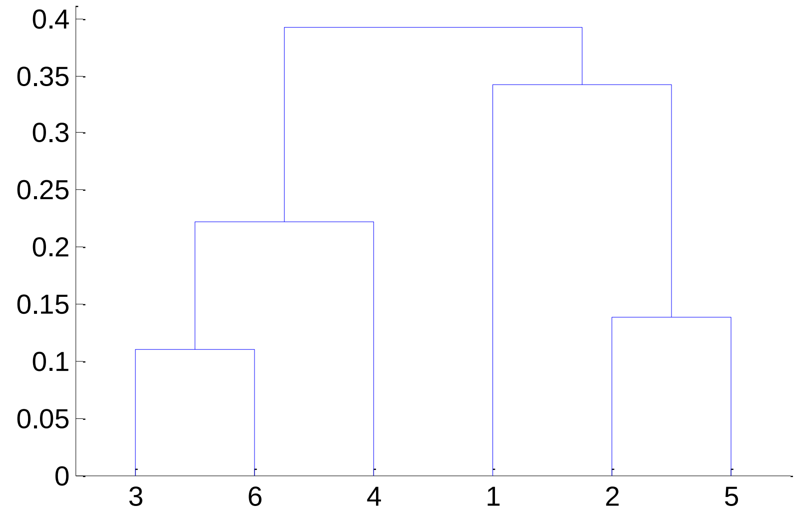
	I1	I2	I3	I4	I5
I1	1.00	0.90	0.10	0.65	0.20
I2	0.90	1.00	0.70	0.60	0.50
I3	0.10	0.70	1.00	0.40	0.30
I4	0.65	0.60	0.40	1.00	0.80
I5	0.20	0.50	0.30	0.80	1.00



Hierarchical Clustering: MAX



Nested Clusters



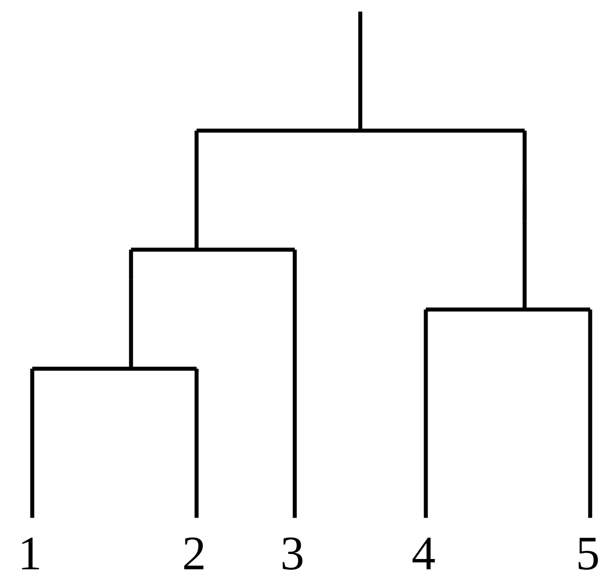
Dendrogram

Cluster Similarity: Group Average

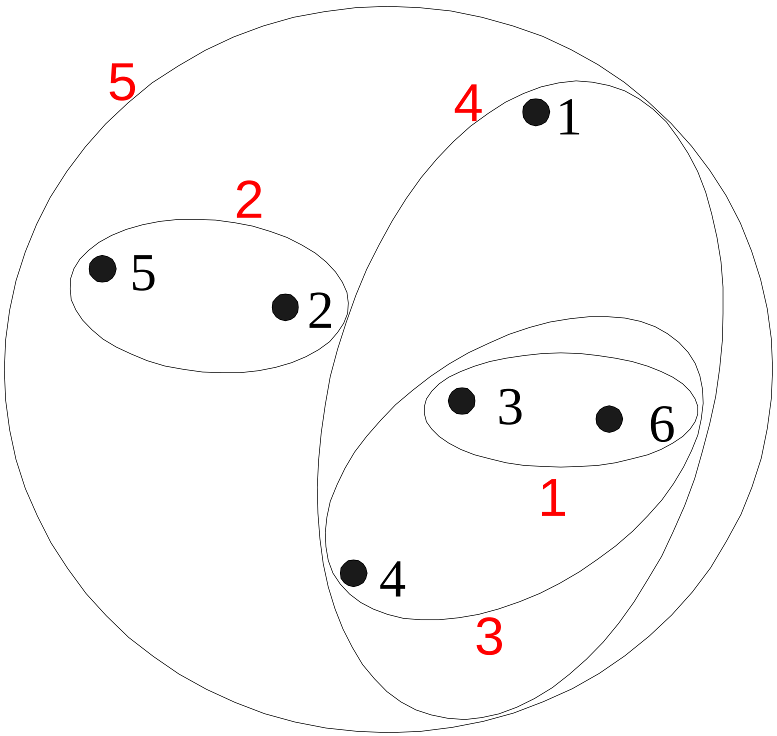
- Proximity of two clusters is the average of pairwise proximity between points in the two clusters.

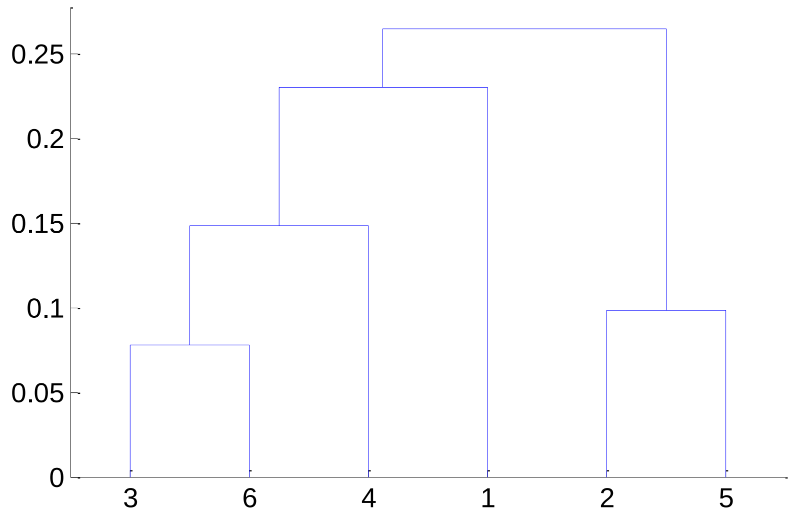
$$\mathbf{proximity}(\mathbf{Cluster}_i, \mathbf{Cluster}_j) = \frac{\sum\limits_{\substack{p_i \in \mathbf{Cluster}_i \\ p_j \in \mathbf{Cluster}_j}} \mathbf{proximity}(p_i, p_j)}{|\mathbf{Cluster}_i| \times |\mathbf{Cluster}_j|}$$

	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

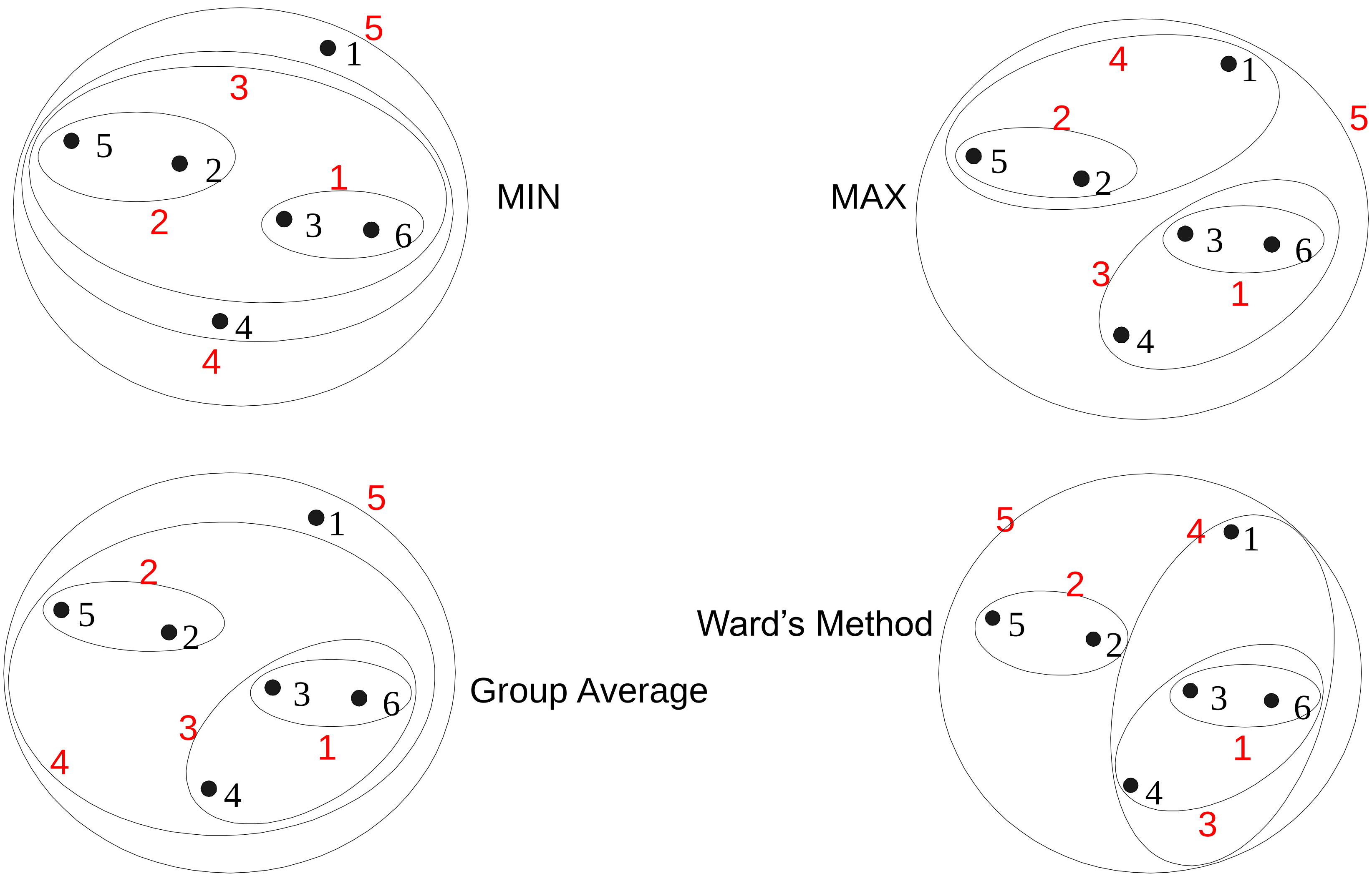
- Strengths
  - Less susceptible to noise and outliers

- Limitations
  - Biased towards globular clusters

Cluster Similarity: Ward's Method

- Proximity 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 Clustering: Comparison



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
  - Difficult to handle different sized clusters and convex shapes
  - Breaking large clusters

Density based Clustering

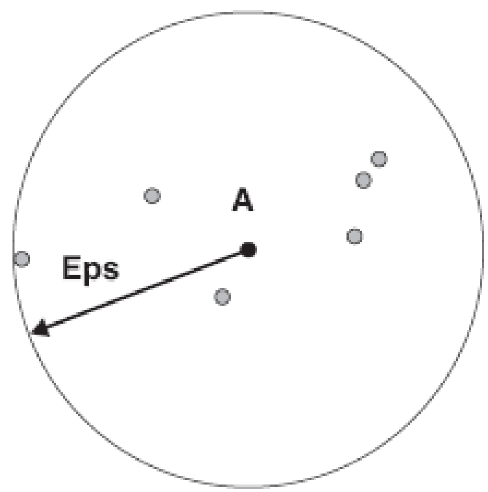
- locates high density regions in low density regions
- requires to define
  - What is the density

DBSCAN

- DBSCAN is a density-based algorithm.
  - Density = number of points within a specified radius (Eps)

DBSCAN

- DBSCAN is a density-based algorithm.
  - Density = number of points within a specified radius (Eps)

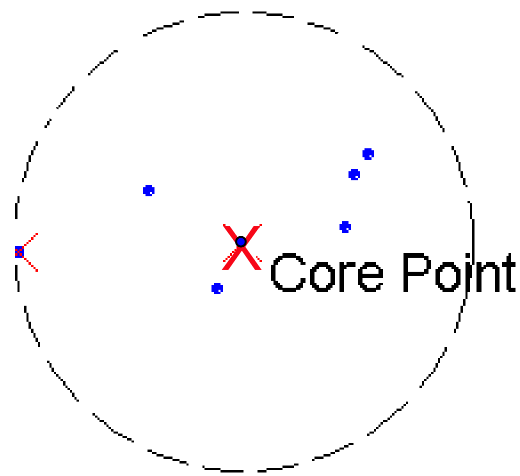


DBSCAN

- DBSCAN is a density-based algorithm.
  - Density = number of points within a specified radius (Eps)
  - density is calculated based on
  - core point
  - border point
  - noise point

DBSCAN

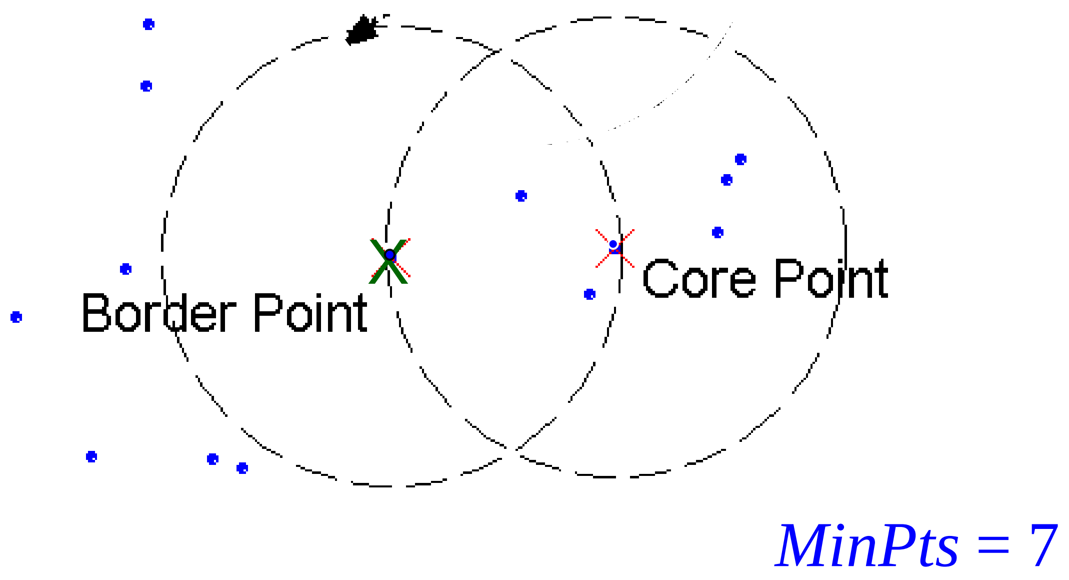
– A point is a core point if it has more than a *specified number of points* (*MinPts*) within *Eps*
  - These are points that are at the interior of a cluster



MinPts = 7

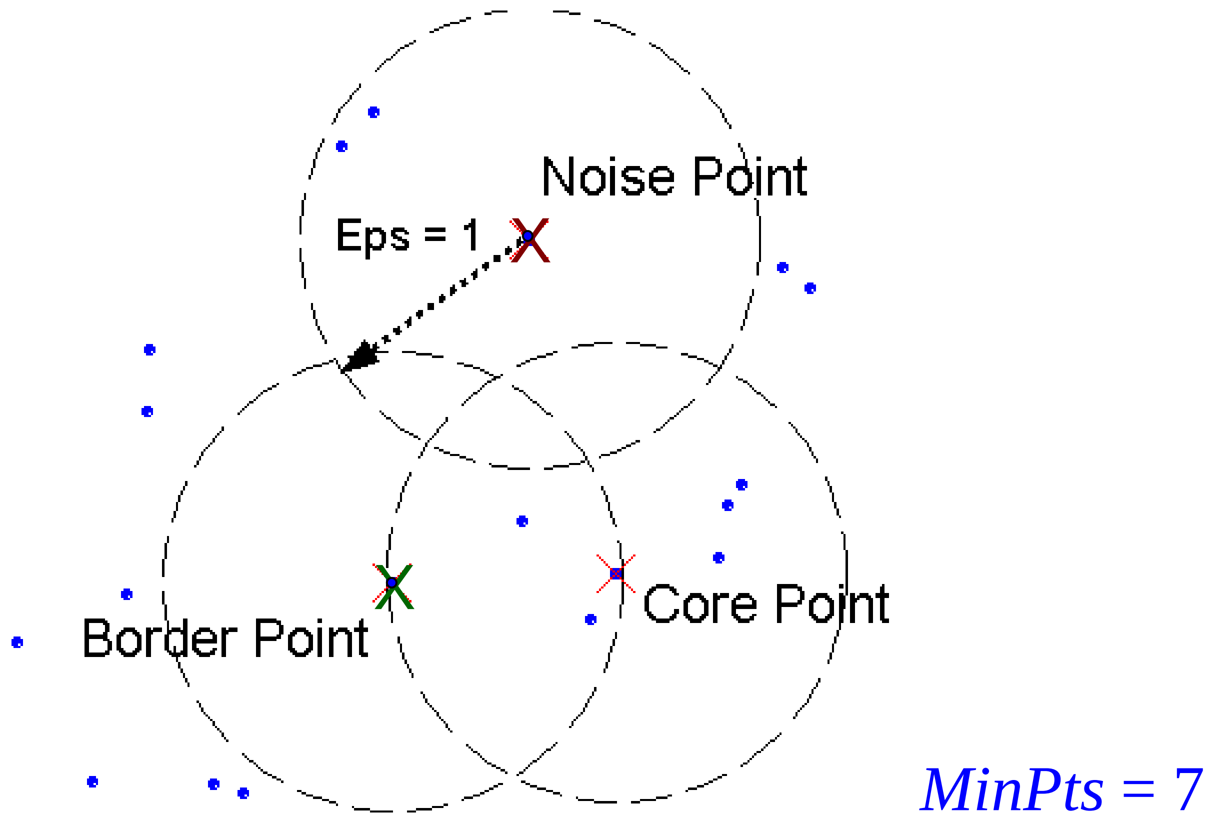
DBSCAN

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



DBSCAN

– A noise point is any point that is not a core point or a border point.



DBSCAN Algorithm

- Eliminate noise points
- Perform clustering on the remaining points
  - Put an edge between all core points which are within *Eps*
  - Make each group of core points as a cluster
  - Assign border point to one of the clusters of its associated core points

DBSCAN: Core, Border and Noise Points

Original Points

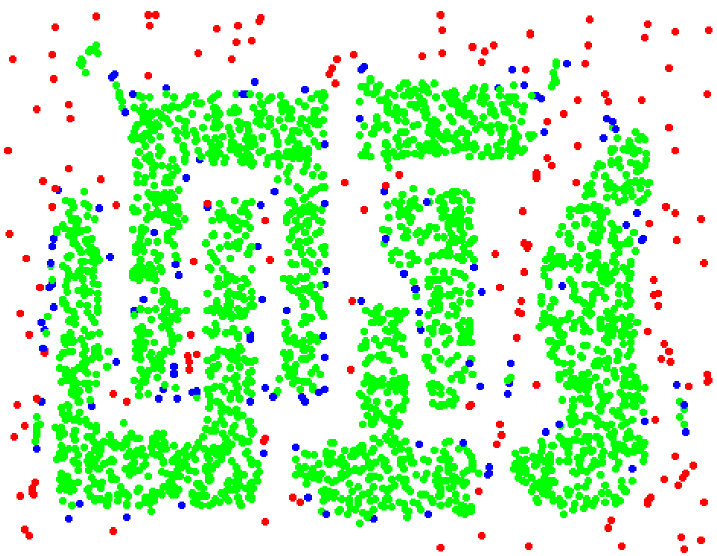
DBSCAN: Core, Border and Noise Points

Original Points

Eps = 10, MinPts = 4

DBSCAN: Core, Border and Noise Points

Original Points

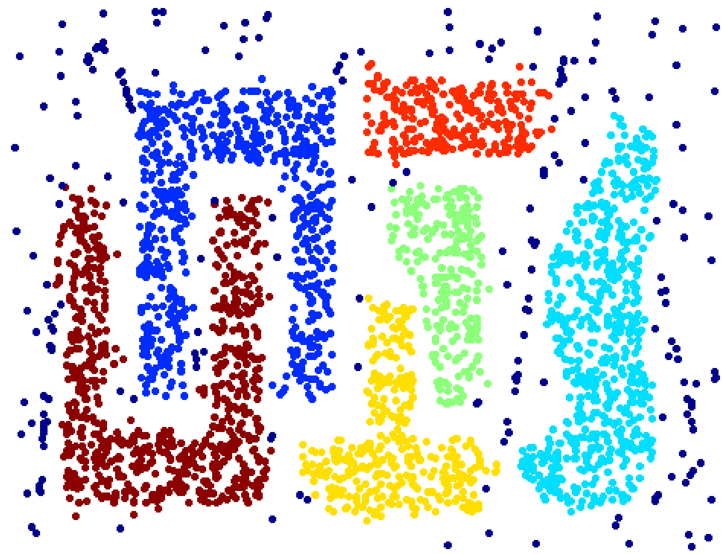


Point types: **core**, **border** and **noise**

Eps = 10, MinPts = 4

When DBSCAN Works Well

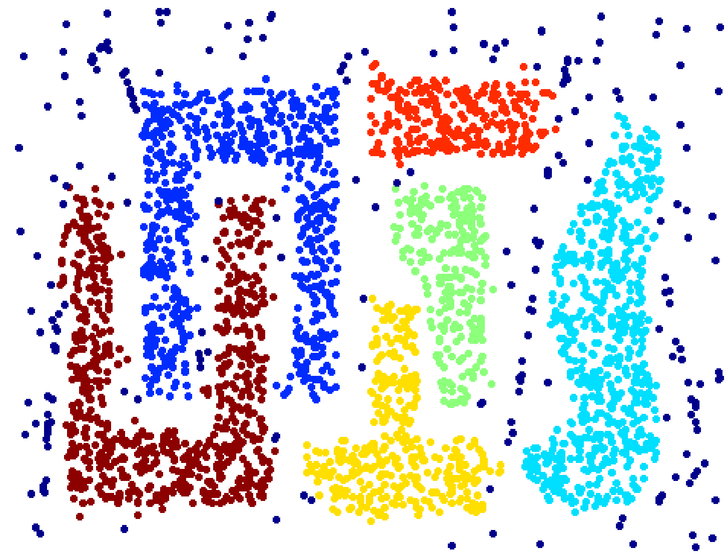
Original Points



Clusters

When DBSCAN Works Well

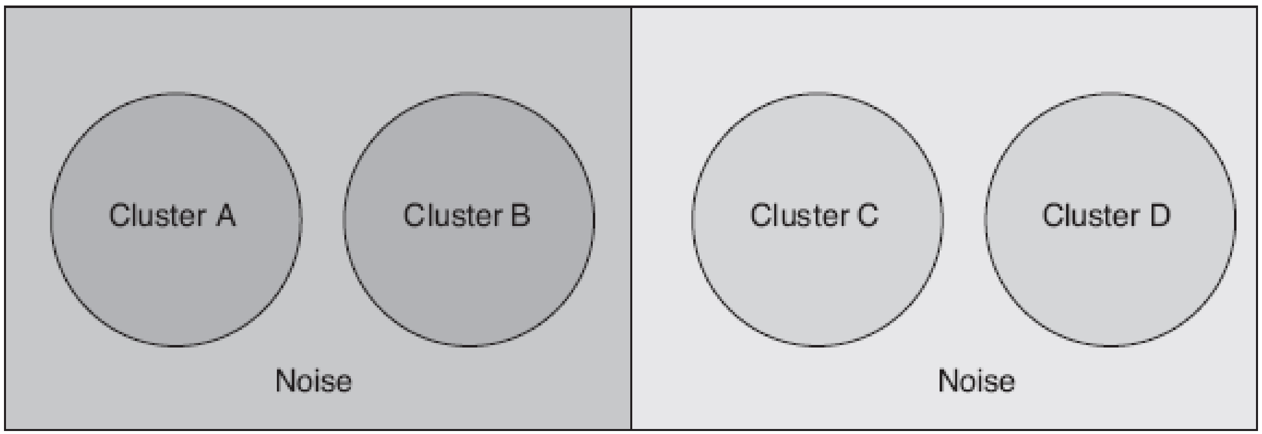
Original Points



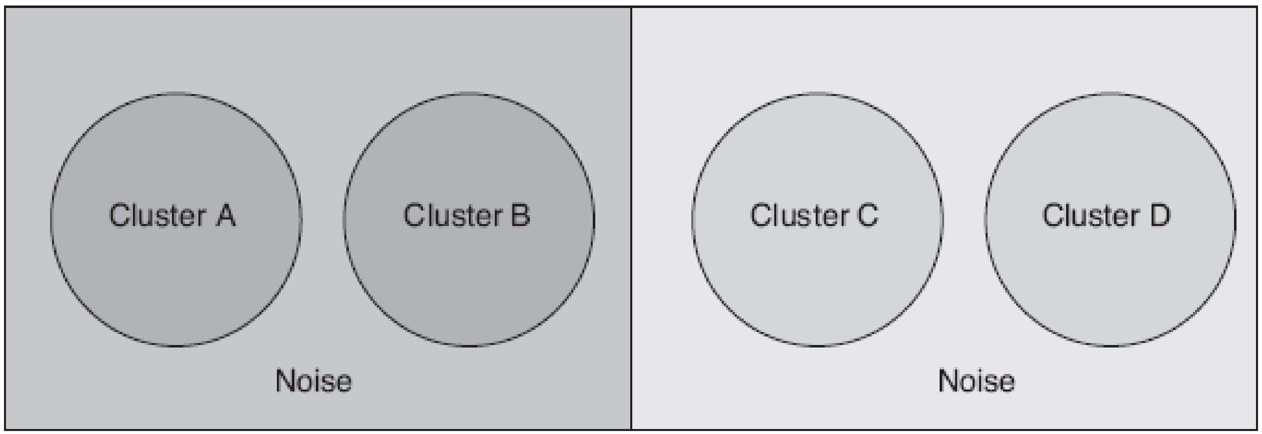
Clusters

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

When DBSCAN Does NOT Work Well

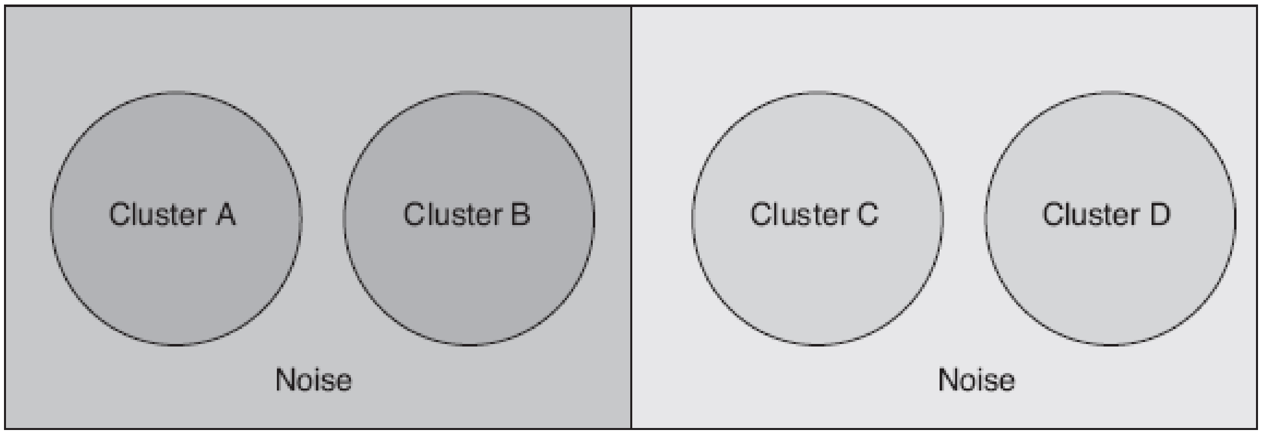


When DBSCAN Does NOT Work Well



- when *Eps* is low enough
  - successfully finds low density clusters C and D
  - considers the left side as a single cluster

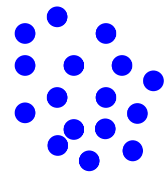
When DBSCAN Does NOT Work Well



- when *Eps* is high enough
  - successfully finds high density clusters A, B
  - considers others as noise

DBSCAN: Determining EPS and MinPts

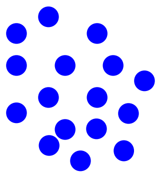
DBSCAN: Determining EPS and MinPts



Note the *k-dist* of these cluster points

K-dist: distance from *k*th nearest neighbour

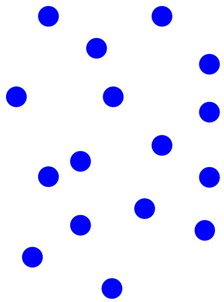
DBSCAN: Determining EPS and MinPts



Note the *k-dist* of these cluster points

K-dist should be *very similar*

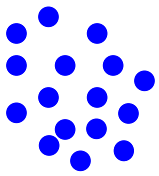
DBSCAN: Determining EPS and MinPts



K-dist should be *very similar*

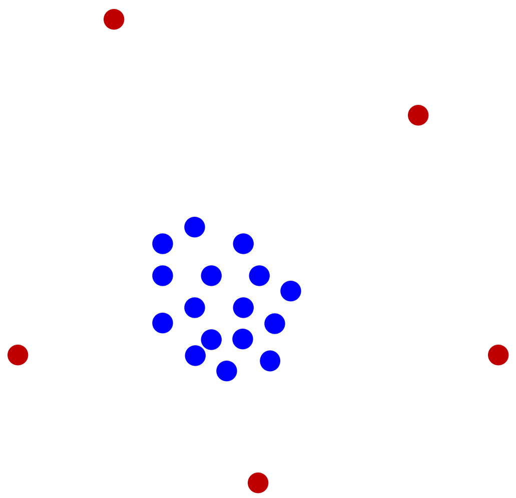
Unless densities change significantly

DBSCAN: Determining EPS and MinPts



K-dist should be *very similar*

DBSCAN: Determining EPS and MinPts



K-dist should be *very similar*

However, *k-dist* for noise/outlier will be different

DBSCAN: Determining EPS and MinPts

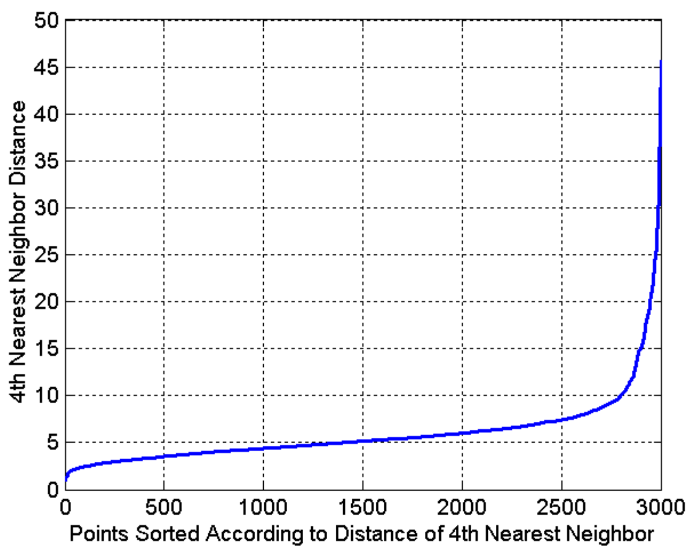
- For points in a cluster, their k^{th} nearest neighbors are at roughly the same distance
- Noise points have the k^{th} nearest neighbor at farther distance
- So, plot sorted distance of every point to its k^{th} nearest neighbor

DBSCAN: Determining EPS and MinPts

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

DBSCAN: Determining EPS and MinPts

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



DBSCAN: Problems and Limitations

- Resistant to noise
- Handle different sizes of clusters
- Problems with the following:
  - Different densities
  - Density/proximity analysis for high dimension