IT/PC/B/T/411

Machine Learning

Clustering: Basic Concepts and Algorithms

Dr. Pawan Kumar Singh

Department of Information Technology

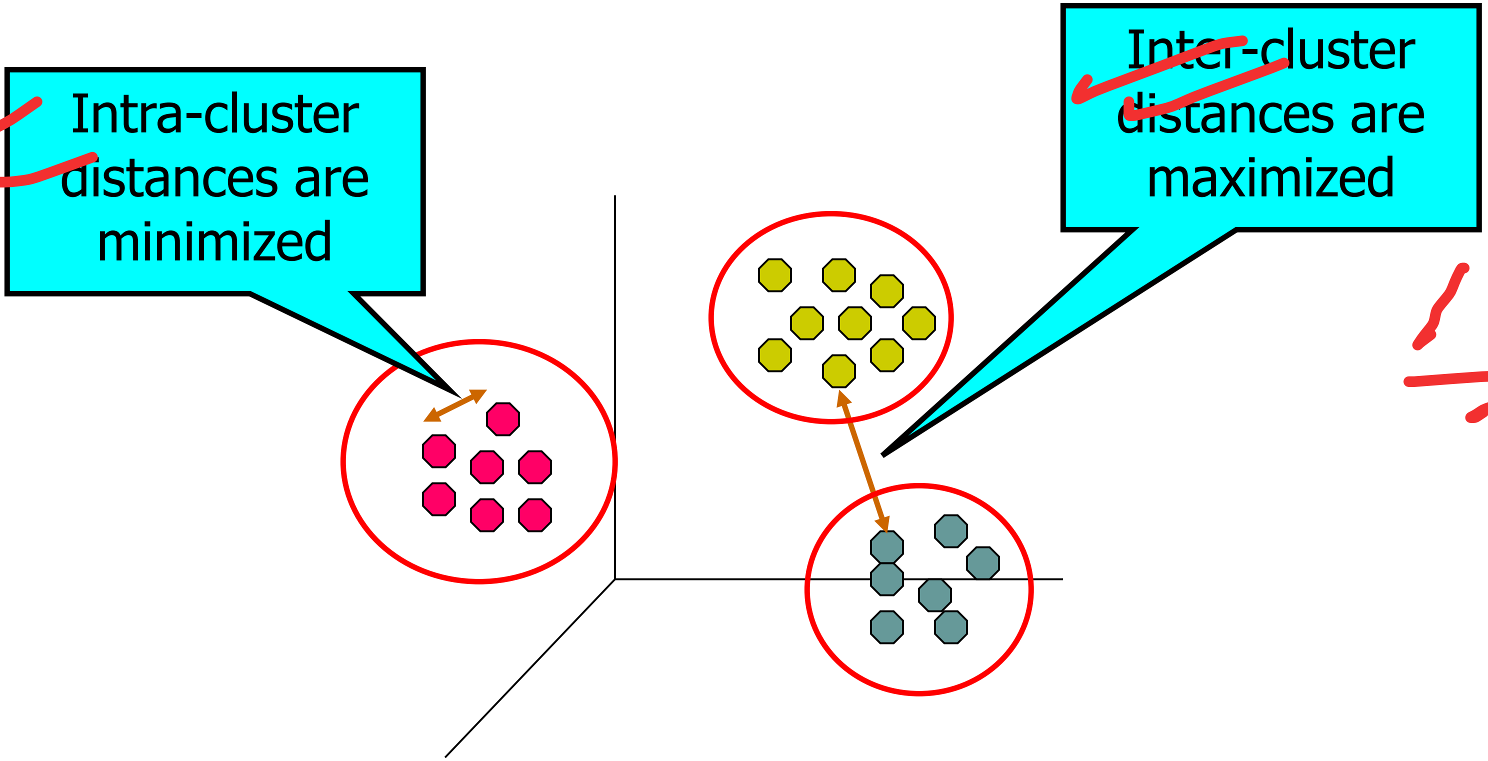
Jadavpur University

pawankrsingh.cse@gmail.com

+91-6291555693

What is Cluster Analysis?

- Given a set of objects, place them in groups such that the objects in a group are similar (or related) to one another and different from (or unrelated to) the objects in other groups



Intra-cluster distances are minimized

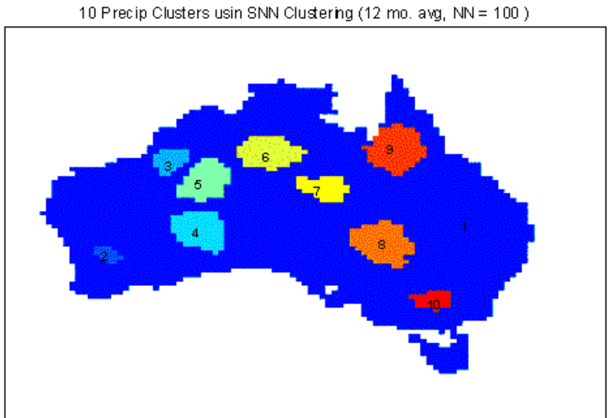
Inter-cluster distances are maximized

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

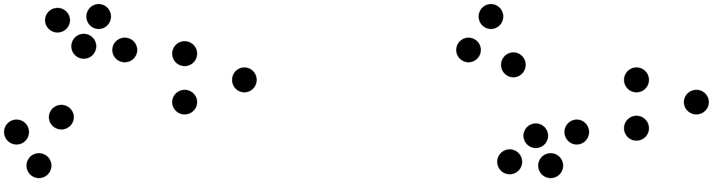
	Discovered Clusters	*Industry Group*
1	Applied-Matl-DOWN,Bay-Network-Down,3-COM-DOWN, Cabletron-Sys-DOWN,CISCO-DOWN,HP-DOWN, DSC-Comm-DOWN,INTEL-DOWN,LSI-Logic-DOWN, Micron-Tech-DOWN,Texas-Inst-Down,Tellabs-Inc-Down, Natl-Semiconduct-DOWN,Oracl-DOWN,SGI-DOWN, Sun-DOWN	Technology1-DOWN
2	Apple-Comp-DOWN,Autodesk-DOWN,DEC-DOWN, 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

- **Summarization**
  - Reduce the size of large data sets

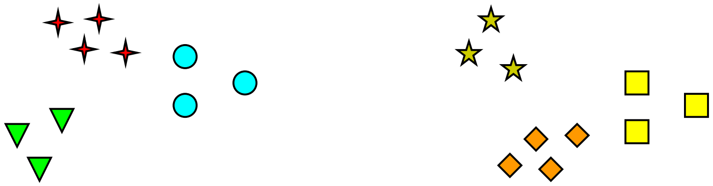

10 Precip Clusters usin SNN Clustering (12 mo. avg, NN = 100)
1
2
3
4
5
6
7
8
9
10

Clustering precipitation in Australia

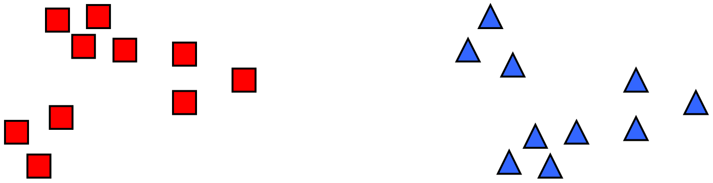
Notion of a Cluster can be Ambiguous



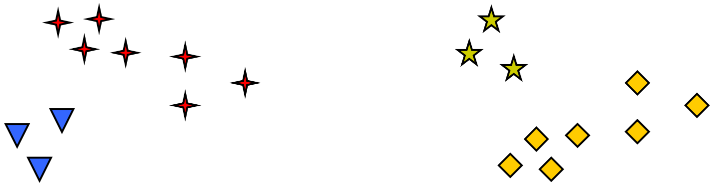
How many clusters?



Six Clusters



Two Clusters



Four Clusters

What is Clustering?

- Clustering is one of the most important research areas in the field of data mining.
- **Cluster analysis** or **clustering** is the task of grouping a set of objects in such a way that objects in the same group (called a **cluster**) are more similar (in some sense or another) to each other than to those in other groups (clusters).
- It is an unsupervised learning technique.
- Data clustering is the subject of active research in several fields such as statistics, pattern recognition and machine learning.
- From a practical perspective clustering plays an outstanding role in data mining applications in many domains.
- **The main advantage of clustering is that interesting patterns and structures can be found directly from very large data sets with little or none of the background knowledge**.
- Clustering algorithms an be applied in many areas, like marketing, biology, libraries, insurance, city-planning, earthquake studies and www document classification.

Applications of Clustering

- Real life examples where we use clustering:
  - **Marketing**
    - Finding group of customers with similar behavior given a large data-base of customers.
    - Data containing their properties and past buying records (Conceptual Clustering).
  - **Biology**
    - Classification of Plants and Animals Based on the properties under observation (Conceptual Clustering).
  - **Insurance**
    - Identifying groups of car insurance policy holders with a high average claim cost (Conceptual Clustering).
  - **City-Planning**
    - Groups of houses according to their house type, value and geographical location it can be both (Conceptual Clustering and Distance Based Clustering)
  - **Libraries**
    - It is used in clustering different books on the basis of topics and information.
  - **Earthquake studies**
    - By learning the earthquake-affected areas we can determine the dangerous zones.

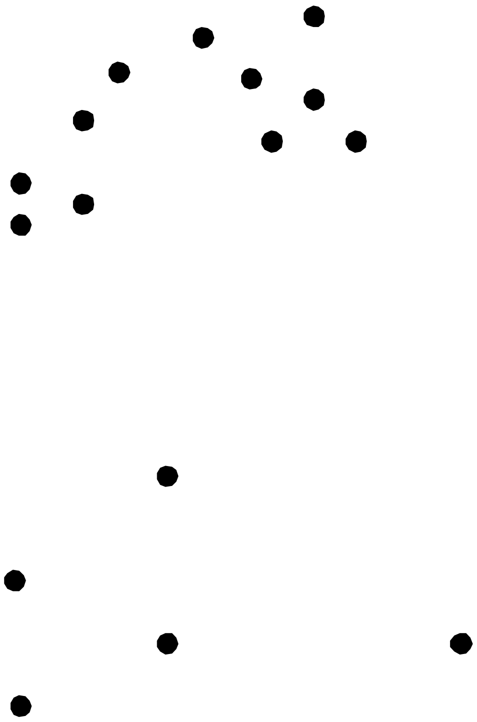
Types of Clustering

- A clustering is a set of clusters
- Important distinction between hierarchical and partitional sets of clusters
  - Partitional Clustering
    - A division of data objects into non-overlapping subsets (clusters)
  - Hierarchical clustering
    - A set of nested clusters organized as a hierarchical tree

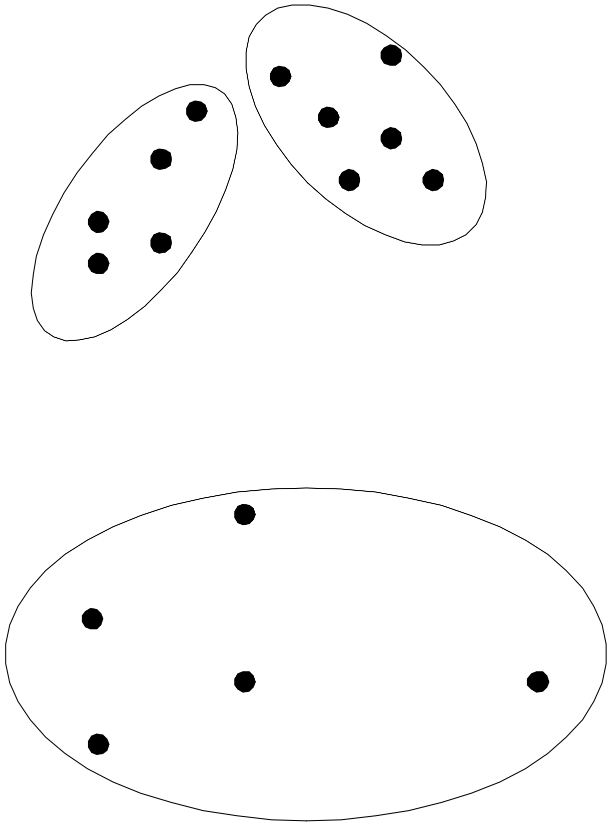
Imp

V.V.I.P

Partitional Clustering

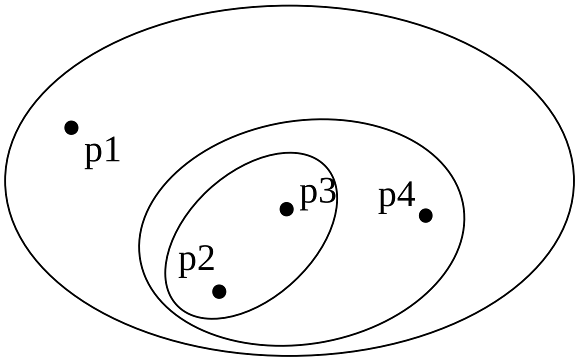


Original Points

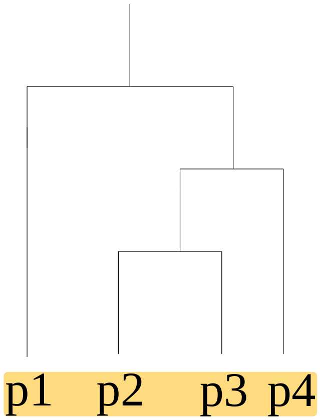


A Partitional Clustering

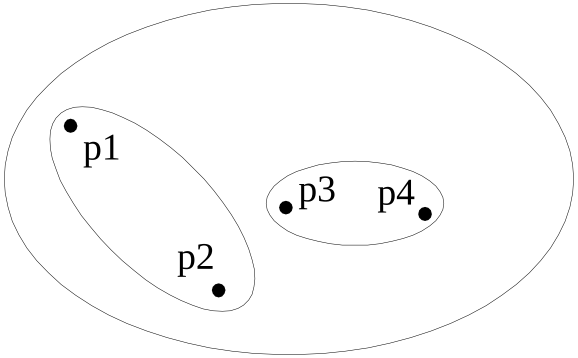
Hierarchical Clustering



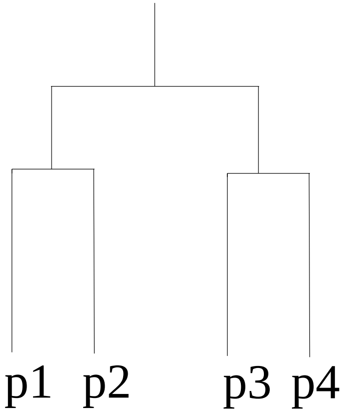
Traditional Hierarchical Clustering



Traditional Dendrogram



Non-traditional Hierarchical Clustering



Non-traditional Dendrogram

Other Distinctions Between Sets of Clusters

- Exclusive versus non-exclusive
  - In non-exclusive clusterings, points may belong to multiple clusters.
    - Can belong to multiple classes or could be ‘border’ points
    - Fuzzy clustering (one type of non-exclusive)
    - 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

Types of Clusters

- Well-separated clusters

- Prototype-based clusters

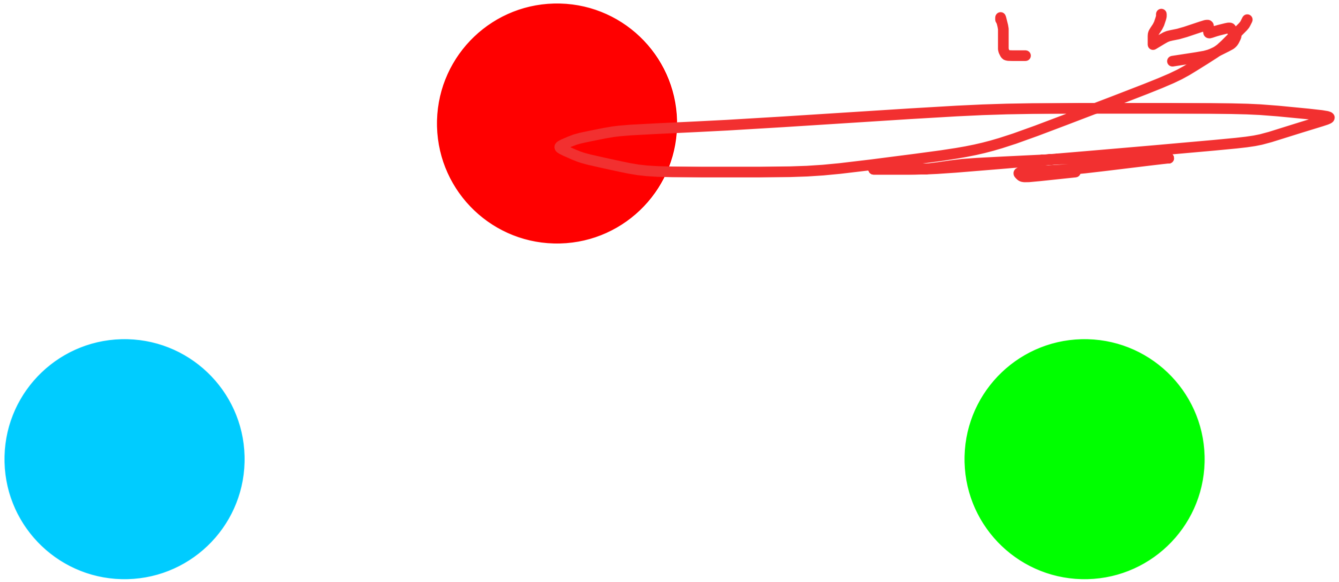
- Contiguity-based clusters

- Density-based clusters

- Described by an Objective Function

Types of Clusters: Well-Separated

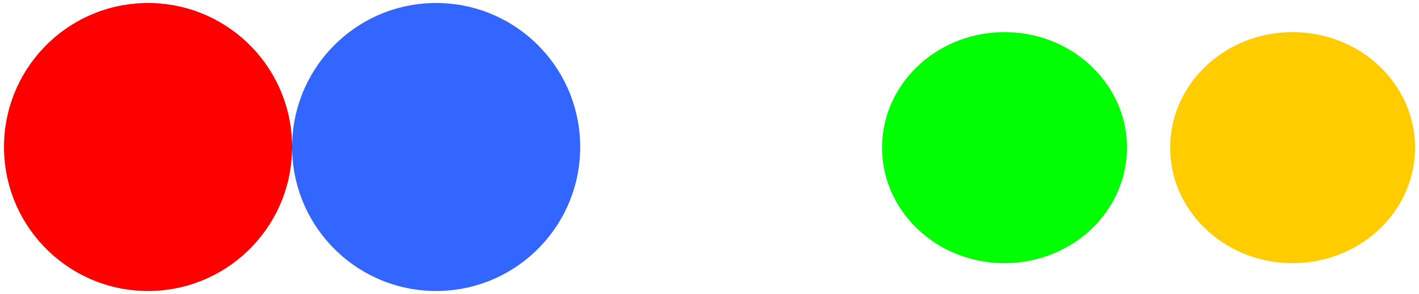
- Well-Separated Clusters:
  - A cluster is a set of points such that any point in a cluster is closer (or more similar) to every other point in the cluster than to any point not in the cluster.



3 well-separated clusters

Types of Clusters: Prototype-Based

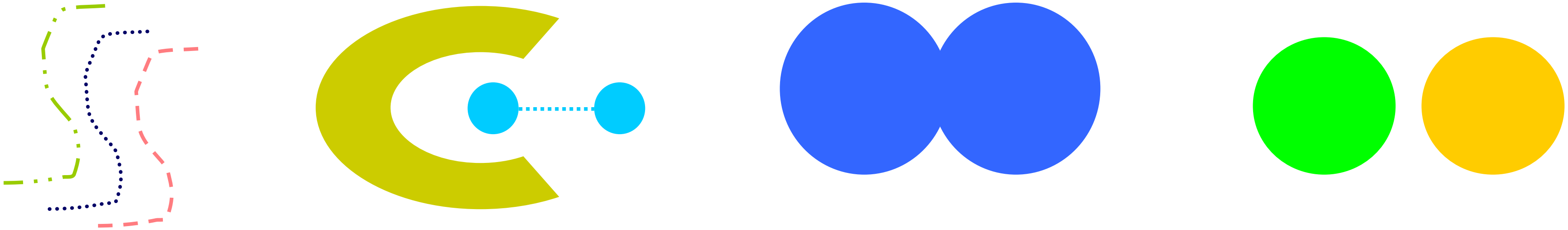
- Prototype-based
  - A cluster is a set of objects such that an object in a cluster is closer (more similar) to the prototype or "center" of a cluster, than to the center of any other cluster
  - The center of a cluster is often a centroid, the average of all the points in the cluster, or a medoid, the most "representative" point of a cluster



4 center-based clusters

Types of Clusters: Contiguity-Based

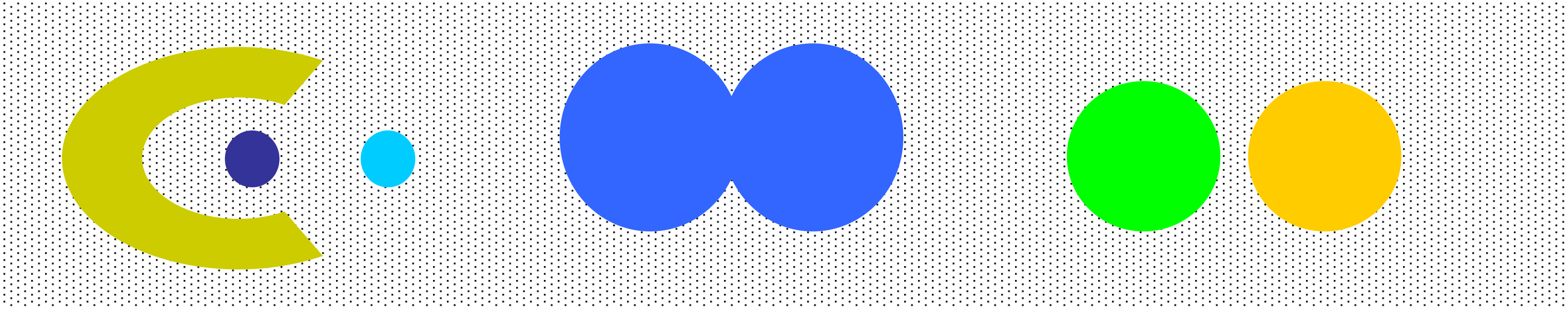
- Contiguous Cluster (Nearest neighbor or Transitive)
  - A cluster is a set of points such that a point in a cluster is closer (or more similar) to one or more other points in the cluster than to any point not in the cluster.



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.



6 density-based clusters

Types of Clusters: Objective Function

- Clusters Defined by an Objective Function
  - Finds clusters that minimize or maximize an objective function.
  - Enumerate all possible ways of dividing the points into clusters and evaluate the `goodness' of each potential set of clusters by using the given objective function. (NP Hard)
  - Can have global or local objectives.
    - Hierarchical clustering algorithms typically have local objectives
    - Partitional algorithms typically have global objectives
  - A variation of the global objective function approach is to fit the data to a parameterized model.
    - Parameters for the model are determined from the data.
      - Mixture models assume that the data is a 'mixture' of a number of statistical distributions.

Characteristics of the Input Data Are Important

- Type of proximity or density measure
  - Central to clustering
  - Depends on data and application

- Data characteristics that affect proximity and/or density are
  - Dimensionality
    - Sparseness
  - Attribute type
  - Special relationships in the data
    - For example, autocorrelation
  - Distribution of the data

- Noise and Outliers
  - Often interfere with the operation of the clustering algorithm
- Clusters of differing sizes, densities, and shapes

Clustering Algorithms

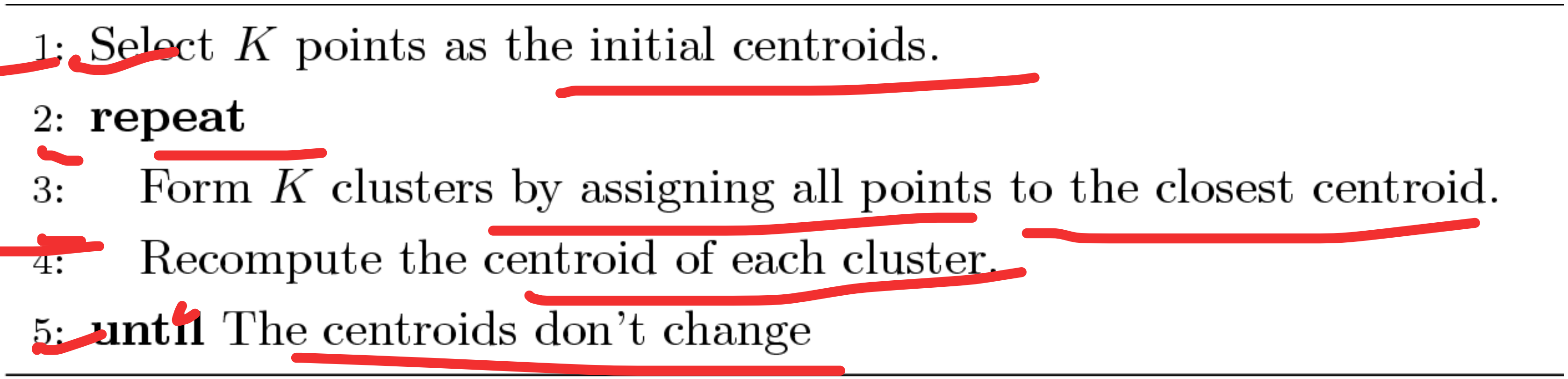
- K-means and its variants
- Hierarchical clustering
- Density-based clustering

Imp

parametric based

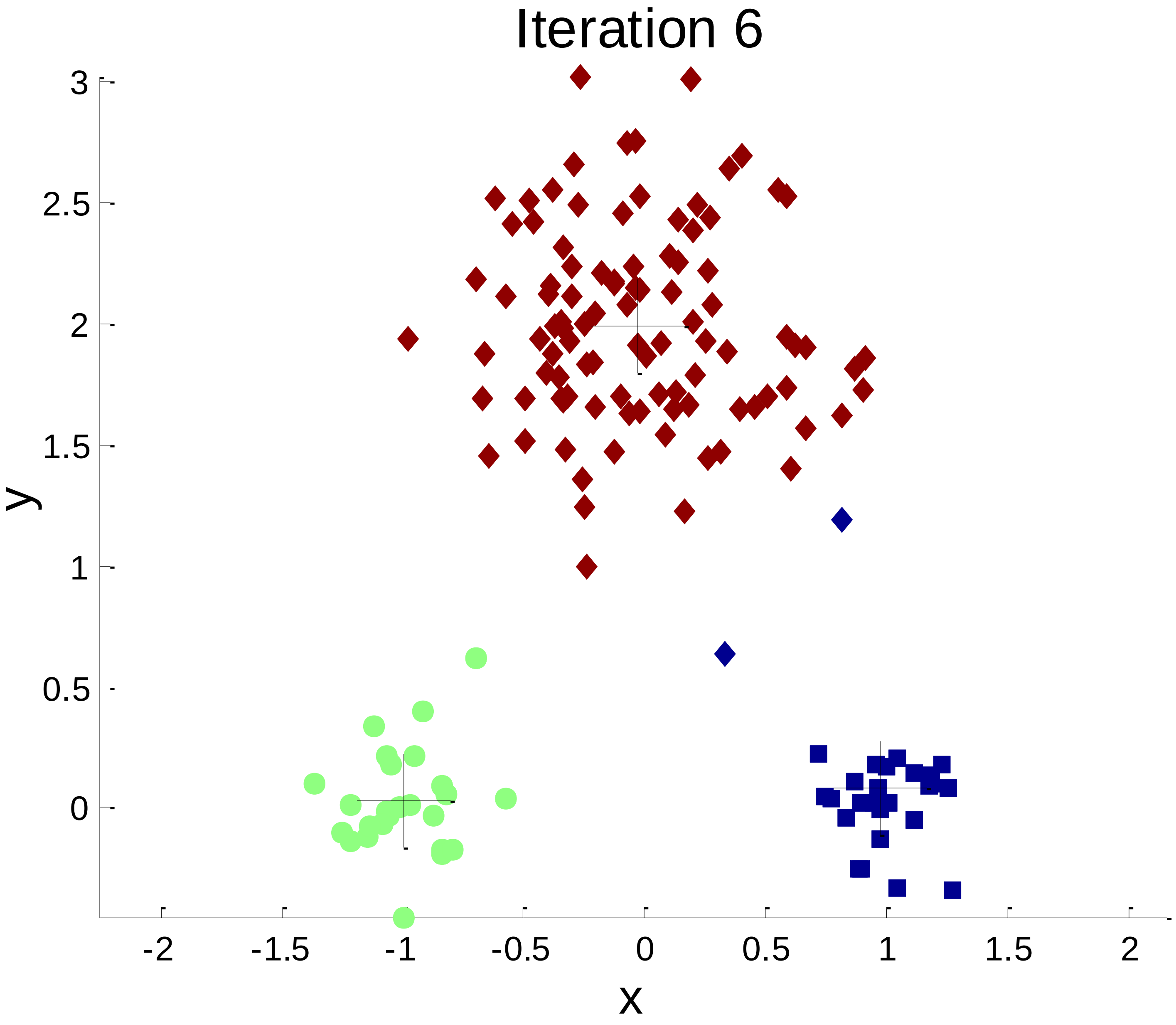
K-means Clustering

- Partitional clustering approach
- Number of clusters, K, must be specified
- Each cluster is associated with a centroid (center point)
- Each point is assigned to the cluster with the closest centroid
- The basic algorithm is very simple

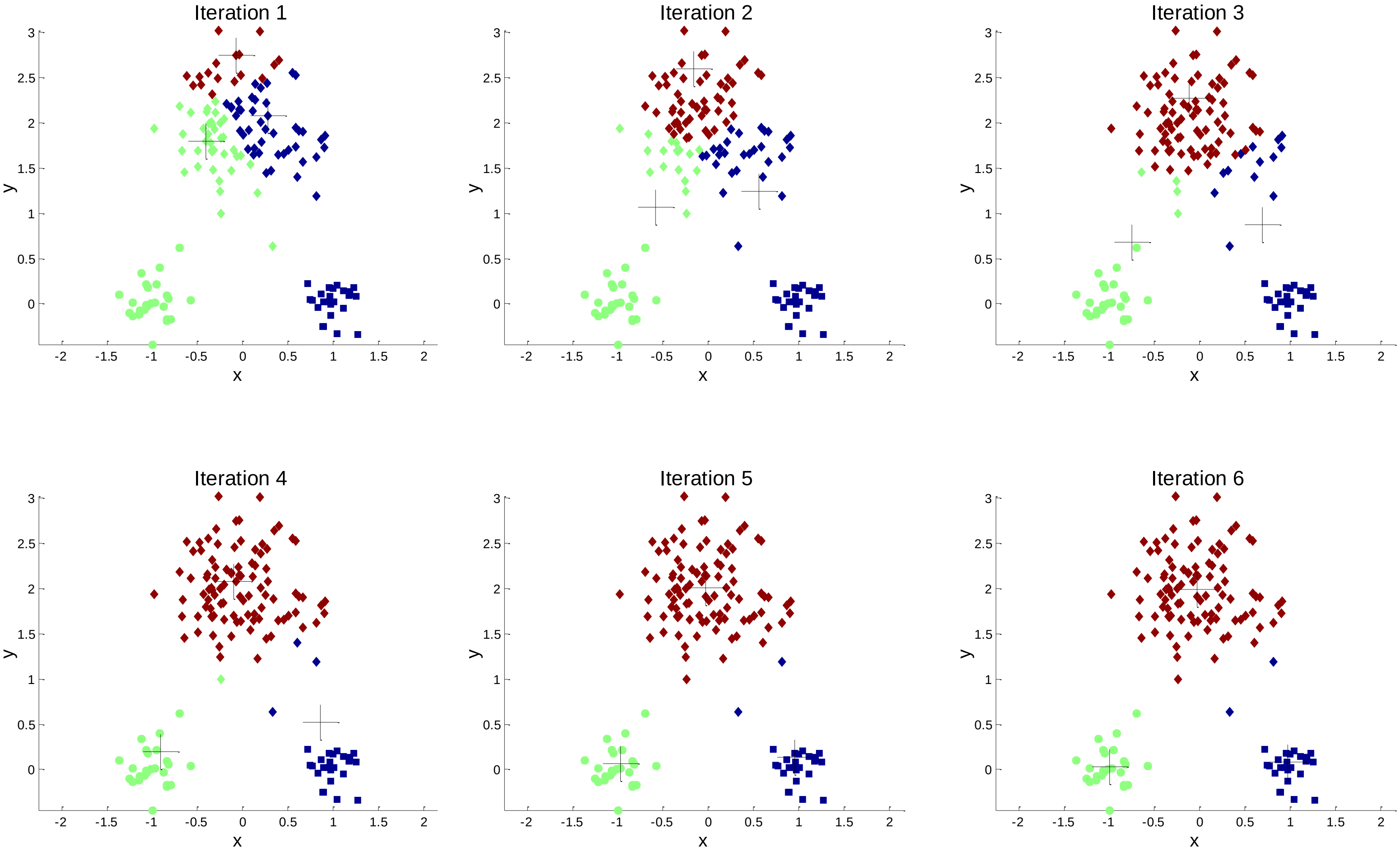


```
Select $K$ points as the initial centroids.
repeat
    Form $K$ clusters by assigning all points to the closest centroid.
    Recompute the centroid of each cluster.
until The centroids don't change
```

Example of K-means Clustering



Example of K-means Clustering



~~K-means Clustering – Details~~

- Simple iterative algorithm.
  - Choose initial centroids;
  - repeat {assign each point to a nearest centroid; re-compute cluster centroids}
  - until centroids stop changing.
- Initial centroids are often chosen randomly.
  - Clusters produced can vary from one run to another
- The centroid is (typically) the mean of the points in the cluster, but other definitions are possible.
- K-means will converge for common proximity measures with appropriately defined centroid.
- 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 points, K = number of clusters,
    I = number of iterations, d = number of attributes

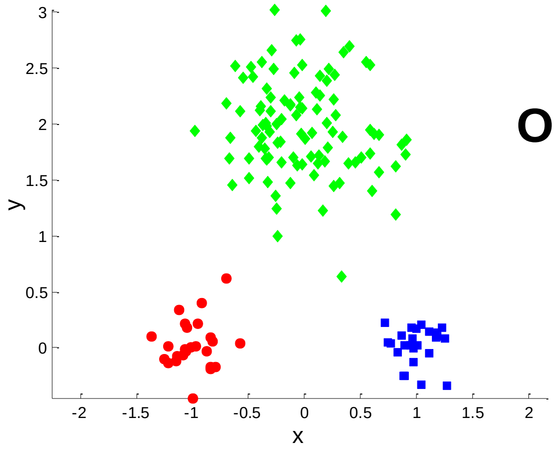
K-means Objective Function

- A common objective function (used with Euclidean distance measure) is Sum of Squared Error (SSE)
  - For each point, the error is the distance to the nearest cluster center
  - To get SSE, we square these errors and sum them.

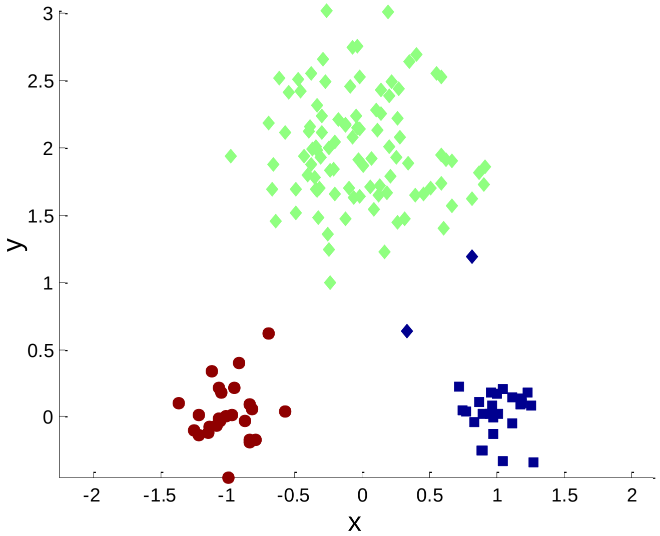
$$SSE = \sum_{i=1}^{K} \sum_{x \in C_i} dist^2(m_i, x)$$

- *x* is a data point in cluster C_i and m_i is the centroid (mean) for cluster C_i
- SSE improves in each iteration of K-means until it reaches a local or global minima.

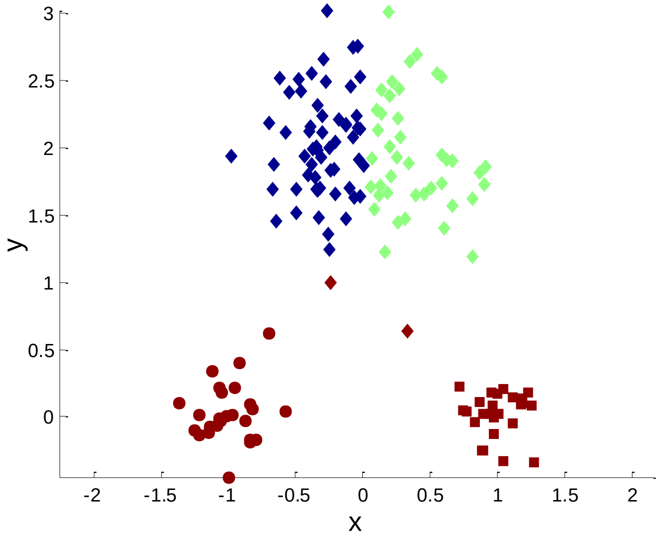
Two different K-means Clusterings



Original Points

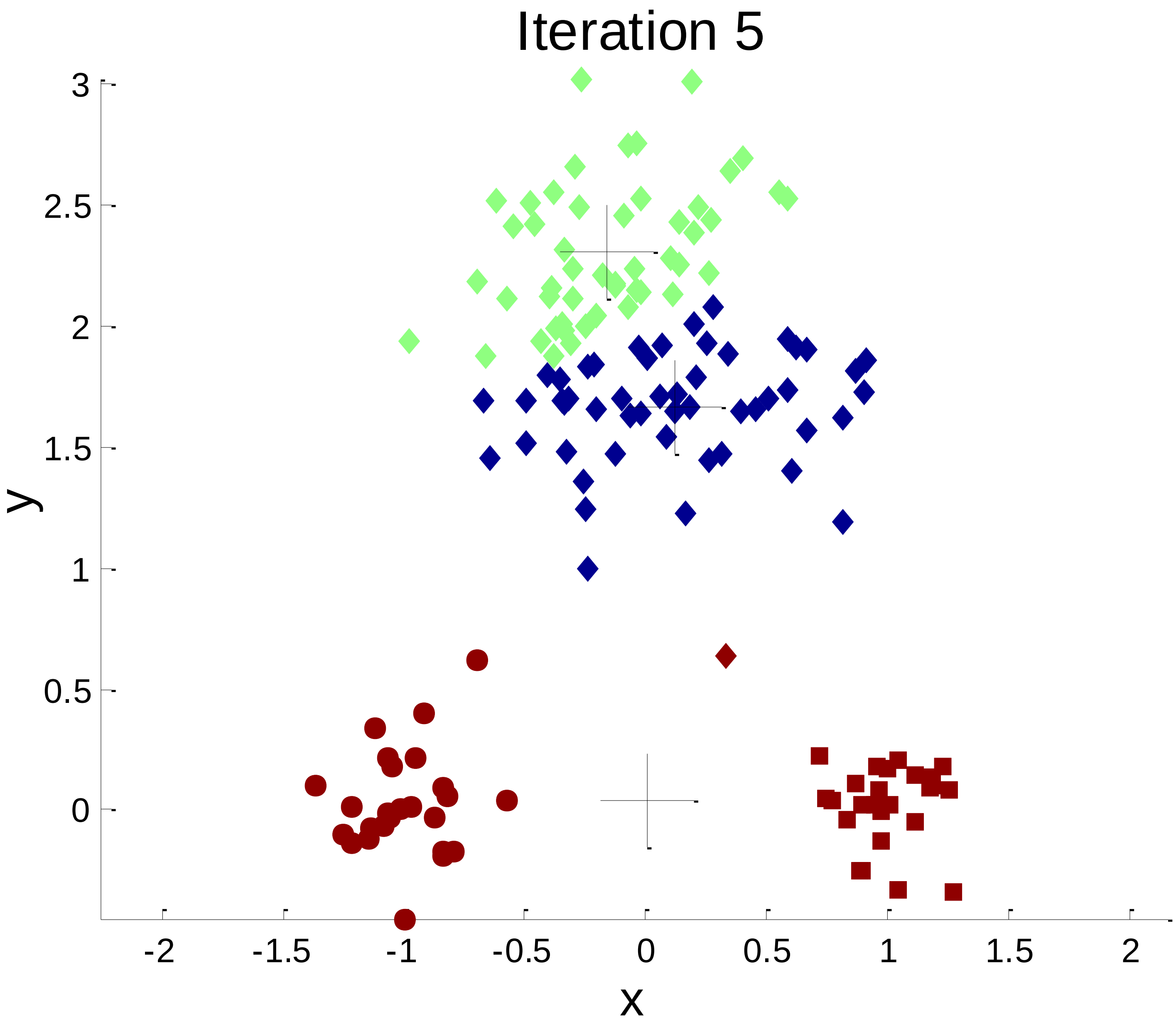


Optimal Clustering

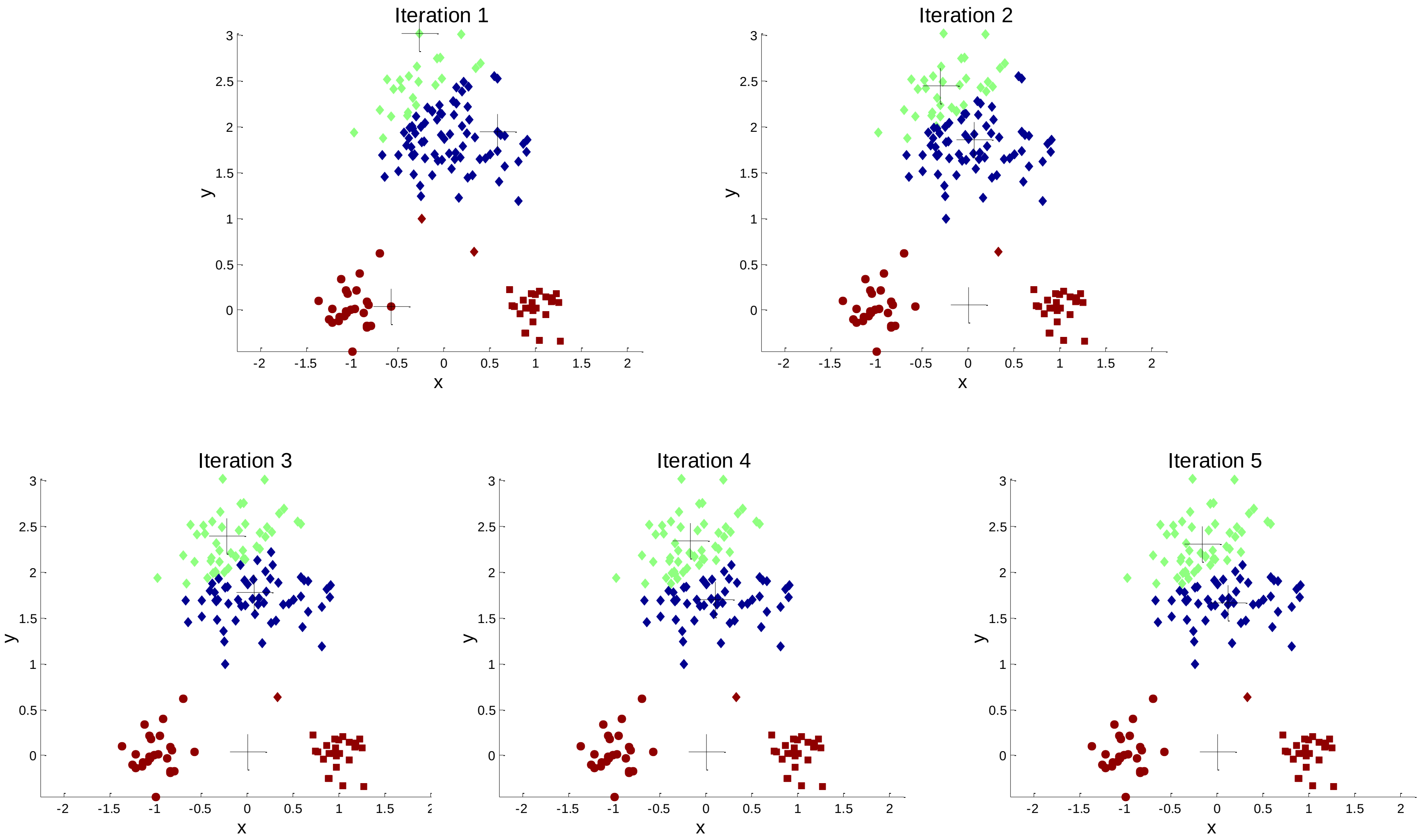


Sub-optimal Clustering

Importance of Choosing Initial Centroids …



Importance of Choosing Initial Centroids …



Importance of Choosing Intial Centroids

- Depending on the choice of initial centroids, B and C may get merged or remain separate

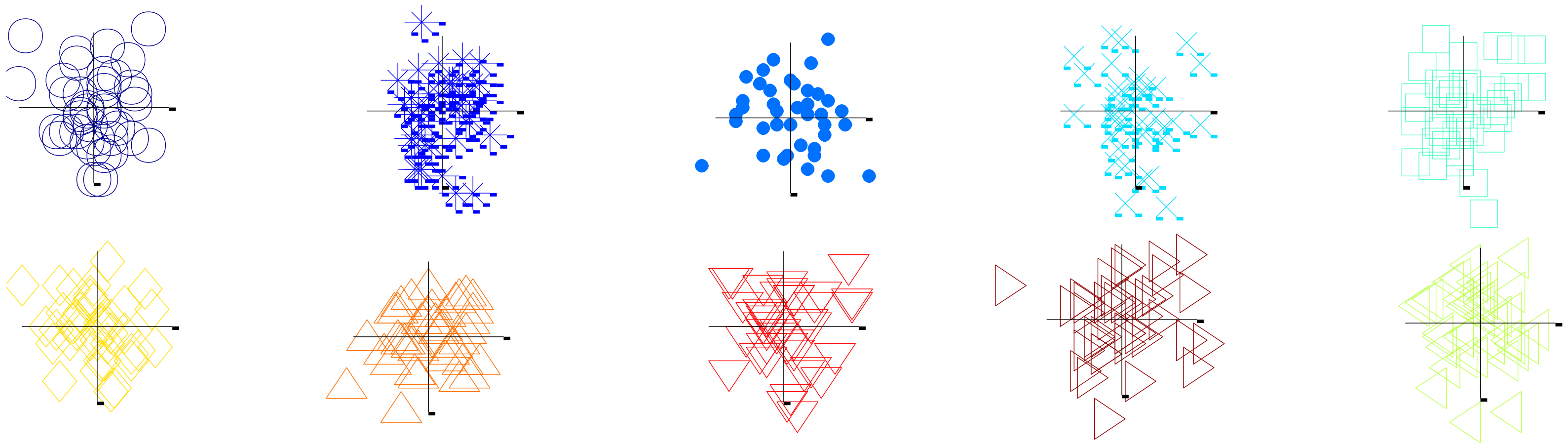
Problems with Selecting Initial Points

- 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

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

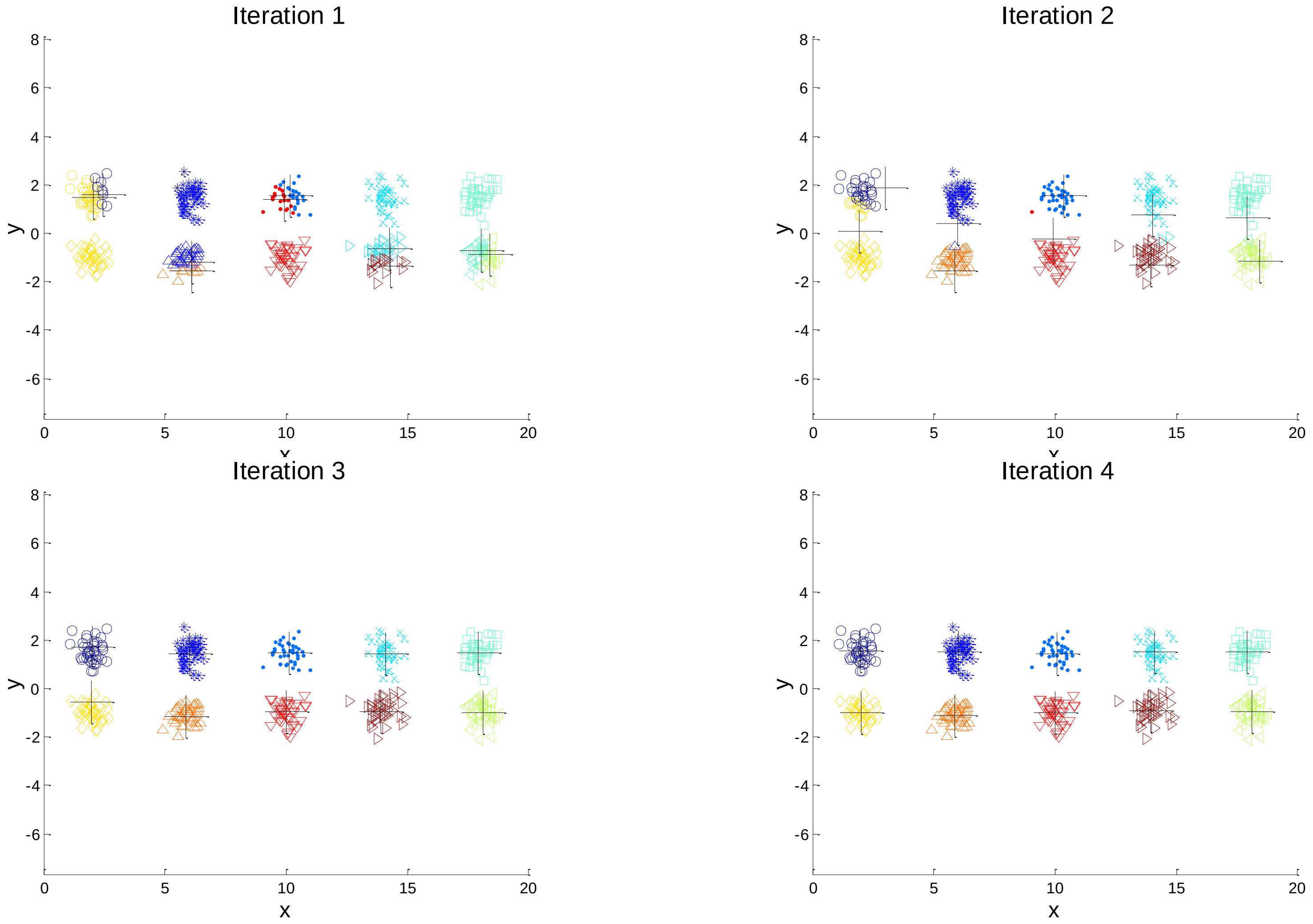
  - For example, if K = 10, then probability = $10!/10^{10}$ = 0.00036
  - Sometimes the initial centroids will readjust themselves in ‘right’ way, and sometimes they don’t
  - Consider an example of five pairs of clusters

10 Clusters Example



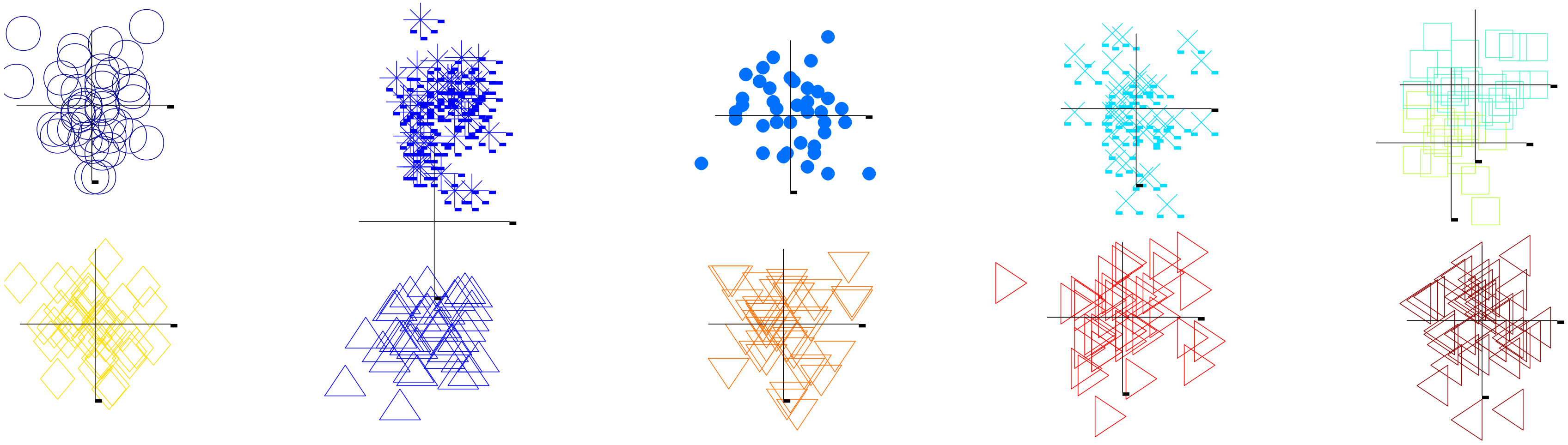
Starting with two initial centroids in one cluster of each pair of clusters

10 Clusters Example



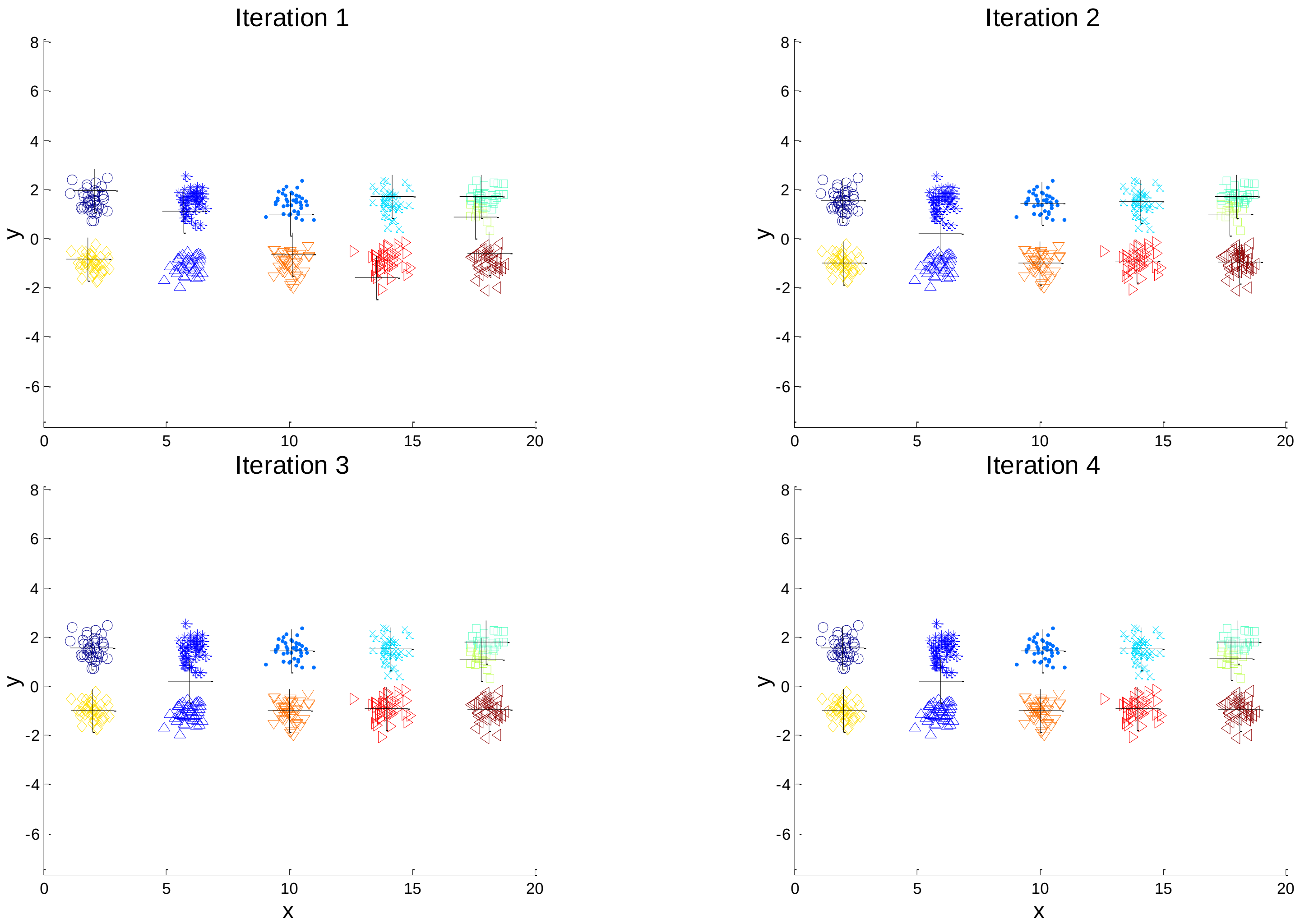
Starting with two initial centroids in one cluster of each pair of clusters

10 Clusters Example



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

10 Clusters Example

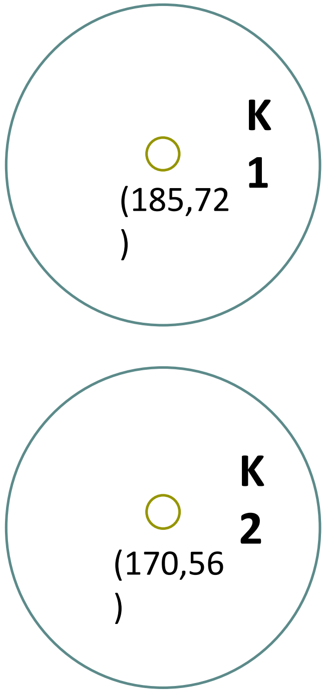


Starting with some pairs of clusters having three initial centroids, while other have only one.

K-Means Algorithm - Example

K-Means Algorithm - Example

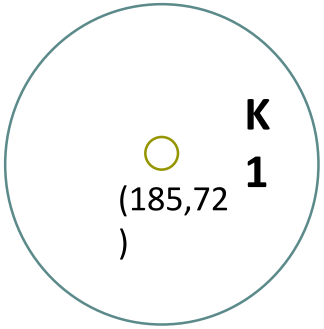
Sr.	Height	Weight
1	185	72
2	170	56
3	168	60
4	179	68
5	182	72
6	188	77
7	180	71
8	180	70
9	183	84
10	180	88
11	180	67
12	177	76

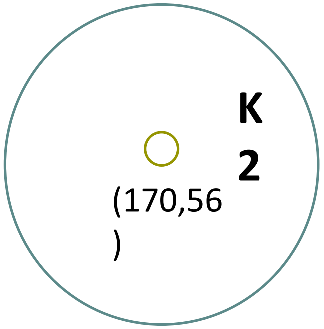


- First we take **K=2** So, two clusters or groups.
- We choose first (185,72) & second (170,56) row as centroid of each cluster or group.
- Now, we have to find Euclidean Distance,
  - $$\boldsymbol{ED} = \sqrt{(\boldsymbol{X_o} - \boldsymbol{X_c})^2 + (\boldsymbol{Y_o} - \boldsymbol{Y_c})^2}$$
- Where
  - X_0 & Y_o = Observed Value
  - X_c & Y_c = Centroid Value

K-Means Algorithm – Example Cont..

Sr.	Height	Weight
1	185	72
2	170	56
3	168	60
4	179	68
5	182	72
6	188	77
7	180	71
8	180	70
9	183	84
10	180	88
11	180	67
12	177	76





⤷ ED From $\mathbf{K1}\ \boldsymbol{to}\ (\mathbf{168}, \mathbf{60})$

$= \sqrt{(X_o - X_c)^2 + (Y_o - Y_c)^2}$

$= \sqrt{(168 - 185)^2 + (60 - 72)^2}$

$= \sqrt{(-17)^2 + (-12)^2}$

$= \sqrt{289 + 144}$

$= \sqrt{433}$

$= 20.80$

⤷ ED From $\mathbf{K2}\ \boldsymbol{to}\ (\mathbf{168}, \mathbf{60})$

$= \sqrt{(X_o - X_c)^2 + (Y_o - Y_c)^2}$

$= \sqrt{(168 - 170)^2 + (60 - 56)^2}$

$= \sqrt{(-2)^2 + (-4)^2}$

$= \sqrt{4 + 16}$

$= \sqrt{20}$

$= 4.48$

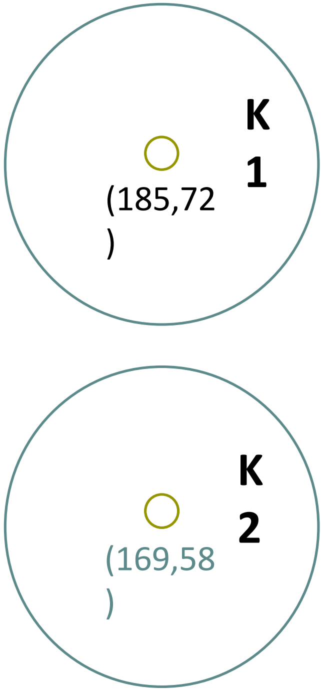
Now, data (168,60) nearer to K2, so it belongs to K2.

K1 = {1}

K2 = {2,3}

K-Means Algorithm – Example Cont..

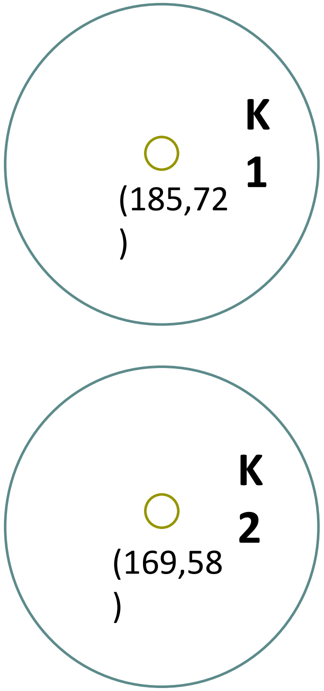
Sr.	Height	Weight
1	185	72
2	**170**	**56**
3	**168**	**60**
4	179	68
5	182	72
6	188	77
7	180	71
8	180	70
9	183	84
10	180	88
11	180	67
12	177	76



Now, New Centroid Calculation
For K2 = {2,3}
So, **K2** = {(170,56),(168,60)}
= 170+168/2 & 56+60/2
We get new centroid **C** = (169,58)

K-Means Algorithm – Example Cont..

Sr.	Height	Weight
1	185	72
2	170	56
3	168	60
4	**179**	**68**
5	182	72
6	188	77
7	180	71
8	180	70
9	183	84
10	180	88
11	180	67
12	177	76



- ED From $\mathbf{K1}\ \boldsymbol{to}\ (\mathbf{179}, \mathbf{68})$

$$= \sqrt{(X_{\mathrm{o}} - X_c)^2 + (Y_o - Y_c)^2}$$
$$= \sqrt{(179 - 185)^2 + (68 - 72)^2}$$
$$= \sqrt{(-6)^2 + (-4)^2}$$
$$= \sqrt{36 + 16}$$
$$= \sqrt{52}$$
$$= 7.21$$

- ED From $\mathbf{K2}\ \boldsymbol{to}\ (\mathbf{179}, \mathbf{68})$

$$= \sqrt{(X_o - X_c)^2 + (Y_o - Y_c)^2}$$
$$= \sqrt{(179 - 169)^2 + (68 - 58)^2}$$
$$= \sqrt{(10)^2 + (10)^2}$$
$$= \sqrt{100 + 100}$$
$$= \sqrt{200}$$
$$= 14.14$$

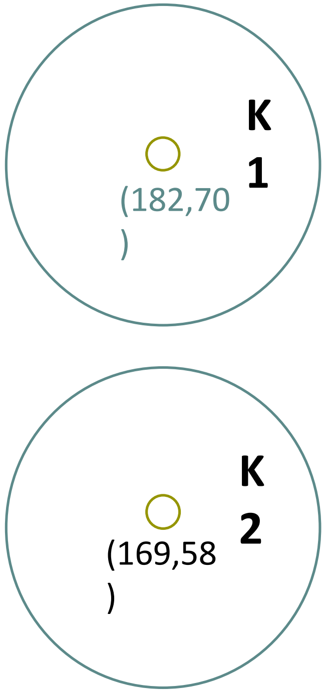
Now, data (179,68) nearer to K1, so it belongs to K1.

K1 = {1,4}

K2 = {2,3}

K-Means Algorithm – Example Cont..

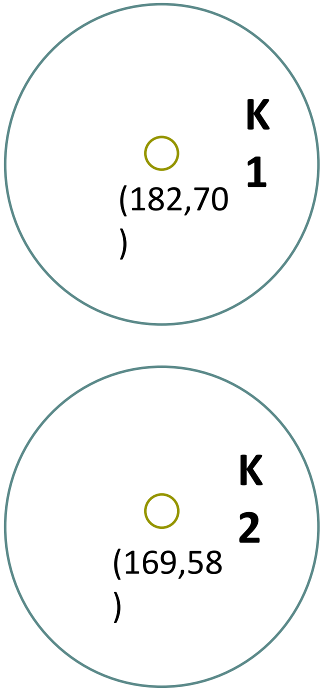
Sr.	Height	Weight
1	**185**	**72**
2	170	56
3	168	60
4	**179**	**68**
5	182	72
6	188	77
7	180	71
8	180	70
9	183	84
10	180	88
11	180	67
12	177	76



Now, New Centroid Calculation
For K1 = {1,4}
So, **K2** = {(185,72),(179,68)}
= 185+179/2 & 72+68/2
We get new centroid **C** = (182,70)

K-Means Algorithm – Example Cont..

Sr.	Height	Weight
1	185	72
2	170	56
3	168	60
4	179	68
5	**182**	**72**
6	188	77
7	180	71
8	180	70
9	183	84
10	180	88
11	180	67
12	177	76



↪ ED From $\mathbf{K1}\ \boldsymbol{to}\ (\mathbf{182},\mathbf{72})$

$$= \sqrt{(X_o - X_c)^2 + (Y_o - Y_c)^2}$$
$$= \sqrt{(182 - 182)^2 + (72 - 70)^2}$$
$$= \sqrt{(0)^2 + (2)^2}$$
$$= \sqrt{0 + 4}$$
$$= \sqrt{4}$$
$$= 2$$

↪ ED From $\mathbf{K2}\ \boldsymbol{to}\ (\mathbf{182},\mathbf{72})$

$$= \sqrt{(X_o - X_c)^2 + (Y_o - Y_c)^2}$$
$$= \sqrt{(182 - 169)^2 + (72 - 58)^2}$$
$$= \sqrt{(-13)^2 + (-14)^2}$$
$$= \sqrt{169 + 196}$$
$$= \sqrt{365}$$
$$= 19.10$$

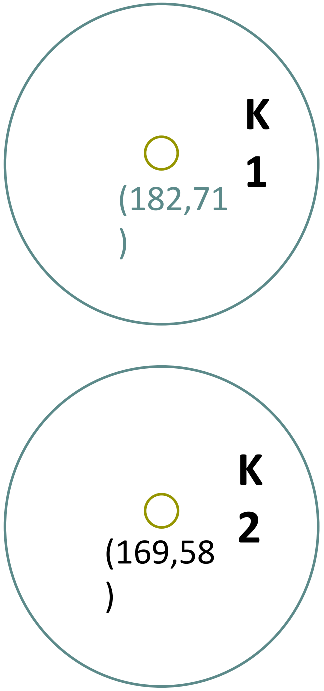
Now, data (182,72) nearer to K1, so it belongs to K1.

K1 = {1,4,5}

K2 = {2,3}

K-Means Algorithm – Example Cont..

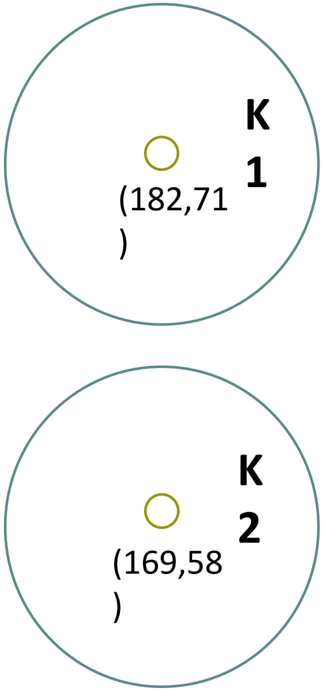
Sr.	Height	Weight
1	**185**	**72**
2	170	56
3	168	60
4	**179**	**68**
5	**182**	**72**
6	188	77
7	180	71
8	180	70
9	183	84
10	180	88
11	180	67
12	177	76



Now, New Centroid Calculation
For K1 = {1,4,5}
So, **K2** = {(185,72),(179,68),(182,72)}
= 185+179+182/3 & 72+68+72/3
We get new centroid **C** = (182,70.666) ~ (182,71)

K-Means Algorithm – Example Cont..

Sr.	Height	Weight
1	185	72
2	170	56
3	168	60
4	179	68
5	182	72
6	188	77
7	180	71
8	180	70
9	183	84
10	180	88
11	180	67
12	177	76



- ED From $\mathbf{K1}\ \boldsymbol{to}\ (\mathbf{188}, \mathbf{77})$

$$= \sqrt{(X_o - X_c)^2 + (Y_o - Y_c)^2}$$
$$= \sqrt{(188 - 182)^2 + (77 - 71)^2}$$
$$= \sqrt{(6)^2 + (6)^2}$$
$$= \sqrt{36 + 36}$$
$$= \sqrt{72}$$
$$= 8.48$$

- ED From $\mathbf{K2}\ \boldsymbol{to}\ (\mathbf{188}, \mathbf{77})$

$$= \sqrt{(X_o - X_c)^2 + (Y_o - Y_c)^2}$$
$$= \sqrt{(188 - 169)^2 + (77 - 58)^2}$$
$$= \sqrt{(19)^2 + (19)^2}$$
$$= \sqrt{361 + 361}$$
$$= \sqrt{722}$$
$$= 26.87$$

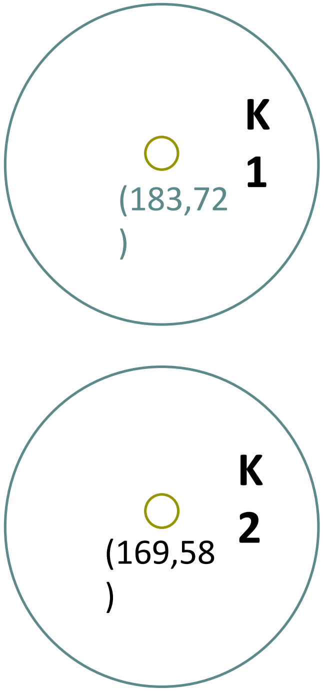
Now, data (188,77) nearer to K1, so it belongs to K1.

K1 = {1,4,5,6}

K2 = {2,3}

K-Means Algorithm – Example Cont..

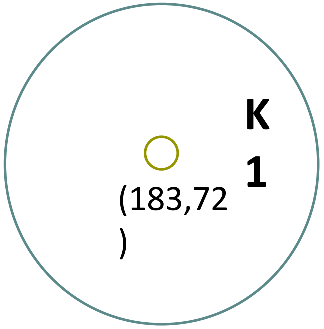
Sr.	Height	Weight
1	**185**	**72**
2	170	56
3	168	60
4	**179**	**68**
5	**182**	**72**
6	**188**	**77**
7	180	71
8	180	70
9	183	84
10	180	88
11	180	67
12	177	76

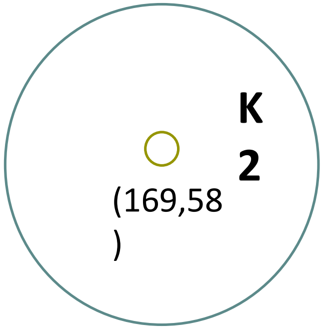


Now, New Centroid Calculation
For K1 = {1,4,5,6}
So, **K2** = {(185,72),(179,68),(182,72),(188,77)}
= 185+179+182+188/4 & 72+68+72+77/7
We get new centroid **C** = (183.50,72.25) ~ (183,72)

K-Means Algorithm – Example Cont..

Sr.	Height	Weight
1	185	72
2	170	56
3	168	60
4	179	68
5	182	72
6	188	77
7	180	71
8	180	70
9	183	84
10	180	88
11	180	67
12	177	76





↳ ED From $\mathbf{K1}\ \boldsymbol{to}\ (\mathbf{180}, \mathbf{71})$

$$= \sqrt{(X_\mathrm{o} - X_c)^2 + (Y_o - Y_c)^2}$$
$$= \sqrt{(180 - 183)^2 + (71 - 72)^2}$$
$$= \sqrt{(-3)^2 + (-1)^2}$$
$$= \sqrt{9 + 1}$$
$$= \sqrt{10}$$
$$= 3.16$$

↳ ED From $\mathbf{K2}\ \boldsymbol{to}\ (\mathbf{180}, \mathbf{71})$

$$= \sqrt{(X_o - X_c)^2 + (Y_o - Y_c)^2}$$
$$= \sqrt{(180 - 169)^2 + (71 - 58)^2}$$
$$= \sqrt{(11)^2 + (13)^2}$$
$$= \sqrt{121 + 169}$$
$$= \sqrt{290}$$
$$= 17.02$$

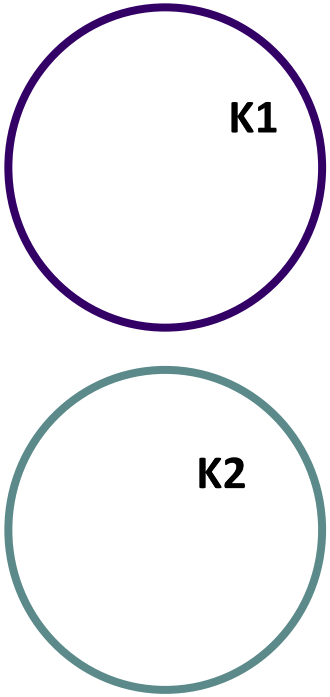
Now, data (180,71) nearer to K1, so it belongs to K1.

K1 = {1,4,5,6,7}

K2 = {2,3}

K-Means Algorithm – Example Cont..

Sr.	Height	Weight
1	185	72
2	170	56
3	168	60
4	179	68
5	182	72
6	188	77
7	180	71
8	180	70
9	183	84
10	180	88
11	180	67
12	177	76



Cluster K1 = {1,4,5,6,7,8,9,10,11,12}

Cluster K2 = {2,3}

K-Means Algorithm Cont..

- Let us assume two clusters, and each individual's scores include two variables.
- **Step-1**
  - Choose the number of clusters.
- **Step-2**
  - Set the initial partition, and the initial mean vectors for each cluster.
- **Step-3**
  - For each remaining individual...
- **Step-4**
  - Get averages for comparison to the Cluster 1:
    - Add individual's A value to the sum of A values of the individuals in Cluster 1, then divide by the total number of scores that were summed.
    - Add individual's B value to the sum of B values of the individuals in Cluster 1, then divide by the total number of scores that were summed.

K-Means Algorithm Cont..

- **Step-5**
  - Get averages for comparison to the Cluster 2:
    - Add individual's A value to the sum of A values of the individuals in Cluster 2, then divide by the total number of scores that were summed.
    - Add individual's B value to the sum of B values of the individuals in Cluster 2, then divide by the total number of scores that were summed.
- **Step-6**
  - If the averages found in Step 4 are closer to the mean values of Cluster 1, then this individual belongs to Cluster 1, and the averages found now become the new mean vectors for Cluster 1.
  - If closer to Cluster 2, then it goes to Cluster 2, along with the averages as new mean vectors.
- **Step-7**
  - If there are more individual's to process, continue again with Step 4. Otherwise go to Step 8.
- **Step-8**
  - Now compare each individual’s distance to its own cluster's mean vector, and to that of the opposite cluster.
  - The distance to its cluster's mean vector should be smaller than it distance to the other vector.
  - If not, relocate the individual to the opposite cluster.

K-Means Algorithm Cont..

- **Step-9**
  - If any relocations occurred in Step 8, the algorithm must continue again with Step 3, using all individuals and the new mean vectors.
  - If no relocations occurred, stop. Clustering is complete.

Solutions to Initial Centroids Problem

- Multiple runs
  - Helps, but probability is not on your side
- Use some strategy to select the k initial centroids and then select among these initial centroids
  - Select most widely separated
    - K-means++ is a robust way of doing this selection
  - Use hierarchical clustering to determine initial centroids
- Bisecting K-means
  - Not as susceptible to initialization issues

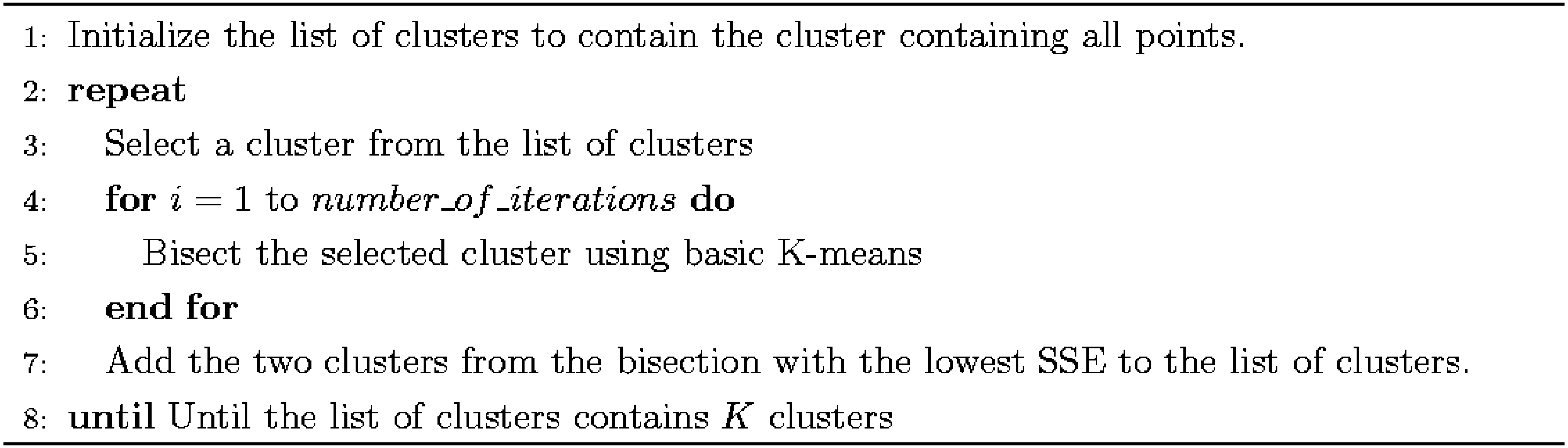
K-means++

- This approach can be slower than random initialization, but very consistently produces better results in terms of SSE
  - The k-means++ algorithm guarantees an approximation ratio O(log k) in expectation, where k is the number of centers
- To select a set of initial centroids, *C*, perform the following

```
1.  Select an initial point at random to be the first centroid
2.  For k – 1 steps
3.      For each of the N points, $x_i$, $1 \leq i \leq N$, find the minimum squared distance to the currently selected centroids, $C_1, \ldots, C_j$, $1 \leq j < k$, i.e., $\min_j d^2( C_j, x_i )$
4.      Randomly select a new centroid by choosing a point with probability proportional to $\frac{\min_j d^2( C_j, x_i )}{\sum_i \min_j d^2( C_j, x_i )}$ is
5.  End For
```

Bisecting K-means

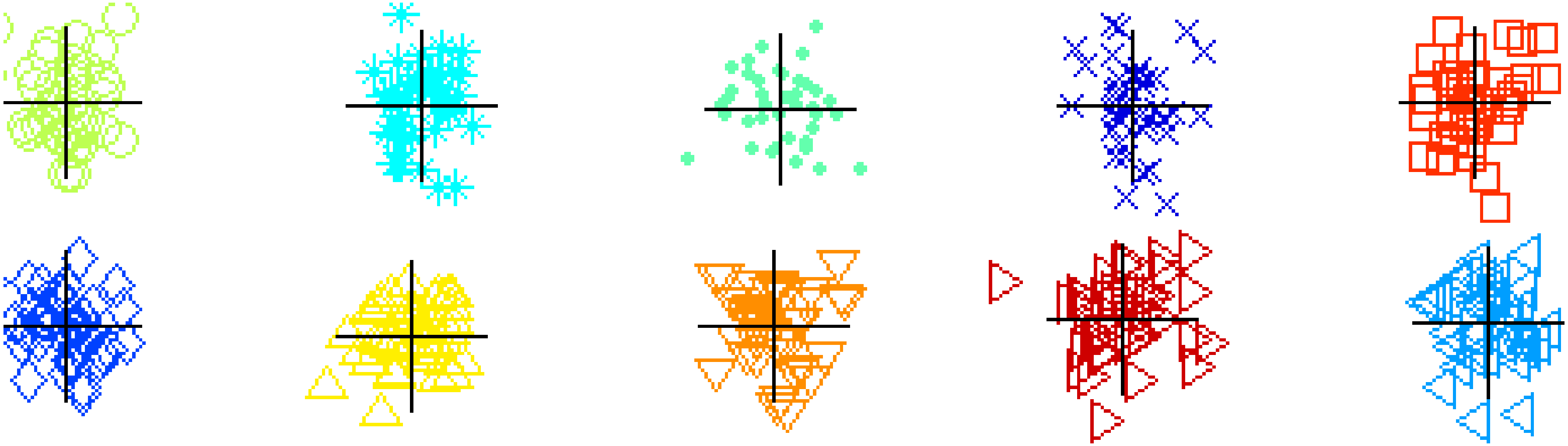
- Bisecting K-means algorithm
  - Variant of K-means that can produce a partitional or a hierarchical clustering



```
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
```

CLUTO: http://glaros.dtc.umn.edu/gkhome/cluto/cluto/overview

Bisecting K-means Example



Limitations of K-means

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

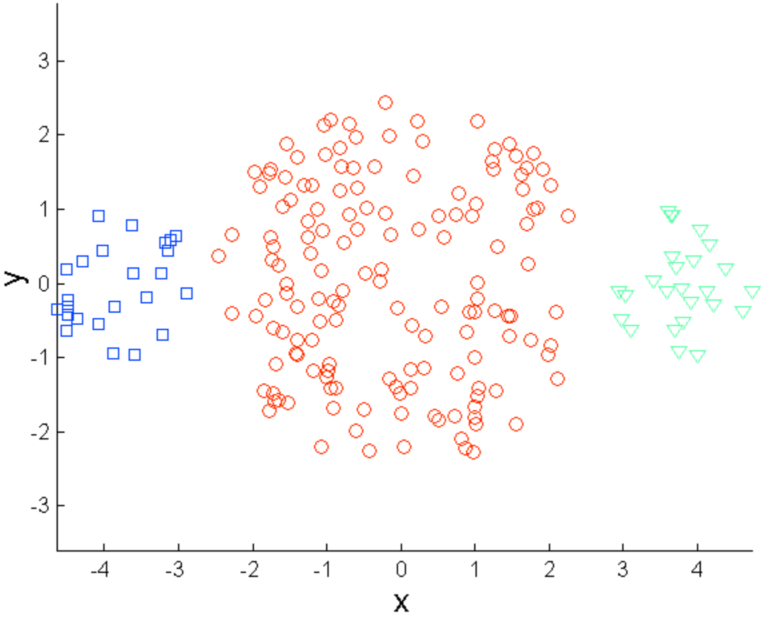
Imp

- K-means has problems when the data contains outliers.
  - One possible solution is to remove outliers before clustering

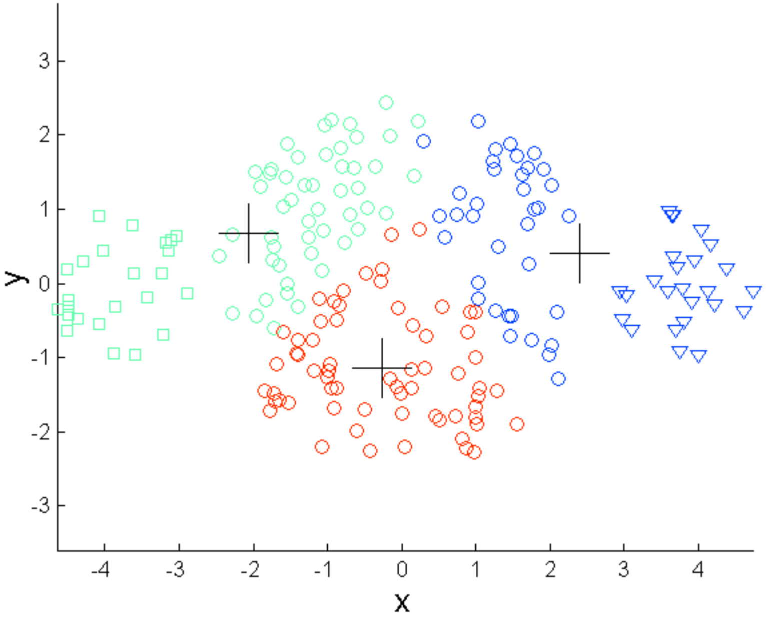
Imp

Imp

Limitations of K-means: Differing Sizes

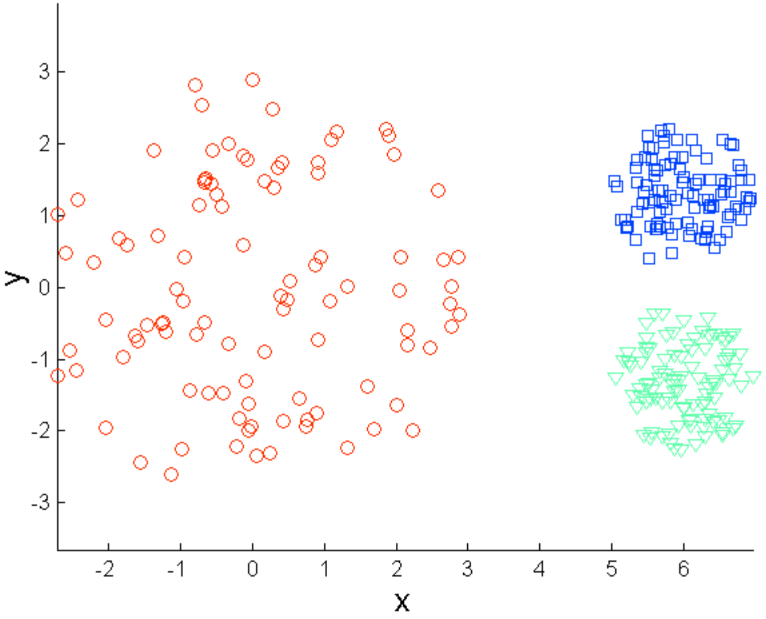


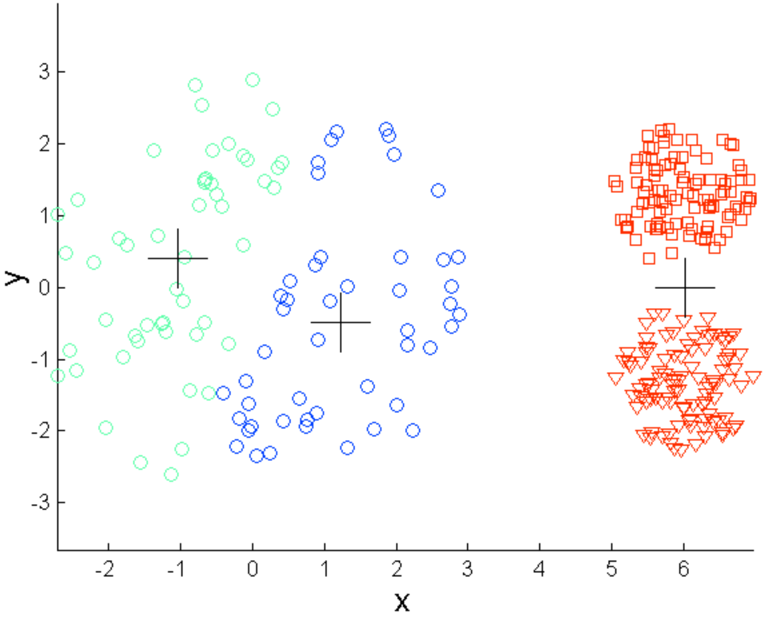
Original Points



K-means (3 Clusters)

Limitations of K-means: Differing Density

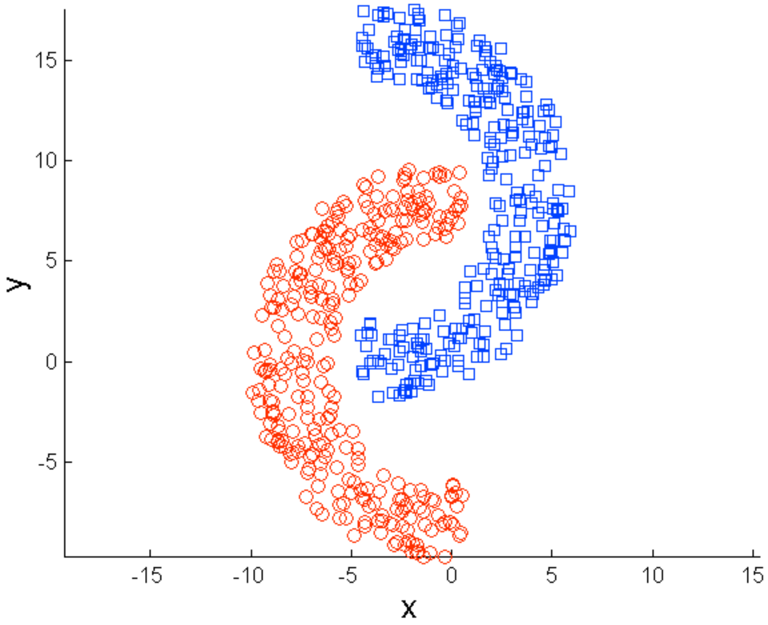




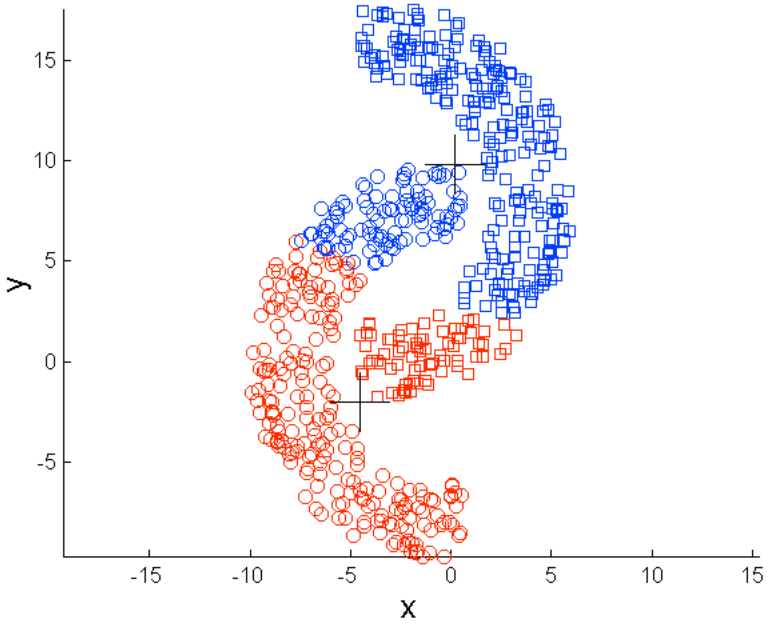
Original Points

K-means (3 Clusters)

Limitations of K-means: Non-globular Shapes

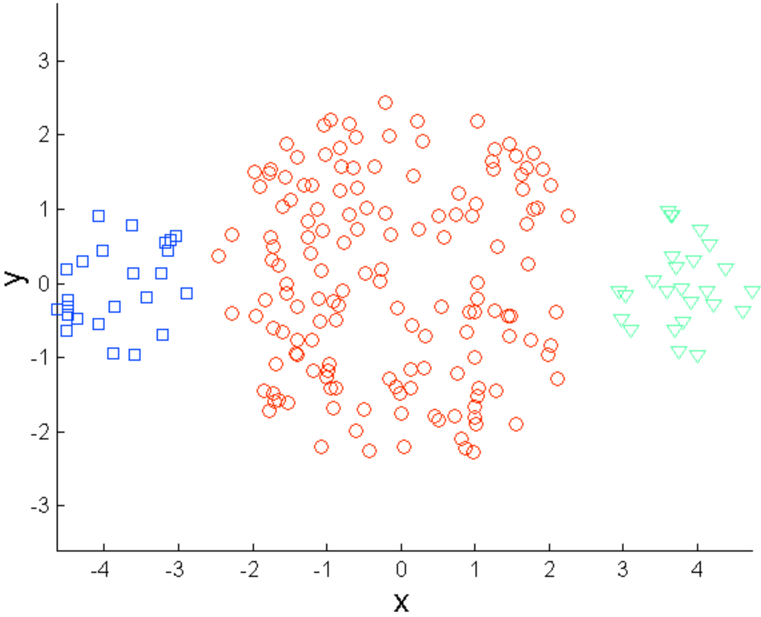


Original Points

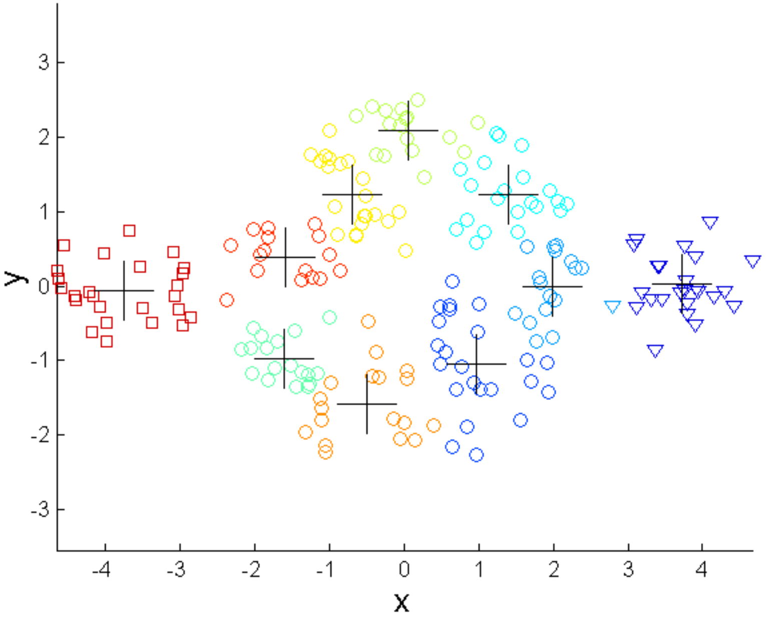


K-means (2 Clusters)

Overcoming K-means Limitations



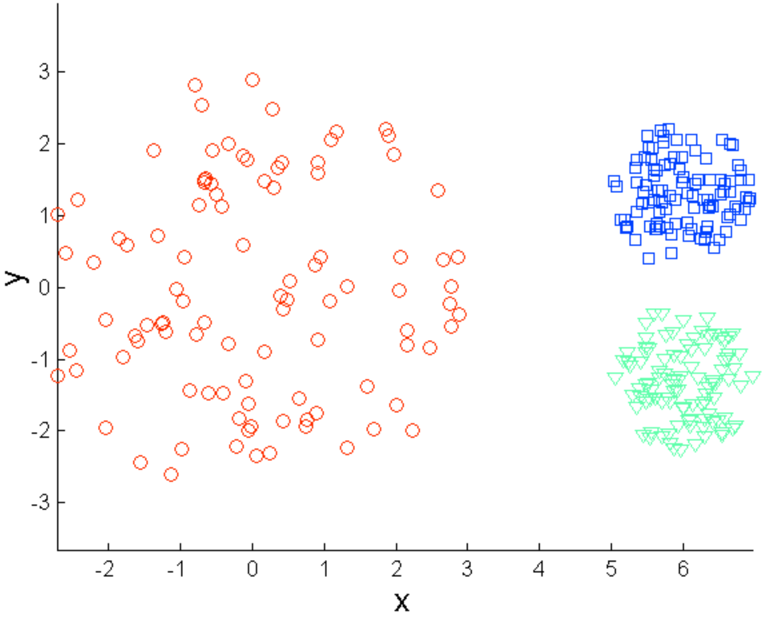
Original Points



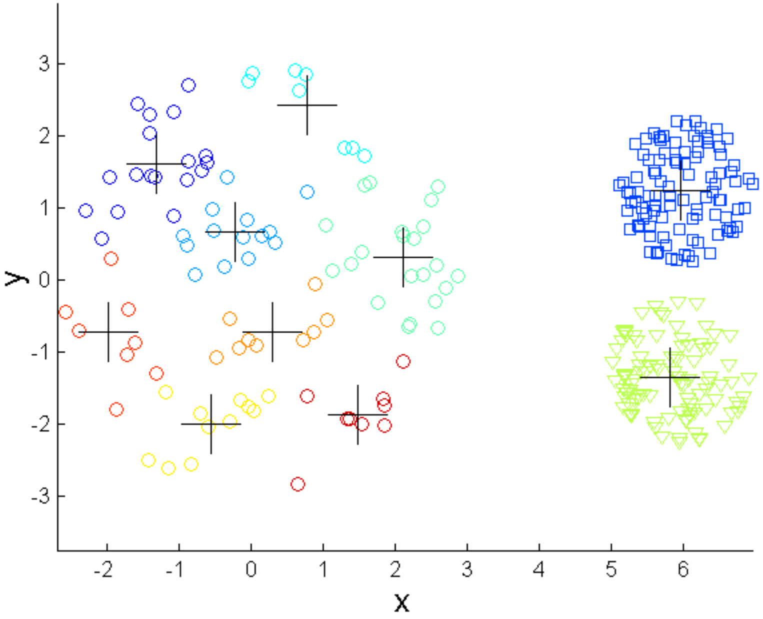
K-means Clusters

One solution is to find a large number of clusters such that each of them represents a part of a natural cluster. But these small clusters need to be put together in a post-processing step.

Overcoming K-means Limitations



Original Points

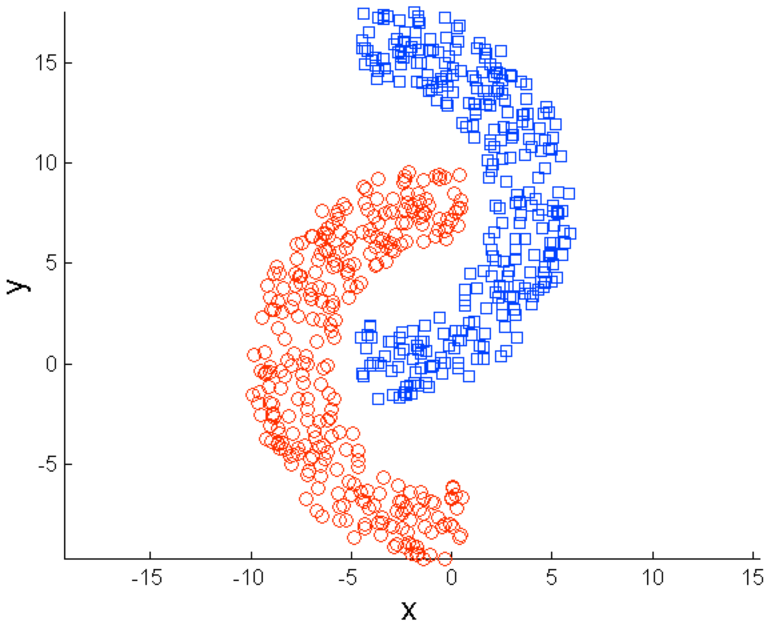


K-means Clusters

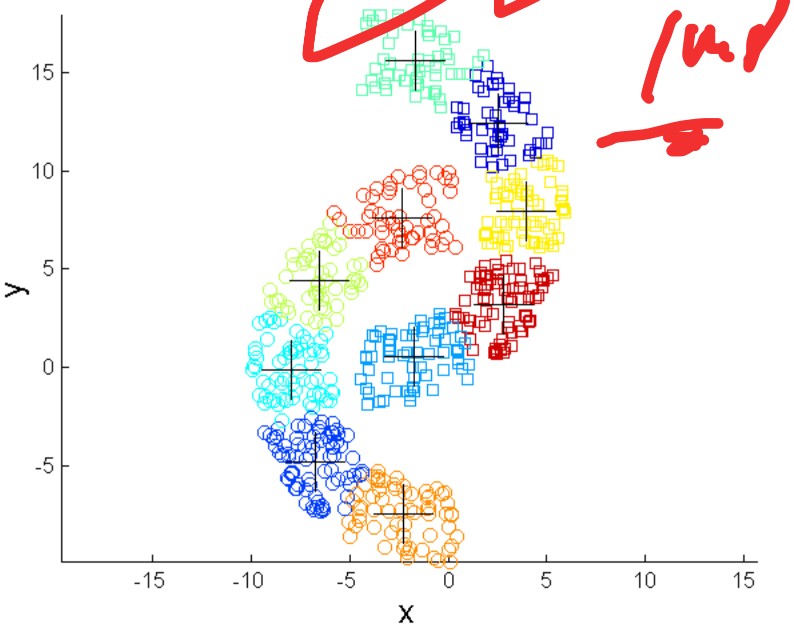
One solution is to find a large number of clusters such that each of them represents a part of a natural cluster. But these small clusters need to be put together in a post-processing step.

Overcoming K-means Limitations

Non- globular shap



Original Points



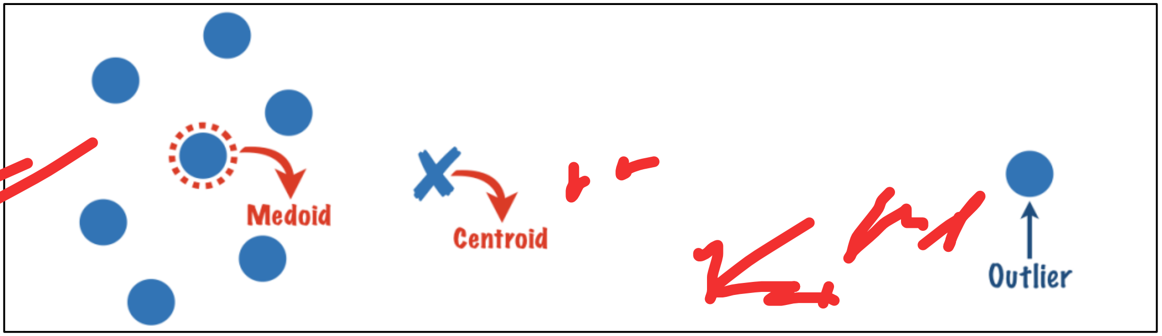
K-means Clusters

One solution is to find a large number of clusters such that each of them represents a part of a natural cluster. But these small clusters need to be put together in a post-processing step.

K-Medoids Clustering Algorithm (PAM)

What is Medoid?

- **Medoids** are similar in concept to means or centroids, but medoids are always restricted to be members of the data set.
- **Medoids** are most commonly used on data when a mean or centroid cannot be defined, such as graphs.
- **Note:** A medoid is not equivalent to a median.



K-Medoids Clustering Algorithm (PAM)

- The **k-medoids algorithm** is a clustering algorithm related to the k-means algorithm also called as the medoid shift algorithm.
- Both the k-means and k-medoids algorithms are partitional (breaking the dataset up into groups).
- In contrast to the k-means algorithm, k-medoids chooses datapoints as centers (medoids or exemplars).
- K-medoids is also a partitioning technique of clustering that clusters the data set of n objects into k clusters with k known a priori.
- It could be more robust to noise and outliers as compared to k-means because it minimizes a sum of general pairwise dissimilarities instead of a sum of squared Euclidean distances.
- A medoid of a finite dataset is a data point from this set, whose average dissimilarity to all the data points is minimal i.e. it is the most centrally located point in the set.

K-Medoids Clustering Algorithm (PAM) Cont..

- It was proposed in 1987 by Kaufman and Rousseeuw.
- A medoid can be defined as the point in the cluster, whose dissimilarities with all the other points in the cluster is minimum.
- The dissimilarity of the medoid(Ci) and object(Pi) or items is calculated by using E = |Pi - Ci|
- The cost in K-Medoids algorithm is given as $C = \sum_{Ci}^{n} \sum_{Pi\text{€}Ci}^{n} |Pi \ - Ci|$
- Steps for k-medoid clustering (**Partitioning Around Medoids (PAM))** algorithm follows as..

1. **Initialize**: randomly select *k* of the *n* data points as the medoids.
2. **Assignment step**: Associate each data point to the closest medoid.
3. **Update step**:
   - For each medoid *m* and each data point *o* associated to *m* swap *m* and *o* and compute the total cost of the configuration (that is, the average dissimilarity of *o* to all the data points associated to *m*).
   - Select the medoid *o* with the lowest cost of the configuration.
4. Repeat alternating steps 2 and 3 until there is no change in the assignments.

$\sum_{Ci}^{n} \sum$

K-Medoids Clustering Algorithm - Example

Sr.	X	Y
0	8	7
1	3	7
2	4	9
3	9	6
4	8	5
5	5	8
6	7	3
7	8	4
8	7	5
9	4	5

Step 1:
Let the randomly selected 2 **medoids**, so select k = 2 and let **C1 -(4, 5)** and **C2 -(8, 5)** are the two medoids.

The dissimilarity of each non-medoid point with the medoids is calculated and

Sr.	X	Y	Dissimilarity From C1	Dissimilarity From C2
0	8	7	|(8-4)|+|(7-5)| = 6	|(8-8)|+|(7-5)| = 2
1	3	7		
2	4	9		
3	9	6		
5	5	8		
6	7	3		
7	8	4		
8	7	5		

K-Medoids Clustering Algorithm – Example Cont..

Sr.	X	Y	Dissimilarity From C1	Dissimilarity From C2
0	8	7	6	**2**
1	3	7	**3**	7
2	4	9	**4**	8
3	9	6	6	**2**
5	5	8	**4**	6
6	7	3	5	**3**
7	8	4	5	**1**
8	7	5	3	**1**

- Each point is assigned to the cluster of that medoid whose dissimilarity is less.
- The points **1, 2, 5 go to cluster C1** and **0, 3, 6, 7, 8 go to cluster C2**.
- The Cost = (3 + 4 + 4) + (2 + 2 + 3 + 1 + 1) = 20

K-Medoids Clustering Algorithm – Example Cont..

Sr.	X	Y	Dissimilarity From C1	Dissimilarity From C2
0	8	7	6	3
1	3	7	3	8
2	4	9	4	9
3	9	6	6	3
4	8	5	4	1
5	5	8	4	7
6	7	3	5	2
8	7	5	3	2

- **Step 3: randomly select one non-medoid point and recalculate the cost.**
- Let the randomly selected point be (8, 4).
- The dissimilarity of each non-medoid point with the medoids – **C1 (4, 5)** and **C2 (8, 4)** is calculated and tabulated.

K-Medoids Clustering Algorithm – Example Cont..

Sr.	X	Y	Dissimilarity From C1	Dissimilarity From C2
0	8	7	6	**3**
1	3	7	**3**	8
2	4	9	**4**	9
3	9	6	6	**3**
4	8	5	4	**1**
5	5	8	**4**	7
6	7	3	5	**2**
8	7	5	3	**2**

- Each point is assigned to that cluster whose dissimilarity is less. **So, the points 1, 2, 5 go to cluster C1** and **0, 3, 4, 6, 8 go to cluster C2**.
- The New cost,

  = (3 + 4 + 4) + (3 + 3 + 1 + 2 + 2) = 22
- Swap Cost = New Cost – Previous Cost

  = 22 – 20

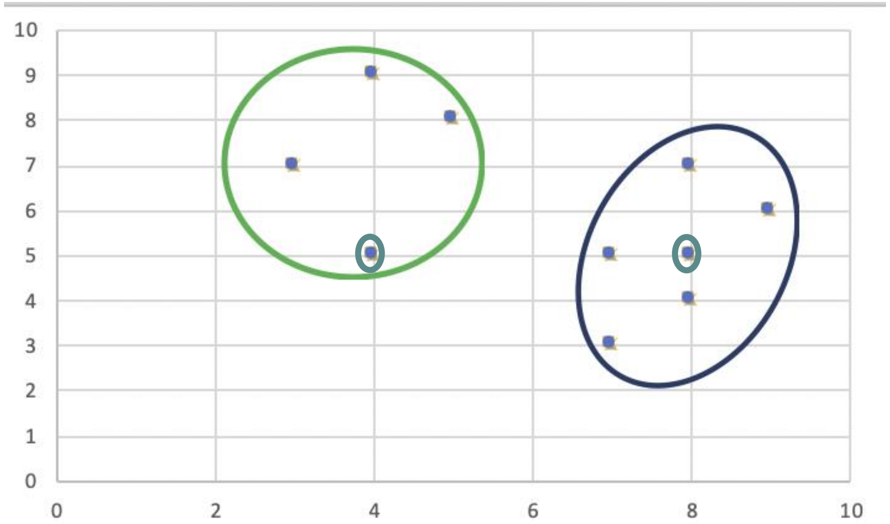
  = 2
- So, 2>0 that is positive, now our previous medoid is best.
- **The total cost of Medoid (8,4) > the total cost when (8,5) was the medoid earlier & it generates the same clusters as earlier.**
- If you get negative then you have to take new medoid and recalculate again.

K-Medoids Clustering Algorithm – Example Cont..

Cost Movement function.

Sr.	X	Y
0	8	7
1	3	7
2	4	9
3	9	6
4	**8**	**5**
5	5	8
6	7	3
7	8	4
8	7	5
9	**4**	**5**

- As the swap cost is not less than zero, we undo the swap.
- Hence (4, 5) and (8, 5) are the final medoids.
- The clustering would be in the following way

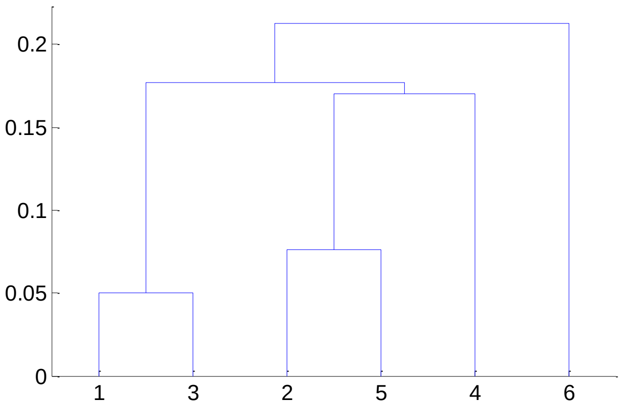


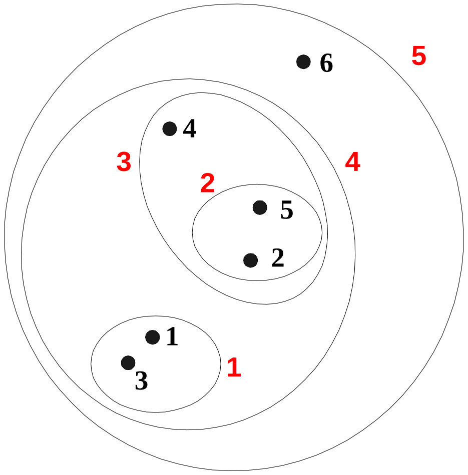
K-Medoids Clustering Algorithm (Try Yourself!!)

Sr.	X	Y
0	2	6
1	3	4
2	3	8
3	4	7
4	6	2
5	6	4
6	7	3
7	7	4
8	8	5
9	7	6

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 an individual 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

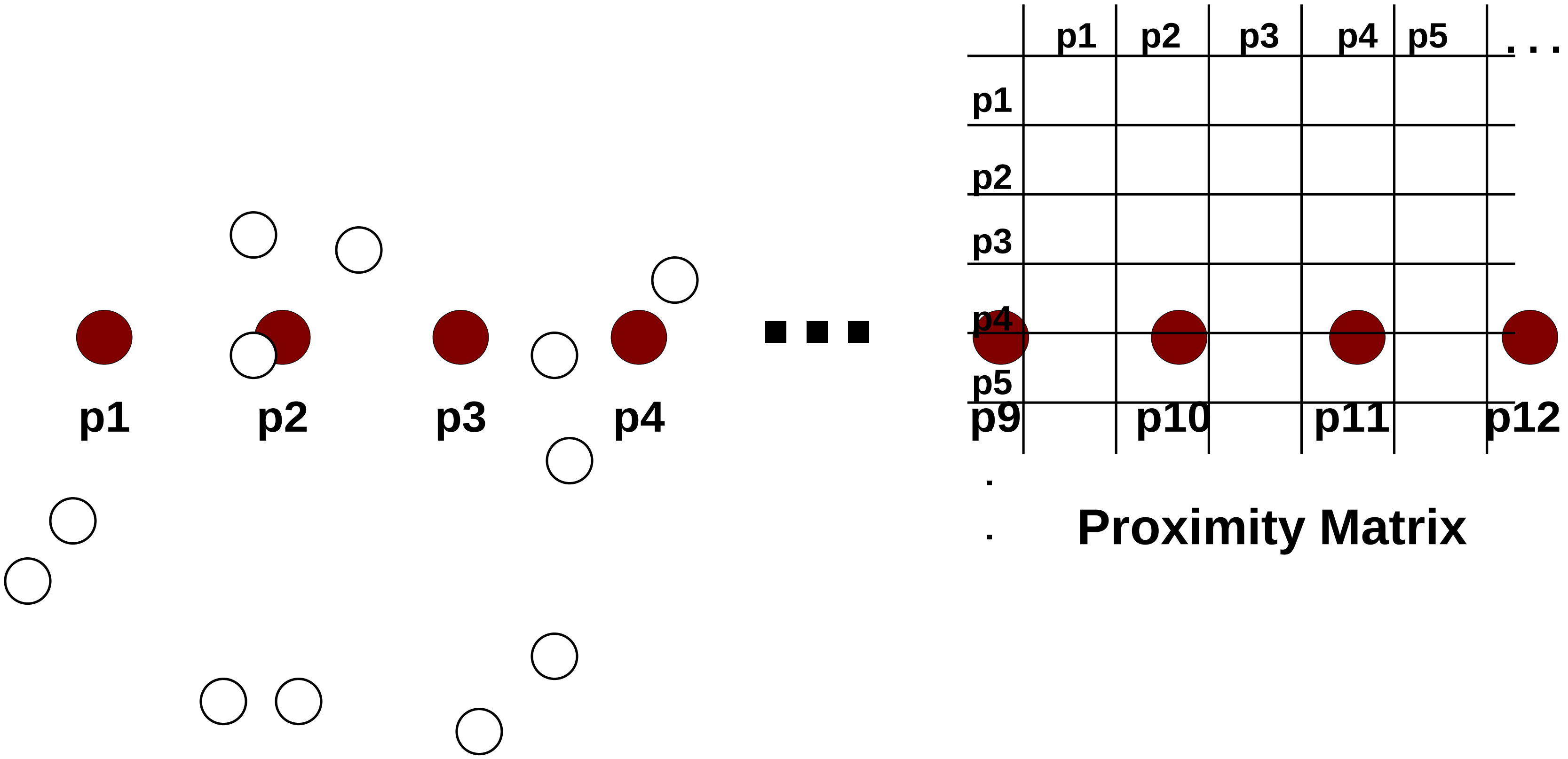
- **Key Idea: Successively merge closest clusters**
- Basic algorithm

```
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

Steps 1 and 2

- Start with clusters of individual points and a proximity matrix

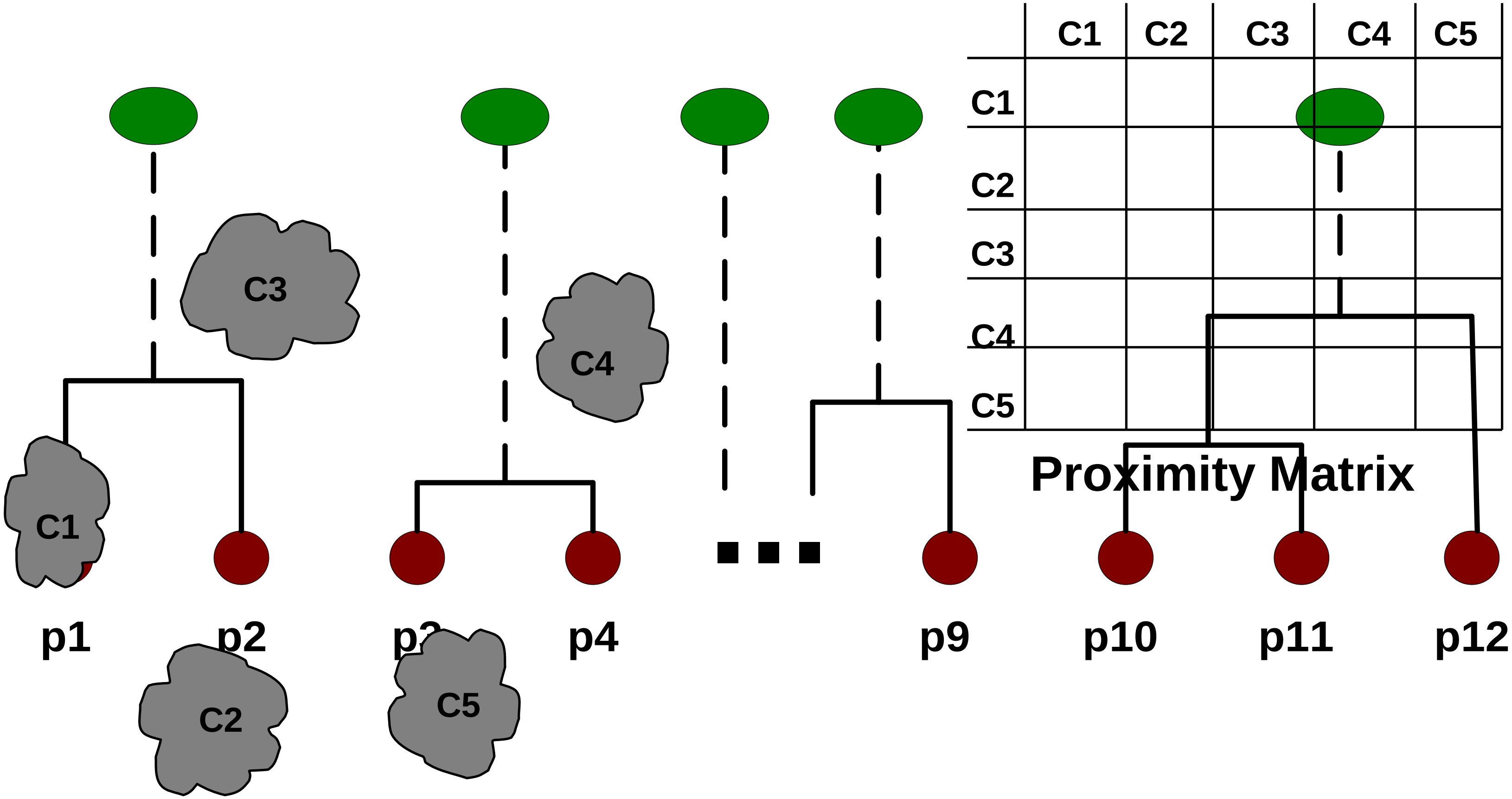


	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						

Proximity Matrix

Intermediate Situation

- After some merging steps, we have some clusters

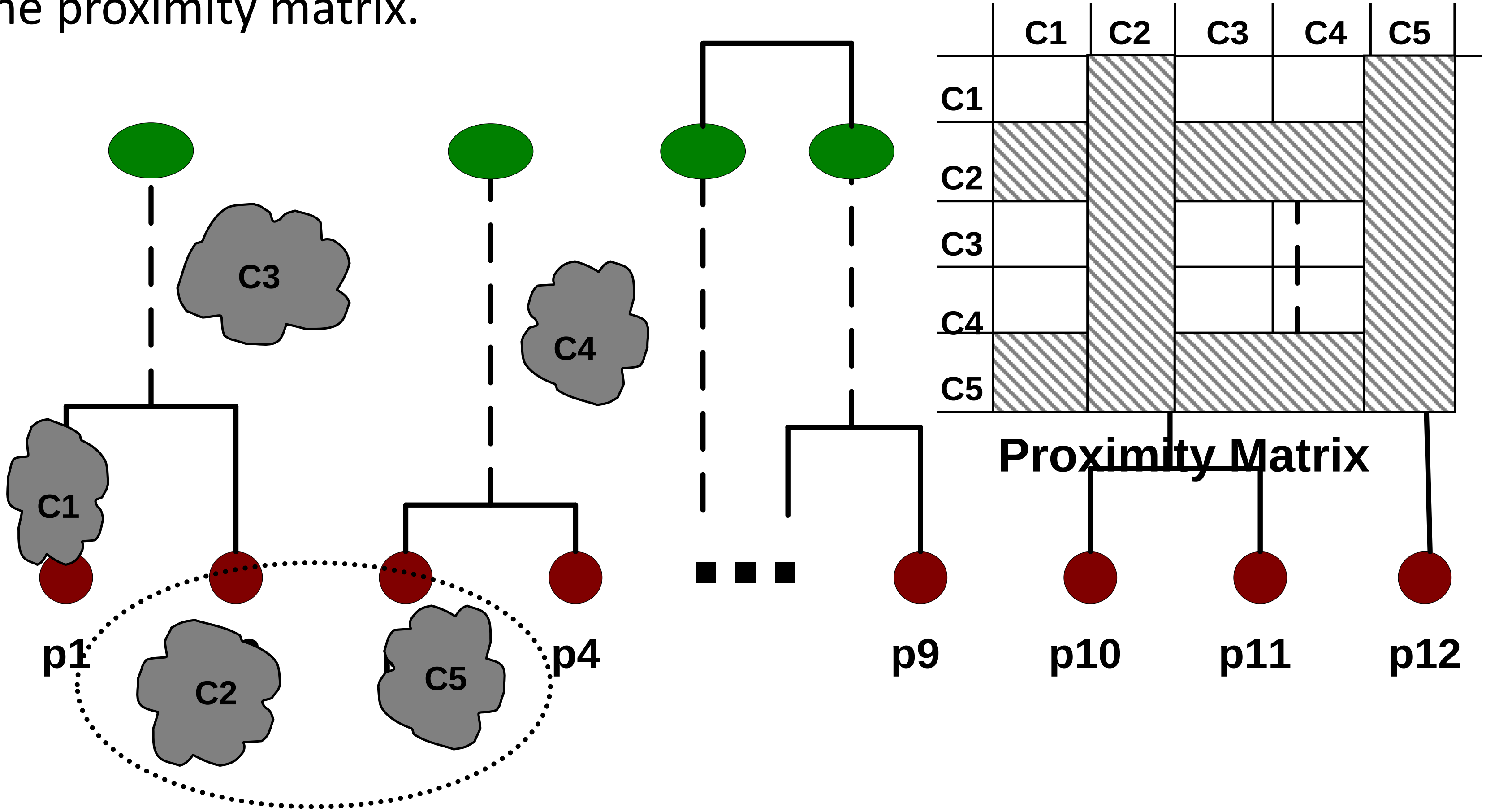


	C1	C2	C3	C4	C5
C1					
C2					
C3					
C4					
C5					

Proximity Matrix

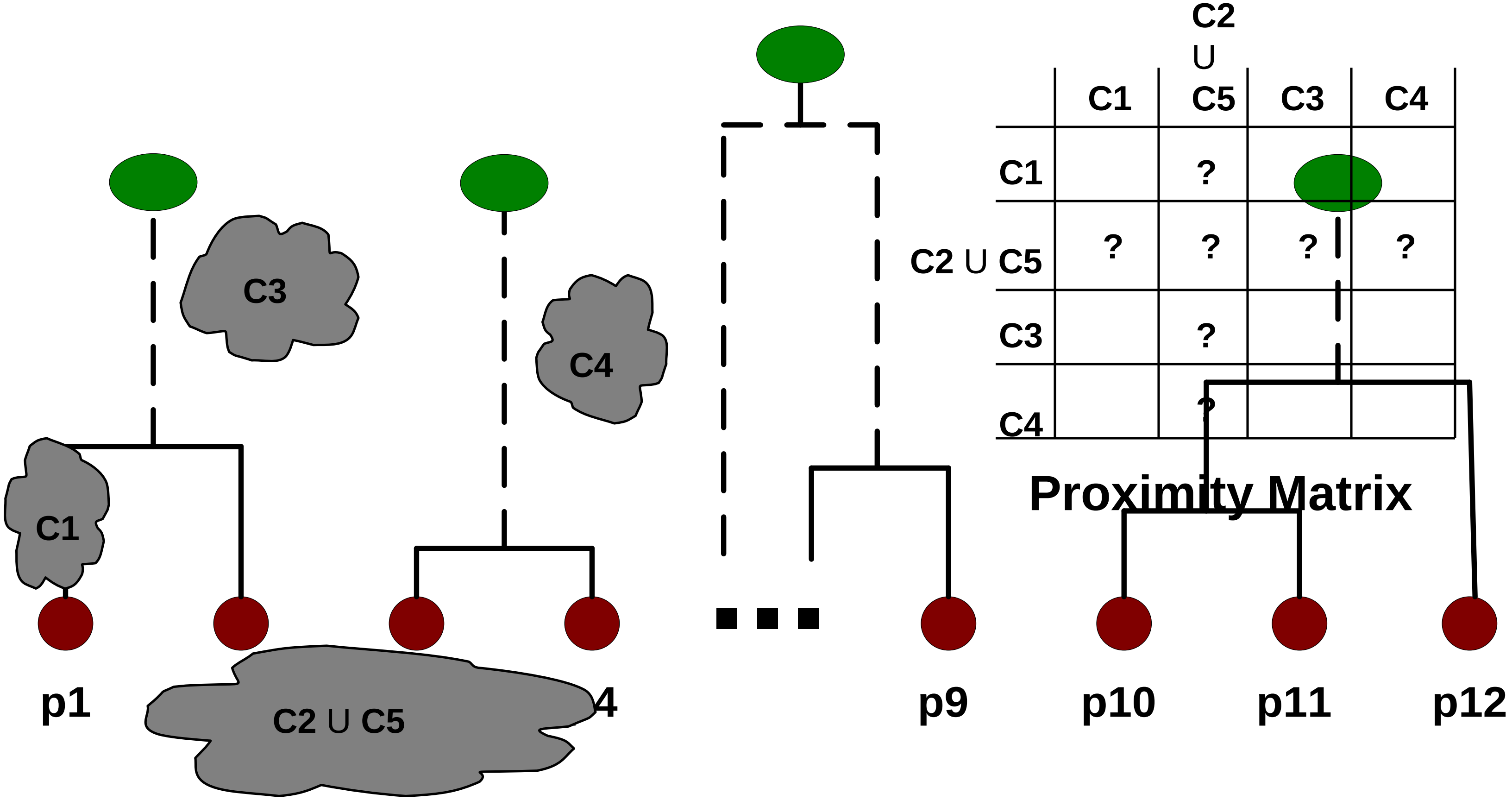
Step 4

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



Step 5

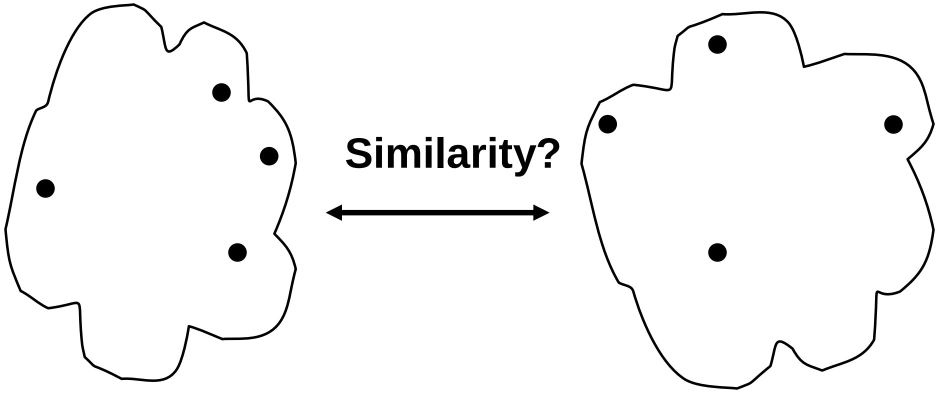
- The question is “How do we update the proximity matrix?”



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

Proximity Matrix

How to Define Inter-Cluster Distance

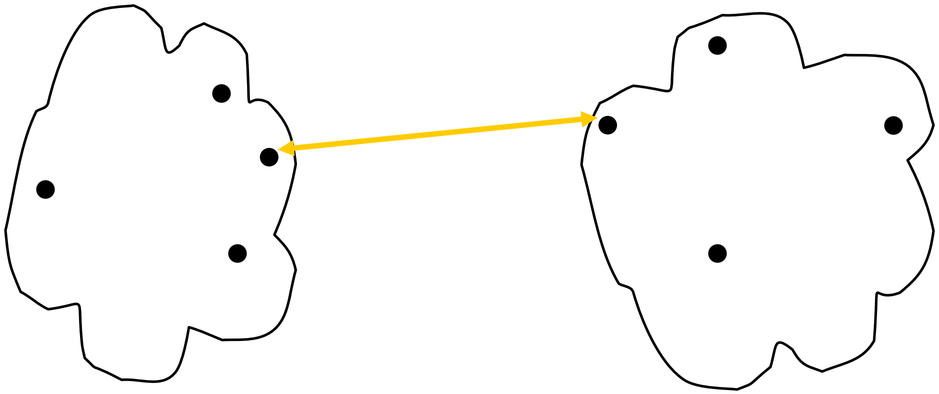


	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

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

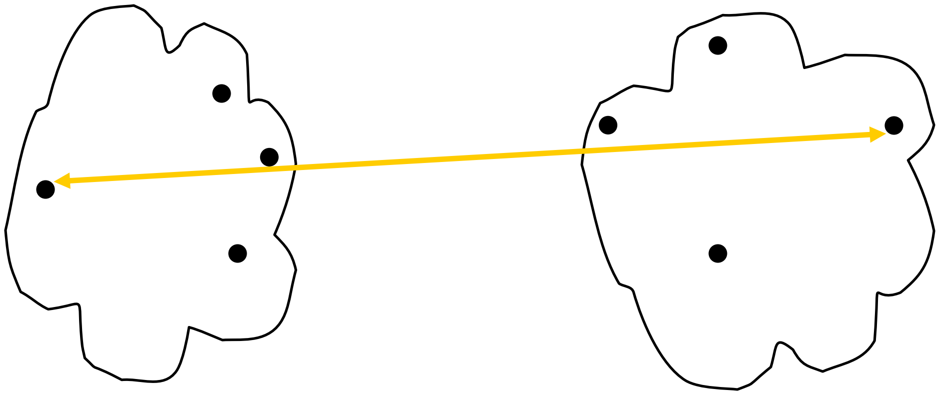


	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

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						

.

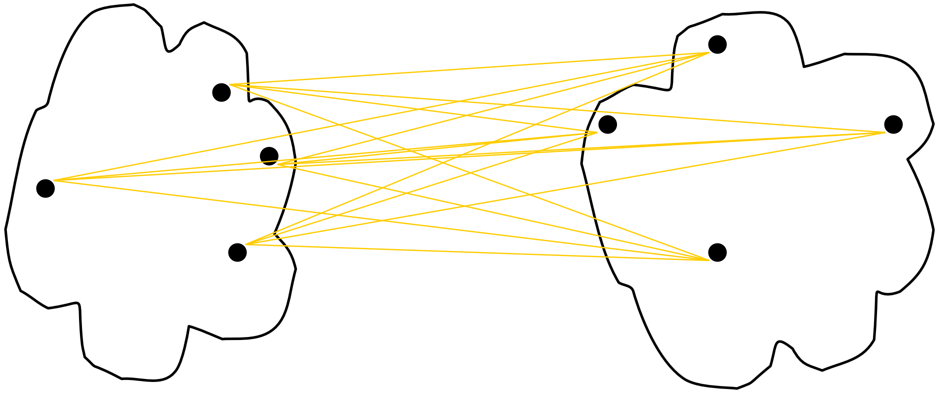
.

.

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

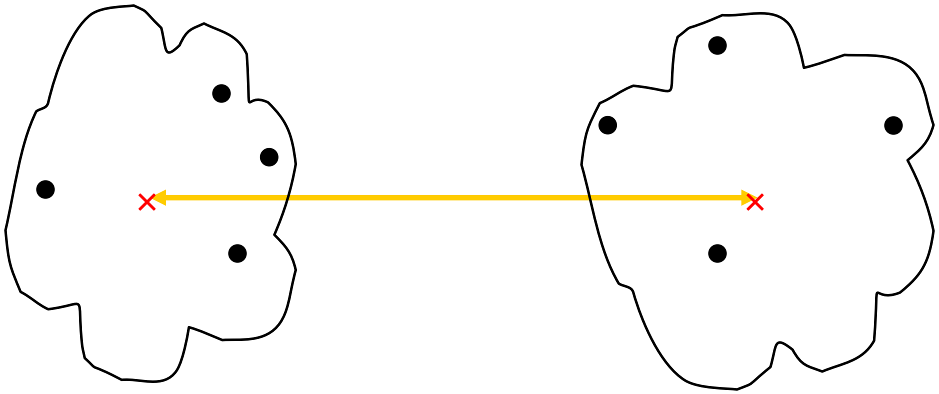


	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

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



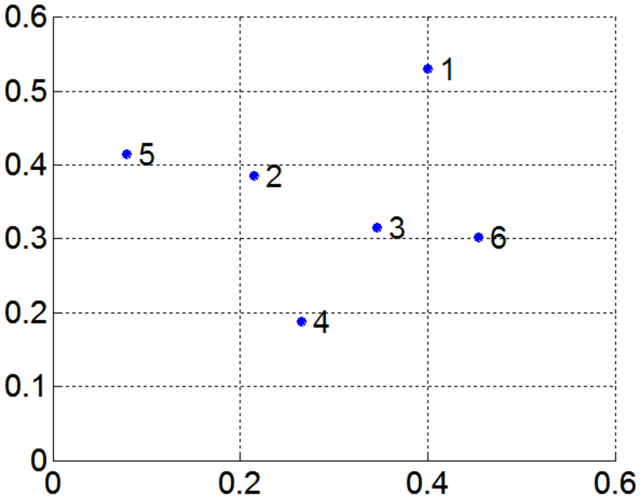
	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
. . .						

Proximity Matrix

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

MIN or Single Link

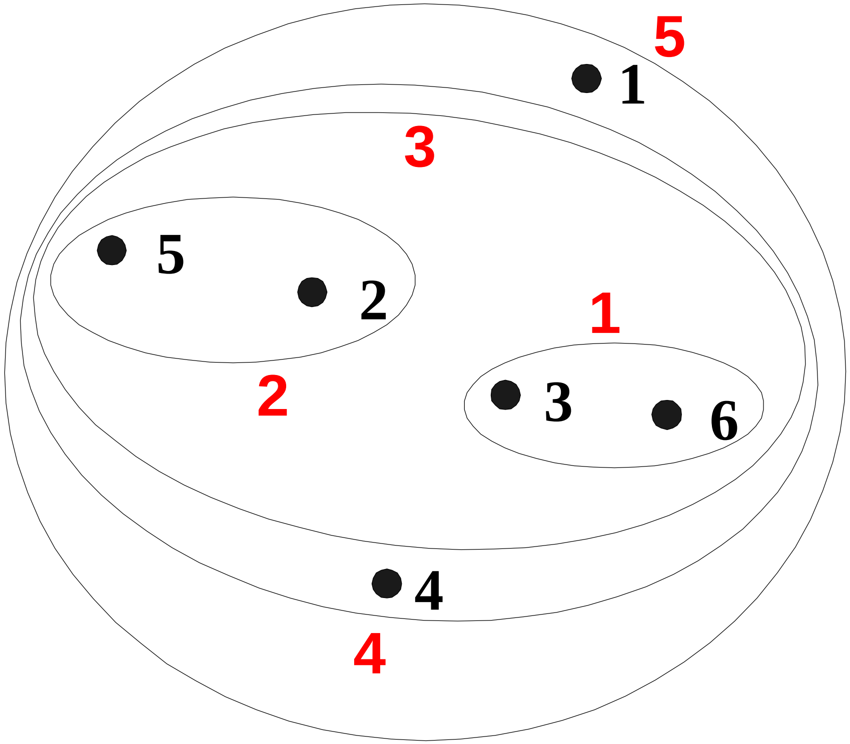
- Proximity of two clusters is based on the two closest points in the different clusters
  - Determined by one pair of points, i.e., by one link in the proximity graph
- Example:

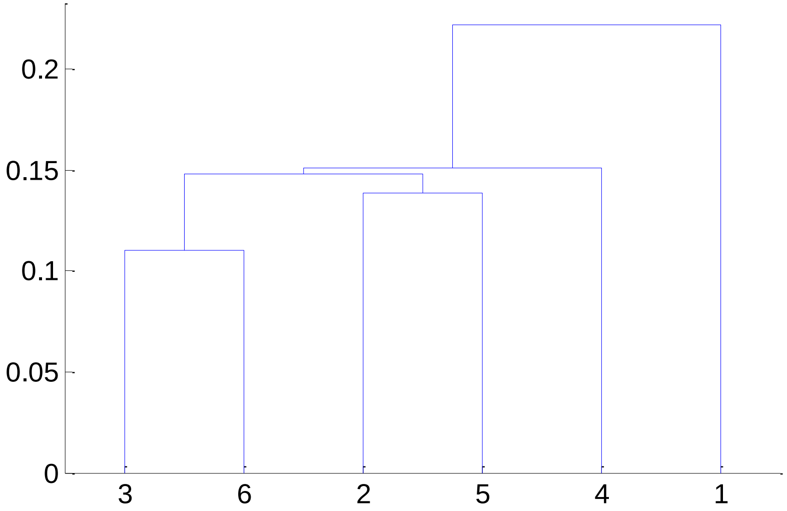


Distance Matrix:

	p1	p2	p3	p4	p5	p6
p1	0.00	0.24	0.22	0.37	0.34	0.23
p2	0.24	0.00	0.15	0.20	0.14	0.25
p3	0.22	0.15	0.00	0.15	0.28	0.11
p4	0.37	0.20	0.15	0.00	0.29	0.22
p5	0.34	0.14	0.28	0.29	0.00	0.39
p6	0.23	0.25	0.11	0.22	0.39	0.00

Hierarchical Clustering: MIN

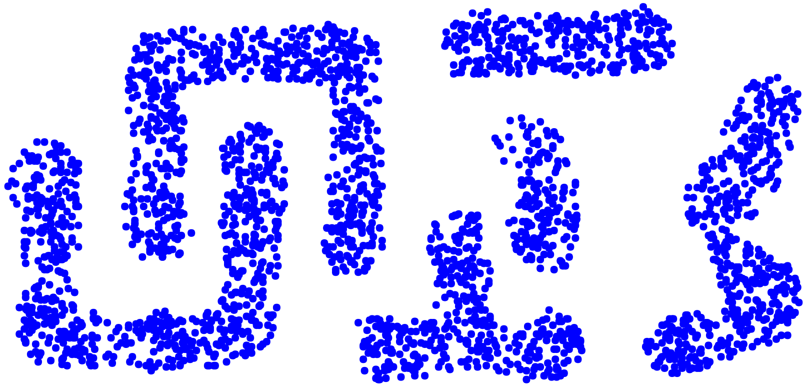




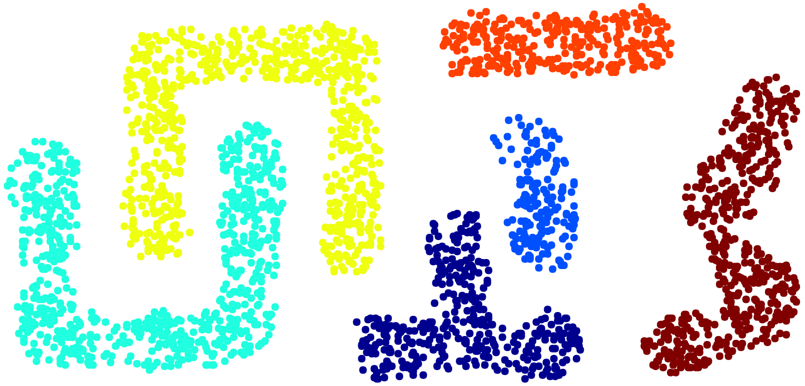
Nested Clusters

Dendrogram

Strength of MIN



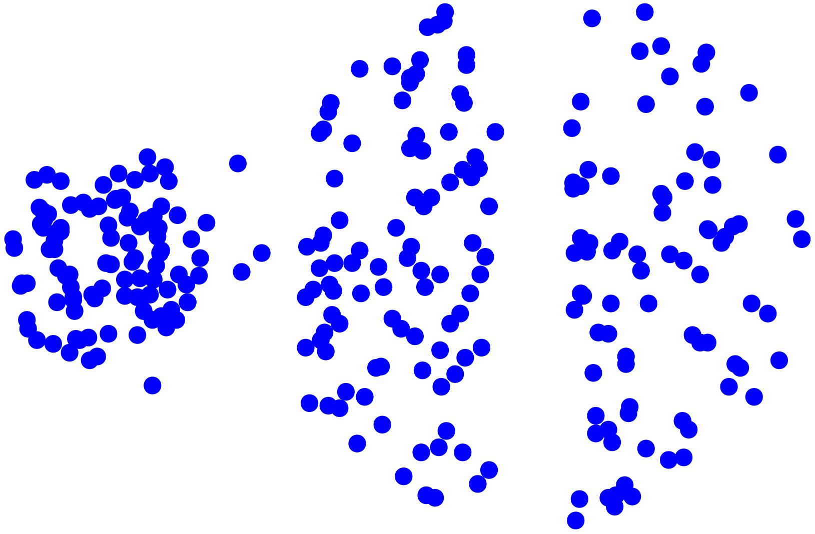
Original Points



Six Clusters

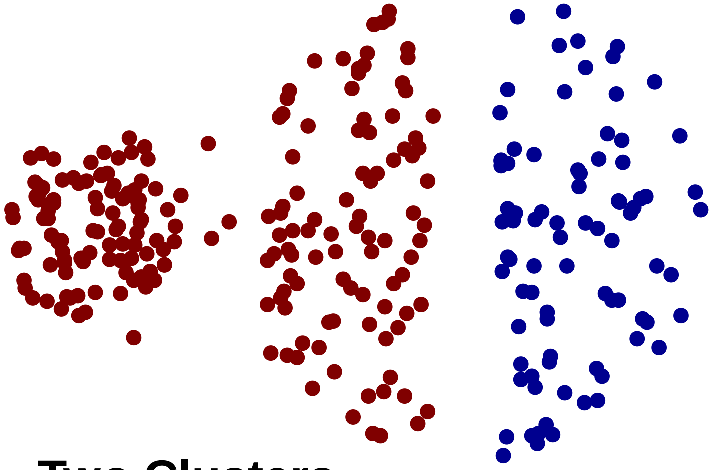
- **Can handle non-elliptical shapes**

Limitations of MIN

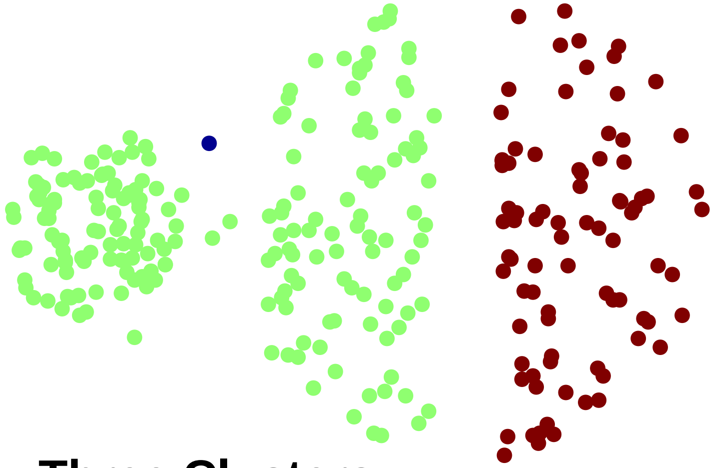


Original Points

- Sensitive to noise



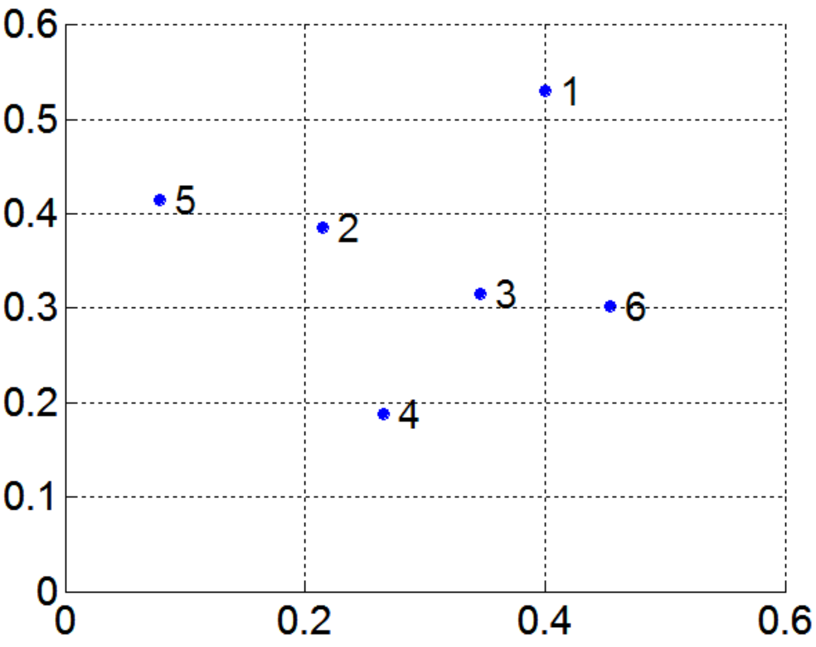
Two Clusters



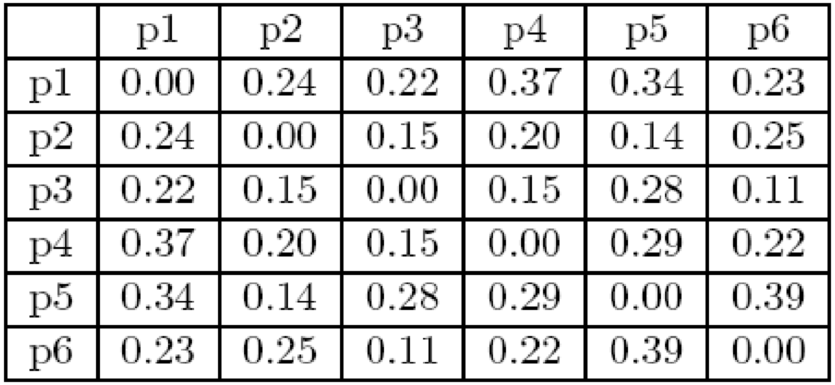
Three Clusters

MAX or Complete Linkage

- Proximity of two clusters is based on the two most distant points in the different clusters
  - Determined by all pairs of points in the two clusters

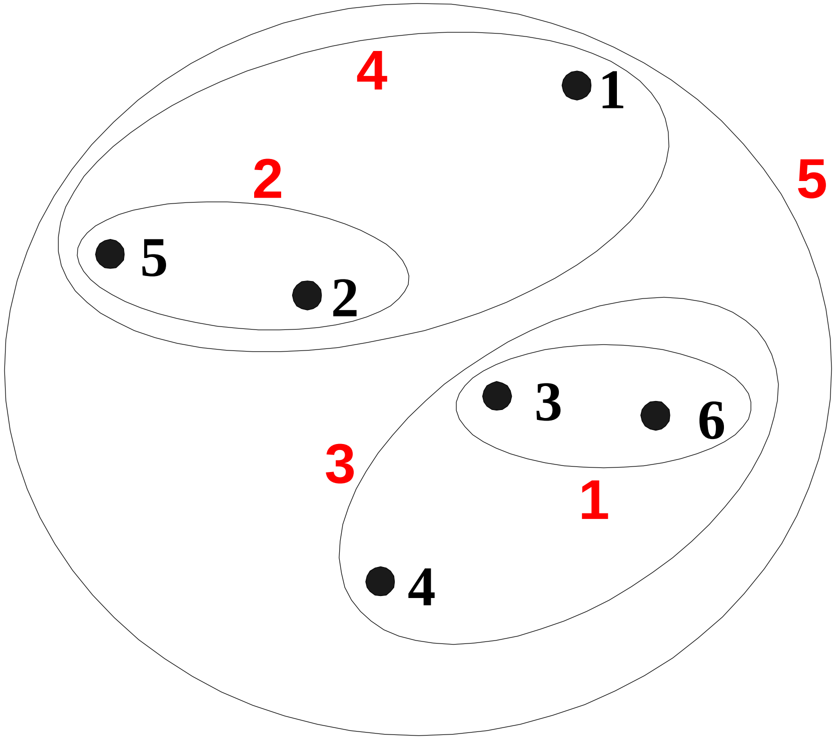


Distance Matrix:

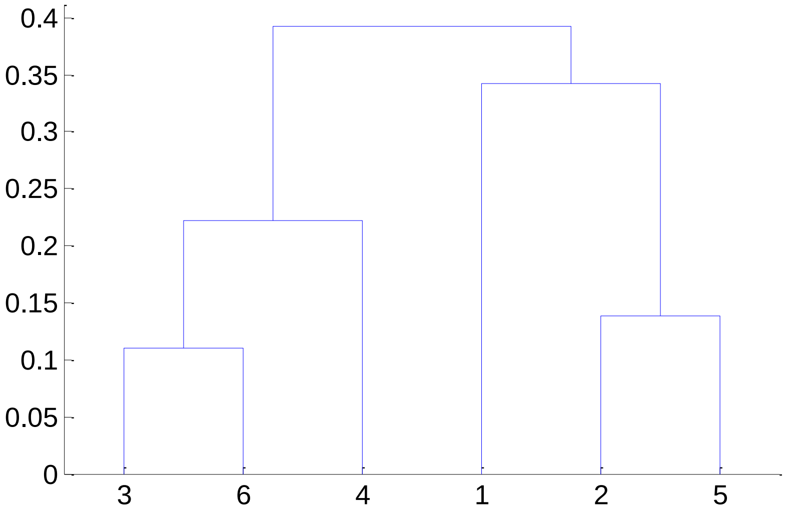


	p1	p2	p3	p4	p5	p6
p1	0.00	0.24	0.22	0.37	0.34	0.23
p2	0.24	0.00	0.15	0.20	0.14	0.25
p3	0.22	0.15	0.00	0.15	0.28	0.11
p4	0.37	0.20	0.15	0.00	0.29	0.22
p5	0.34	0.14	0.28	0.29	0.00	0.39
p6	0.23	0.25	0.11	0.22	0.39	0.00

Hierarchical Clustering: MAX

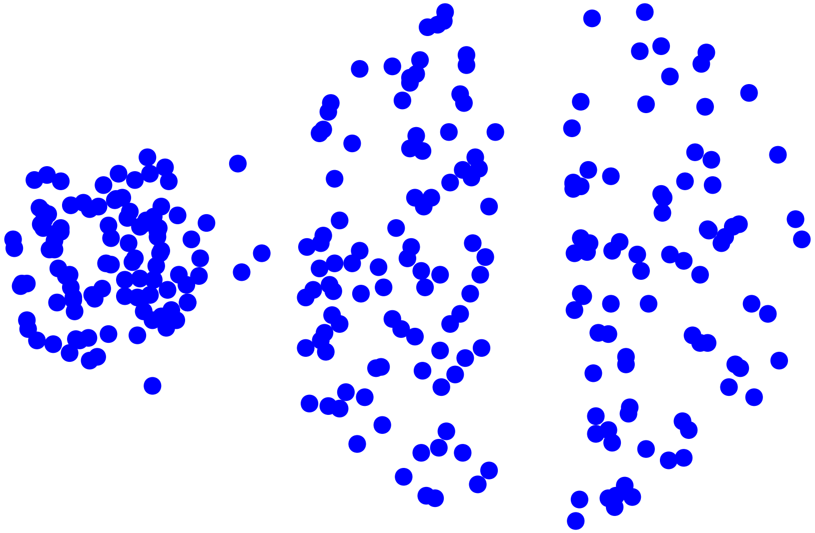


Nested Clusters

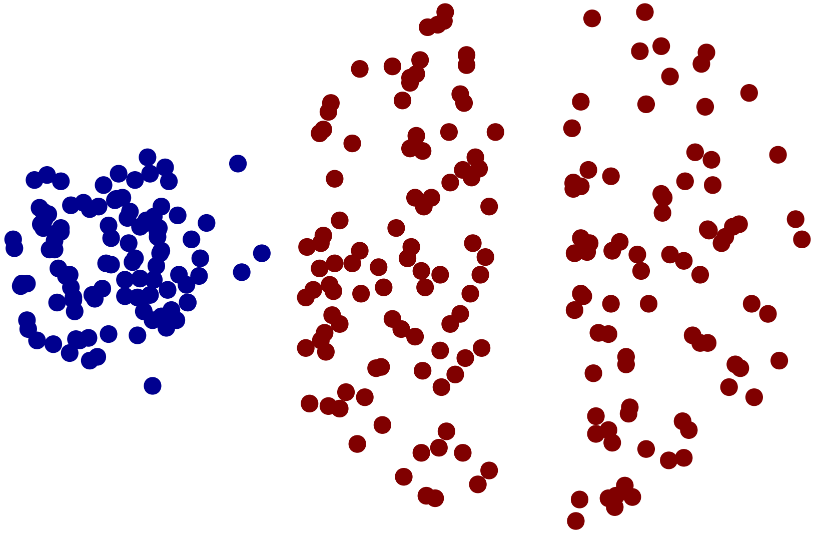


Dendrogram

Strength of MAX



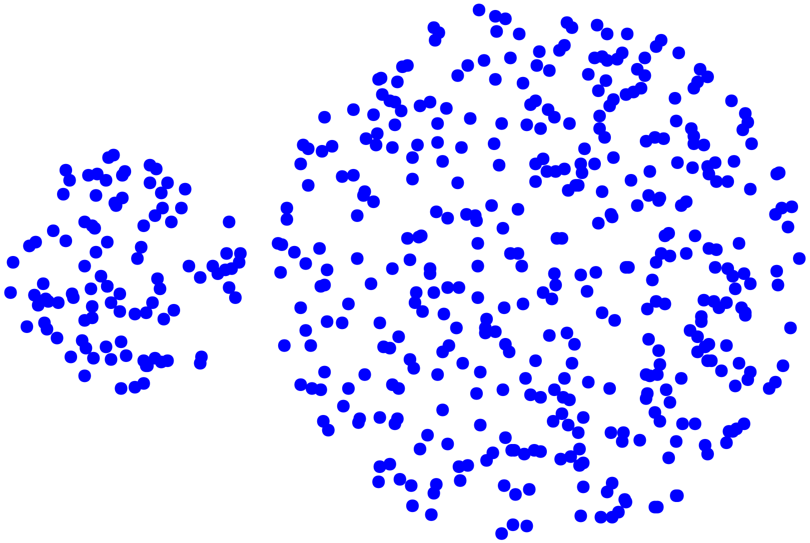
Original Points



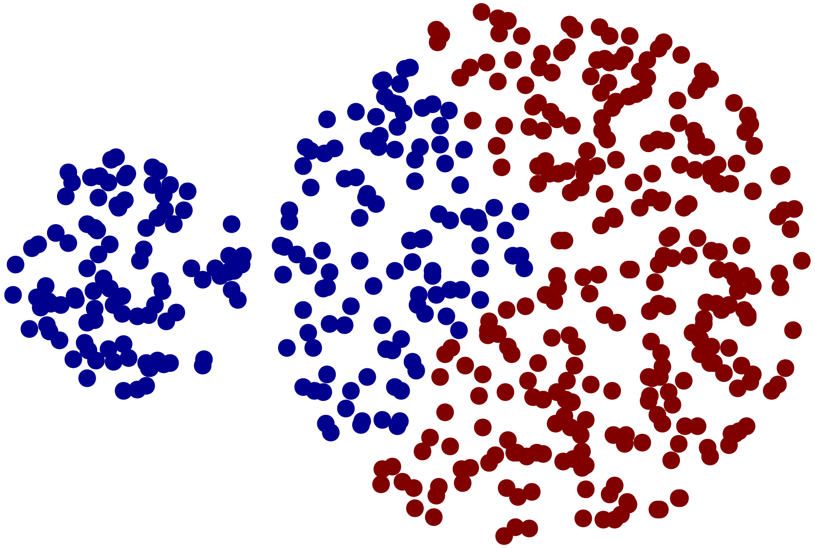
Two Clusters

- Less susceptible to noise

Limitations of MAX



Original Points



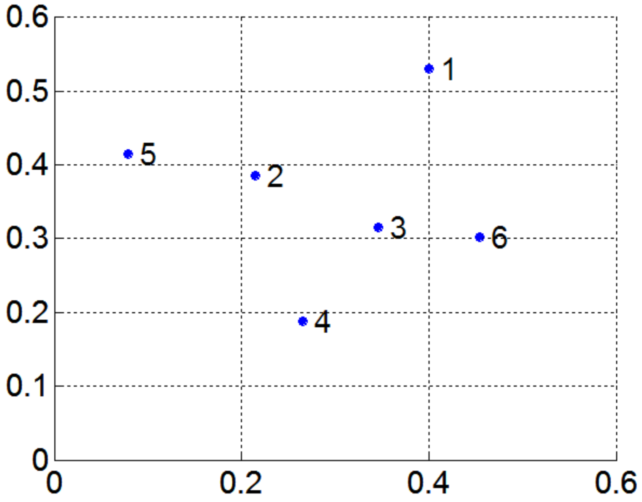
Two Clusters

- Tends to break large clusters
- Biased towards globular clusters

Group Average

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

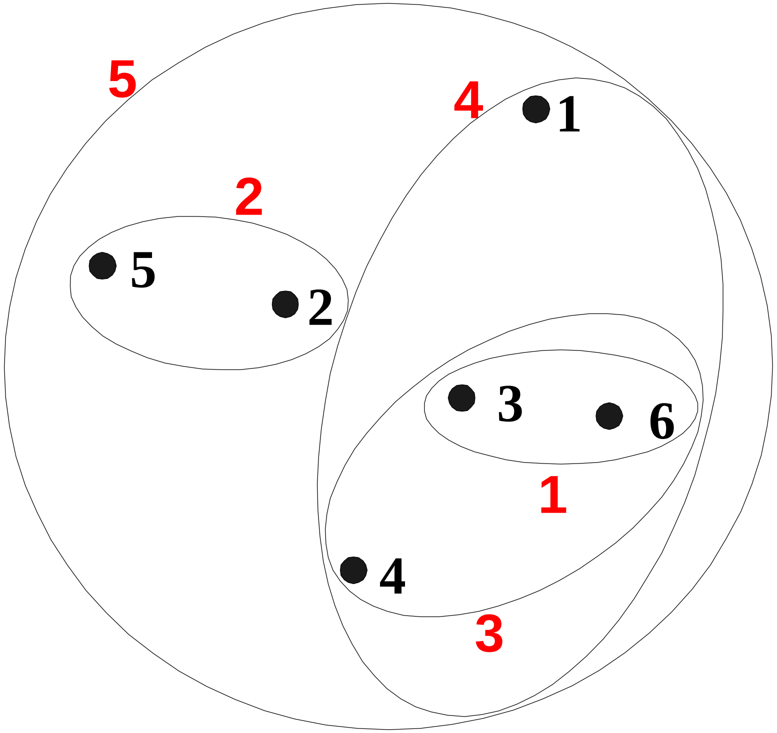
$$\mathbf{proximity}(\mathbf{Cluster}_i, \mathbf{Cluster}_j) = \frac{\sum\limits_{\substack{p_i \in \mathrm{Cluster}_i \\ p_j \in \mathrm{Cluster}_j}} \mathbf{proximity}(p_i, p_j)}{|\mathbf{Cluster}_i| \times |\mathbf{Cluster}_j|}$$

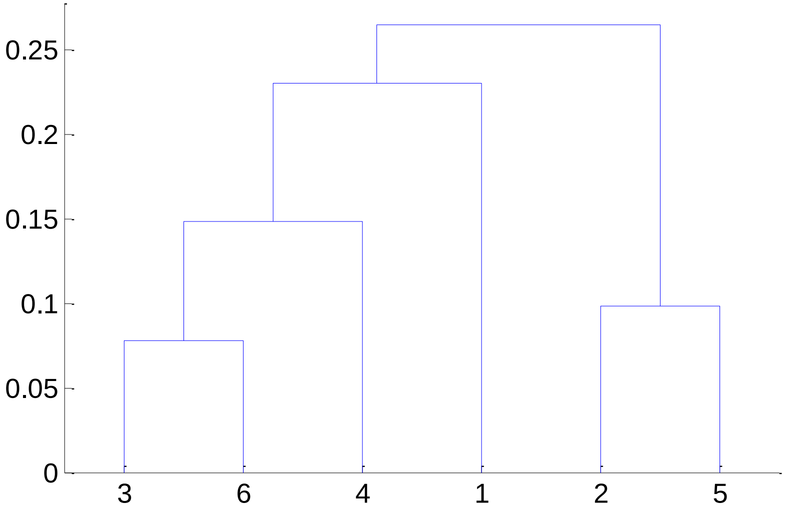


Distance Matrix:

	p1	p2	p3	p4	p5	p6
p1	0.00	0.24	0.22	0.37	0.34	0.23
p2	0.24	0.00	0.15	0.20	0.14	0.25
p3	0.22	0.15	0.00	0.15	0.28	0.11
p4	0.37	0.20	0.15	0.00	0.29	0.22
p5	0.34	0.14	0.28	0.29	0.00	0.39
p6	0.23	0.25	0.11	0.22	0.39	0.00

Hierarchical Clustering: Group Average





Nested Clusters

Dendrogram

Hierarchical Clustering: Group Average

- Compromise between Single and Complete Link

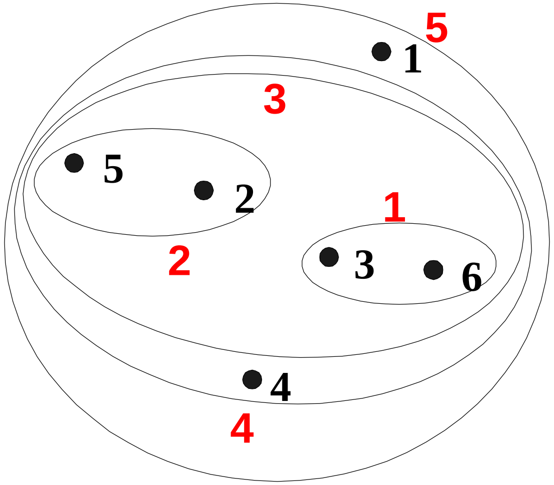
- Strengths
  - Less susceptible to noise

- Limitations
  - Biased towards globular clusters

Cluster Similarity: Ward's Method

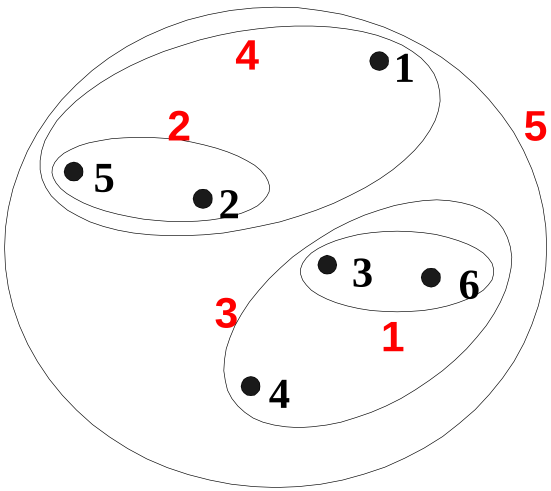
- Similarity of two clusters is based on the increase in squared error when two clusters are merged
  - Similar to group average if distance between points is distance squared
- Less susceptible to noise
- Biased towards globular clusters
- Hierarchical analogue of K-means
  - Can be used to initialize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

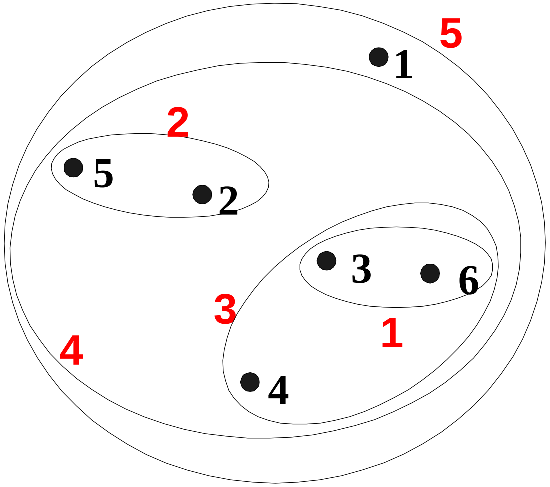
Hierarchical Clustering: Comparison



MIN

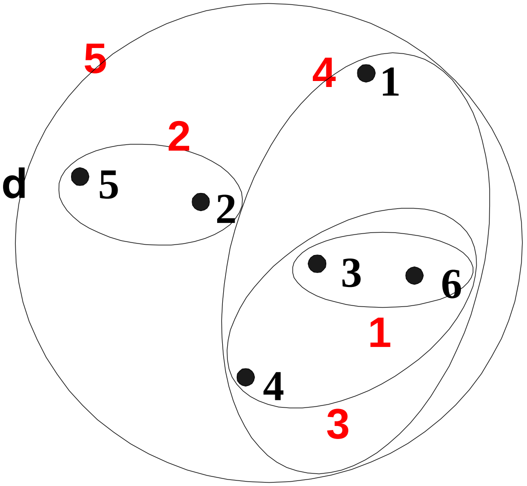
MAX





Group Average

Ward's Method



Hierarchical Clustering Example

Agglomerative Hierarchical Clustering - Example

	X	Y
P1	0.40	0.53
P2	0.22	0.38
P3	0.35	0.32
P4	0.26	0.19
P5	0.08	0.41
P6	0.45	0.30

- Calculate Euclidean distance, create the distance matrix.
- Distance [(x,y),(a,b)] = $\sqrt{(x - a)^2 + (y - b)^2}$

↪ ED $\mathbf{P1}$ & $\boldsymbol{P2}$ $(\mathbf{0.40}, \mathbf{0.53}), (\mathbf{0.22}, \mathbf{0.38})$

$= \sqrt{(0.40 - 0.22)^2 + (0.53 - 0.38)^2}$

$= \sqrt{(0.18)^2 + (0.15)^2}$

$= \sqrt{0.0324 + 0.0225}$

$= \sqrt{0.0549}$

$= 0.23$

	P1	P2	P3	P4	P5	P6
P1	0					
P2	0.23	0				
P3			0			
P4				0		
P5					0	
P6						0

Agglomerative Hierarchical Clustering - Example

	X	Y
P1	0.40	0.53
P2	0.22	0.38
P3	0.35	0.32
P4	0.26	0.19
P5	0.08	0.41
P6	0.45	0.30

- ED $\mathbf{P1}$ & $\boldsymbol{P3}$ $\mathbf{(0.40, 0.53), (0.35, 0.32)}$

$$= \sqrt{(0.40-0.35)^2+(0.53-0.32)^2}$$

$$= \sqrt{(0.05)^2+(0.21)^2}$$

$$= \sqrt{0.0025+0.0441}$$

$$= \sqrt{0.0466}$$

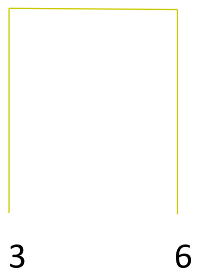
$$= 0.22$$

	P1	P2	P3	P4	P5	P6
P1	0					
P2	0.23	0				
P3	0.22		0			
P4				0		
P5					0	
P6						0

Agglomerative Hierarchical Clustering - Example

	X	Y
P1	0.40	0.53
P2	0.22	0.38
P3	0.35	0.32
P4	0.26	0.19
P5	0.08	0.41
P6	0.45	0.30

	P1	P2	P3	P4	P5	P6
P1	0					
P2	0.23	0				
P3	0.22	0.15	0			
P4	0.37	0.20	0.15	0		
P5	0.34	0.14	0.28	0.29	0	
P6	0.23	0.25	0.11	0.22	0.39	0



Agglomerative Hierarchical Clustering - Example

- To Update the distance matrix MIN[dist(P3,P6),P1]
- MIN (dist(P3,P1),(P6,P1))
- Min[(0.22,0.23)]
- 0.22

- To Update the distance matrix MIN[dist(P3,P6),P2]
- MIN (dist(P3,P2),(P6,P2))
- Min[(0.15,0.25)]
- 0.15

	P1	P2	P3	P4	P5	P6
P1	0					
P2	0.23	0				
P3	**0.22**	**0.15**	0			
P4	0.37	0.20	0.15	0		
P5	0.34	0.14	0.28	0.29	0	
P6	0.23	0.25	0.11	0.22	0.39	0

Agglomerative Hierarchical Clustering - Example

- To Update the distance matrix MIN[dist(P3,P6),P4]
- MIN (dist(P3,P4),(P6,P4))
- Min[(0.15,0.22)]
- 0.15

- To Update the distance matrix MIN[dist(P3,P6),P5]
- MIN (dist(P3,P5),(P6,P5))
- Min[(0.28,0.39)]
- 0.28

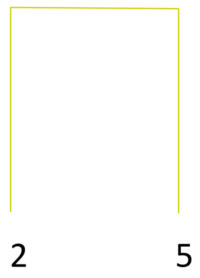
	P1	P2	P3	P4	P5	P6
P1	0					
P2	0.23	0				
P3	0.22	0.15	0			
P4	0.37	0.20	**0.15**	0		
P5	0.34	0.14	**0.28**	0.29	0	
P6	0.23	0.25	0.11	0.22	0.39	0

Agglomerative Hierarchical Clustering - Example

- The Updated distance matrix for cluster P3, P6

	P1	P2	P3,P6	P4	P5
P1	0				
P2	0.23	0			
P3,P6	0.22	0.15	0		
P4	0.37	0.20	0.15	0	
P5	0.34	0.14	0.28	0.29	0

	P1	P2	P3	P4	P5	P6
P1	0					
P2	0.23	0				
P3	0.22	0.15	0			
P4	0.37	0.20	0.15	0		
P5	0.34	0.14	0.28	0.29	0	
P6	0.23	0.25	0.11	0.22	0.39	0



Agglomerative Hierarchical Clustering - Example

- To Update the distance matrix MIN[dist(P2,P5),P1]
- MIN (dist(P2,P1),(P5,P1))
- Min[(0.23,0.34)]
- 0.23

- To Update the distance matrix MIN[dist(P2,P5),(P3,P6)
- MIN [(dist(P2,(P3,P6)),(P5,(P3,P6))]
- Min[(0.15,0.28)]
- 0.15

	P1	**P2**	**P3,P6**	**P4**	**P5**
P1	0				
P2	0.23	0			
P3,P6	0.22	0.15	0		
P4	0.37	0.20	0.15	0	
P5	0.34	0.14	0.28	0.29	0

Agglomerative Hierarchical Clustering - Example

- To Update the distance matrix MIN[dist(P2,P5),P4]
- MIN (dist(P2,P4),(P5,P4))
- Min[(0.20,0.29)]
- 0.20

	P1	P2	P3,P6	P4	P5
P1	0				
P2	0.23	0			
P3,P6	0.22	0.15	0		
P4	0.37	0.20	0.15	0	
P5	0.34	0.14	0.28	0.29	0

Agglomerative Hierarchical Clustering - Example

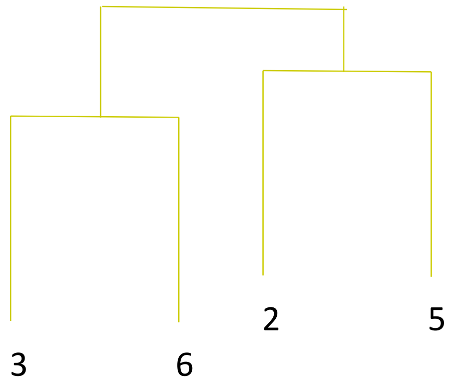
	P1	P2	P3,P6	P4	P5
P1	0				
P2	0.23	0			
P3,P6	0.22	0.15	0		
P4	0.37	0.20	0.15	0	
P5	0.34	0.14	0.28	0.29	0

	P1	P2,P5	P3,P6	P4
P1	0			
P2,P5	0.23	0		
P3,P6	0.22	0.15	0	
P4	0.37	0.20	0.15	0

Agglomerative Hierarchical Clustering - Example

	P1	P2,P5	P3,P6	P4
P1	0			
P2,P5	0.23	0		
P3,P6	0.22	0.15	0	
P4	0.37	0.20	0.15	0

	P1	P2,P5	P3,P6	P4
P1	0			
P2,P5	0.23	0		
P3,P6	0.22	0.15	0	
P4	0.37	0.20	0.15	0

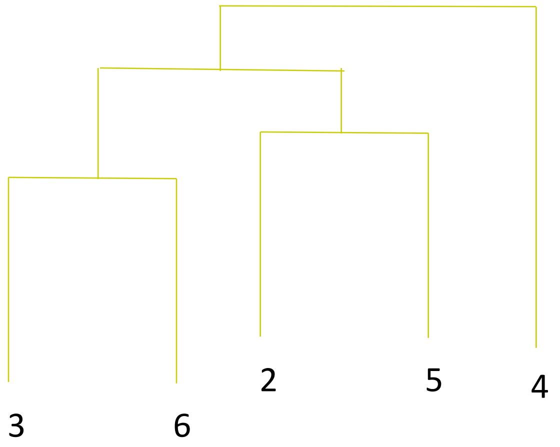


Agglomerative Hierarchical Clustering - Example

	P1	P2,P5	P3,P6	P4
P1	0			
P2,P5	0.23	0		
P3,P6	0.22	0.15	0	
P4	0.37	0.20	0.15	0

To Update the distance matrix MIN[dist(P2,P5),(P3,P6)),P1]
MIN (dist(P2,P5),P1),((P3,P6),P1)]
Min[(0.23,0.22)]
0.22

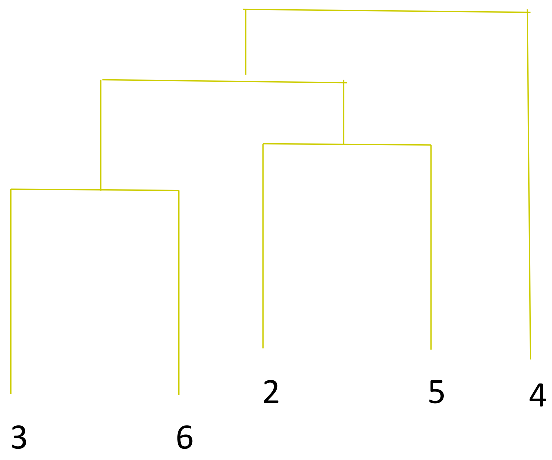
To Update the distance matrix MIN[dist(P2,P5),(P3,P6)),P4]
MIN (dist(P2,P5),P4),((P3,P6),P4)]
Min[(0.20,0.15)]
0.15



	P1	P2,P5,P3,P6	P4
P1	0		
P2,P5,P3,P6	0.22	0	
P4	0.37	0.15	0

Agglomerative Hierarchical Clustering - Example

	P1	P2,P5,P3,P6	P4
P1	0		
P2,P5,P3,P6	0.22	0	
P4	0.37	0.15	0

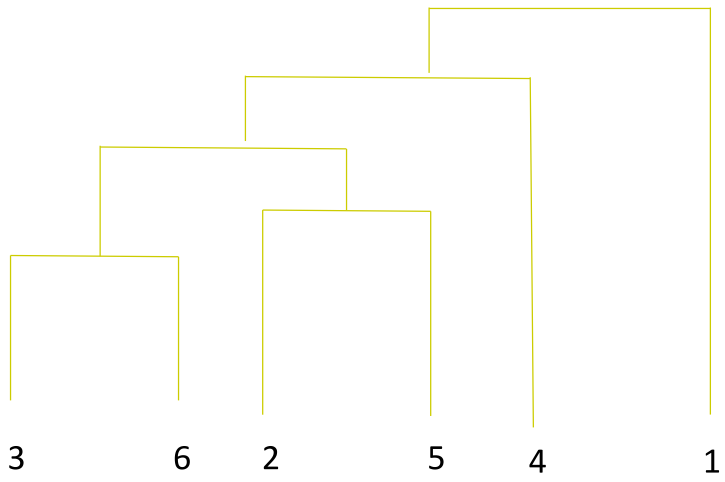


To Update the distance matrix MIN[dist(P2,P5,P3,P6),P4]
MIN (dist(P2,P5,P3,P6),P1),(P4,P1)]
Min[(0.22,0.37)]
0.22

	P1	P2,P5,P3,P6,P4
P1	0	
P2,P5,P3,P6,P4	0.22	0

Agglomerative Hierarchical Clustering - Example

	X	Y
P1	0.40	0.53
P2	0.22	0.38
P3	0.35	0.32
P4	0.26	0.19
P5	0.08	0.41
P6	0.45	0.30



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 with some cleverness

Hierarchical Clustering: Problems and Limitations

- Once a decision is made to combine two clusters, it cannot be undone

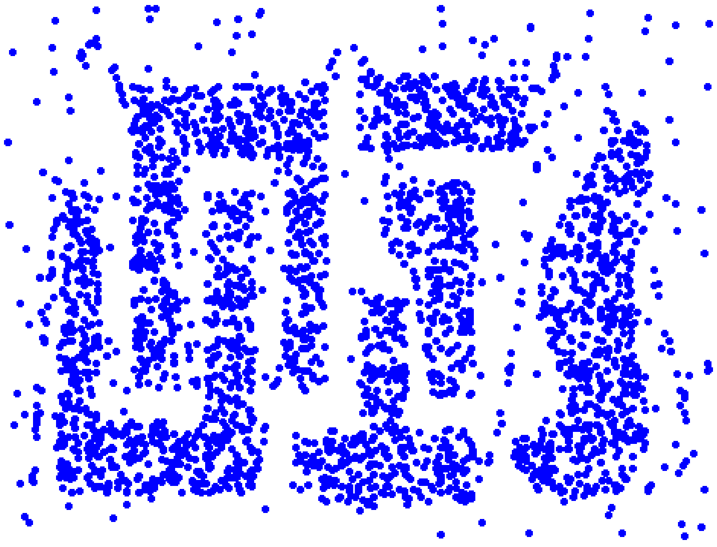
- No global objective function is directly minimized

- Different schemes have problems with one or more of the following:
  - Sensitivity to noise
  - Difficulty handling clusters of different sizes and non-globular shapes
  - Breaking large clusters

Density Based Clustering

- Clusters are regions of high density that are separated from one another by regions on low density.

Resilient to noise



DBSCAN

MinPts = 3

- DBSCAN is a density-based algorithm.
  - Density = number of points within a specified radius (Eps)
  - A point is a core point if it has at least a specified number of points (MinPts) within Eps
    - These are points that are at the interior of a cluster
    - Counts the point itself
  - A border point is not a core point, but is in the neighborhood of a core point

(MinPts - 1)

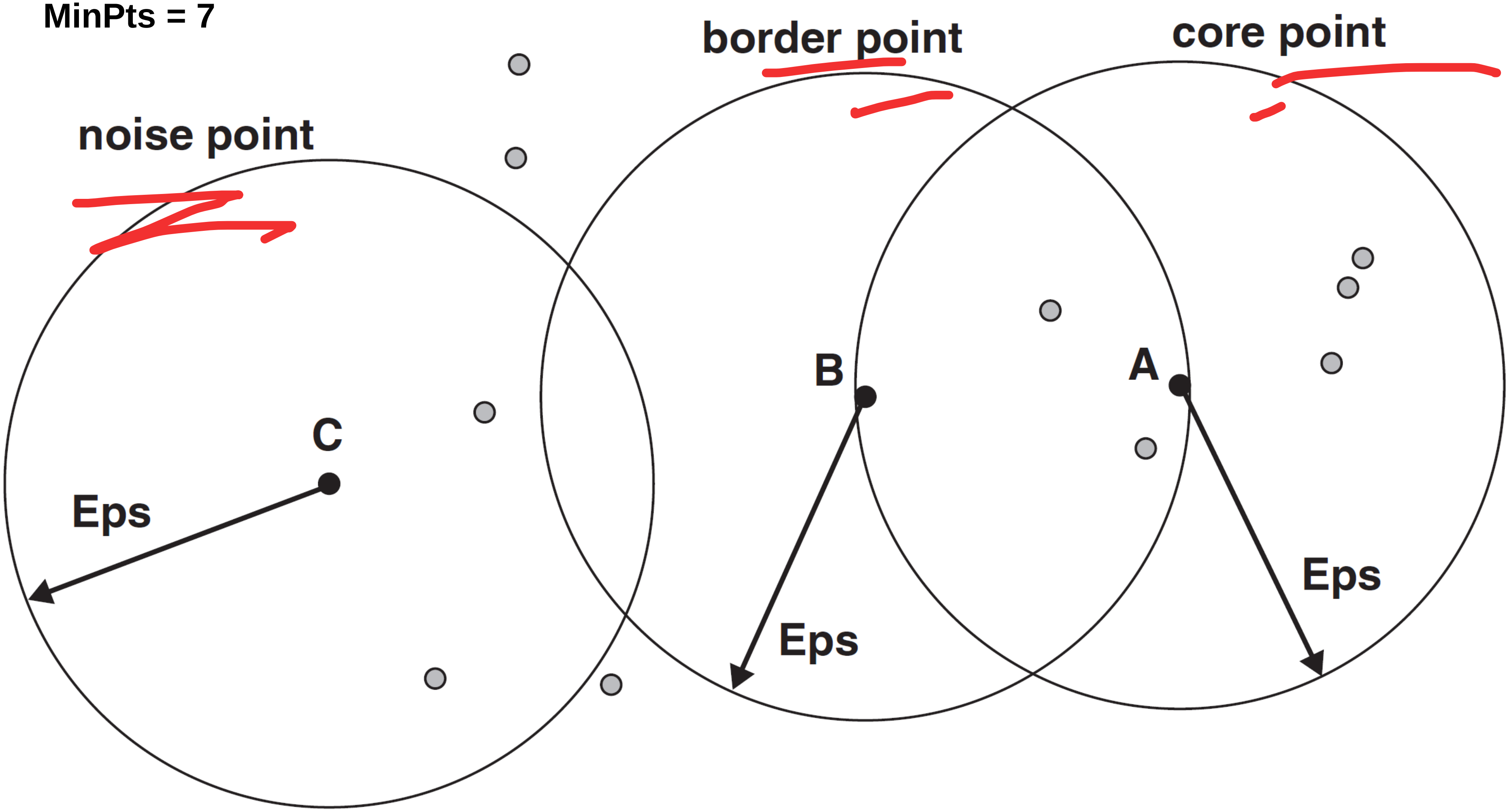
  - A noise point is any point that is not a core point or a border point

No. neighbour points

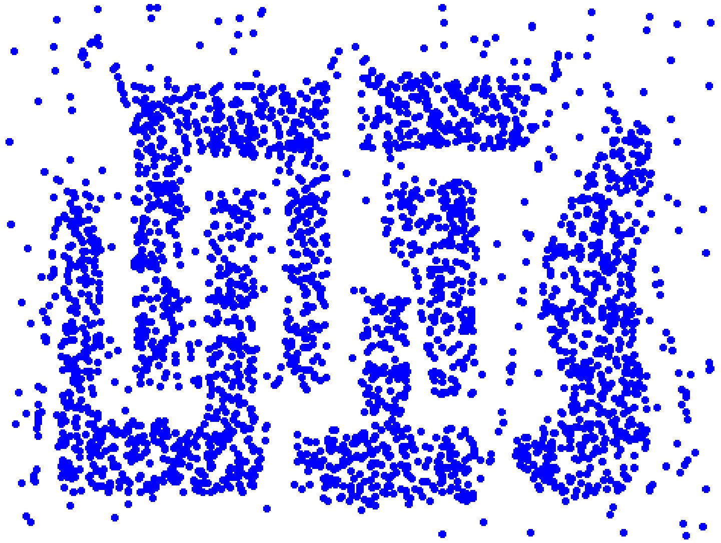
a1 = Eps

DBSCAN: Core, Border, and Noise Points

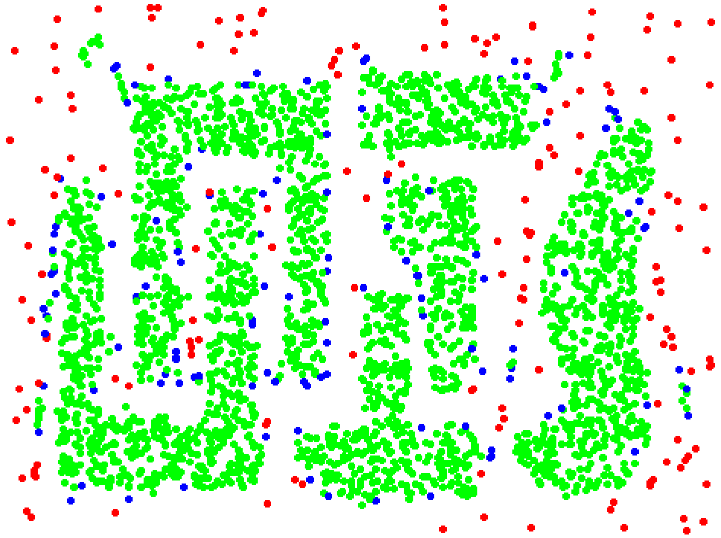
MinPts = 7



DBSCAN: Core, Border and Noise Points



Original Points



Point types: core, border and noise

Eps = 10, MinPts = 4

DBSCAN Algorithm

- Form clusters using core points, and assign border points to one of its neighboring clusters

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

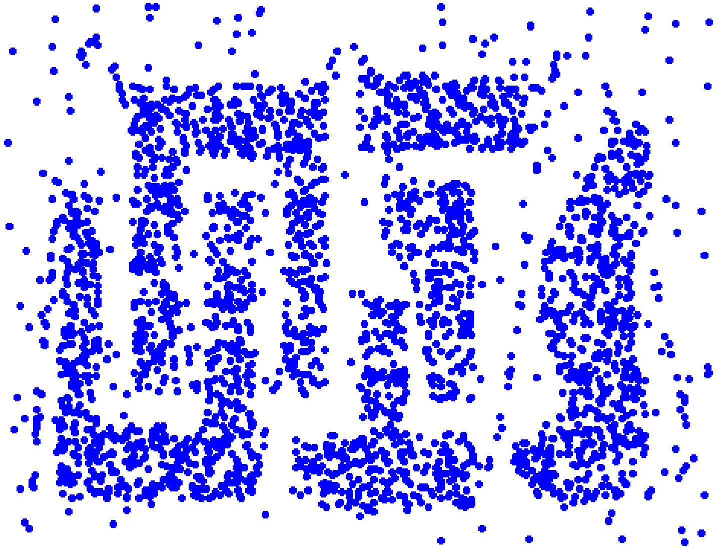
2: Eliminate noise points.

3: Put an edge between all core points within a distance *Eps* of each other.

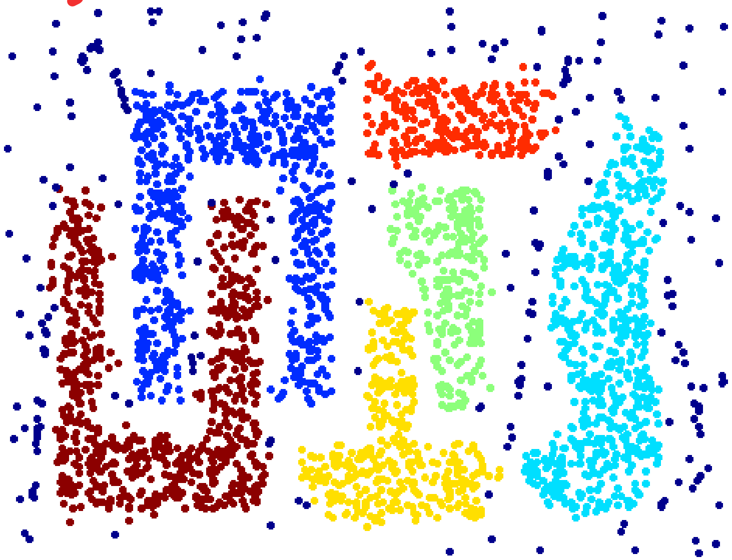
4: Make each group of connected core points into a separate cluster.

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

When DBSCAN Works Well



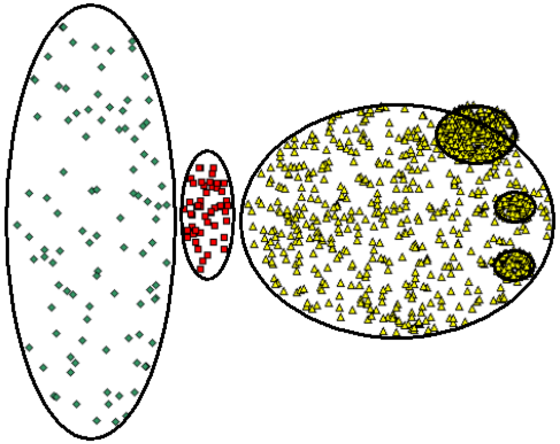
Original Points



Clusters (dark blue points indicate noise)

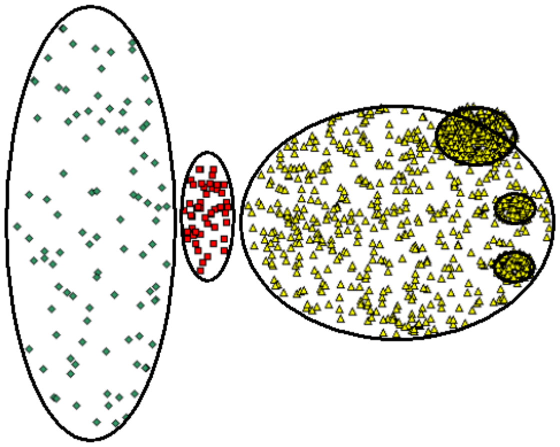
- ~~Can handle clusters~~ of different shapes and sizes
- Resista~~nt to noise~~

When DBSCAN Does NOT Work Well

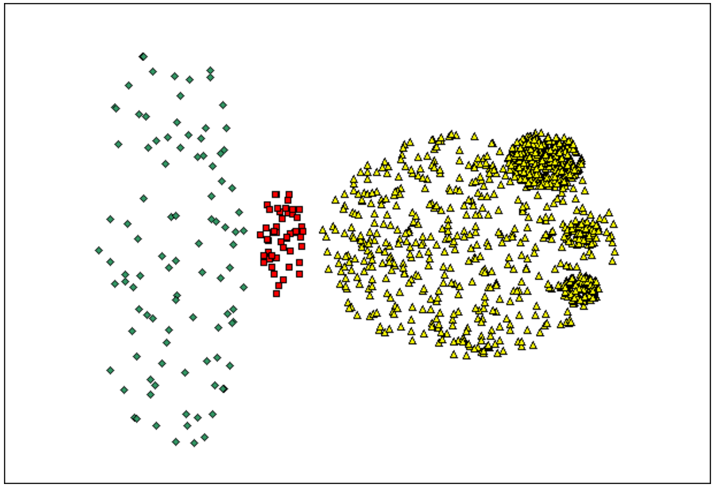


Original Points

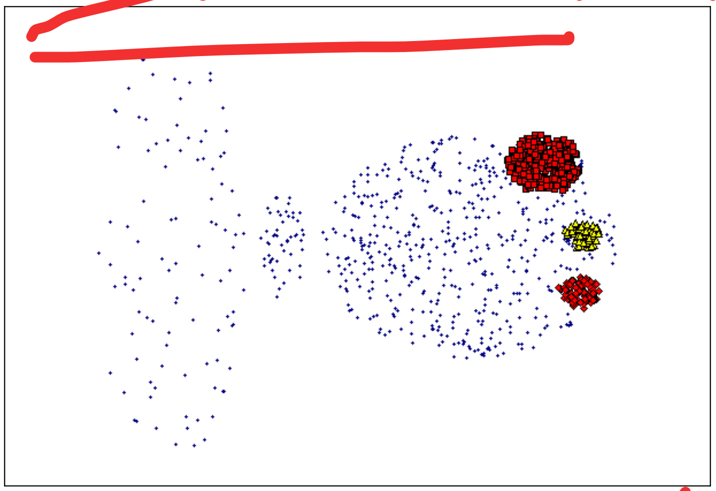
When DBSCAN Does NOT Work Well



Original Points



(MinPts=4, Eps=9.92).

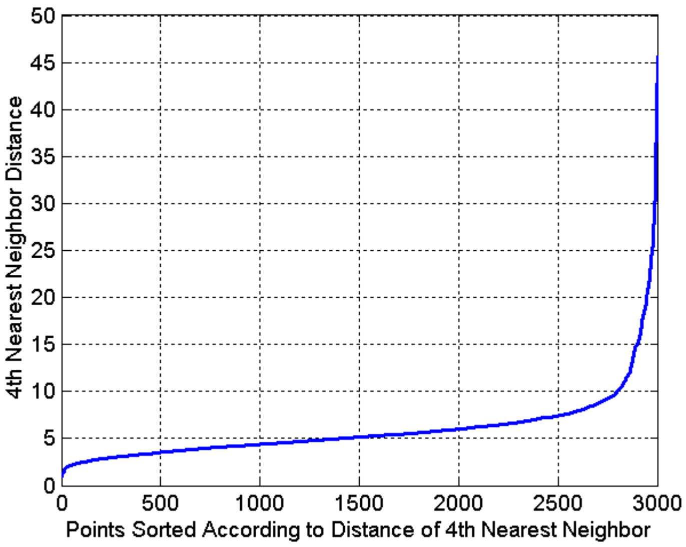


(MinPts=4, Eps=9.75)

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

DBSCAN: Determining EPS and MinPts

- Idea is that for points in a cluster, their kth nearest neighbors are at close distance
- Noise points have the kth nearest neighbor at farther distance
- So, plot sorted distance of every point to its kth nearest neighbor

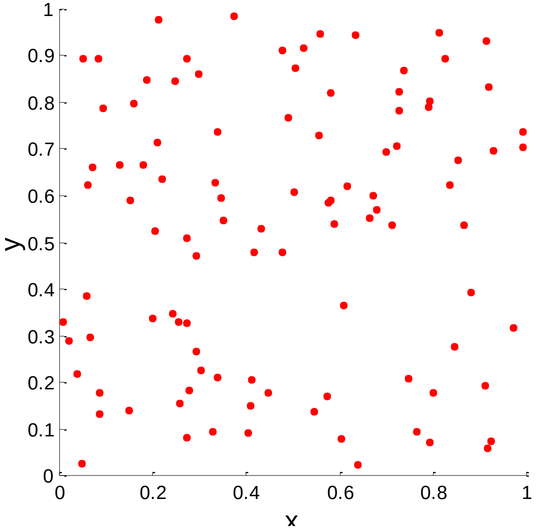


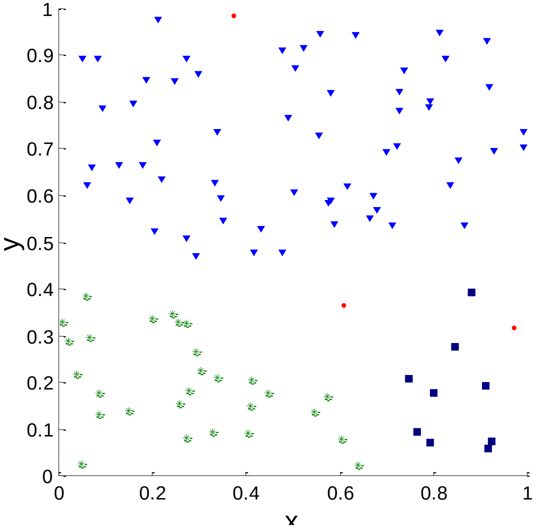
Cluster Validity

- 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"!
  - In practice the clusters we find are defined by the clustering algorithm
- Then why do we want to evaluate them?
  - To avoid finding patterns in noise
  - To compare clustering algorithms
  - To compare two sets of clusters
  - To compare two clusters

Clusters found in Random Data

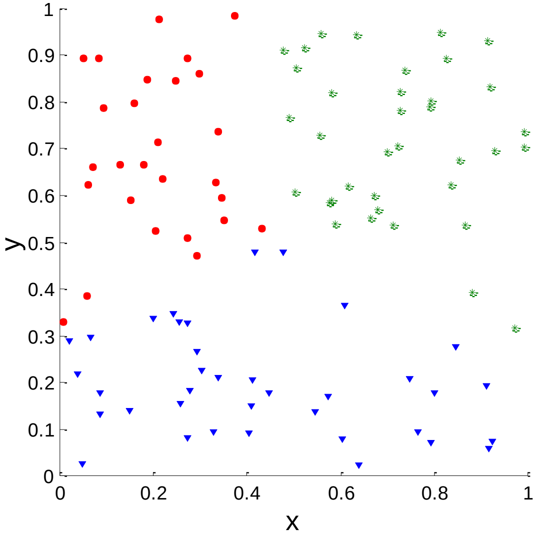
Random Points

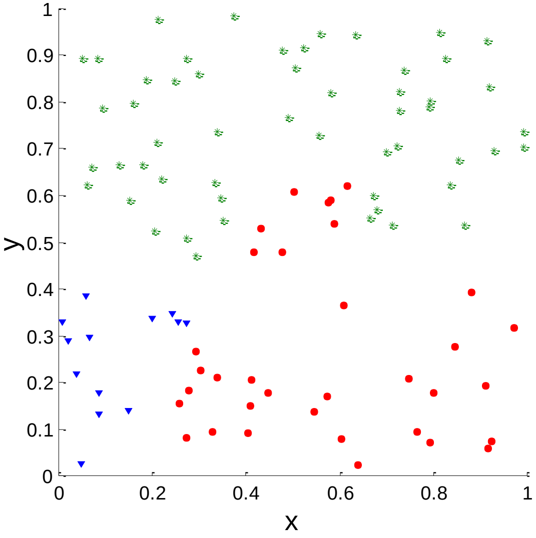




DBSCAN

K-means





Complete Link

Measures of Cluster Validity

- Numerical measures that are applied to judge various aspects of cluster validity, are classified into the following two types.
  - Supervised: Used to measure the extent to which cluster labels match externally supplied class labels.
    - Entropy
    - Often called *external indices* because they use information external to the data
  - Unsupervised: Used to measure the goodness of a clustering structure *without* respect to external information.
    - Sum of Squared Error (SSE)
    - Often called *internal indices* because they only use information in the data

- You can use supervised or unsupervised measures to compare clusters or clusterings

Unsupervised Measures: Cohesion and Separation

- Cluster Cohesion: Measures how closely related are objects in a cluster
  - Example: SSE
- Cluster Separation: Measure how distinct or well-separated a cluster is from other clusters
- Example: Squared Error
  - Cohesion is measured by the within cluster sum of squares (SSE)

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

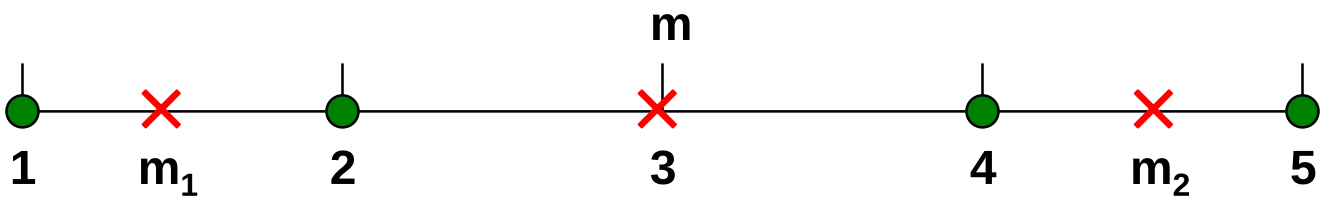
  - Separation is measured by the between cluster sum of squares

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

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

Unsupervised Measures: Cohesion and Separation

- Example: SSE
  - SSB + SSE = constant



K=1 cluster:

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

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

$$Total = 10 + 0 = 10$$

K=2 clusters:

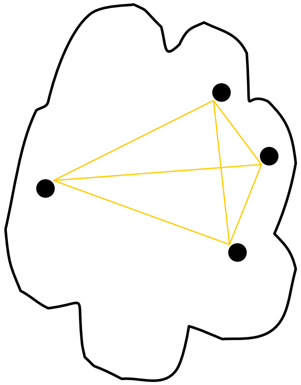
$$SSE = (1-1.5)^2 + (2-1.5)^2 + (4-4.5)^2 + (5-4.5)^2 = 1$$

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

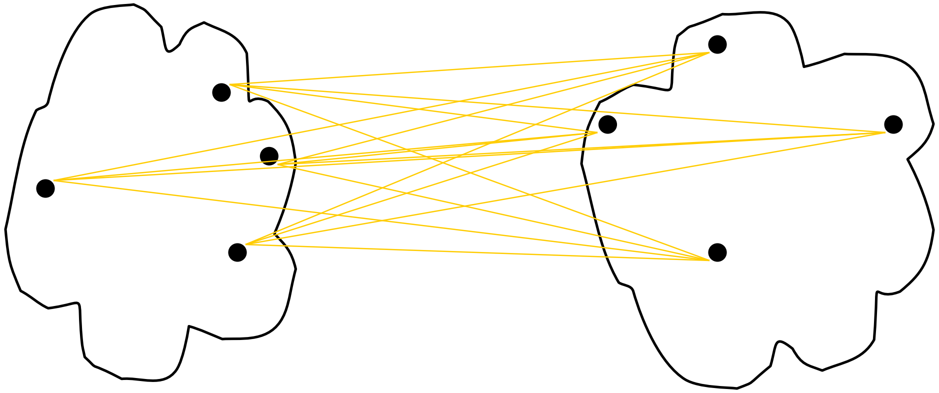
$$Total = 1 + 9 = 10$$

Unsupervised Measures: Cohesion and Separation

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



cohesion



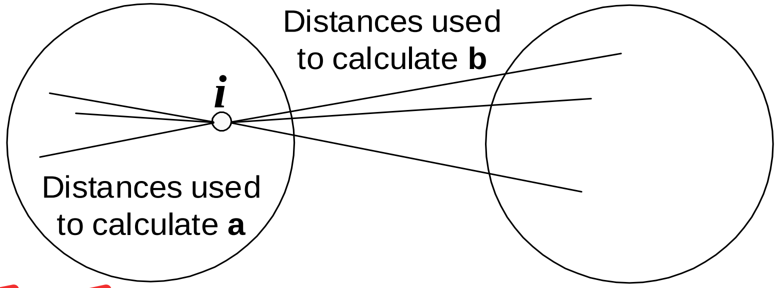
separation

Unsupervised Measures: Silhouette Coefficient

- Silhouette coefficient combines ideas of both cohesion and separation, but for individual points, as well as clusters and clusterings
- For an individual point, ***i***
  - Calculate ***a*** = average distance of ***i*** to the points in its cluster
  - Calculate ***b*** = min (average distance of ***i*** to points in another cluster)
  - The silhouette coefficient for a point is then given by

$$s = (b - a) / \max(a,b)$$

  - Value can vary between -1 and 1
  - Typically ranges between 0 and 1.
  - The closer to 1 the better.


Distances used to calculate **b**
i
Distances used to calculate **a**

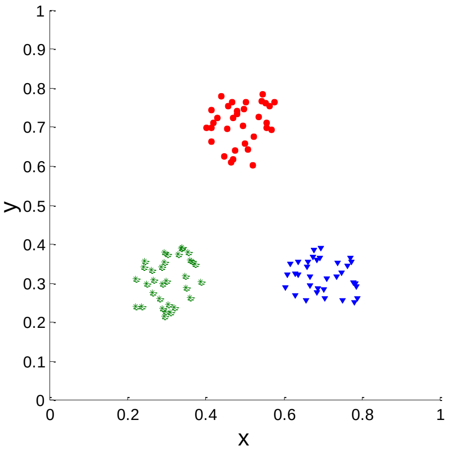
- Can calculate the average silhouette coefficient for a cluster or a clustering

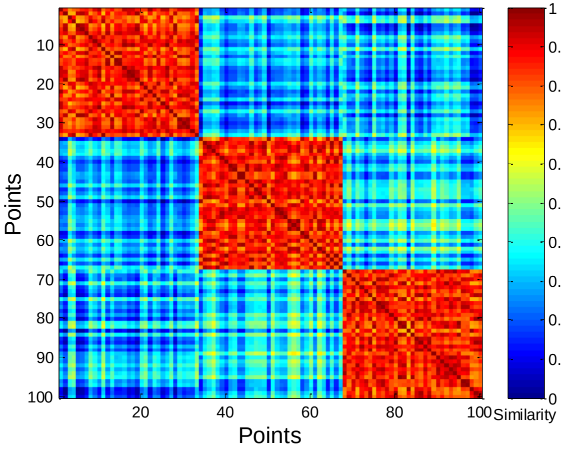
Measuring Cluster Validity Via Correlation

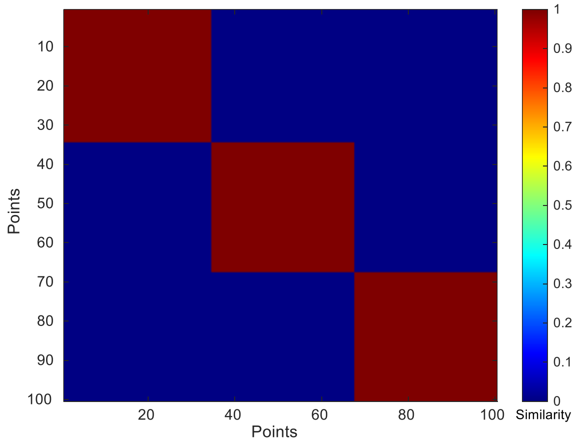
- Two matrices
  - Proximity Matrix
  - Ideal Similarity Matrix
    - One row and one column for each data point
    - An entry is 1 if the associated pair of points belong to the same cluster
    - An entry is 0 if the associated pair of points belongs to different clusters
- Compute the correlation between the two matrices
  - Since the matrices are symmetric, only the correlation between n(n-1) / 2 entries needs to be calculated.
- High magnitude of correlation indicates that points that belong to the same cluster are close to each other.
  - Correlation may be positive or negative depending on whether the similarity matrix is a similarity or dissimilarity matrix
- Not a good measure for some density or contiguity based clusters.

Measuring Cluster Validity Via Correlation

- Correlation of ideal similarity and proximity matrices for the K-means clusterings of the following well-clustered data set.



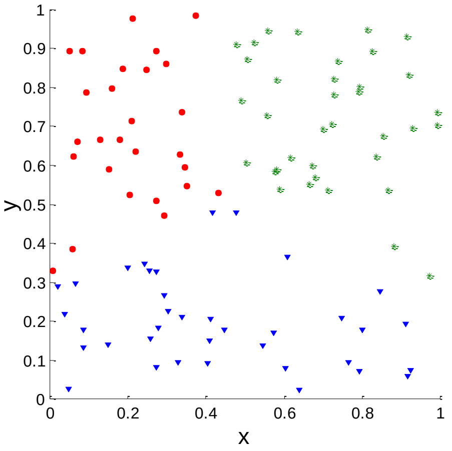


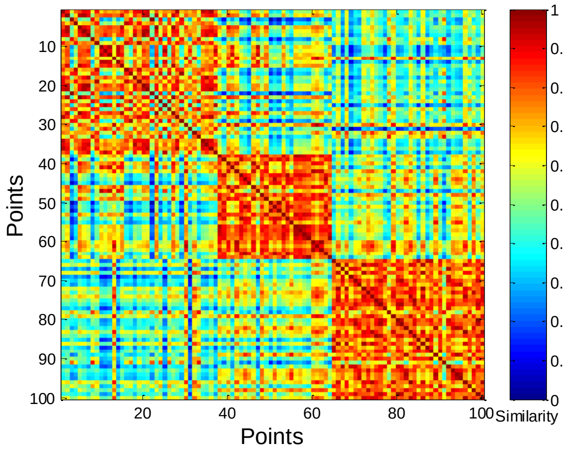


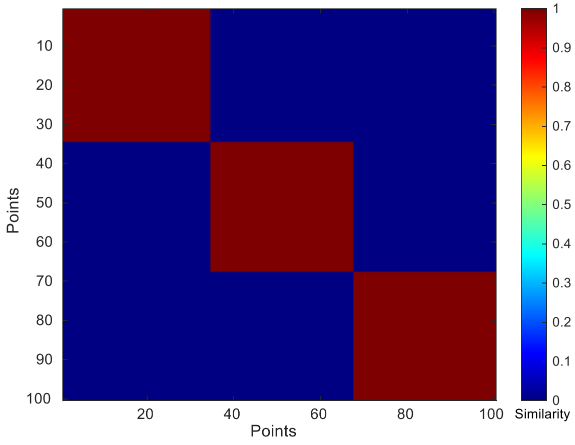
Corr = 0.9235

Measuring Cluster Validity Via Correlation

- Correlation of ideal similarity and proximity matrices for the K-means clusterings of the following random data set.





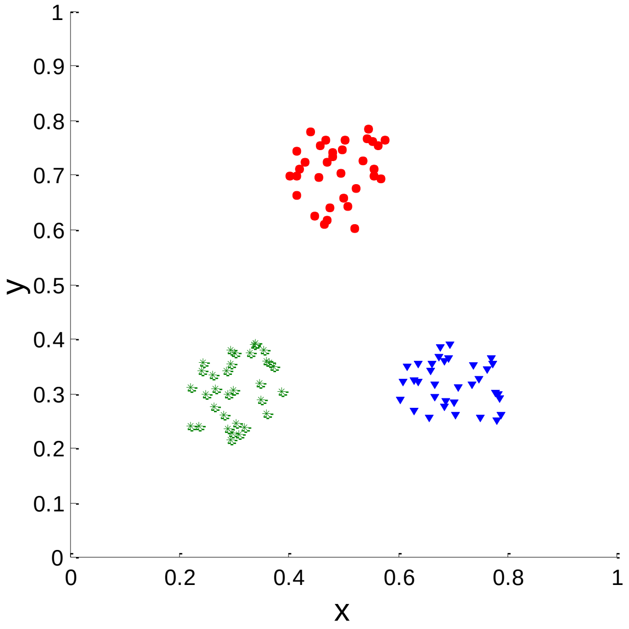


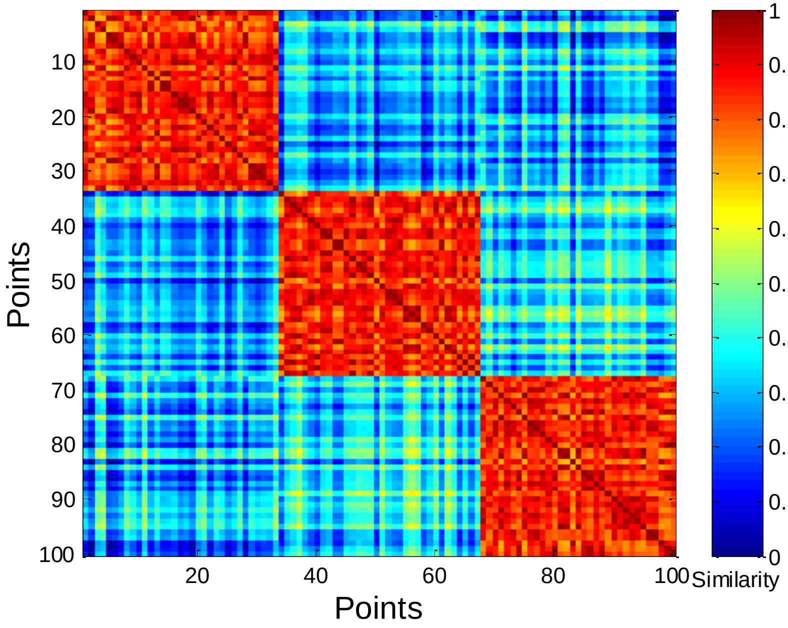
K-means

Corr = 0.5810

Judging a Clustering Visually by its Similarity Matrix

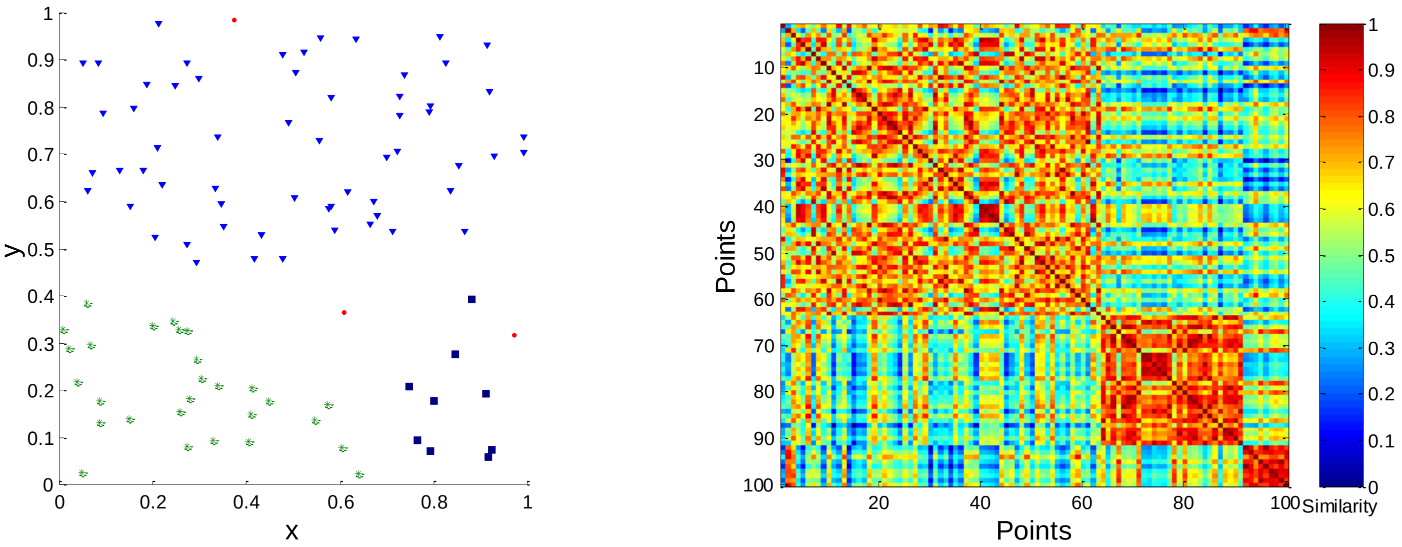
- Order the similarity matrix with respect to cluster labels and inspect visually.





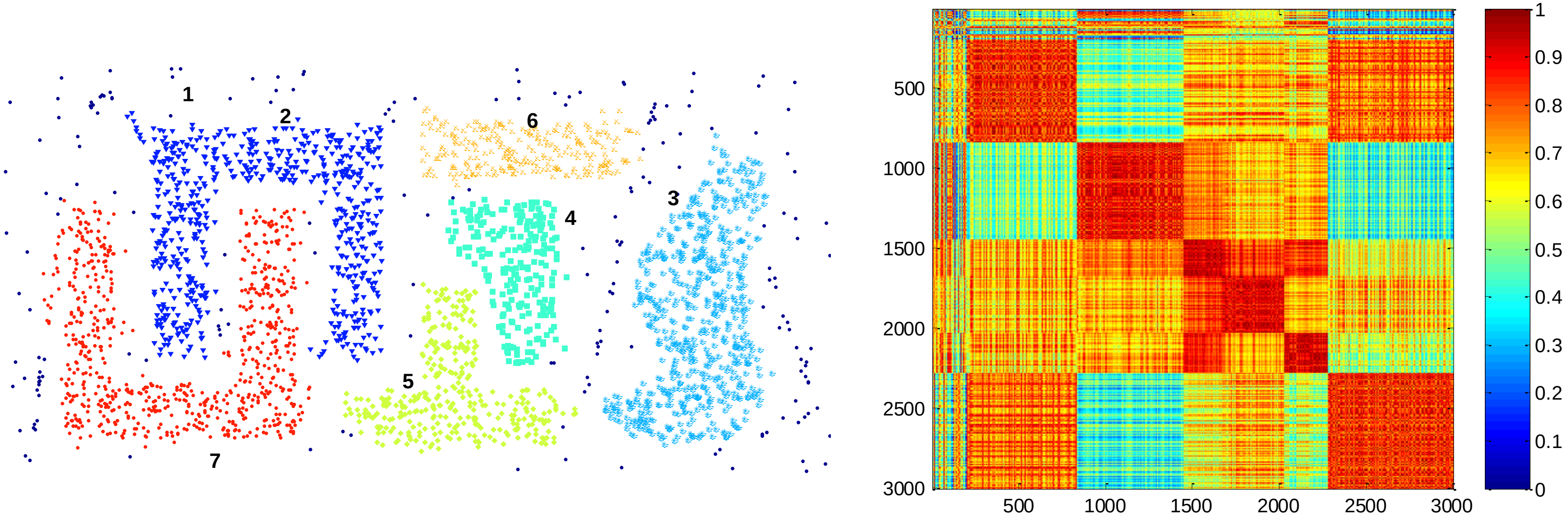
Judging a Clustering Visually by its Similarity Matrix

- Clusters in random data are not so crisp



DBSCAN

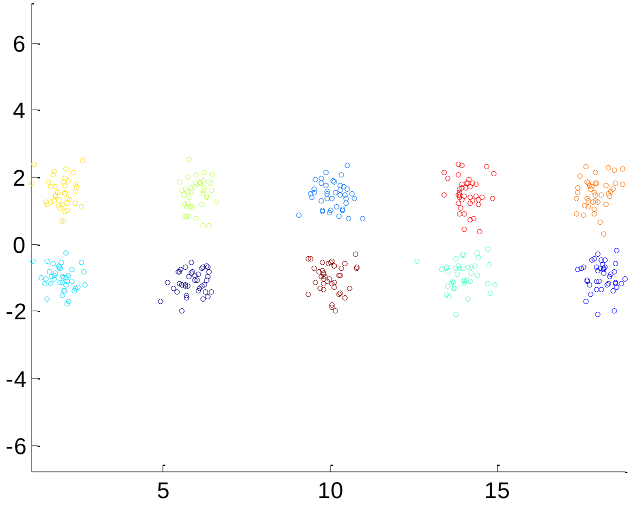
Judging a Clustering Visually by its Similarity Matrix

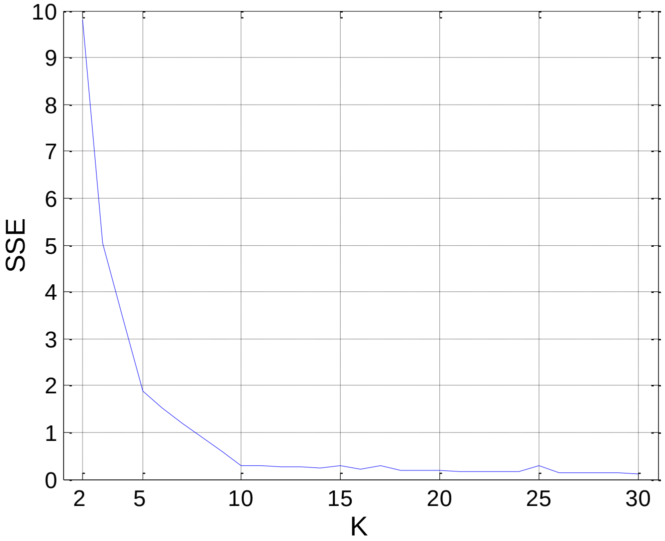


DBSCAN

Determining the Correct Number of Clusters

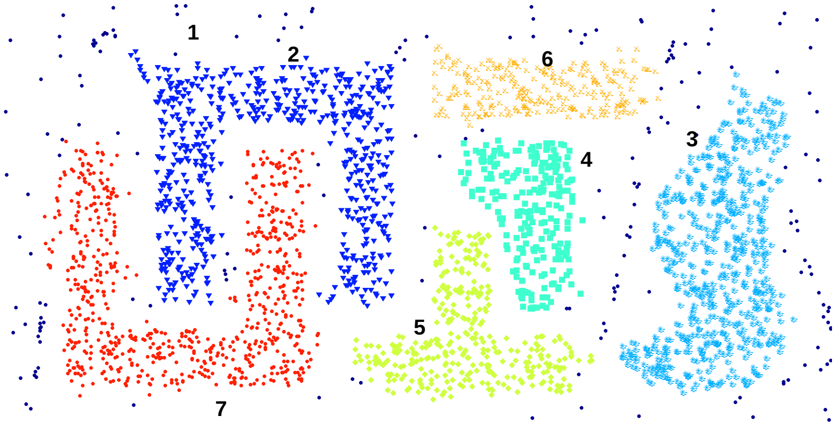
- SSE is good for comparing two clusterings or two clusters
- SSE can also be used to estimate the number of clusters

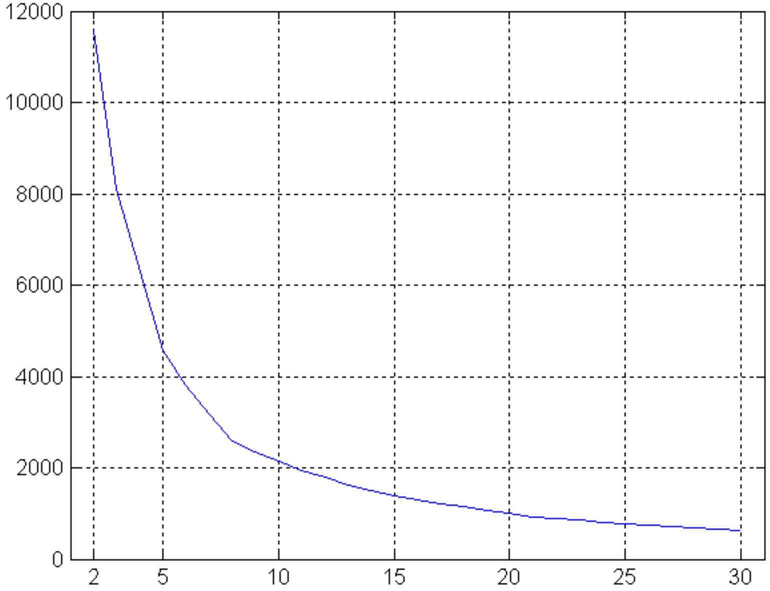




Determining the Correct Number of Clusters

- SSE curve for a more complicated data set





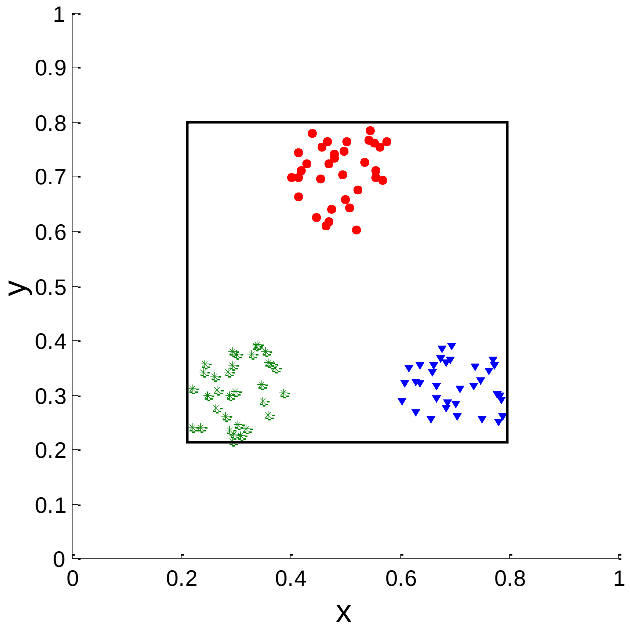
SSE of clusters found using K-means

Assessing the Significance of Cluster Validity Measures

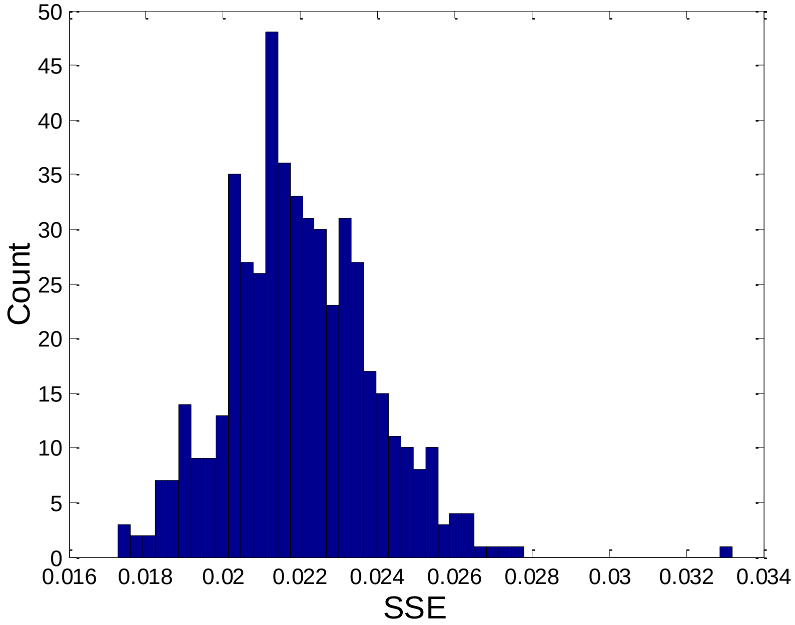
- Need a framework to interpret any measure.
  - For example, if our measure of evaluation has the value, 10, is that good, fair, or poor?
- Statistics provide a framework for cluster validity
  - The more “atypical” a clustering result is, the more likely it represents valid structure in the data
  - Compare the value of an index obtained from the given data with those resulting from random data.
    - If the value of the index is unlikely, then the cluster results are valid

Statistical Framework for SSE

- Example
  - Compare SSE of three cohesive clusters against three clusters in random data



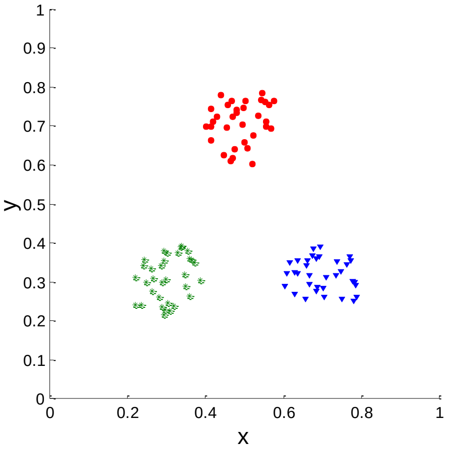
SSE = 0.005



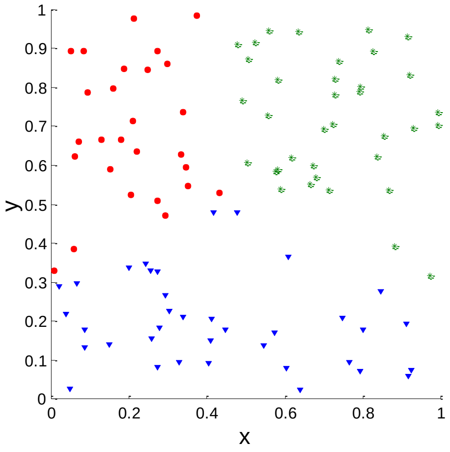
Histogram shows SSE of three clusters in 500 sets of random data points of size 100 distributed over the range 0.2 – 0.8 for x and y values

Statistical Framework for Correlation

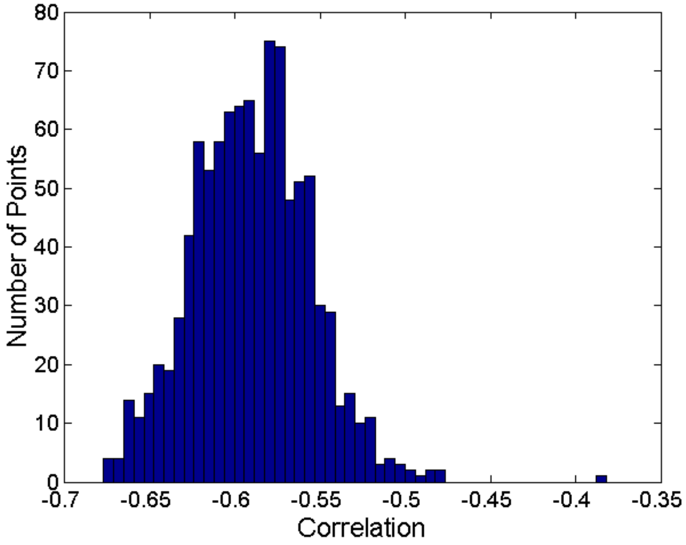
- Correlation of ideal similarity and proximity matrices for the K-means clusterings of the following two data sets.



Corr = -0.9235



Corr = -0.5810



Histogram of correlation for 500 random data sets of size 100 with *x* and *y* values of points between 0.2 and 0.8.

Correlation is negative because it is calculated between a distance matrix and the ideal similarity matrix. Higher magnitude is better.

Final Comment on Cluster Validity

“The validation of clustering structures is the most difficult and frustrating part of cluster analysis.

Without a strong effort in this direction, cluster analysis will remain a black art accessible only to those true believers who have experience and great courage.”

***Algorithms for Clustering Data*, Jain and Dubes**

- H. Xiong and Z. Li. *Clustering Validation Measures*. In C. C. Aggarwal and C. K. Reddy, editors, Data Clustering: Algorithms and Applications, pages 571–605. Chapman & Hall/CRC, 2013.