Managing Big Data

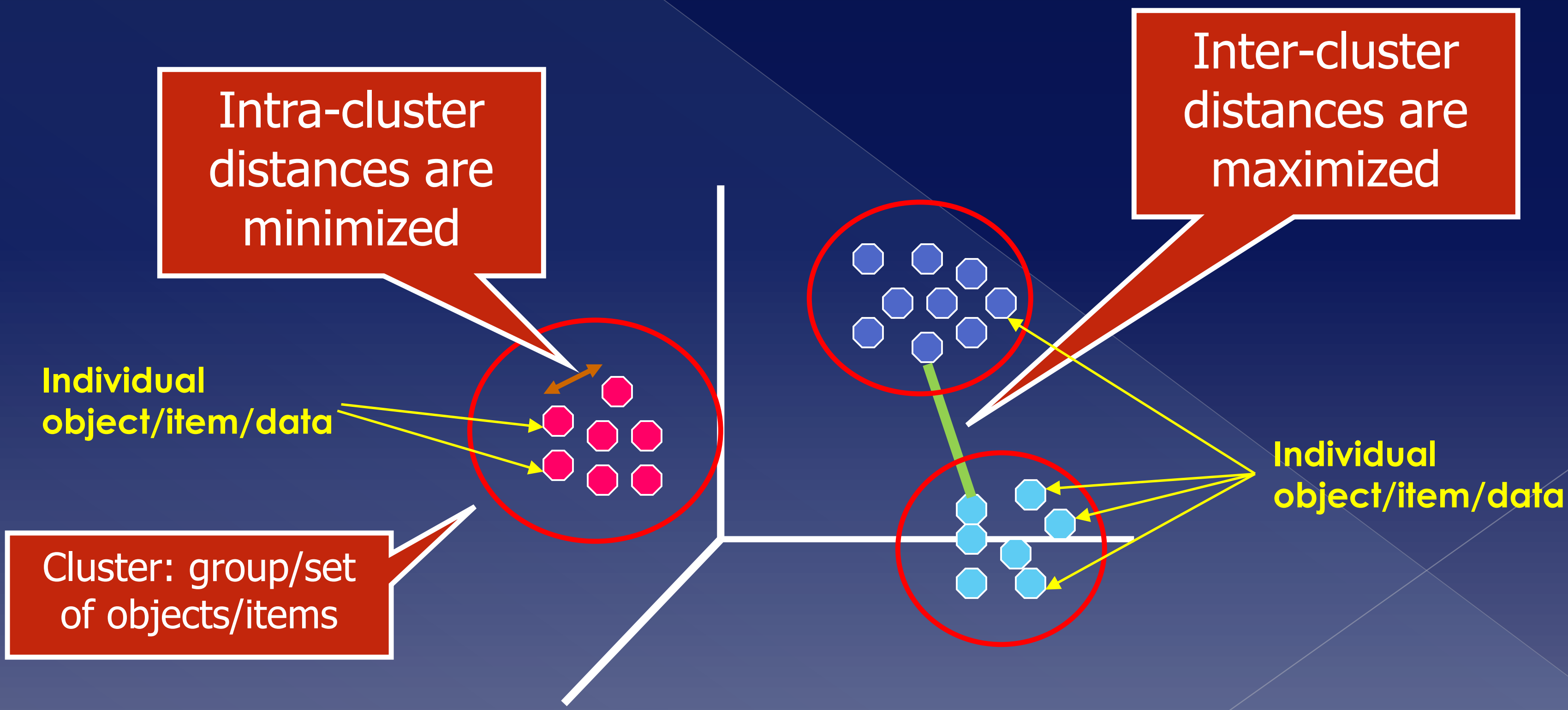
Cluster Analysis: Basic Concepts and Algorithms

Manolis Tzagarakis
Assistant Professor
Department of Economics
University of Patras

tzagara@upatras.gr
2610 969845
google:tzagara
Facebook: tzagara
SkypeID: tzagara
QuakeLive: DeusEx

Cluster analysis

- What is cluster analysis?
  - **Finding groups/sets of objects** such that the **objects in a group/set will be <u>similar (or related)</u> to one another** and **<u>different from (or unrelated to)</u> the objects in other groups**



Cluster analysis

- What is cluster analysis?
  - **Clustering** is an (somehow) **endemic characteristic** of humans
    - E.g. even children can make groups out of photos (buildings, cars, humans, plants etc)
  - In clustering, **discovered groups (also called clusters)** are **potential categories** and can be assigned class labels
  - The basic approach is to **create such groupings solely based on the values of attributes of the data**
    - Assuming **data represented as (a_1, a_2, a_3, … a_n)**

Cluster analysis

- What is cluster analysis?
  - The idea is that **items/data/objects in the same group** share **<u>some</u> conceptual similarity**
    - **Hence, can be (somehow) classified**

Cluster analysis

- Why use cluster analysis (aka clustering)
  - **Understanding**
    - E.g. Group related documents for browsing
    - E.g. group genes and proteins that have similar functionality
    - E.g. group stocks with similar price fluctuations
  - **Summarization**
    - Reduce the size of large data sets (a preprocessing step)
  - **Data Exploration**
    - Get some insights into distribution of data
    - Understand patterns in data

Cluster analysis

- Early applications: **John Snow (father of Epidemiology), London, 1854**

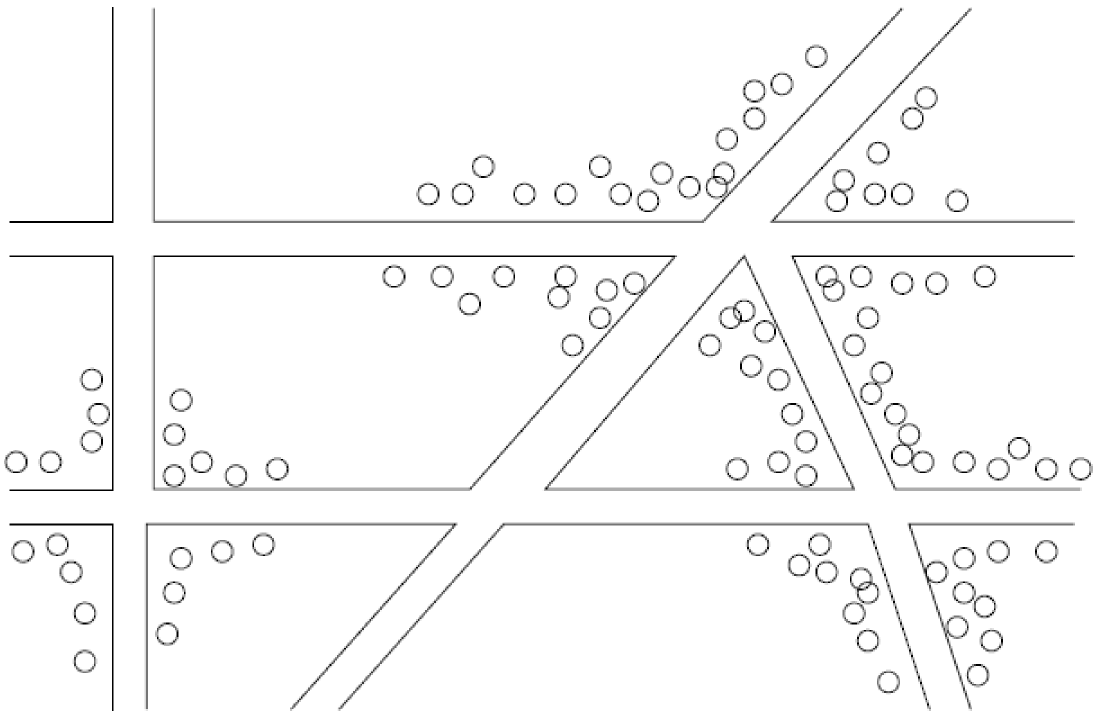


Figure 1.1: Plotting cholera cases on a map of London

Tracing Cholera cases in Soho, London in 1854.

Inspired fundamental changes in the water and waste systems of London

Cluster analysis

- Application domains (where it's useful)
  - **Marketing**: finding groups of customers with similar behavior given a large database of customer data containing their properties and past buying records
  - **Insurance:** identifying groups of motor insurance policy holders with a high average claim cost; identifying frauds
  - **City-planning:** identifying groups of houses according to their house type, value and geographical location
  - **Earthquake studies:** clustering observed earthquake epicenters to identify dangerous zones
  - **Tax evasion:** case selection of taxpayers with high probability of cheating
  - **Recommendation Systems**: providing personalized services to users based on the preferences of similar users

Clustering vs Classification?

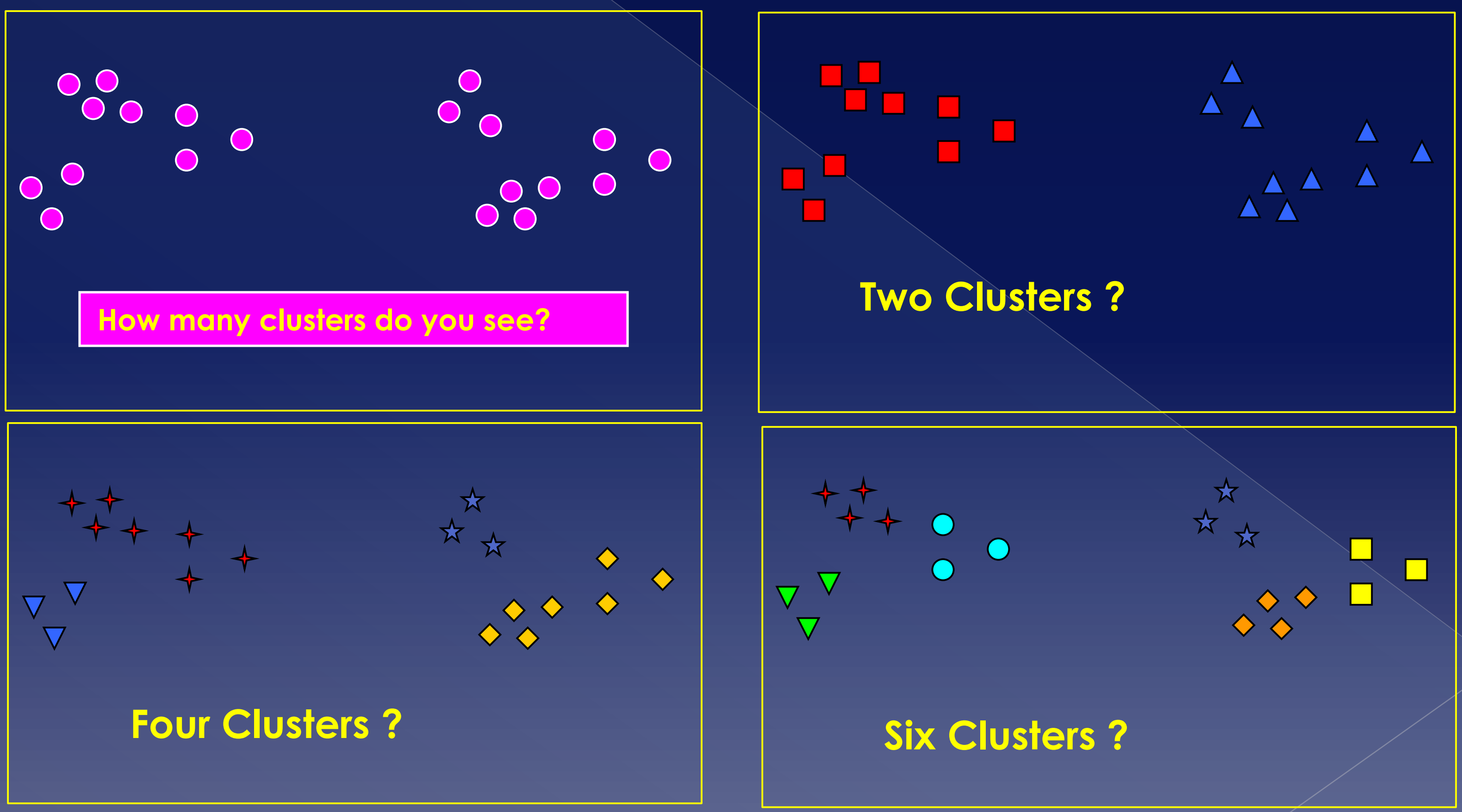
- **Classification**
  - **Classification** has an **existing labeled** (i.e. class known) set as training set. Grouping structure is learned => **Supervised learning**
    - Supervised = existing classes distinct and already known
  - **Classification** tries to **predict the class** of (unknown) data based on the model
- **Clustering**
  - **Clustering**, classes of data items in the beginning **unknown** => **Unsupervised learning**
    - Unsupervised = classes unknown in the beginning
  - **Clustering** attempts to **group items/objects into "natural" classes**, when no classes are available
  - **Clustering** automatically decides on the grouping structure i.e. **automatically tries to find the classes**

What is **not** cluster analysis

- **Simple segmentation**
  - E.g. dividing students into different registration groups alphabetically, by last name
- **Results of a query**
  - Groupings are a result of an external specification
    - i.e. not based on attributes of data
- **Graph partitioning**
  - Some mutual relevance and synergy, but areas are not identical

Cluster analysis

- **Identifying clusters** (i.e. groups of objects) **not always easy**



Depends on "resolution" !

Cluster analysis

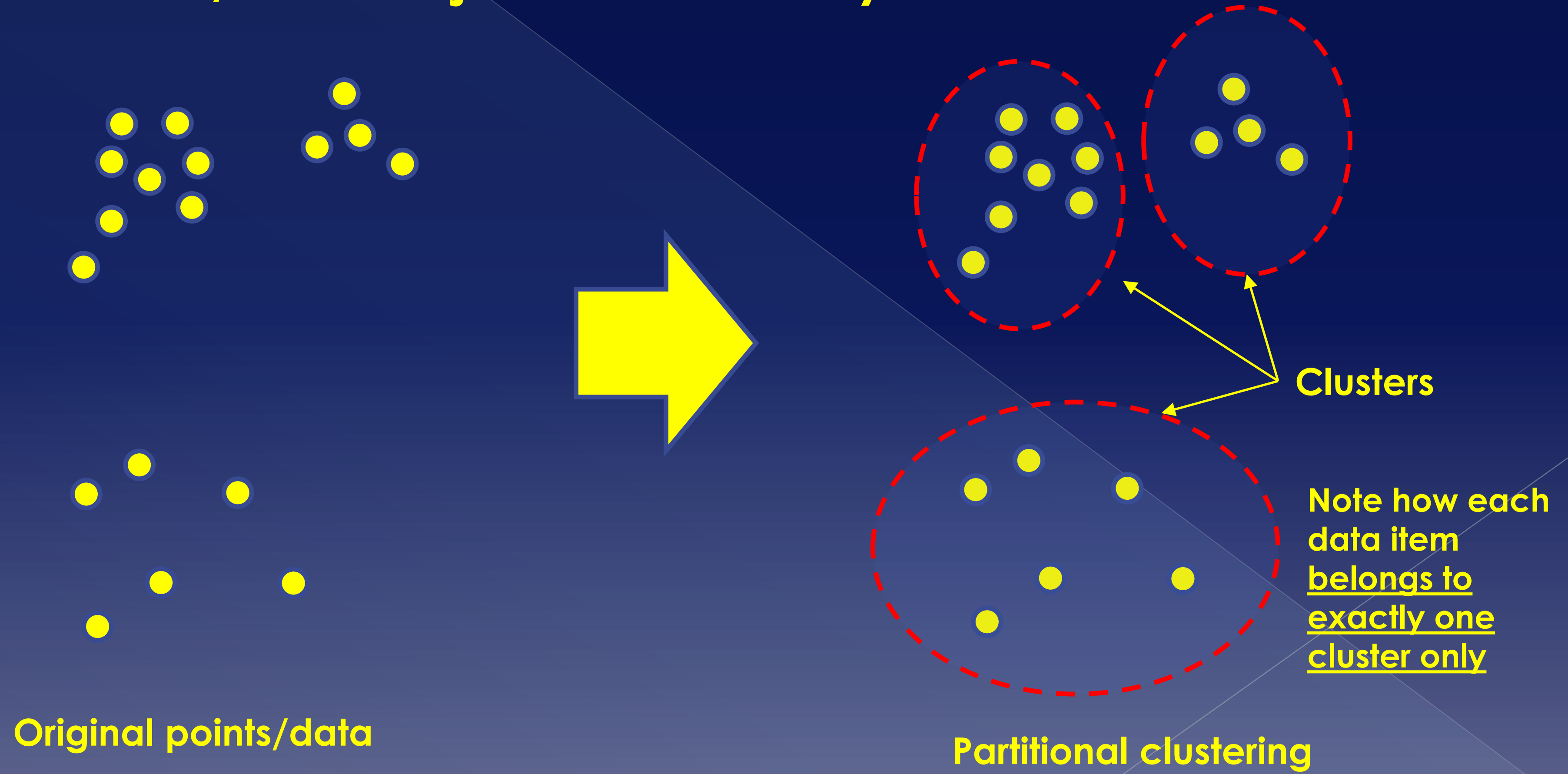
- Clusters are in general fuzzy (i.e. with not clear, well defined boundaries)
  - **Properly defining clusters** depends on the **nature of the problem** and **the desired outcome (what the goal of our clustering is)**

Cluster analysis

- A **clustering** is a **set of clusters (groups)**
- Different **types of clustering, based on the kind of clustering (at large scale) the algorithms produce:**
  - **Partitional clustering**
  - **Hierarchical clustering**

Cluster analysis

- **Partitional clustering**
  - A division of **items/data objects into non-overlapping subsets (clusters)** such that **each item/data object is in exactly one subset**

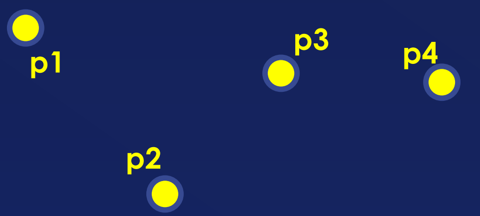


Cluster analysis

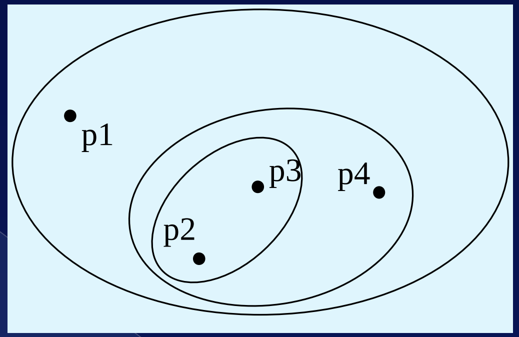
- **Hierarchical clustering**
  - Creates a **set of nested clusters** organized as a **hierarchical tree**
    - **Tree visualized as dendrogram**

Cluster analysis

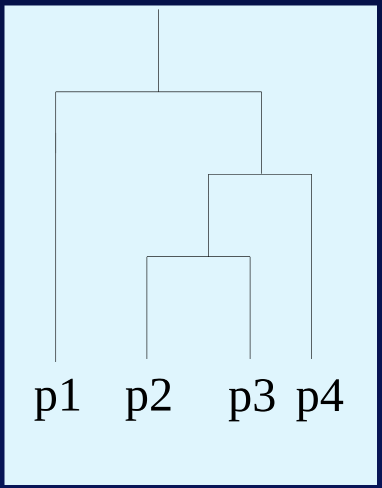
- Examples of Hierarchical Clustering



Original points/data

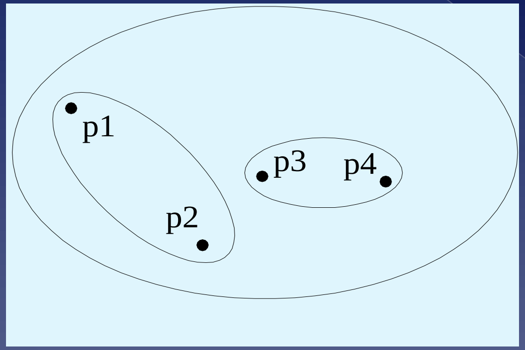


Traditional Hierarchical Clustering

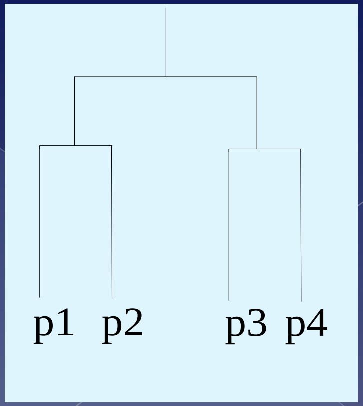


Traditional Dendrogram

Original points/data



Non-traditional Hierarchical Clustering



Non-traditional Dendrogram

Cluster analysis

- Other types of clustering
  - **Exclusive versus non-exclusive**
    - In non-exclusive clustering, points may belong to multiple clusters
    - Can represent multiple classes or 'border' points
  - **Fuzzy vs non-fuzzy**
    - In fuzzy clustering, a point belongs to every cluster with some weight between 0 and 1
    - Weights must sum to 1
    - Probabilistic clustering has similar characteristics

Cluster analysis

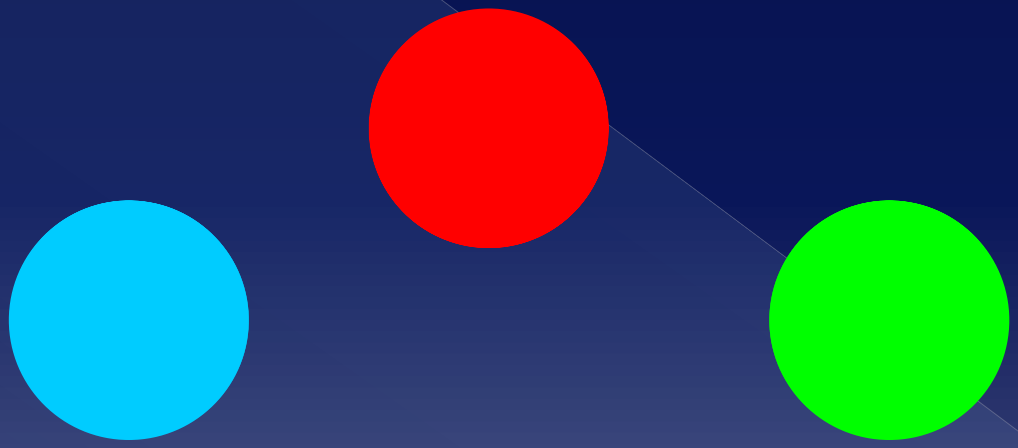
- Other types of clustering (cont.)
  - **Partial versus complete**
    - In some cases, we only want to cluster some (subset) of the data
    - Some data into clusters; others not
      - Some data maybe noise, outliers etc
  - **Heterogeneous versus homogeneous**
    - Cluster of widely different sizes, shapes, and densities

Cluster analysis

- We talked about **types of clustering**. There are also **types of clusters, based on what kind of clusters the algorithms look for**:
  - **Well separated**
  - **Center-based**
  - **Contiguous (Nearest neighbor/Transitive)**
  - **Density-based**
  - **Property or Conceptual**
  - **Described by an Objective Function**

Cluster analysis

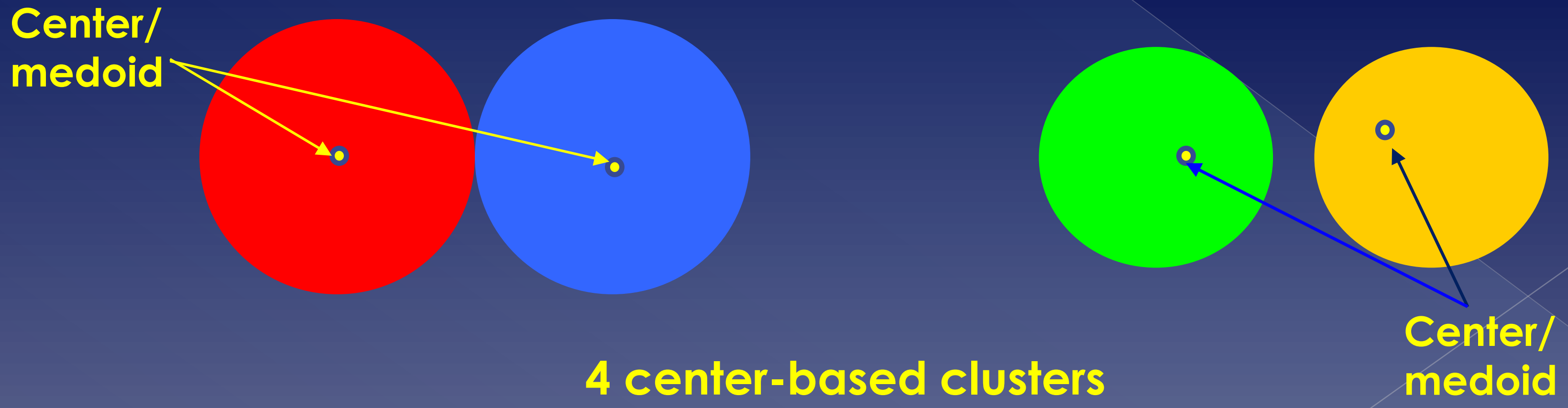
- **Well separated clusters**
  - A cluster is a set of points such that any point in a cluster is closer (or more similar) to every other point in the cluster than to any point not in the cluster



3 well-separated clusters

Cluster analysis

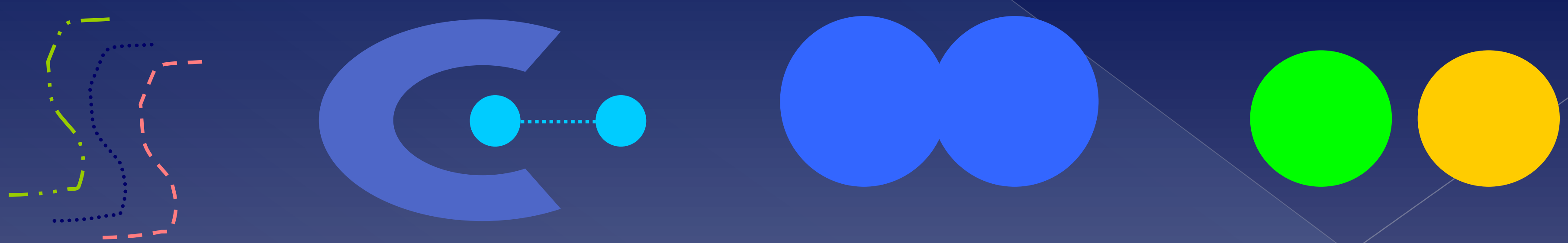
- **Center-based clusters**
  - A cluster is a set of objects such that an object in a cluster is **closer (more similar) to the <u>"center"</u> of a cluster**, than to the center of any other cluster
  - The <u>**center**</u> of a **cluster is often a centroid, the average of all the points in the cluster**,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



4 center-based clusters

Cluster analysis

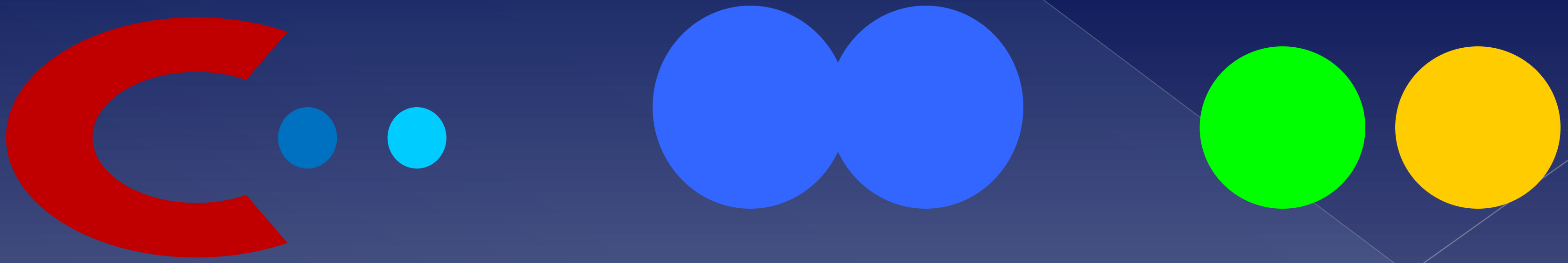
- **Contiguous (Nearest neighbor/Transitive)**
  - A cluster is a set of points such **that a point in a cluster is closer (or more similar) to <u>one</u> or more other points** in the cluster than to any point not in the cluster.



8 contiguous clusters

Cluster analysis

- **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

Cluster analysis

- **Property or Conceptual**
  - Finds **clusters that share some common property** or represent **a particular concept**

Cluster analysis

- **Clusters Defined by an Objective Function**
  - Finds clusters that **minimize or maximize an objective function.**
  - **How?** 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.

Cluster analysis

- **Objective Function:** Map the clustering problem to a different domain and solve a related problem in that domain
  - **Proximity matrix** defines a weighted graph, where the nodes are the points being clustered, and the weighted edges represent the proximities between points
  - **Clustering** is **equivalent to breaking the graph** into connected components, one for each cluster.
  - Want to minimize the edge weight between clusters and maximize the edge weight within clusters

Cluster analysis

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

Cluster analysis

- Overview: **Basic ingredients** needed for cluster analysis
  - **Objects/Items/Data (of course)**
    - In the form of attribute/values: **(a_1, a_2, a_3,…a_n)**
    - Attributes can be of **any type**: **nominal, ordinal, interval, ratio**
  - **Distance measure**
    - To measure similarity/distance and decide when two items are close together
  - **Clustering algorithm**
    - Attempts to minimize distances of items within groups/clusters and maximize distances between groups/clusters
  - **Preprocessing**
    - Scaling: Normalize/Standardize attributes (e.g. min-max, z-score) to avoid influence of some attributes on the distance measure (similar to the issues in k-NN classification)

Cluster analysis

- **Distance measure**
  - Must be a **metric**, i.e. satisfying

$$
\begin{aligned}
&1.\ d(x,y) \geq 0 \\
&2.\ d(x,y) = 0 \text{ iff } x = y \\
&3.\ d(x,y) = d(y,x) \\
&4.\ d(x,z) \leq d(x,y) + d(y,z)
\end{aligned}
$$

  - Using the same **distance measures** seen in classification problems
    - **Manhattan**
    - **Euclidean (most common)**
    - **Cosine similarity**
    - **Jaccard coefficient, etc….**

Cluster analysis

- **Distance measure (cont.)**
  - When data has **attributes of all types** e.g. **(Steak, Blue, 1.78, 67, 0.5)**
    - **Normalize/standardize** using min-max, z-score (like in the case of e.g. K-NN)
    - Calculate distance for **each attribute with the proper distance metric**
    - Use **weighted formula** to combine effects

Cluster analysis

- Clustering Algorithms
  - **K-means and variants**
  - **Hierarchical clustering**
  - **Density-based clustering**

K-means Algorithm

K-means

- K-means is a **partitional**, **center-based** clustering algorithm
  - **Partitional** = **no hierarchies**, data point belongs to **exactly one cluster**
  - **Center-based** = data points **closest to "center" of cluster**
- K-means **uses the Euclidean distance as a distance metric**
  - Hence, **appropriate only for numerical vectors**
  - Note: Variations of **K-means for vectors with qualitative attributes available e.g. K-modes**

K-means

- The "**K**" in "**K**-means" is the **number of desired clusters**
  - Given as **input to the algorithm by the user** e.g. K=3, K=4 etc
- Basic idea of K-means:
  - Choose initially **K centers (centroids) at random and cluster data around these centers**
  - Iteratively, **calculate new centers of clusters** (centers shift in data space!)
  - Stop when **centers do not shift anymore**
    - Or shift below a threshold

K-means

- K-means algorithm in a nutshell

```
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
```

K-means

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

K-means

- How to calculate the various steps?

Euclidean distance of each point to centroid: $d(x,y) = \sqrt{\sum_{k=1}^{n}(x_k - y_k)^2}$

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

Find new centroid by computing mean of points belonging to cluster (m_i number of items in cluster, C_i old cluster) :

$$c_i = \frac{1}{m_i}\sum_{x \in C_i} x$$

Example: if (1,1), (2,3), (6,2) in cluster, the mean is:
(1+2+6)/3 = 3 /*avg 1st dimension*/
(1+3+2)/3 = 2 /*avg 2nd dimension*/
Hence new mean of cluster is point: (3,2)

K-means

- Example: K-means, K=2

Height	Weight
185	72
170	56
168	60
179	68
182	72

Data set (Height, Weight)

- Assume K=2, i.e. cluster data set of people into 2 (K=2) clusters
  - K always given as input
- **Step 1:** select 2 initial centroids.
  - Various ways to do it
    - **Select 2 (=K) points of the data space randomly** e.g. Height=190, Weight=102 and Height=169, Weight=59 (note: not in dataset)
    - **Select 2 (=K) arbitrary points from the dataset**
      - E.g. select first two observation as centroids, Height=185, Weight=72 and Height=170, Weight=56
      - **We use this!**

Centroids		
	Height	**Width**
Cluster 1	185	72
Cluster 2	170	56

K-means

- Example: K-means, K=2

Height	Weight
185	72
170	56
168	60
179	68
182	72

Data set (Height, Weight)

Centroids			
	Height	Width	Data in cluster
Cluster 1	185	72	(185, 72) ,(179,68), (182,72)
Cluster 2	170	56	(170,56), (168,60)

- **Step 2:** Calculate distance of all other data points from the 2 centroids and add data to closest cluster
  - **Use Euclidean distance**

168,60: distance from cluster 1 = sqrt((185-168)^2 + (72-60)^2) = 20.82
168,60: distance from cluster 2 = sqrt((170-168)^2 + (56-60)^2) = 4.47 (PUT in this cluster)
179, 68: distance from cluster 1 = sqrt((185-179)^2 + (72-68)^2) = 7.21 (Put in this cluster)
179, 68: distance from cluster 2 = sqrt((170-179)^2 + (56-68)^2) = 15
182,72: distance from cluster 1 = sqrt((185-182)^2 + (72-72)^2) = 3 (PUT in this cluster)
182,72: distance from cluster 2 = sqrt((170-182)^2 + (56-72)^2) = 20

K-means

- Example: K-means, K=2

Height	Weight
185	72
170	56
168	60
179	68
182	72

Data set (Height, Weight)

Centroids			
	Height	Width	Data in cluster
Cluster 1	185	72	(185, 72), (179,68), (182,72)
Cluster 2	170	56	(170,56), (168,60)

- **Step 3:** Calculate new centroids from data in cluster

Cluster 1: Height : (185+179+182)/3 = 182, Weight: (72+68+72)/3 = 70.6
Cluster 2: Height: (170+168)/2 = 169, Weight: (56+60)/2 = 58

NEW Centroids		
	Height	Width
Cluster 1	182	70.6
Cluster 2	169	58

K-means

- Example: K-means, K=2

Height	Weight
185	72
170	56
168	60
179	68
182	72

Data set (Height, Weight)

NEW Centroids		
	Height	**Width**
Cluster 1	182	70.6
Cluster 2	169	58

- **Step 4:** Have centroids moved (or has data moved clusters)? Yes. Hence continue iteration

K-means

- Example: K-means, K=2

Height	Weight
185	72
170	56
168	60
179	68
182	72

Data set (Height, Weight)

Centroids			
	Height	**Width**	**Data in cluster**
Cluster 1	182	70.6	(185,72), (179,68), (182,72)
Cluster 2	169	58	(170,56), (168,60)

- **Step 5: :** Calculate distance of all other data points from the 2 new centroids and add data to closest cluster

185,72: distance from cluster 1 = sqrt((182-185)^2 + (70.6-72)^2) = 3.31 (PUT in this cluster)
185,72: distance from cluster 2 = sqrt((169-185)^2 + (58-72)^2) = 21.26
170, 56: distance from cluster 1 = sqrt((182-170)^2 + (70.6-56)^2) = 18.89
170, 56: distance from cluster 2 = sqrt((169-170)^2 + (58-56)^2) = 2.23 (PUT in this cluster)
168,60: distance from cluster 1 = sqrt((182-168)^2 + (70.6-60)^2) = 17.56
168,60: distance from cluster 2 = sqrt((169-168)^2 + (58-60)^2) = 2.23 (PUT in this cluster)
179,68: distance from cluster 1 = sqrt((182-179)^2 + (70.6-68)^2) = 3.96 (PUT in this cluster)
179,68: distance from cluster 2 = sqrt((169-179)^2 + (58-68)^2) = 14.14
182,72: distance from cluster 1 = sqrt((182-182)^2 + (70.6-72)^2) = 1.4 (PUT in this cluster)
182,72: distance from cluster 2 = sqrt((169-182)^2 + (58.6-72)^2) = 18.66

K-means

- Example: K-means, K=2

Height	Weight
185	72
170	56
168	60
179	68
182	72

Data set (Height, Weight)

Centroids			
	Height	**Width**	**Data in cluster**
Cluster 1	182	70.6	(185,72), (179,68), (182,72)
Cluster 2	169	58	(170,56), (168,60)

- **Step 6:** Calculate new centroids from data in cluster

Cluster 1: Height : (185+179+182)/3 = 182, Weight: (72+68+72)/3 = 70.6
Cluster 2: Height: (170+168)/2 = 169, Weight: (56+60)/2 = 58

NEW Centroids		
	Height	**Width**
Cluster 1	182	70.6
Cluster 2	169	58

K-means

- Example: K-means, K=2

Height	Weight
185	72
170	56
168	60
179	68
182	72

Data set (Height, Weight)

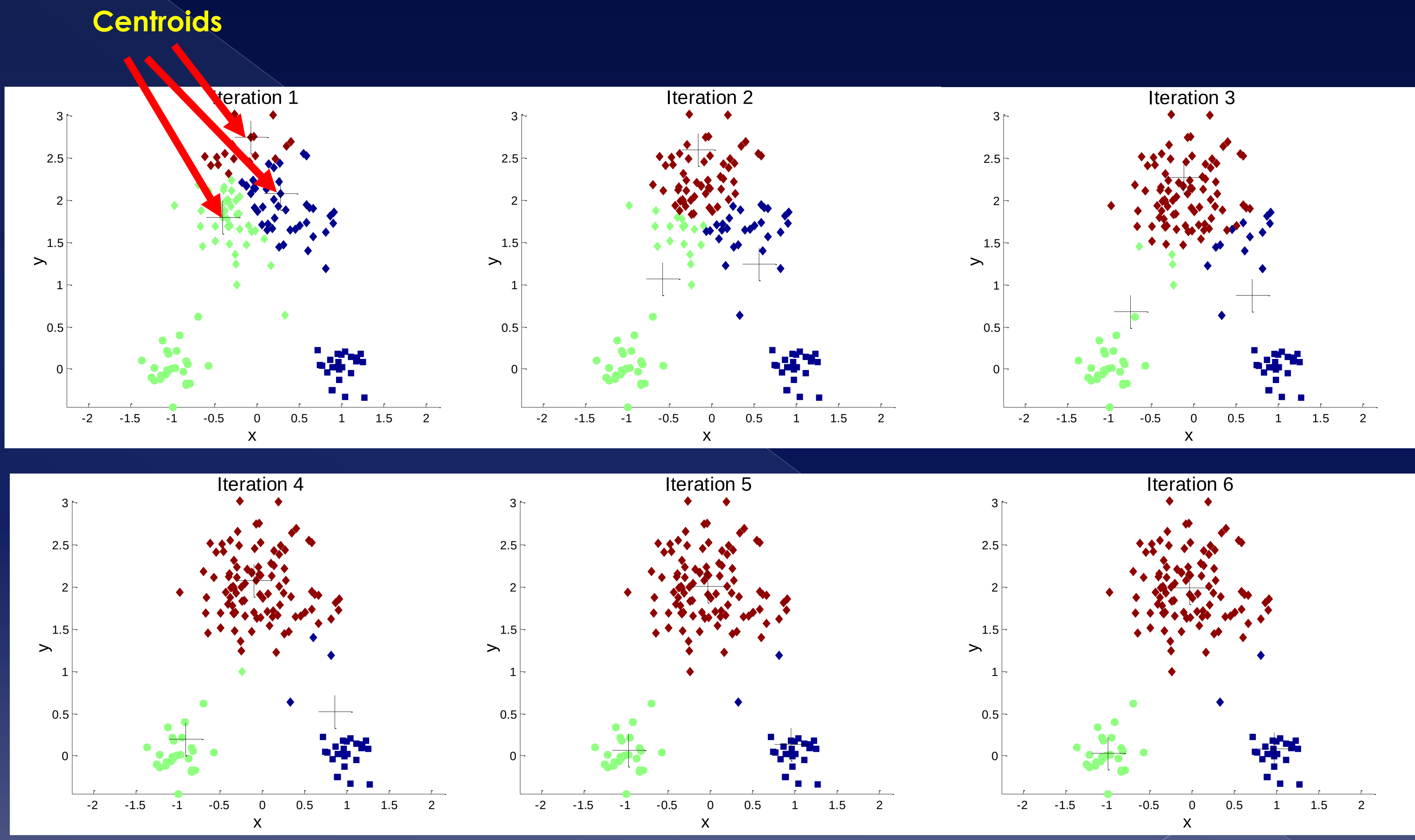
NEW Centroids		
	Height	**Width**
Cluster 1	182	70.6
Cluster 2	169	58

- **Step 7:** Have centroids moved (or has data moved clusters)? **NO. Sweet! K-means terminates**

The final two clusters of our data set are:

Cluster 1: (185,72), (179,68), (182,72)
Cluster 2: (170,56), (168,60)

K-means - Visualized



Note how centroids shift/move at each iteration as a result of step 4 of algorithm i.e. recomputing the centroid of each cluster by calculating the mean of points of cluster.

K-means

- Why does K-means work?
  - It **minimizes an objective function**
    - **Objective function** = equation to be optimized (i.e. minimized, maximized) given some constraints
  - K-means attempts to **minimize the Sum of Squared Error (SSE)** i.e. minimize:

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

K-means

- SSE
  - **dist** = Euclidean distance of point from nearest center c_i (center of cluster C_i)

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

Looks familiar? Yup, basically variance across all clusters

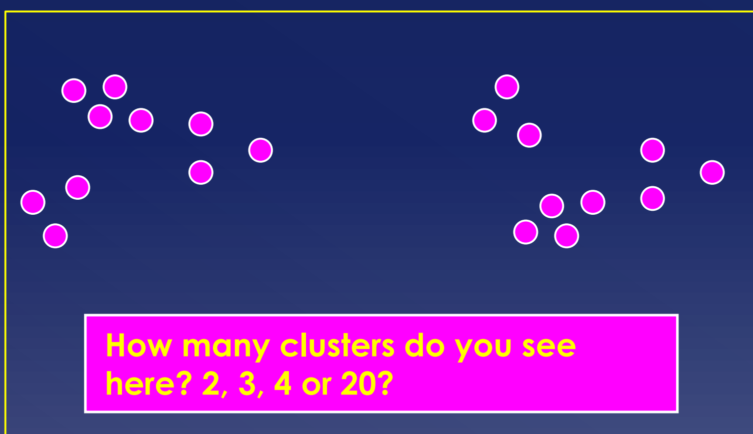
  - In essence SSE **attempts to minimize variance across all clusters**
  - Way to define the **quality of clustering**

K-means

- SSE
  - We can use **SSE as away to evaluate clustering**
    - E.g. Given two clusters, we can choose the one with the smallest error
  - Technique to reduce SSE: **increase number of clusters K**
    - A good clustering with **smaller K** can have a **lower SSE than a poor clustering with higher K**

K-means

- Really, **no good way to pick appropriate K**
  - Depends on the **level of granularity you look at the data!**



Reasoning to chose K

Depends on the level you look at it
1) Look at it from a very top level? Then probably you'll say 2 clusters

2) Look at it from a lower level? Then probably you'll say 4 clusters

3) Look at it from an even lower level? Then probably you'll say 20 clusters (each point defines its own)

In terms of a dataset: you can view the same dataset from very different levels. Are you interested in big-effects on your data (top level view) or are you interested at fine grained effects (lower levels)?

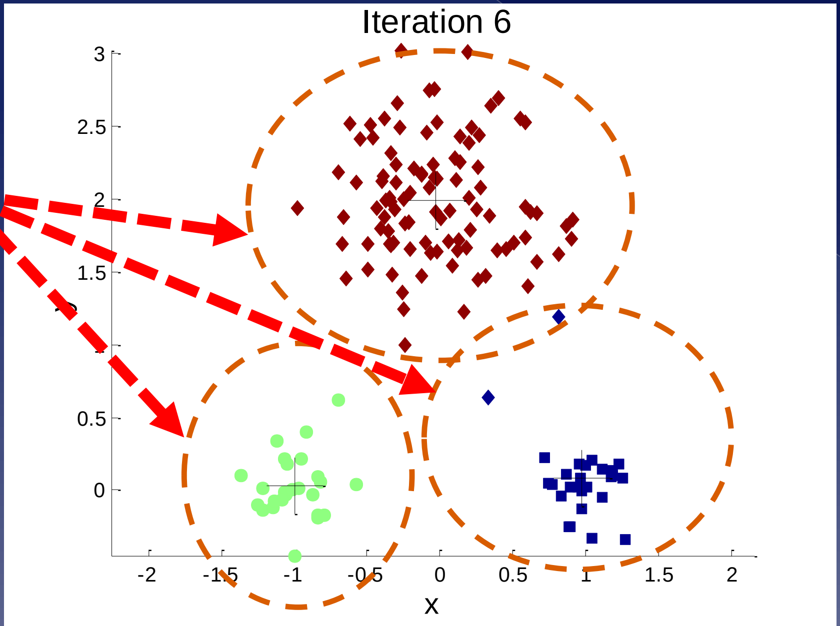
K-means

- But there is **one empirical way** of somehow **estimating a suitable K value**
  - The "Elbow method"
  - **"Elbow method"**
    - Calculate the **percentage of variance explained as a function of the number of clusters K. Choose** a number of clusters **K so that adding another cluster doesn't give much better modeling** (i.e. does not explain a lot better) of the data.
    - But **why the name "Elbow method"** ???
      - Because the graph makes an elbow (see next slides)

K-means

- Which metric to use to assess quality of clustering? In R and Python you may see:
  - **Within-Sum-of-Squares (WSS)**: Total distance of data points from respective cluster centroid.

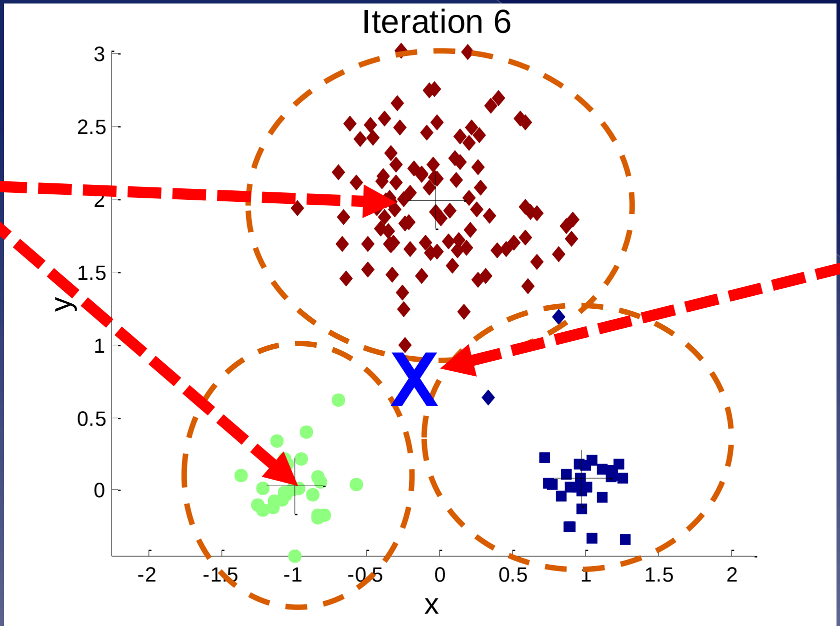
For each cluster, add all distances between data points and cluster it belongs to.
Do this for all clusters and add up the individual Within cluster distances.



K-means

- Which metric to use to assess quality of clustering? In R and Python you may see:
  - **Total-Sum-of-Squares (TSS)**: Total distance of data points from **global mean** of data
    - for a given dataset this is constant!

Cluster centroid

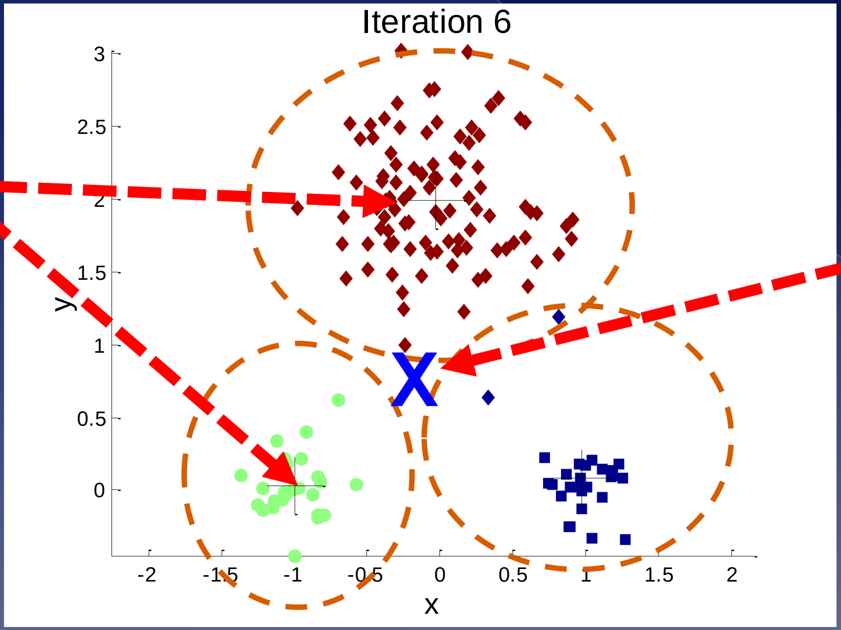


Position of *global mean of data*.
TSS calculates the distances of ALL data points from the global mean, and then adds these distances up.

K-means

- Which metric to use to assess quality of clustering? In R and Python you may see:
  - **Between-Sum-of-Squares (TSS)**: total weighted distance of various cluster centroids to the global mean of data

Cluster centroid

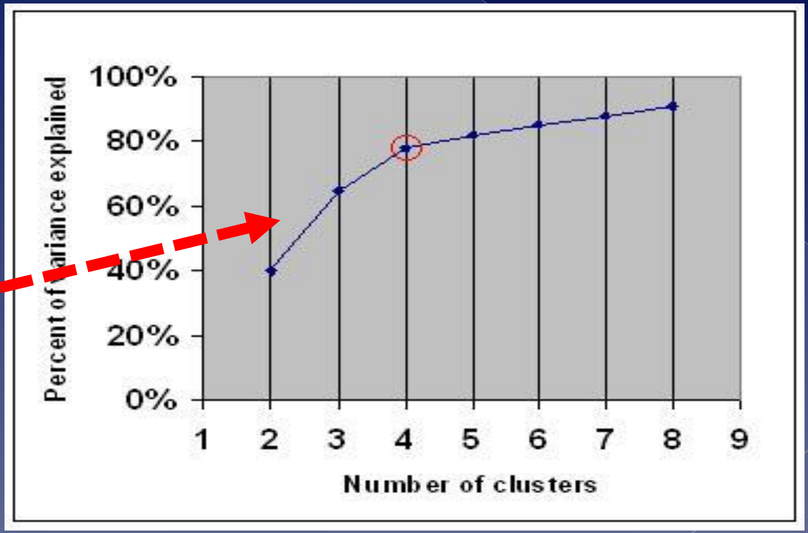


Position of *global mean of data*.
BSS sums up distances between CENTROIDS and GLOBAL MEAN

K-means

- Which metric to use to assess quality of clustering? In R and Python you may see:
  - **R^2 (R-squared)**: defined as BSS / TSS

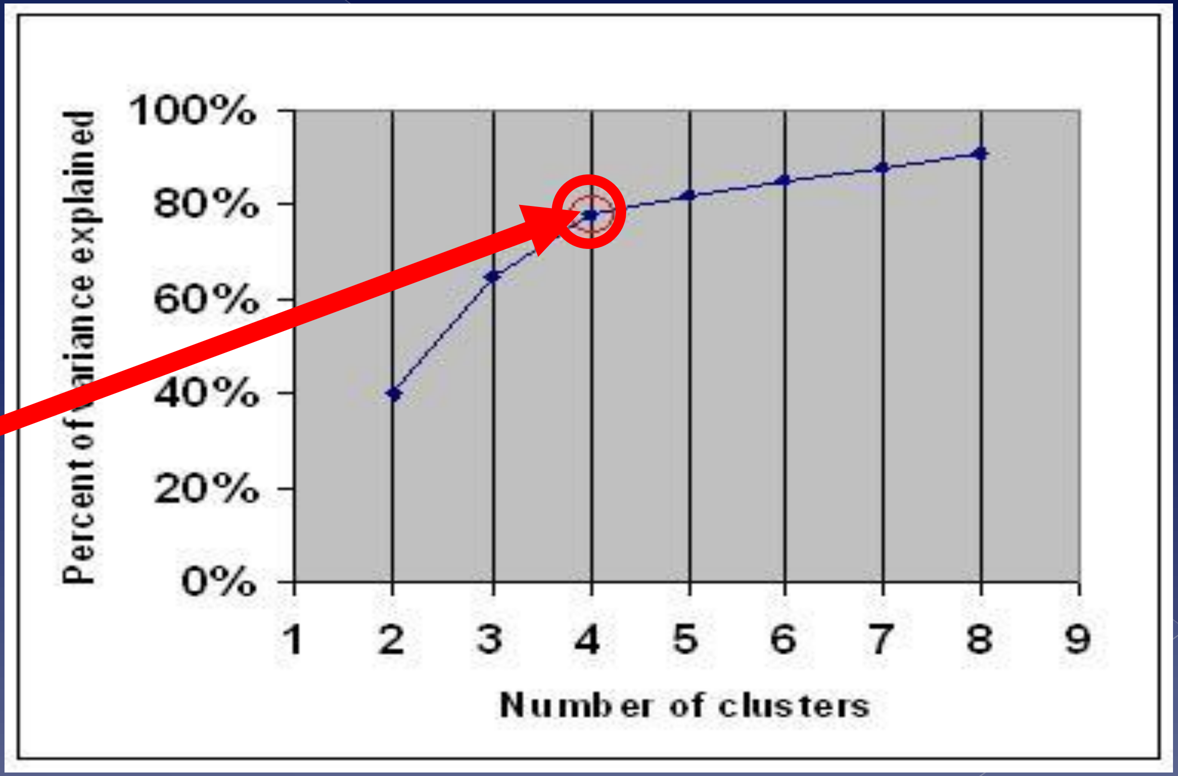
May use this metric to evaluate clustering and apply "ELBOW" method
NOTE: if this increases, this means better clustering.



Using R-squared, elbow method will look like this. Don't get confused. This is normal since the ratio BSS/TSS captures the variance explained. Hence, higher is better.

K-means

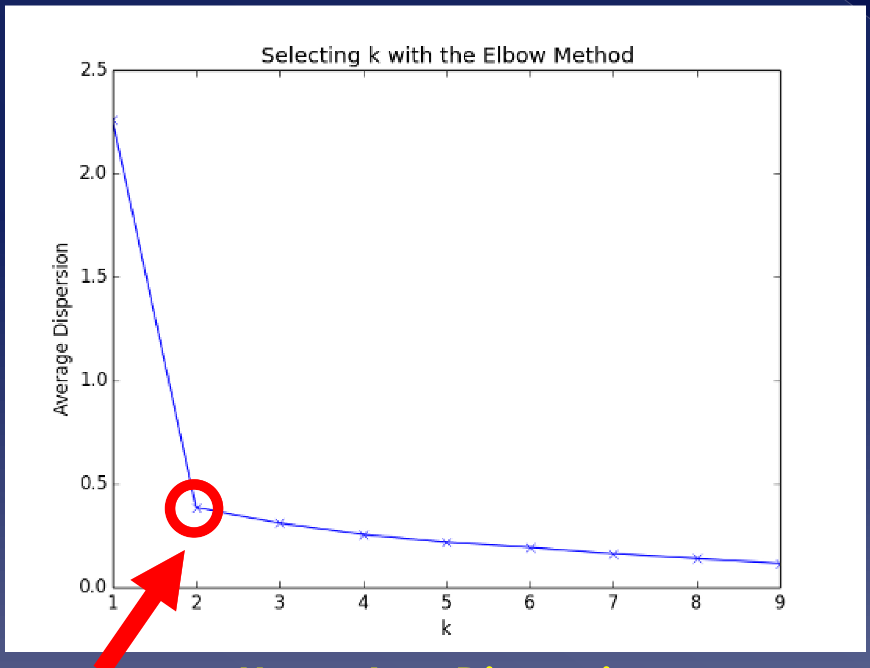
- "Elbow" because **when you plot the pct of variance explained for various K you'll see an elbow ("knick") in the graph**. That's one **ok-ish value for K**

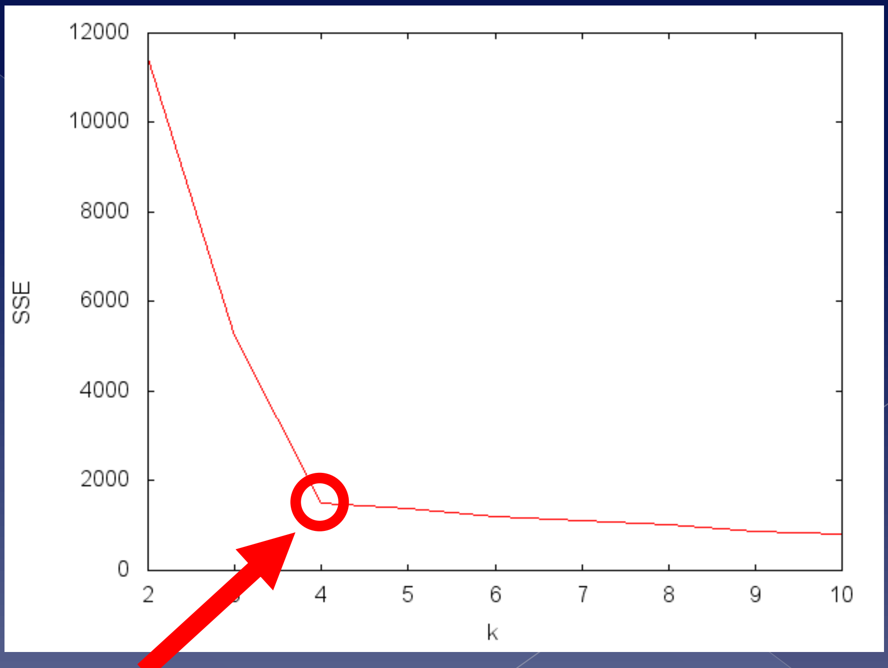


Look an Elbow! Hence 4 is ok-ish for K.

K-means

- In the "Elbow method", pct of variance explained **not the only measure**. You can use others as well (e.g. **Avg dispersion, Within-SS, BSS, ratio Between SS / Total SS** etc)





Elbow. Hence chose K=2

Here: Avg Dispersion

Elbow hence chose K=4

Here: Within SS

K-means

- How to **implement the "Elbow method"?**
  - Simple: **Execute K-means clustering** for your data **for all values of K from 2 until some max that you set (say 200)**. After **each execution of K-means**, **store your desired metric** (e.g. SSE, average dispersion, Pct of variance explained etc)
  - **Plot these values that you got from each execution of K-means**
  - Look for **the Elbow is**.
  - Choose **value K corresponding to Elbow**.
  - **Execute K-means again** with the choosen K value

K-means

- How to **solve the problem of choosing the proper K value?**
  - **Sorry, can't**. No convincing algorithms exist for selecting **the exactly appropriate value of K**
    - "Elbow method" is just **one method to somehow get an approximation of K**.
  - However, **Hierarchical Clustering is a way of addressing this concern**
    - In a **different way though**

K-means

- **Application of K-means involves pre- and post-processing steps**
  - **Pre-processing**
    - Normalize the data
    - Eliminate outliers
  - **Post-processing**
    - Eliminate small clusters that may represent outliers
    - Split 'loose' clusters, i.e., clusters with relatively high SSE
    - Merge clusters that are 'close' and that have relatively low SSE
    - Can use these steps during the clustering process

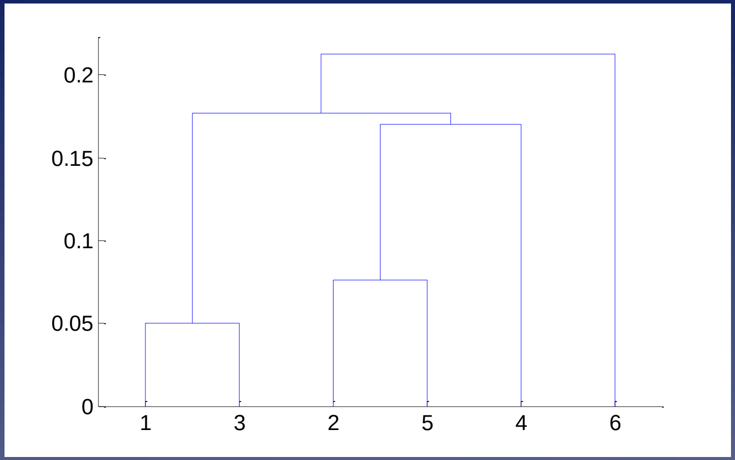
K-means

- **Pros/Cons of K-means?**
  - **Pros**
    - **Simple**
    - **Computationally fast**, even for many variables (than hierarchical clustering)
    - Produces in general **tighter clusters**
  - **Cons**
    - **Sensitive to initial K values**
    - **Different initial partitions** can produce **different clusters**
    - Does **not work well** with **clusters of different sizes and densities**
    - In it's current form, **works only for numerical data** (not nominal or ordinal values)
      - Although variations have been proposed e.g. K-modes

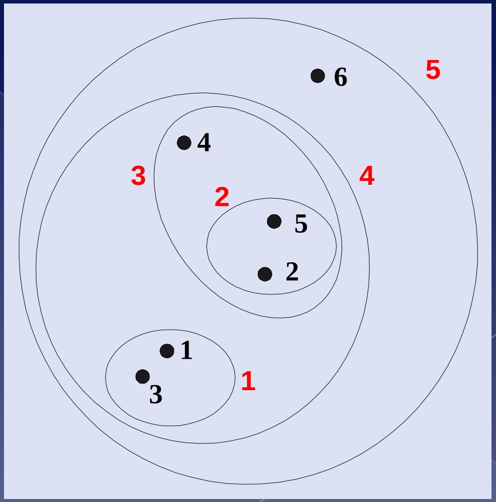
Hierarchical Clustering

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**



Dendrogram



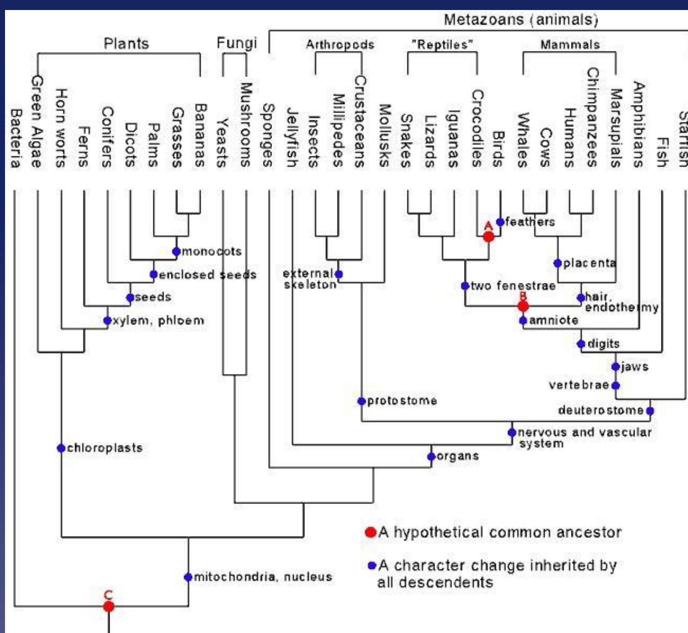
Nested clusters

Hierarchical Clustering

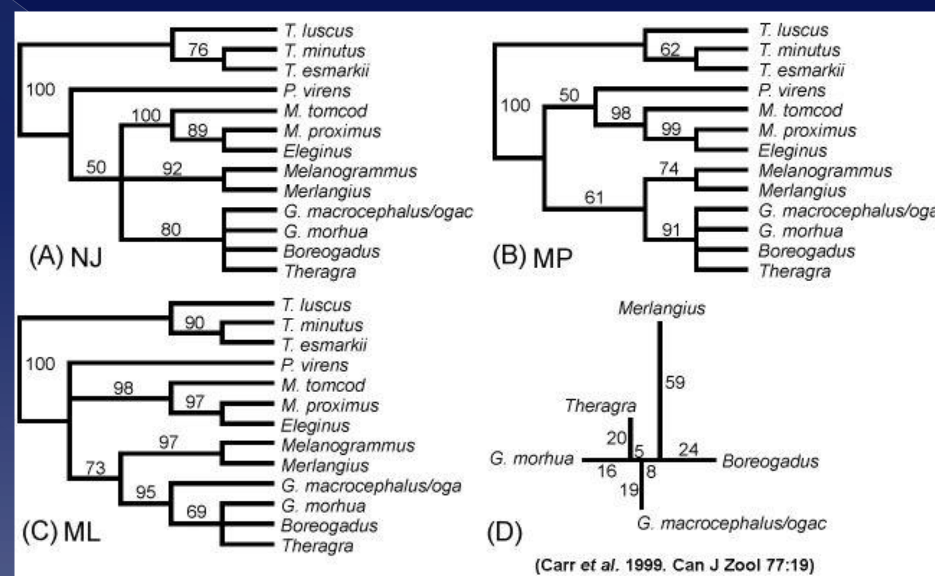
- Strengths of Hierarchical Clustering
  - Do **not have to assume any particular number of clusters** (in contrast to K-means) i.e. **solves the problem of choosing the appropriate value for K, for which no good solutions exist.**
    - Interesting fact: you can create any desired number of partitional clusters **by 'cutting' the dendogram at the proper level**
  - They may **correspond to meaningful taxonomies**
    - Example in biological sciences (e.g., animal kingdom, phylogeny reconstruction, …)

Hierarchical Clustering

- Strengths of Hierarchical Clustering (cont)
  - **More informative than “flat” clusters** (partitional)



Taxonomies



Phylogeni reconstruction

Hierarchical Clustering

- Types of Hierarchical Clustering
  - Based on the way they proceed to create clusters and clusters of clusters
    - **Agglomerative (Bottom-up)**
      - Basic idea
        - Start with the points as individual clusters (i.e. each point is one cluster)
        - At each step, merge the closest pair of clusters until only one cluster (or k clusters) left
    - **Divisive (Up-Down)**
      - Basic idea
        - Start with one, all-inclusive, big cluster
        - At each step, split a cluster until each cluster contains a point (or there are k clusters)

Agglomerative Clustering

- General **outline of Agglomerative Clustering algorithm**:

```
1. Compute the proximity/distance matrix
2. Let each data point be a cluster
3. Repeat
4.      Merge the two closest clusters
5.      Update the proximity/distance matrix
6. Until only a single cluster remains
```

- Important step is the **computation of the proximity/distance matrix and distance between clusters**
  - There are many possible ways

Agglomerative Clustering

- Proximity/Distance matrix?
  - A two **dimensional matrix** containing **the distances, taken pairwise, between the elements of a set**

	p1	p2	p3
p1	0	13.3	3.9
p2	13.3	0	5.6
p3	3.9	5.6	0

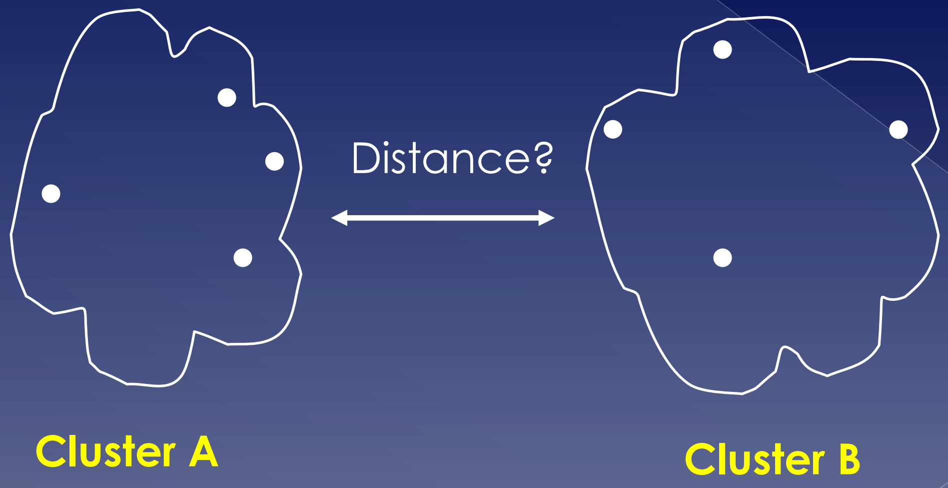
Proximity/Distance matrix of 3 points. Here distance measure e.g. Euclidean

d(p1,p2) = 13.3
d(p3,p2) = 5.6
etc

In general, distance measure can be anything appropriate: Euclidean, Manhattan, Minkowski etc

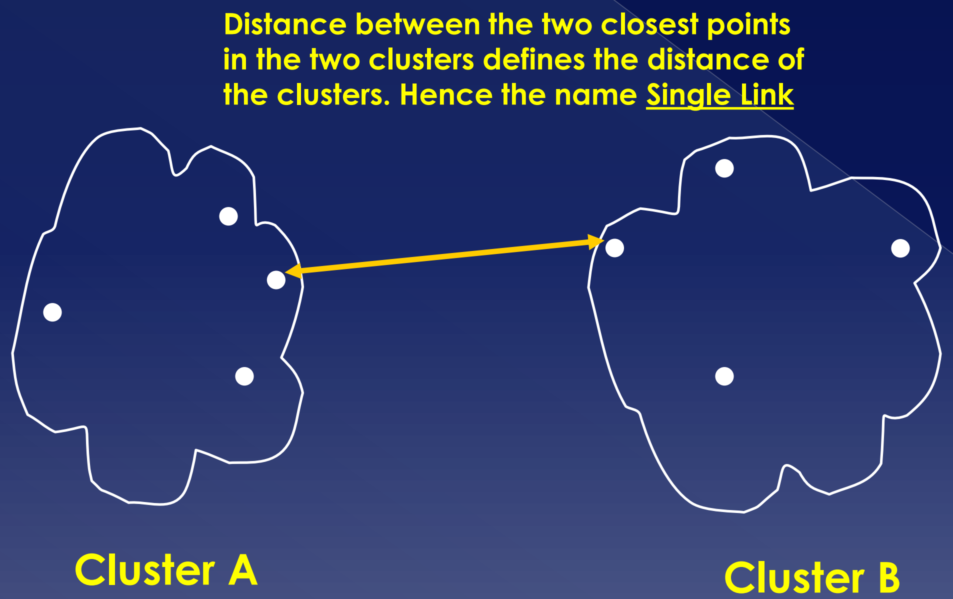
Agglomerative Clustering

- Distance matrix between points is easy. But **Agglomerative clustering requires also distance between clusters (see steps 4 and 5 of algorithm) – Inter-cluster distance**
  - How to **define inter-cluster distance i.e. distance between set of points?**
  - Many different ways



Agglomerative Clustering

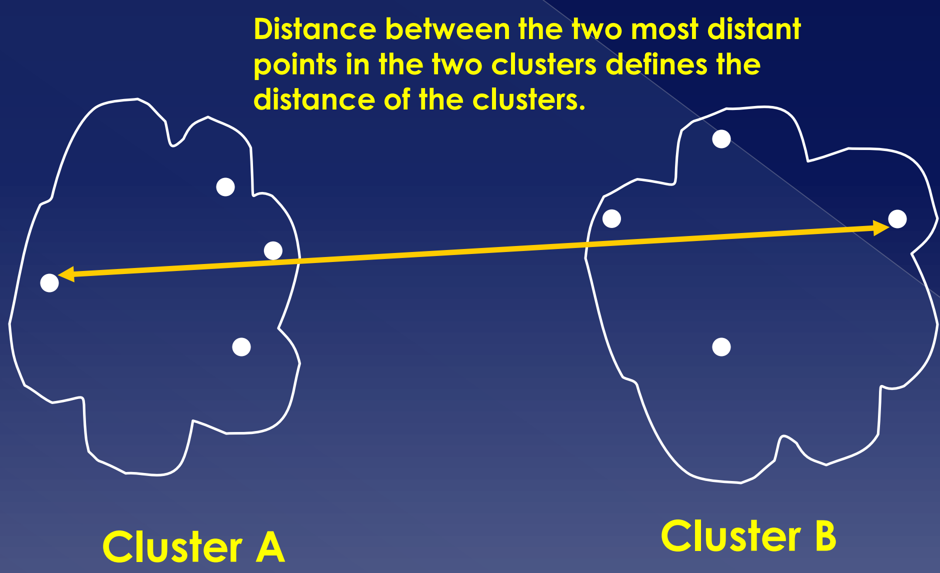
- Measuring distance between clusters
  - **Minimum distance/MIN method (or Single Link)**
    - Distance between clusters is the distance **of the two closest points** in the different clusters



To find these points, determine distance of all pairs of points in the two clusters and get pair with minimum distance. This distance will be the distance of the clusters.

Agglomerative Clustering

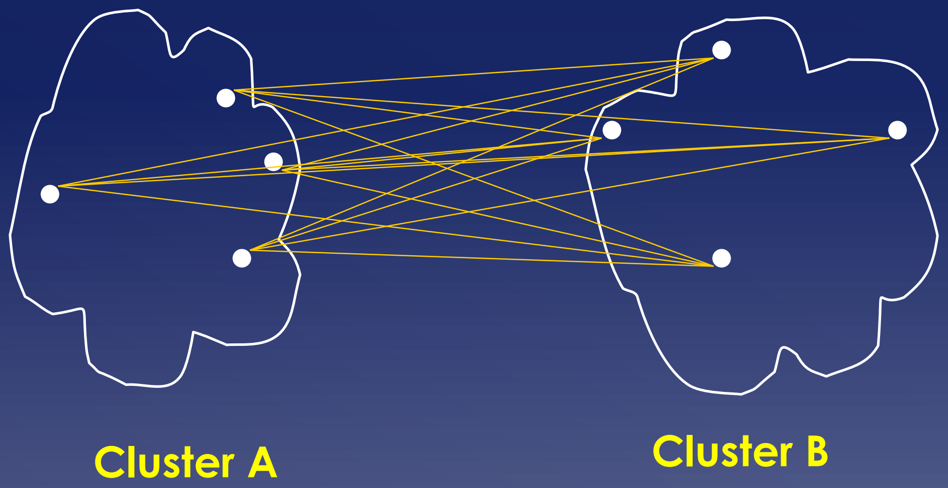
- Measuring distance between clusters
  - **Maximum distance (or Complete linkage)**
    - Distance of two clusters is based **on the two most distant points** in the different clusters



To find these points, determine distance of all pairs of points in the two clusters and get pair with maximum distance. This will be the distance of the clusters

Agglomerative Clustering

- Measuring distance between clusters
  - **Group Average**
    - The **average distance between any pair of points in the two clusters**

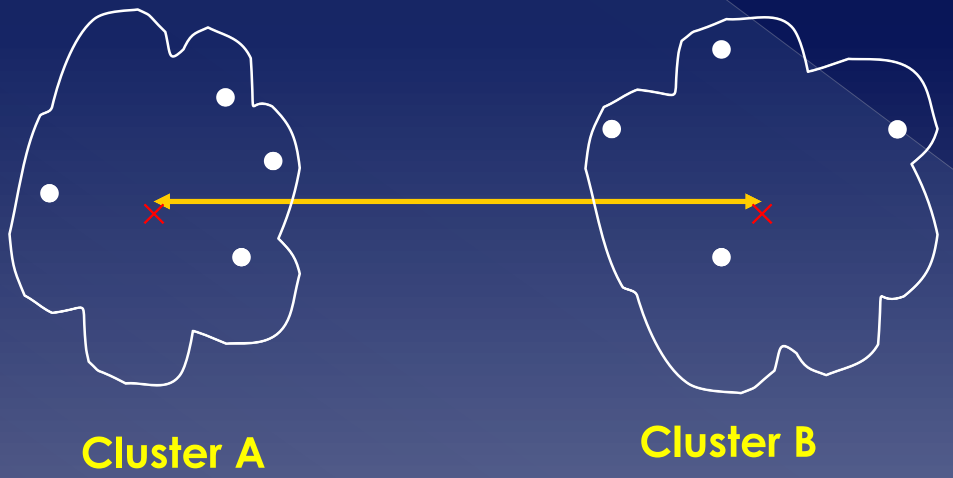


$$d(A,B) = \frac{\sum_{x\in A, y\in B} d(x,y)}{|A||B|}$$

Formula for Group Average distance of clusters A and B

Agglomerative Clustering

- Measuring distance between clusters
  - **Centroid distance**
    - The distance **between the centroids of the two clusters**



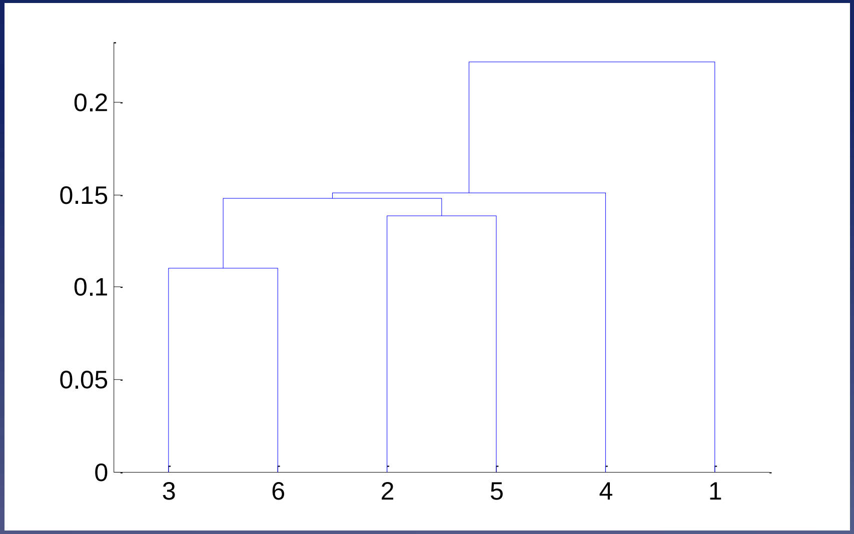
Agglomerative Clustering

- Measuring distance between clusters
  - **Other methods driven by an objective function**
    - E.g. **Ward's method** which aims to **minimize squared error**

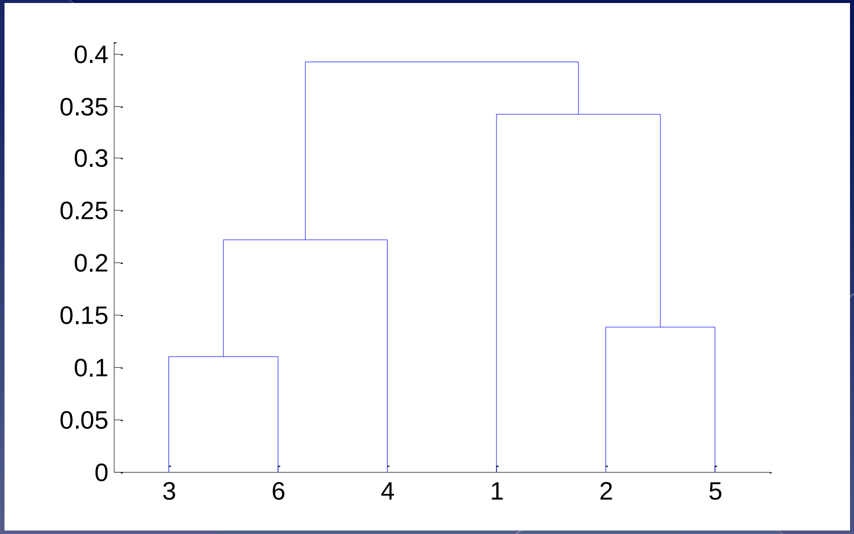
Agglomerative Clustering

- Does distance measuring method influence outcome of hierarchical clustering?
  - Yes!

Hierarchical clustering of the same dataset with different distance measures



Using MIN (Single Link)



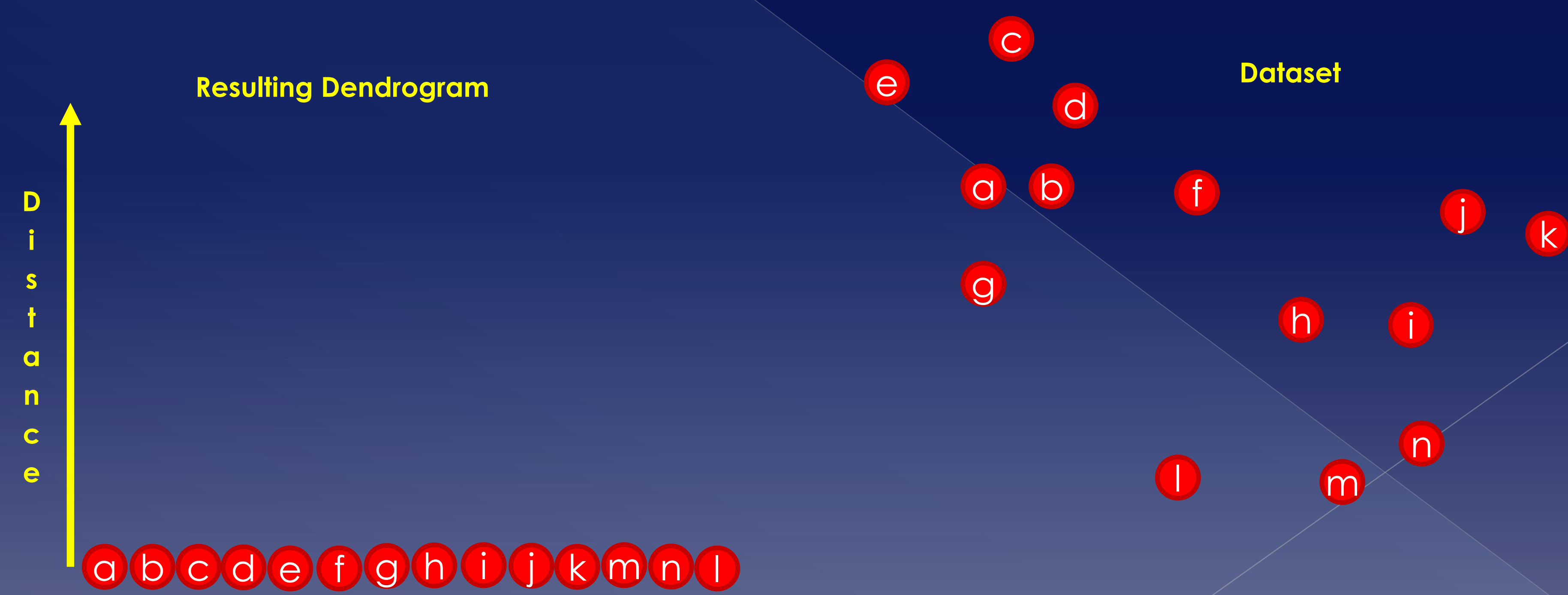
Using MAX (Complete Link)

Agglomerative Clustering

- Pro and Cons of cluster distance measures
  - **MIN**
    - **Pro :** Can handle non-elliptical shapes
    - **Cons:** Sensitive to noise and outliers
  - **MAX**
    - **Pro :** Less susceptible to noise and outliers
    - **Con:** Breaks large clusters, Biased towards globular (=spherical) clusters
  - **GROUP AVERAGE**
    - **Pro:** Less susceptible to noise and outliers
    - **Con:** Biased towards globular clusters

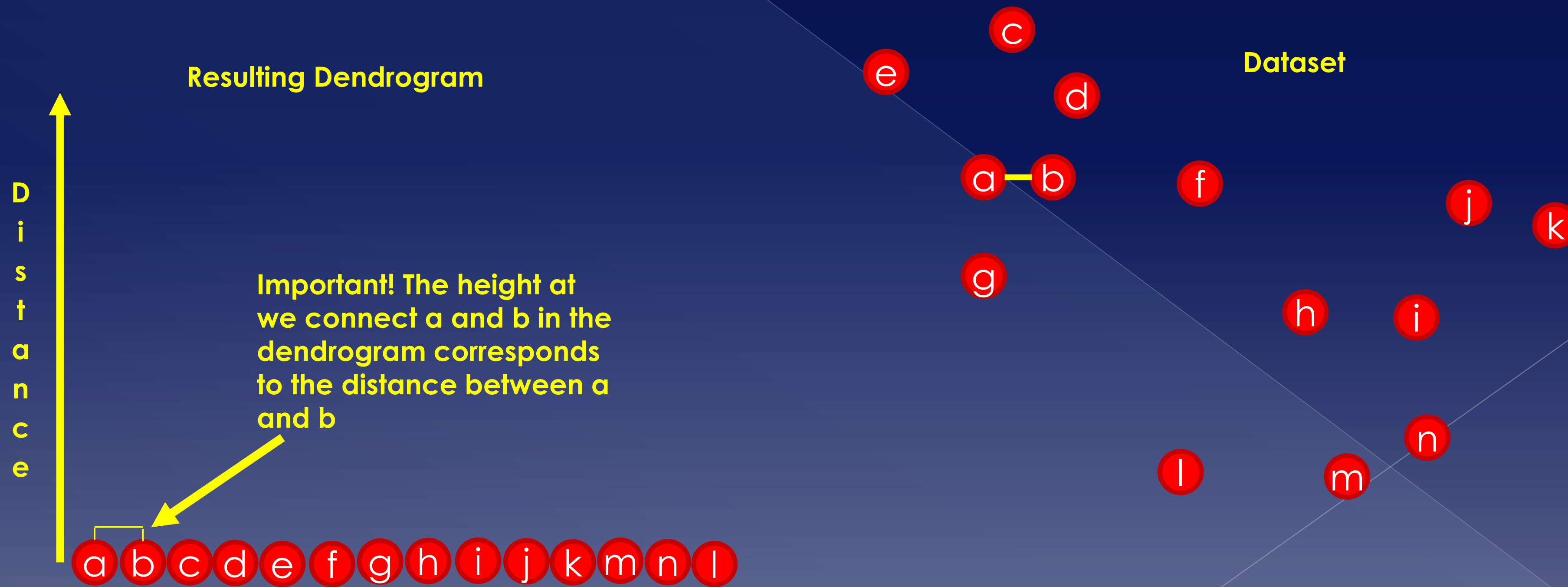
Agglomerative Clustering

- **Demonstrating the idea of Agglomerative Clustering** and **how to construct the dendrogram (note: no numbers yet)**
  - Assume **MIN (Single Link) method for cluster distance measure**



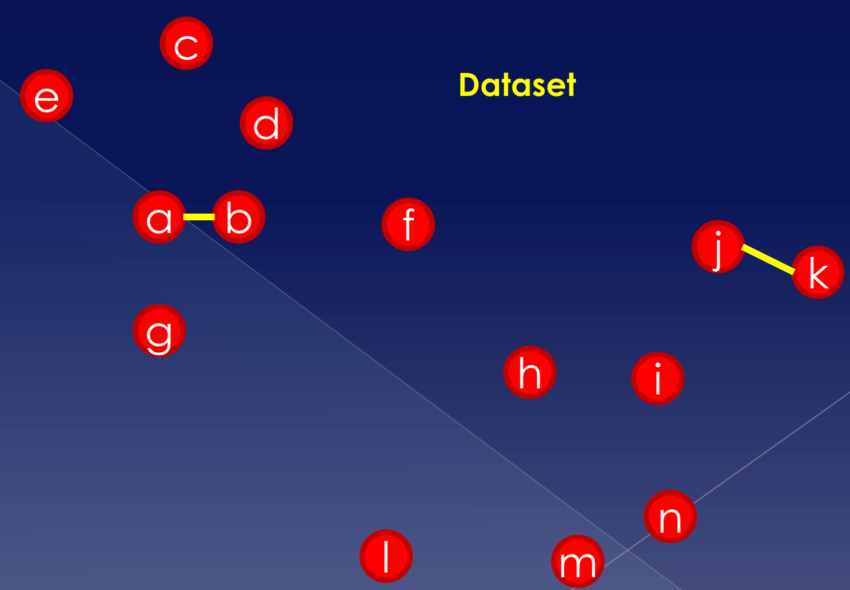
Agglomerative Clustering

- Initially, each point is its own cluster. Then find two points who are the closest and merge them. Lets say a, b closest. Connect them in the dendrogram



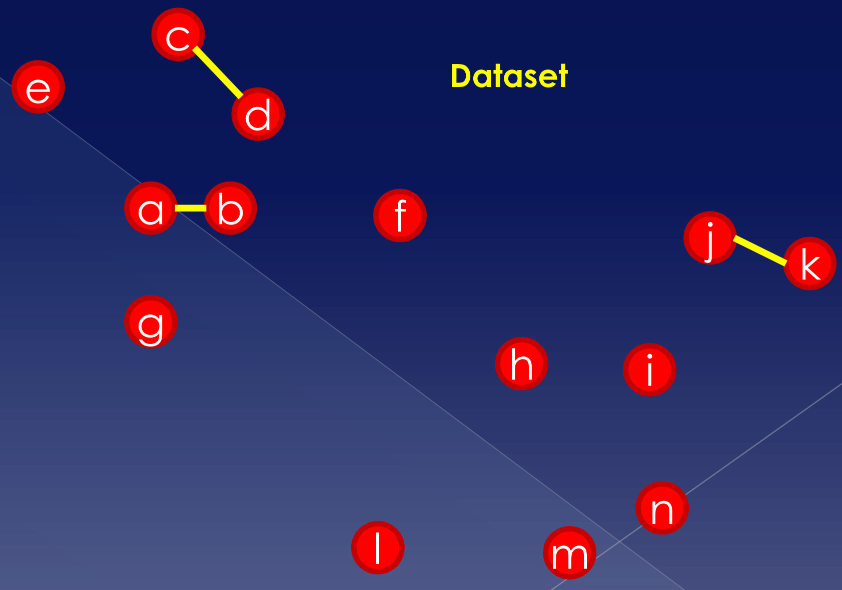
Agglomerative Clustering

- Look for next closest pair of clusters and connect them in the dendrogram e.g. j and k



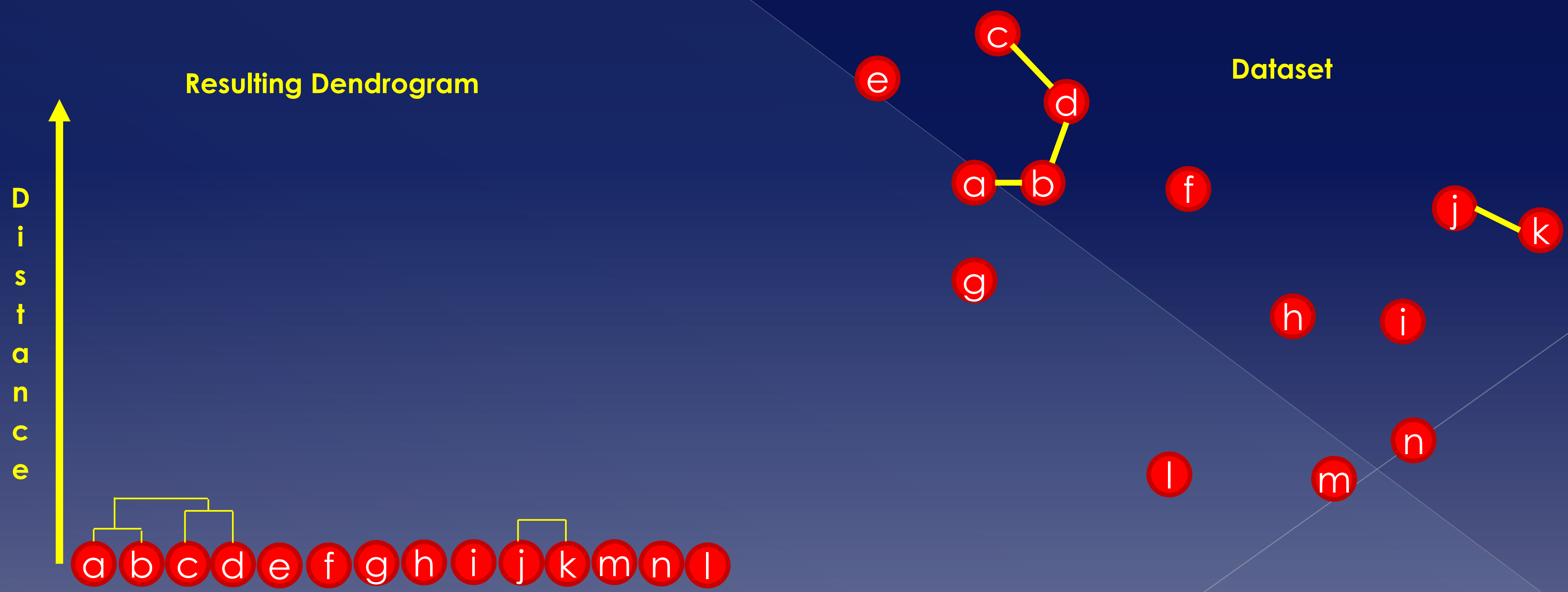
Agglomerative Clustering

- Next closest pair, e.g. c and d



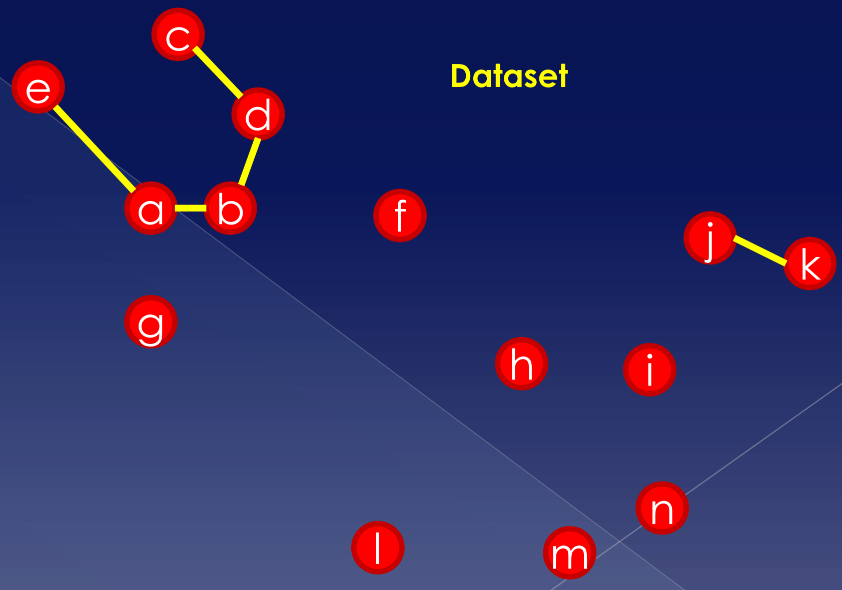
Agglomerative Clustering

- Next closest pair, e.g. b and d. But these belong to clusters already hence merge clusters in dendrogram



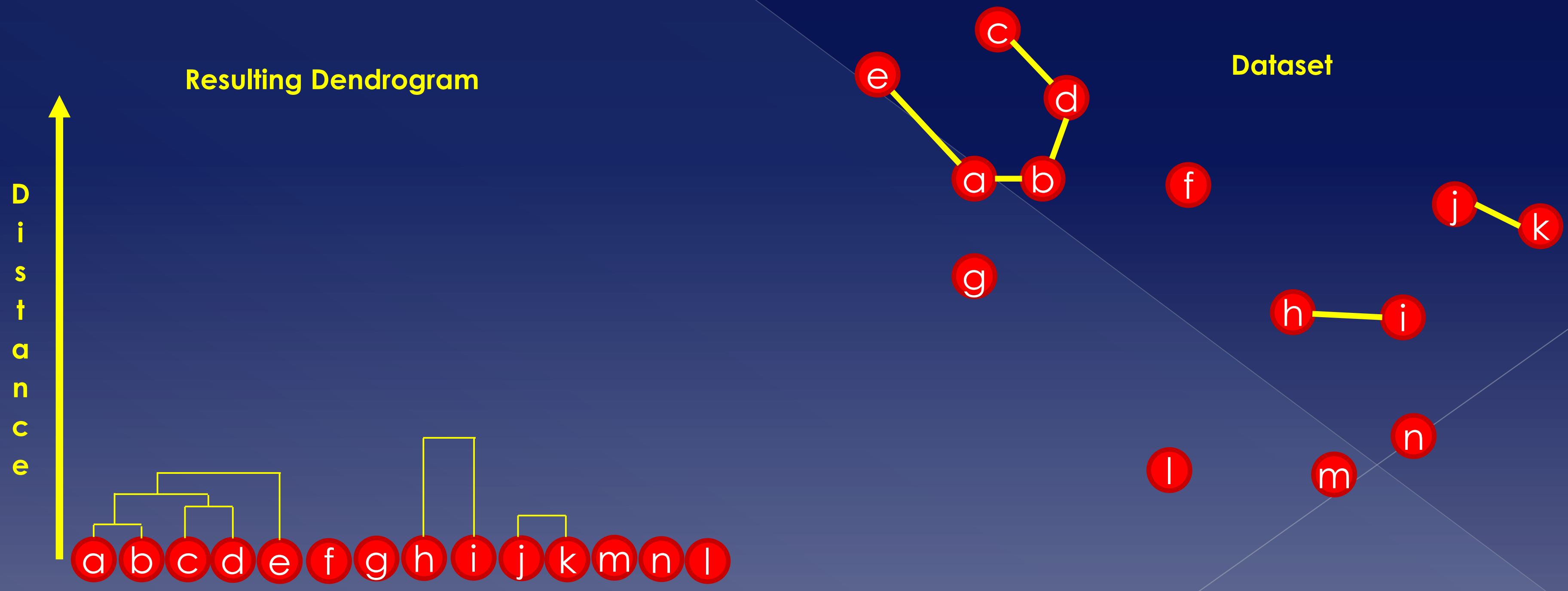
Agglomerative Clustering

- Next closest pair, e.g. e and a (merge clusters)



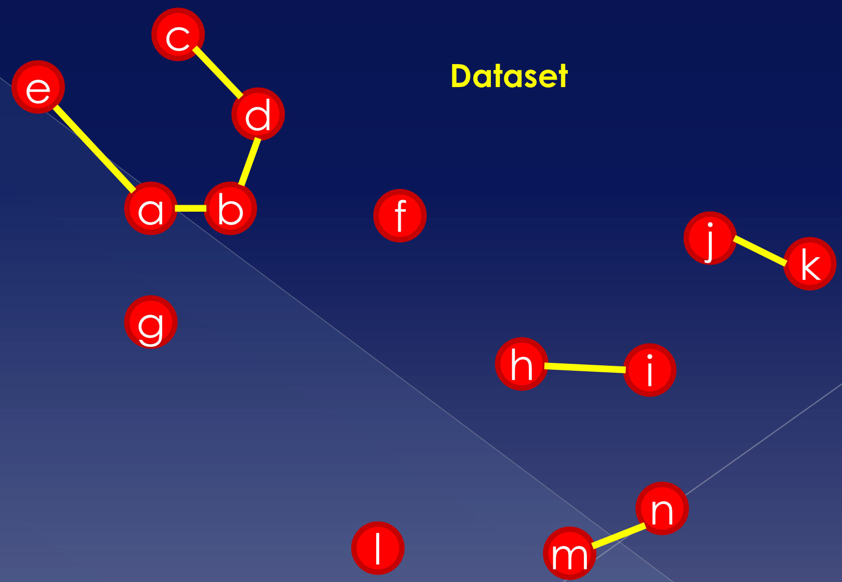
Agglomerative Clustering

- Next closest pair, e.g. h and i



Agglomerative Clustering

- Next closest pair, m and n

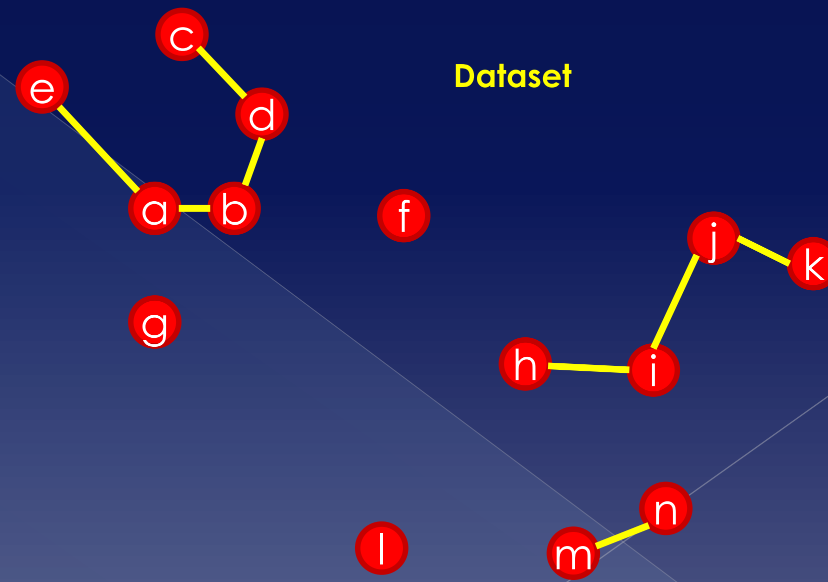


Agglomerative Clustering

- Next closest pair, i and j (merge clusters)

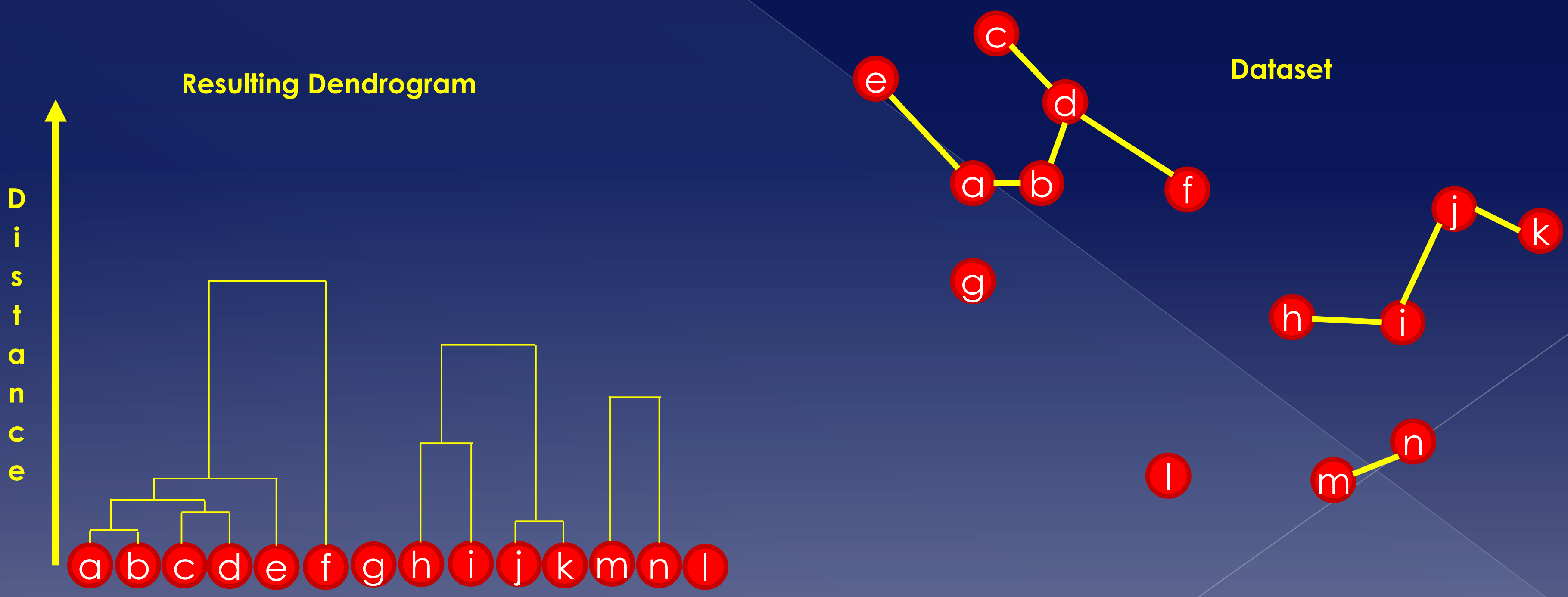
Resulting Dendrogram

Dataset



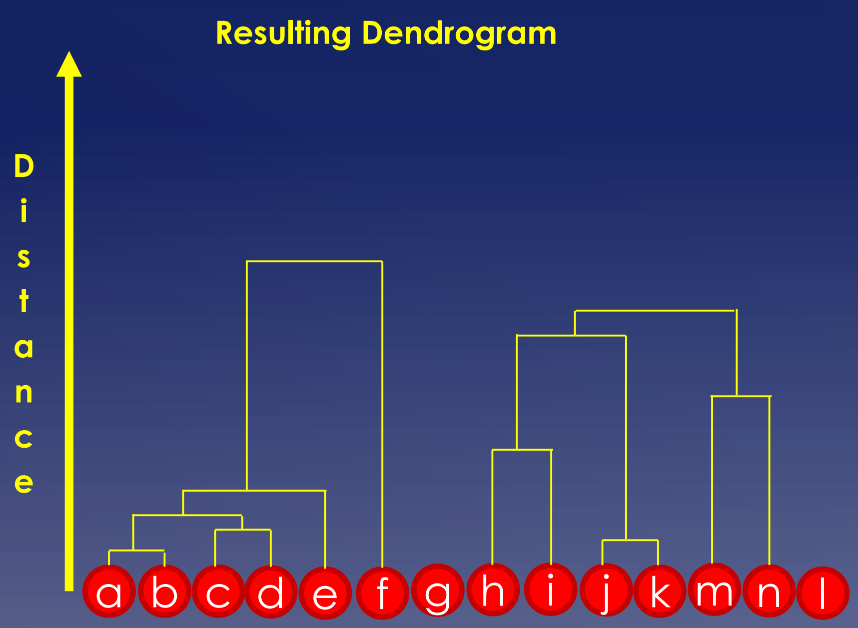
Agglomerative Clustering

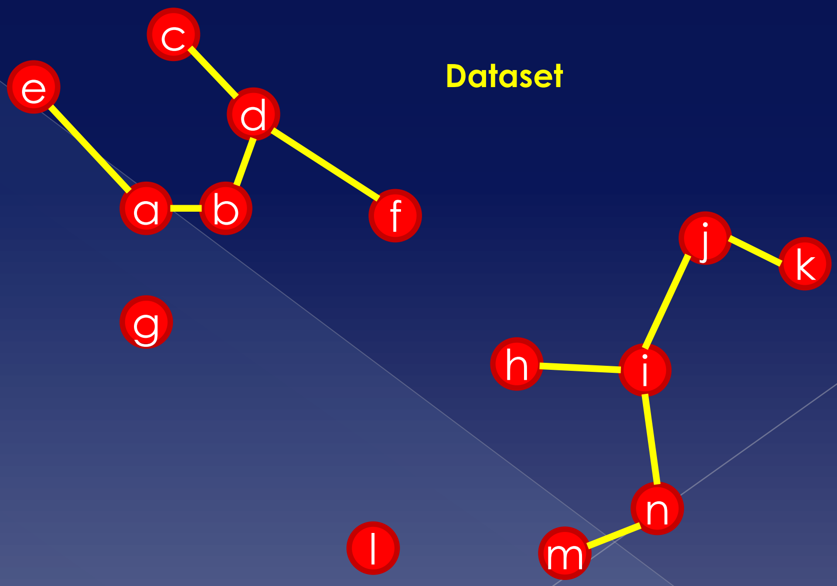
- Next closest pair, e.g. f and d (merge clusters)



Agglomerative Clustering

- Next closest pair, e.g. n and i (merge clusters)

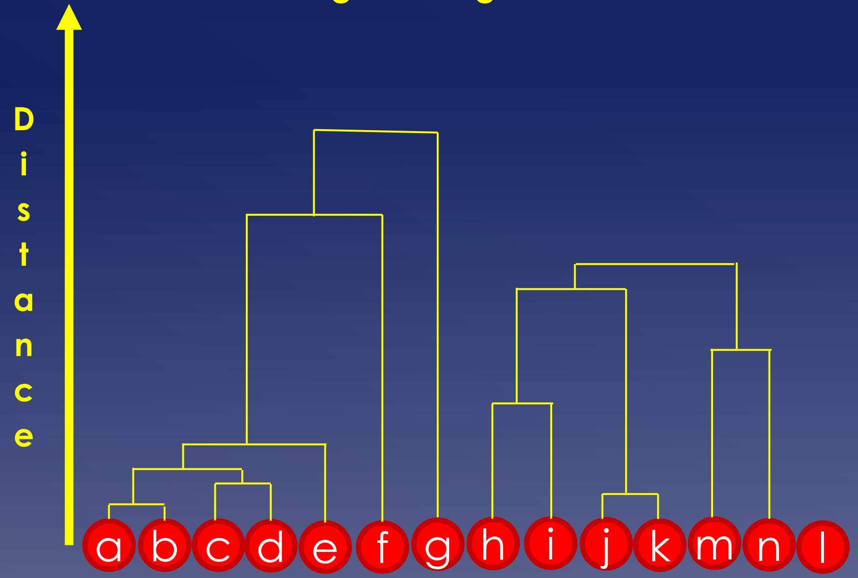




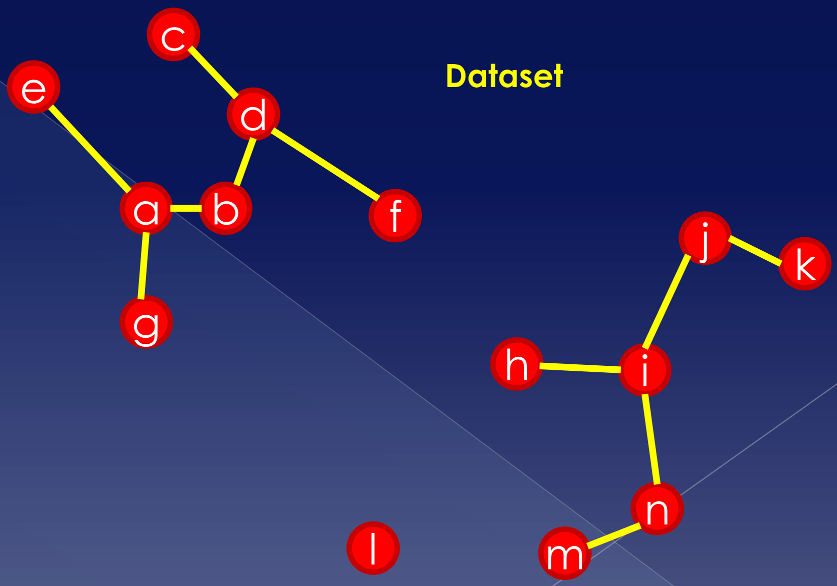
Agglomerative Clustering

- Next closest pair, e.g. g and a (merge clusters)

Resulting Dendrogram



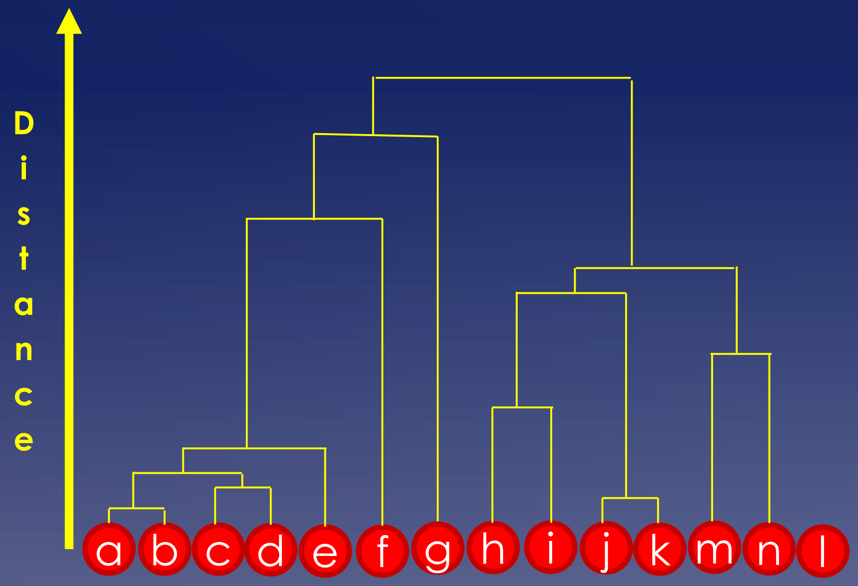
Dataset



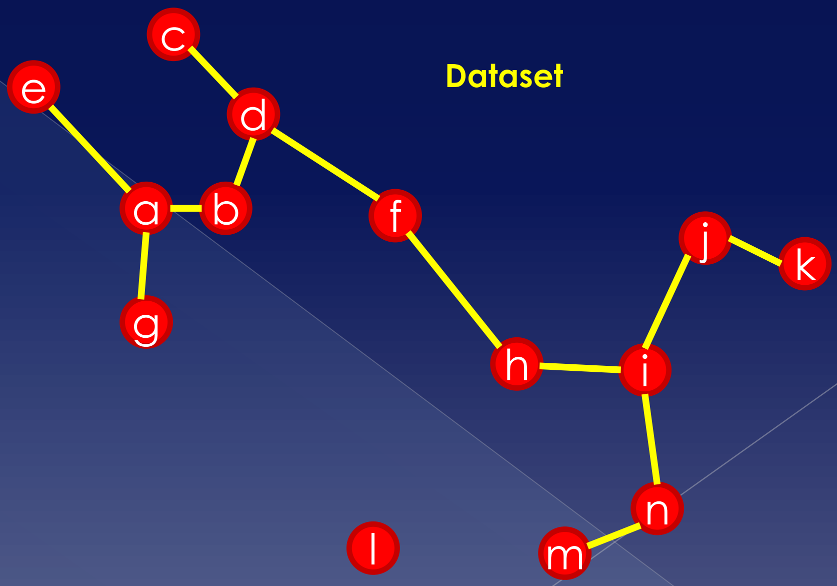
Agglomerative Clustering

- Next closest pair, e.g. f and h

Resulting Dendrogram



Dataset

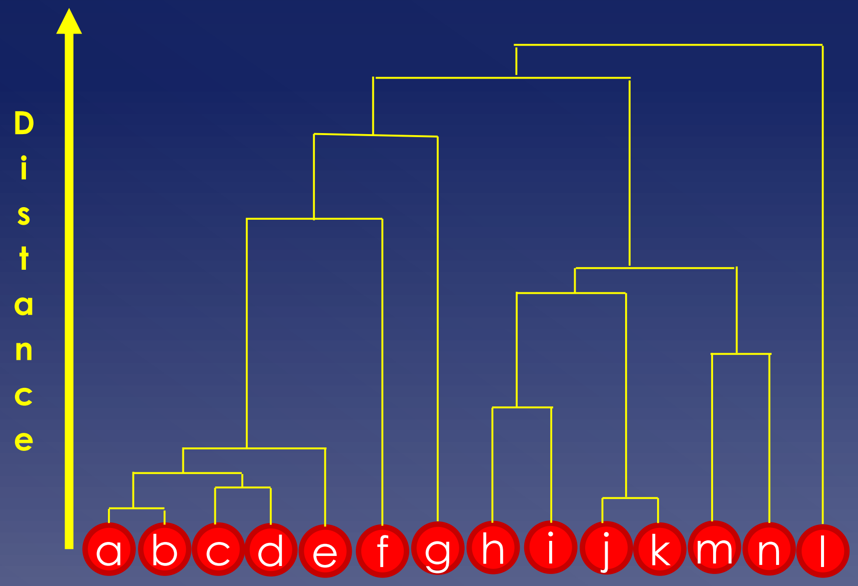


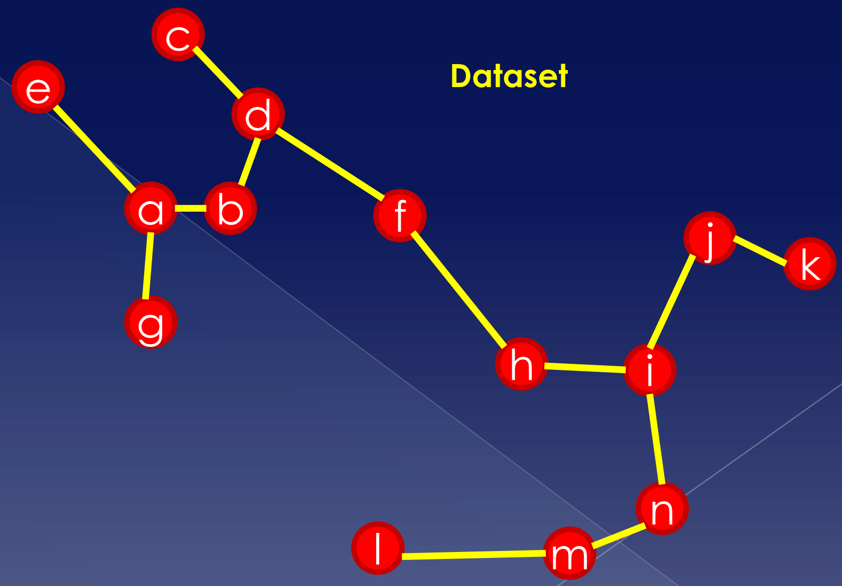
Agglomerative Clustering

- Next closest pair, l and m

Single "Big" Cluster created!
Process terminates

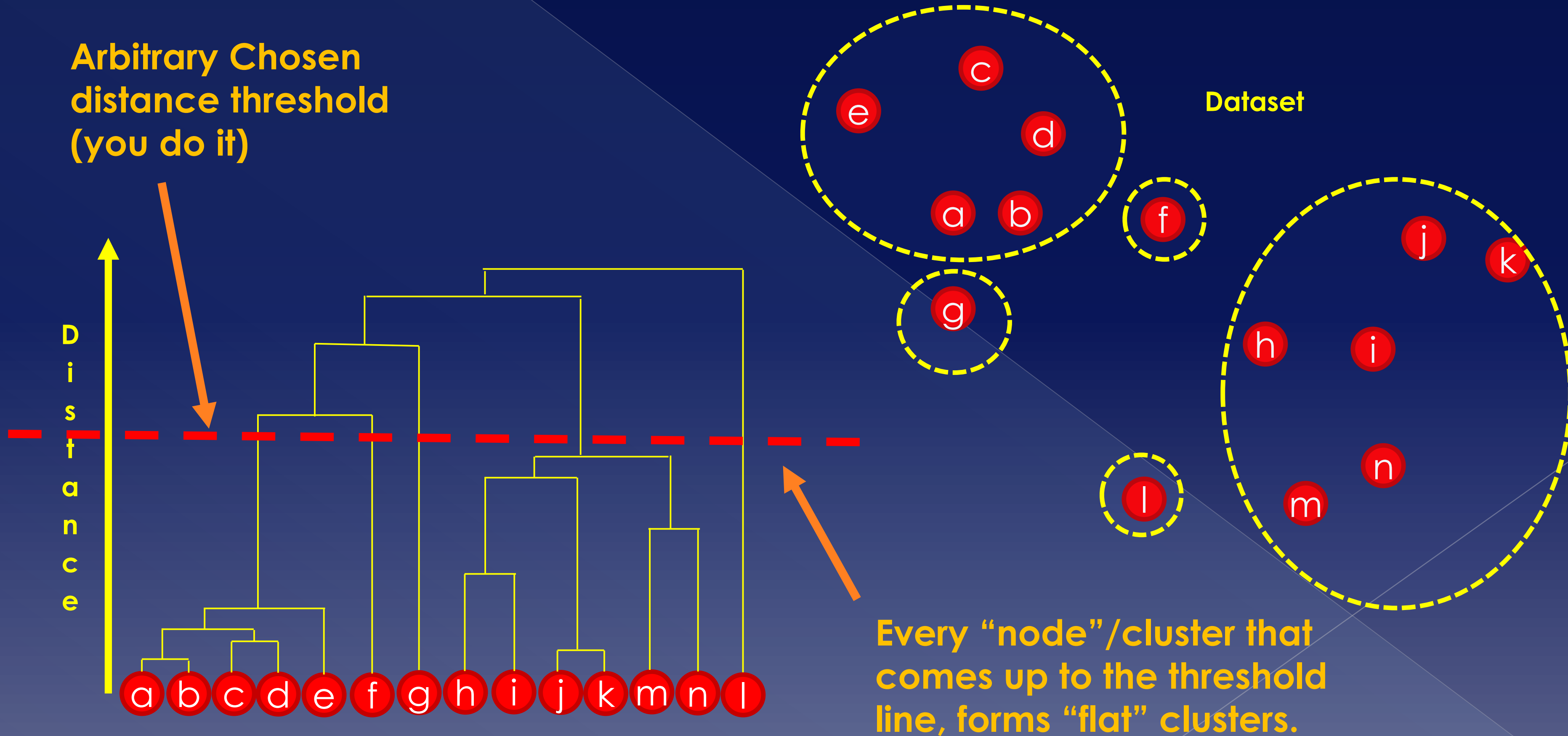
Resulting Dendrogram





Agglomerative Clustering

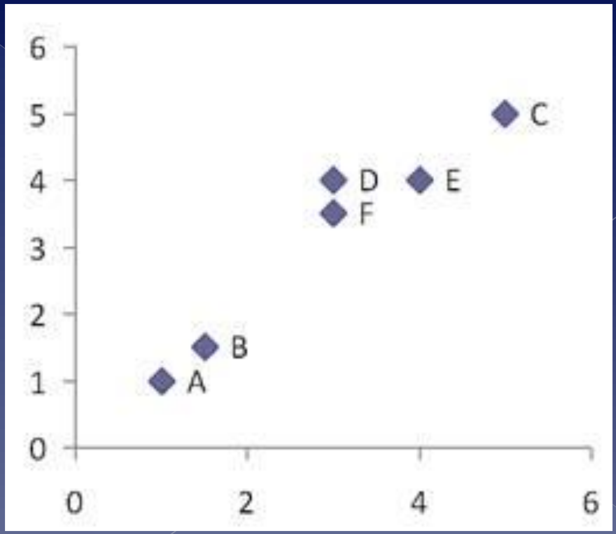
- **Interesting aspect of Dendrograms**
  - You can create "flat"/partitional clusters **by choosing a distance threshold in the dendrogram!**



Agglomerative Clustering

- **Concrete example of Agglomerate clustering** algorithm with distance matrix (yes, with numbers)
  - **Assume 6 points** in a two dimensional space, on which we execute agglomerative clustering
  - **Assume MIN (single linkage) method for cluster proximity**

	X	Y
A	1	1
B	1.5	1.5
C	5	5
D	3	4
E	4	4
F	3	3.5



Agglomerative Clustering

- Example: Step 1 of algorithm
  - Calculate distance matrix for these 6 points
    - Initially use Euclidean distance

Proximity/Distance matrix – INITIAL DISTANCE MATRIX

	A	B	C	D	E	F
A	0	0.71	5.66	3.61	4.24	3.20
B	0.71	0	4.95	2.92	3.54	2.50
C	5.66	4.95	0	2.24	1.41	2.50
D	3.61	2.92	2.24	0	1.00	0.50
E	4.24	3.54	1.41	1.00	0	1.12
F	3.20	2.50	2.50	0.50	1.12	0

NOTE: We call points A,B,C… clusters now as each point defines a cluster (with a single point in it) and agglomerative clustering proceeds bottom-up.

Agglomerative Clustering

- Example: Step 2 of algorithm
  - **All points A,B,C,D,…. are considered clusters now**, with exactly 1 point in each, as each point defines a cluster and agglomerative clustering proceeds bottom-up

Proximity/Distance matrix

	A	B	C	D	E	F
A	0	0.71	5.66	3.61	4.24	3.20
B	0.71	0	4.95	2.92	3.54	2.50
C	5.66	4.95	0	2.24	1.41	2.50
D	3.61	2.92	2.24	0	1.00	0.50
E	4.24	3.54	1.41	1.00	0	1.12
F	3.20	2.50	2.50	0.50	1.12	0

Called "clusters" now

Total of 6 clusters

Agglomerative Clustering

- Example: Inside step 3 repeat. Execute step 4 of algorithm
  - Find in distance matrix clusters with minimum distance. Here F,D

	A	B	C	D	E	F
A	0	0.71	5.66	3.61	4.24	3.20
B	0.71	0	4.95	2.92	3.54	2.50
C	5.66	4.95	0	2.24	1.41	2.50
D	3.61	2.92	2.24	0	1.00	0.50
E	4.24	3.54	1.41	1.00	0	1.12
F	3.20	2.50	2.50	0.50	1.12	0

Minimum distance

Agglomerative Clustering

- Example: Inside step 3. Execute step 4 of algorithm
  - Merge clusters D and F to create one new cluster (D, F)

	A	B	C	(D,F)	E
A	0	0.71	5.66	???	4.24
B	0.71	0	4.95	???	3.54
C	5.66	4.95	0	???	1.41
(D,F)	???	???	???	0	???
E	4.24	3.54	1.41	???	0

Unknown distances. Need to calculate them

Agglomerative Clustering

- Example: Inside step 3. Execute step 5 of algorithm
  - Update distance matrix with new distances
    - **Using the MIN method! Look up initial distance matrix**

	A	**B**	**C**	**(D,F)**	**E**
A	0	0.71	5.66	**3.20**	4.24
B	0.71	0	4.95	**2.50**	3.54
C	5.66	4.95	0	**2.24**	1.41
(D,F)	**3.20**	**2.50**	**2.24**	0	**1.00**
E	4.24	3.54	1.41	**1.00**	0

d(DF, A) = min(d(D,A), d(F,A)) = min(3.61, 3.20) = 3.20
d(DF, B) = min(d(D,B), d(F,B)) = min(2.92, 2.50) = 2.50
d(DF, C) = min(d(D,C), d(F,C)) = min(2.24, 2.50) = 2.24
d(E, DF) = min(d(E,D) , d(E,F)) = min(1.00, 1.12) = 1.00

Calculated using MIN method. Look up distance of every combination of points from the initial distance matrix

Agglomerative Clustering

- Example: Inside step 3. Execute step 6 of algorithm
  - Do we have one single cluster? No! We have 5. Hence continue

	A	**B**	**C**	**(D,F)**	**E**
A	0	0.71	5.66	**3.20**	4.24
B	0.71	0	4.95	**2.50**	3.54
C	5.66	4.95	0	**2.24**	1.41
(D,F)	**3.20**	**2.50**	**2.24**	0	**1.00**
E	4.24	3.54	1.41	**1.00**	0

Agglomerative Clustering

- Example: Inside step 3. Execute step 4 of algorithm
  - Find in distance matrix clusters with minimum distance. Here A,B

	A	**B**	**C**	**(D,F)**	**E**
A	0	0.71	5.66	**3.20**	4.24
B	0.71	0	4.95	**2.50**	3.54
C	5.66	4.95	0	**2.24**	1.41
(D,F)	**3.20**	**2.50**	**2.24**	0	**1.00**
E	4.24	3.54	1.41	**1.00**	0

Agglomerative Clustering

- Example: Inside step 3. Execute step 4 of algorithm
  - Merge clusters A and B to create one new cluster (A, B)

	(A,B)	C	(D,F)	E
(A,B)	0	???	???	???
C	???	0	2.24	1.41
(D,F)	???	2.24	0	1.0
E	???	1.41	1.00	0

Agglomerative Clustering

- Example: Inside step 3. Execute step 5 of algorithm
  - Update distance matrix with new distances

	(A,B)	**C**	**(D,F)**	**E**
(A,B)	0	**4.95**	**2.50**	**3.54**
C	**4.95**	0	2.24	1.41
(D,F)	**2.50**	2.24	0	1.0
E	**3.54**	1.41	1.00	0

d(C, AB) = min(d(C,A), d(C,B)) = min(5.66, 4.95) = 4.95
d(DF, AB) = min(d(D,A), d(D,B), d(FA), d(FB)) = min(3.61, 2.92, 3.20, 2.50) = 2.50
d(DF, C) = min(d(D,C), d(F,C)) = min(2.24, 2.50) = 2.24
d(E, AB) = min(d(E,A) , d(E,B)) = min(4.24, 3.54) = 3.54

Using MIN method. Look up distance of every combination of points from the initial distance matrix

Agglomerative Clustering

- Example: Inside step 3. Execute step 6 of algorithm
  - Do we have one single cluster? No! We have 4. Hence continue

	(A,B)	C	(D,F)	E
(A,B)	0	4.95	2.50	3.54
C	4.95	0	2.24	1.41
(D,F)	2.50	2.24	0	1.0
E	3.54	1.41	1.00	0

Agglomerative Clustering

- Example: Inside step 3. Execute step 4 of algorithm
  - Find in distance matrix clusters with minimum distance. Here (D,F) , E

	(A,B)	C	(D,F)	E
(A,B)	0	4.95	2.50	3.54
C	4.95	0	2.24	1.41
(D,F)	2.50	2.24	0	1.0
E	3.54	1.41	1.00	0

Agglomerative Clustering

- Example: Inside step 3. Execute step 4 of algorithm
  - Merge two cluster (D,F) and E (note: keep subclusters!)

	(A,B)	**C**	**((D,F), E)**
(A,B)	0	4.95	???
C	4.95	0	???
((D,F), E)	???	???	0

Agglomerative Clustering

- Example: Inside step 3. Execute step 5 of algorithm
  - Update distance matrix with new distances using MIN method

	(A,B)	C	((D,F), E)
(A,B)	0	4.95	**2.50**
C	4.95	0	**1.41**
((D,F), E)	**2.50**	**1.41**	0

d(AB, (DF)E) = min(d(A,D), d(A,F), d(A,E), d(B,D), d(B,F), d(B,E)) = min(3.61, 3.20, 4.24, 2.92, 2.50, 3.54) = 2.50
d((DF)E, C) = min(d(D,C), d(F,C), d(E,C)) = min(2.24, 2.50,1.41) = 1.41

Agglomerative Clustering

- Example: Inside step 3. Execute step 6 of algorithm
  - Do we have one single cluster? No! We have 3. Hence continue

	(A,B)	C	((D,F), E)
(A,B)	0	4.95	**2.50**
C	4.95	0	**1.41**
((D,F), E)	**2.50**	**1.41**	0

Agglomerative Clustering

- Example: Inside step 3. Execute step 4 of algorithm
  - Find in distance matrix clusters with minimum distance. Here ((D,F) , E) and C

	(A,B)	**C**	**((D,F), E)**
(A,B)	0	4.95	**2.50**
C	4.95	0	**1.41**
((D,F), E)	**2.50**	**1.41**	0

Agglomerative Clustering

- Example: Inside step 3. Execute step 4 of algorithm
  - Merge two cluster ((D,F) , E) and C (note: keep subclusters!)

	(A,B)	(((D,F), E), C)
(A,B)	0	???
(((D,F), E), C)	???	0

Agglomerative Clustering

- Example: Inside step 3. Execute step 5 of algorithm
  - Update distance matrix with new distances using MIN method

	(A,B)	(((D,F), E), C)
(A,B)	0	**2.50**
(((D,F), E), C)	**2.50**	0

d((((DF)E)C), (AB)) = min(d(D,A), d(D,B), d(F,A), d(F,B), d(E,A), d(E,B), d(C,A),d(C,B)) = min(3.61, 2.92, 3.20, 2.50, 4.24, 3.54, 5.66, 4.95) = 2.50

Agglomerative Clustering

- Example: Inside step 3. Execute step 6 of algorithm
  - Do we have one single cluster? No! We have 2. Hence continue

	(A,B)	(((D,F), E), C)
(A,B)	0	2.50
(((D,F), E), C)	2.50	0

Agglomerative Clustering

- Example: Inside step 3. Execute step 4 of algorithm
  - Find in distance matrix clusters with minimum distance. Note: Don't need to because only 2 clusters left. Simply merge them into a single one. **Algorithm terminates.**

	((((D,F), E), C), (A,B))
((((D,F), E), C), (A,B))	0

Important: Distance of clusters (((D,F), E), C) and (A,B) is 2.50 (see previous distance matrix)

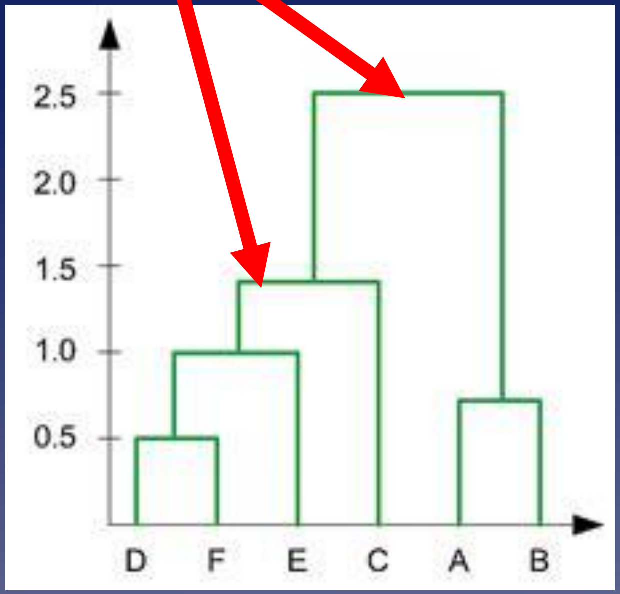
Agglomerative Clustering

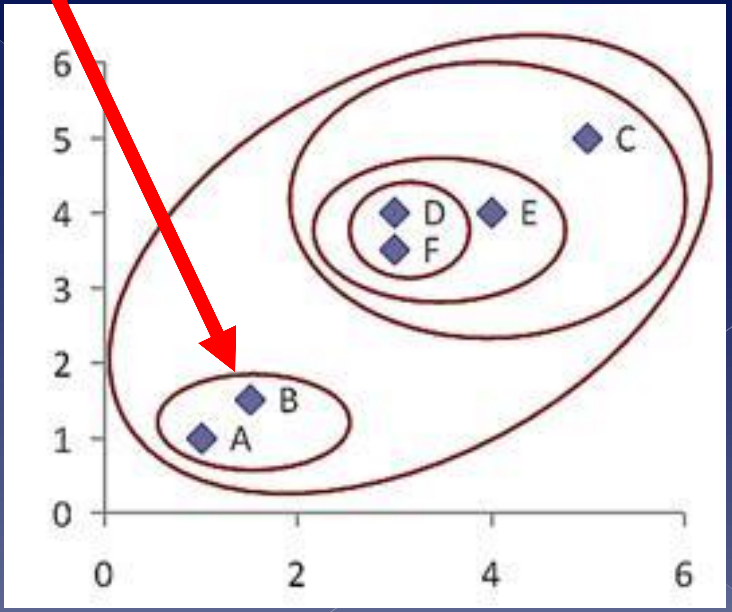
- Example: Based on distance matrix draw now dendrogram or Nested classes

Result of hierarchical clustering of dataset: ((((D,F), E), C), (A,B))
Note: Parentheses indicate subclusters

Again, height at which clusters merge in dendrogram is the clusters' distance!

This line indicates cluster (A,B)





Agglomerative Clustering

- Time and Space complexity
  - **$O(n^2)$ space complexity** since it uses the proximity matrix
    - n = number of points
  - **$O(n^3)$ time complexity** in many cases
    - n = number of points
    - 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 in some situations

Agglomerative Clustering

- **Problems and limitations** of Hierarchical Clustering
  - Once a **decision is made** to combine two clusters, it **cannot be undone**
  - **No objective function** is directly minimized
  - **Different schemes** (e.g. different distance measures) **have problems with one or more of the following**:
    - Sensitivity to noise and outliers
    - Difficulty handling different sized clusters and convex shapes
    - Breaking large clusters

Agglomerate clustering in R

Part 1/4

```
#### Agglomerate Hierarchical Clustering (file hierarchicalClustering.R) ####

#
# Read the file that contains taxpayers' data.
# IMPORTANT! Change path to file if it resides on a different folder on
# your machine.
#
taxpayers <- read.csv("taxpayers.csv")

#
# Take a quick look at some descriptive statistics of the data to see
# if our data looks fine for hierarchical clustering
#
summary(taxpayers)

# Something is not ok. Attributes/Variables have different scales. Since
# we will be using Euclidean distance in the distance matrix, this may
# introduce bias. Hence, try to normalize each value of attribute to an
# a scale from 0 to 1.

# We will use min-max normalization. It's easy ans works (for most cases).
# Define the function norm that will normalize a value using the min-max
# method
#
norm <- function(x){ return( (x-min(x)) / (max(x)-min(x)) ) }

# continued on next slide…
```

Agglomerate clustering in R

Part 2/4

```
#
# Pass now each attribute of the dataset through the norm function
#

# This will normalize attibute Income
taxpayers[,"Income"] <-norm(taxpayers$Income)
# This will normalize attibute Spending
taxpayers[,"Spending"] <-norm(taxpayers$Spending)
# This will normalize attibute YearsWorking
taxpayers[,"YearsWorking"] <-norm(taxpayers$YearsWorking)
# This will normalize attibute NumChildren
taxpayers[,"NumChildren"] <-norm(taxpayers$NumChildren)

#
#Take a look again. Are we ok?
#
summary(taxpayers)

#
# Hey nice! Seems we are ok. Data has been normalized.
#

# continued on next slide…
```

Agglomerate clustering in R

Part 3/4

```
#
# Now, calculate first the initial distance matrix for all data points,
# but remove attribute Name, which is the first attribute.
# Use the R function dist() to calculate the entire distance matrix based on the Euclidean
# distance. To tell R to take into consideration all attributes except
# the first one (which is the Name), we simply say taxpayers[-1] meaning "all except first".
#
distanceMatrix <- dist(taxpayers[-1])

#
# Distance matrix calculated. We can now proceed to execute
# Agglomerate hierarchical clustering using the hclust function
#
#
# The hclust() function executes hierarchical clustering.
# hclust() takes a shitload of arguments, but the important ones
# are two: 1) the distance matrix and 2) the distance measure for clusters
# First argument of hclust is the distanceMatrix that has been calculated previously.
# If no argument for the distance measure of clusters is given,
# the "Complete Linkage" measure is assumed (i.e. the default).
# If you want to use a different method, e.g. MIN, provide argument method="single"
# See help (?hclust) for more options
taxpayersHClustering <-hclust(distanceMatrix)

# continued on next slide…
```

Agglomerate clustering in R

Part 4/4

```
#
# Hierarchical clustering finished. Plot the dendrogram using the
# plot() function. Second parameter labels= tells R to display labels
# (in our case the Names) on the horizontal axis.
#
plot(taxpayersHClustering, labels=taxpayers$Name)

#
# You can also get more fancy and add rectangles identifying more clearly
# the clusters like so
# Argument 8 tells rect.hclust() how many clusters to wrap in rectangles or
# equivalently at which height of the dendrogram to indicate clusters
rect.hclust(taxpayersHClustering, 8)
```


"This is Ripley, last survivor of the Nostromo, signing off."