



# Информациски системи и големи податоци

Unsupervised Learning

- Clustering

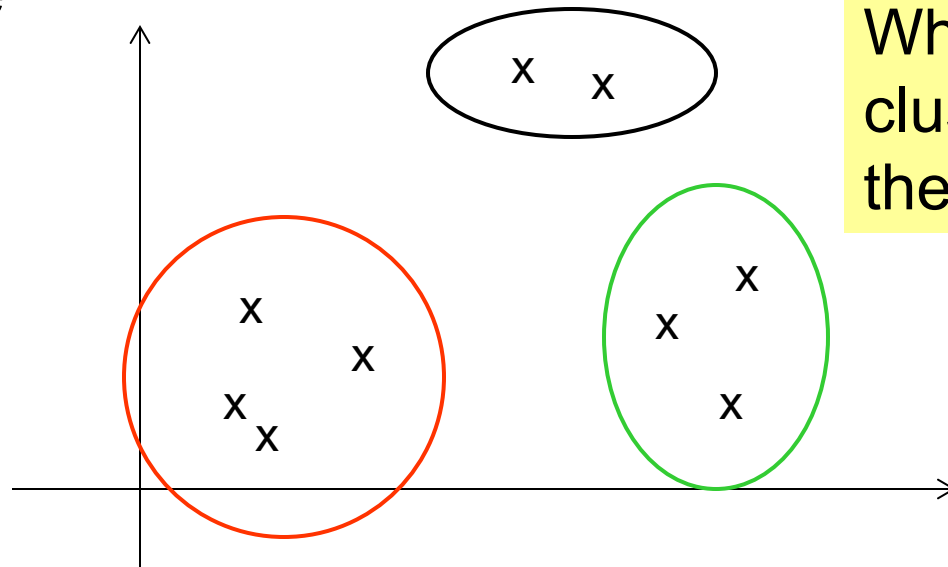


# Overview

- Motivation – why do clustering?
- Similarity measures
- Clustering algorithms
  - Hierarchical agglomerative clustering
  - K-means clustering and quality measures

# Introduction

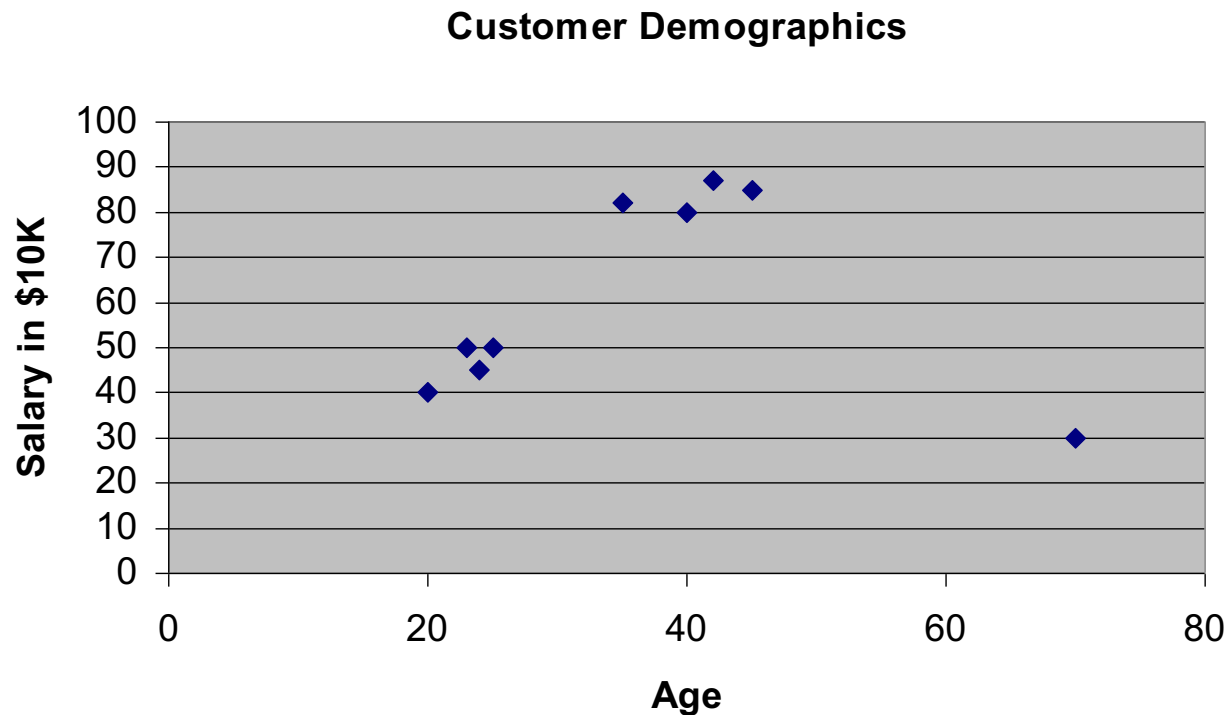
- The goal of clustering is to
  - group data points that are close (or **similar**) to each other
  - identify such groupings (or clusters) in an **unsupervised** manner
    - Unsupervised: no information is provided to the algorithm on which data points belong to which clusters
- Example



What should the clusters be for these data points?

# Clustering (Contd.)

- Example input database: Two numerical variables
- How many groups are here?



Age	Salary
20	40
25	50
24	45
23	50
40	80
45	85
42	87
35	82
70	30

# Clustering

- **Output:** (k) **groups** of records called **clusters**, such that the records within a group are more similar to records in other groups
  - Representative points for each cluster
  - Labeling of each record with each cluster number
  - Other description of each cluster
- *This is unsupervised learning:* No record labels are given to learn from
- Usage:
  - Exploratory data mining
  - Preprocessing step (e.g., outlier detection)

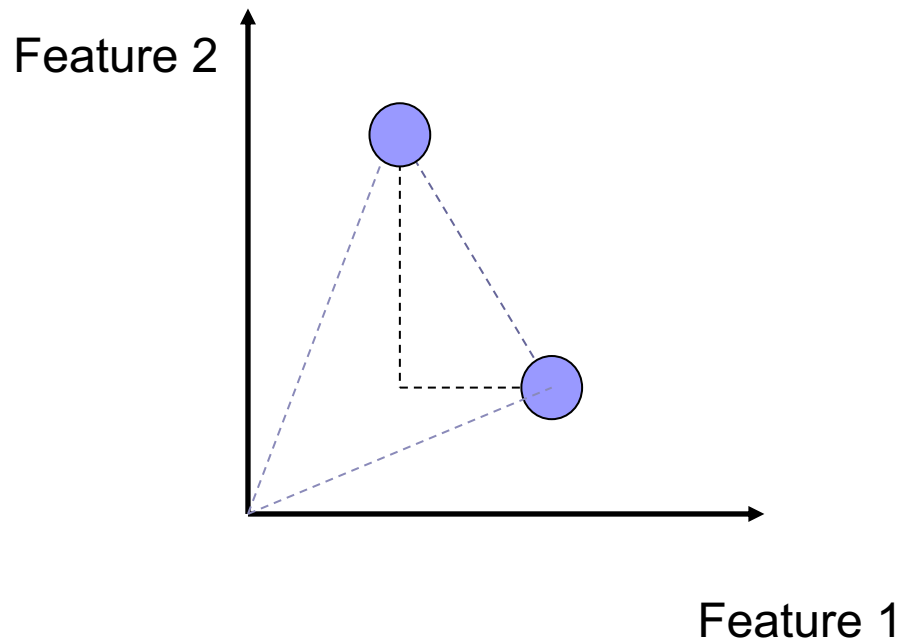


# What is clustering?

- A way of grouping together data samples that are ***similar*** in some way - according to some criteria that you pick
- A form of ***unsupervised learning*** – you generally don't have examples demonstrating how the data *should* be grouped together
- So, it's a method of ***data exploration*** – a way of looking for patterns or structure in the data that are of interest

# Similarity Measures

Euclidean  
Manhattan  
Cosine



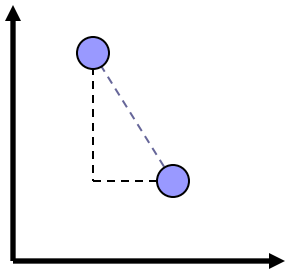


# How do we define “similarity”?

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



# Euclidean distance



$$d_{euc}(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$$

- Here  $n$  is the number of dimensions in the data vector



# Distance measure

- Euclidean distance

$$d(g_1, g_2) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$$

- Manhattan distance

$$d(g_1, g_2) = \sum_{i=1}^n |(x_i - y_i)|$$

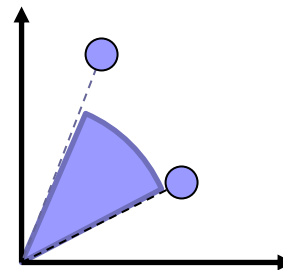
- Minkowski distance

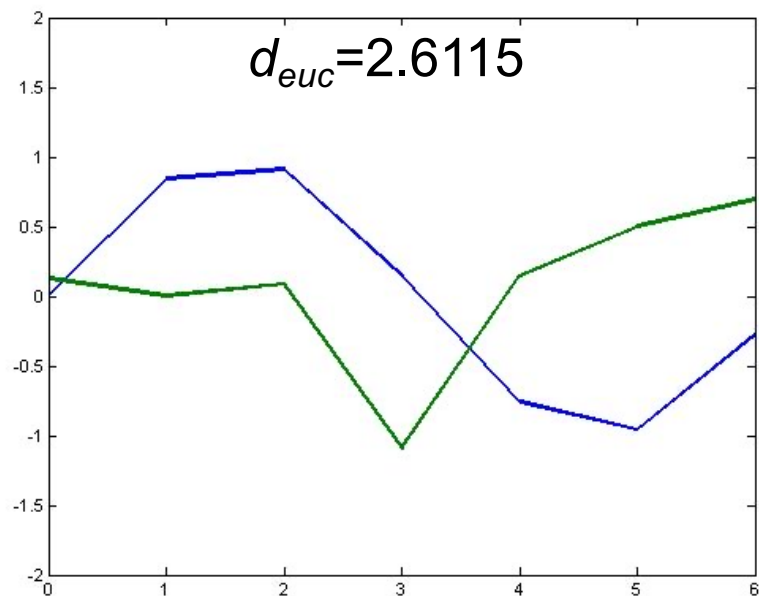
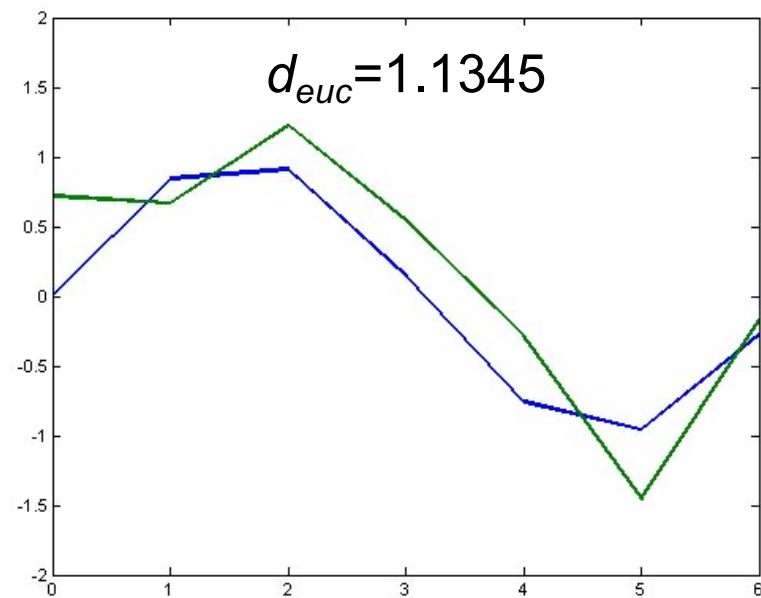
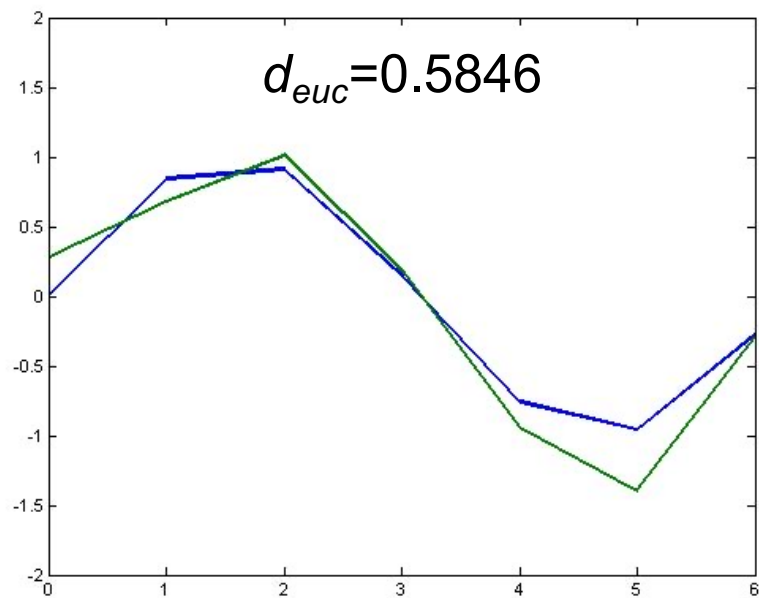
$$d(g_1, g_2) = \sqrt[m]{\sum_{i=1}^n (x_i - y_i)^m}$$

# Distance measure

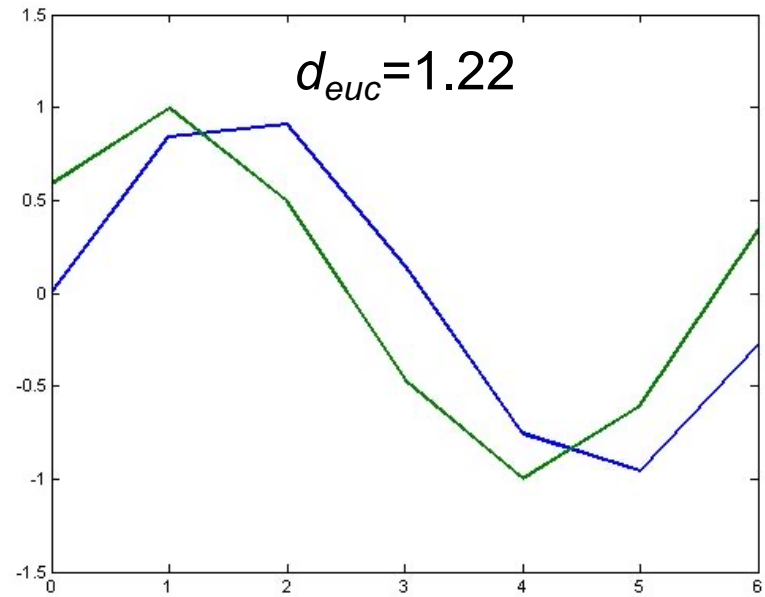
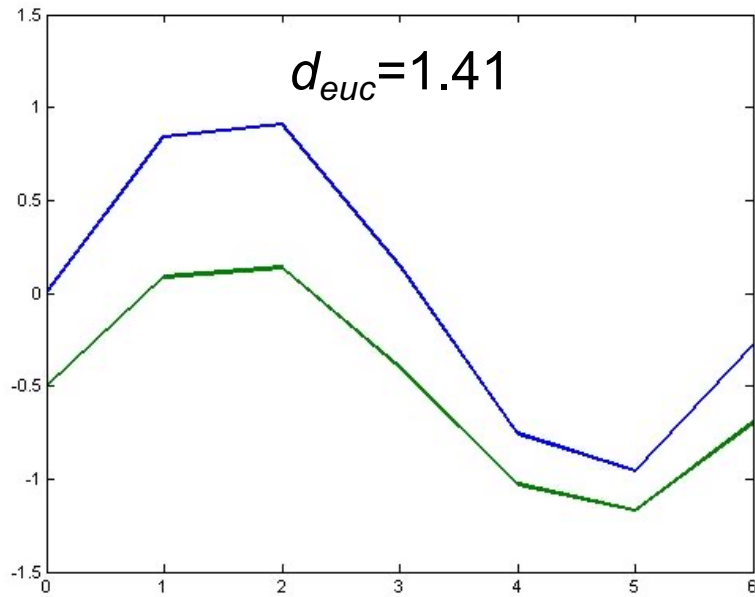
- Cosine similarity - angle

$$\frac{x \bullet y}{\sqrt{x \bullet x} \sqrt{y \bullet y}}$$





These examples of Euclidean distance match our intuition of dissimilarity pretty well...

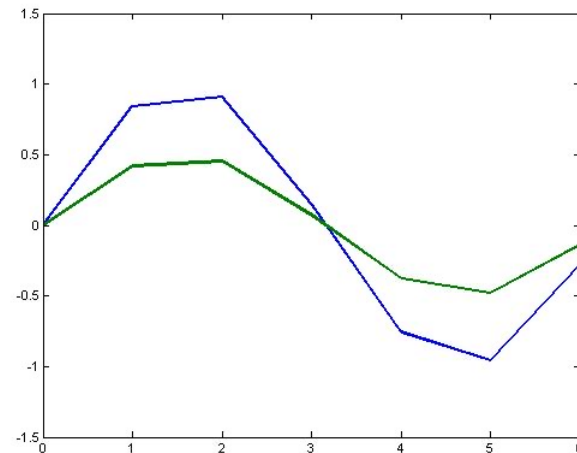
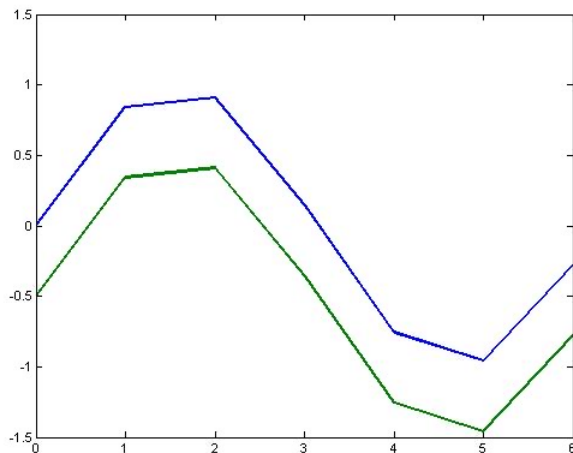


...But what about these?

What might be going on with the time series on the left? On the right?

# Correlation

- We might care more about the overall shape of time series rather than the actual magnitudes
- That is, we might want to consider time series similar when they are “up” and “down” together
- When might we want this kind of measure? What experimental issues might make this appropriate?



# Correlation distance

## ■ Correlation distance

$$r_{xy} = \frac{Cov(X, Y)}{\sqrt{Var(X) \cdot Var(Y)}}$$

- Cov(X, Y) stands for covariance of X and Y
  - degree to which two different variables are related
- Var(X) stands for variance of X
  - measurement of a sample differ from their mean

# Correlation distance

- Variance

$$Var(X) = \sqrt{\frac{\sum_{i=1}^n (x_i - \bar{X})^2}{n-1}}$$

- Covariance

$$CoVar(X, Y) = \frac{\sum_{i=1}^n (x_i - \bar{X})(y_i - \bar{Y})}{n-1}$$

- ☐ Positive covariance

- two variables vary in the same way

- ☐ Negative covariance

- one variable might increase when the other decreases

- ☐ Covariance is only suitable for heterogeneous pairs



# Correlation distance

## ■ Correlation

$$r_{xy} = \frac{Cov(X, Y)}{\sqrt{Var(X) \cdot Var(Y)}}$$

- maximum value of 1 if X and Y are perfectly correlated
- minimum value of -1 if X and Y are exactly opposite
- $d(X, Y) = 1 - r_{xy}$

<http://rpsychologist.com/d3/correlation/>



# Summary of similarity measures

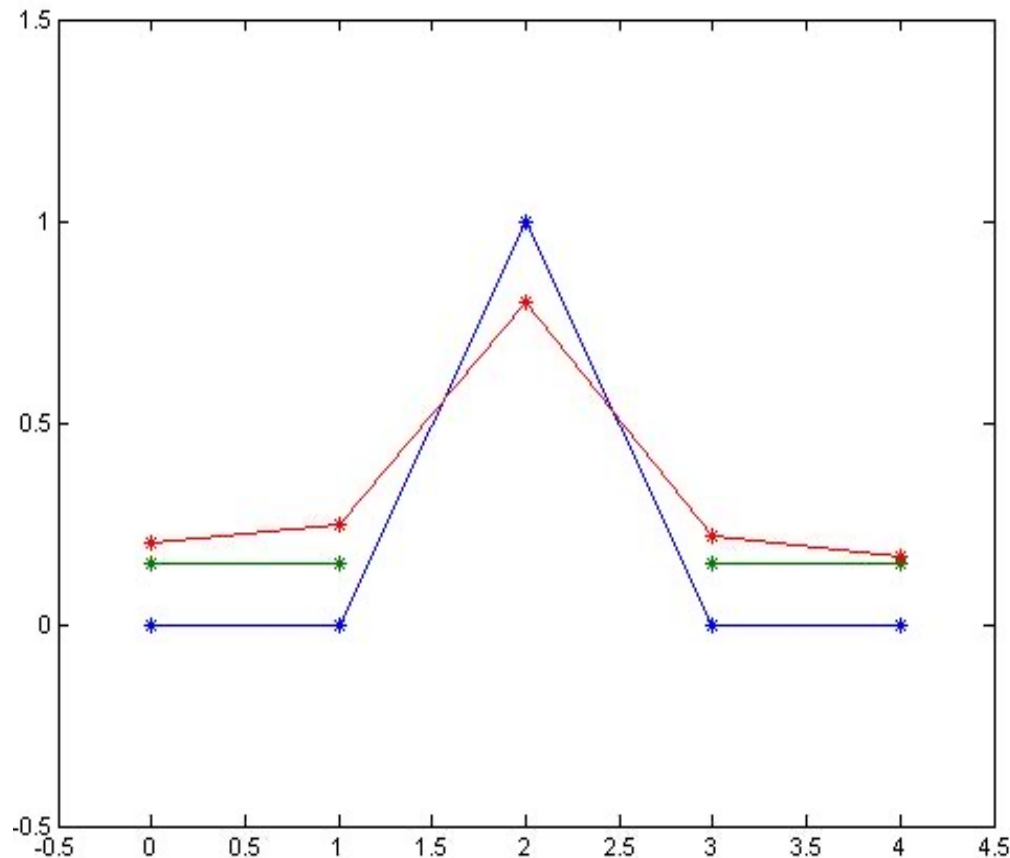
- Using different measures for clustering can yield different clusters
- Euclidean distance and correlation distance are the most common choices of similarity measure for microarray data
- Euclidean vs Correlation Example
  - $g1 = (1, 2, 3, 4, 5)$
  - $g2 = (100, 200, 300, 400, 500)$
  - $g3 = (5, 4, 3, 2, 1)$
  - Which time series are similar according to the two different measures?



# Missing Values

- A common problem w/ microarray data
- One approach with Euclidean distance or PLC is just to ignore missing values (i.e., pretend the data has fewer dimensions)
- There are more sophisticated approaches that use information such as continuity of a time series or related genes to estimate missing values – better to use these if possible

# Missing Values (cont.)



The green profile is missing the point in the middle

If we just ignore the missing point, the green and blue profiles will be perfectly correlated (also smaller Euclidean distance than between the red and blue profiles)




# Clustering Algorithms

# Approaches

- **Hierarchical:** Assume there is one cluster per point, and repeatedly merge nearby clusters using some distance threshold
- **Centroid-based:** Assume we have  $k$  clusters, guess at the centers, assign points to nearest center, e.g., **K-means**;



# Hierarchical

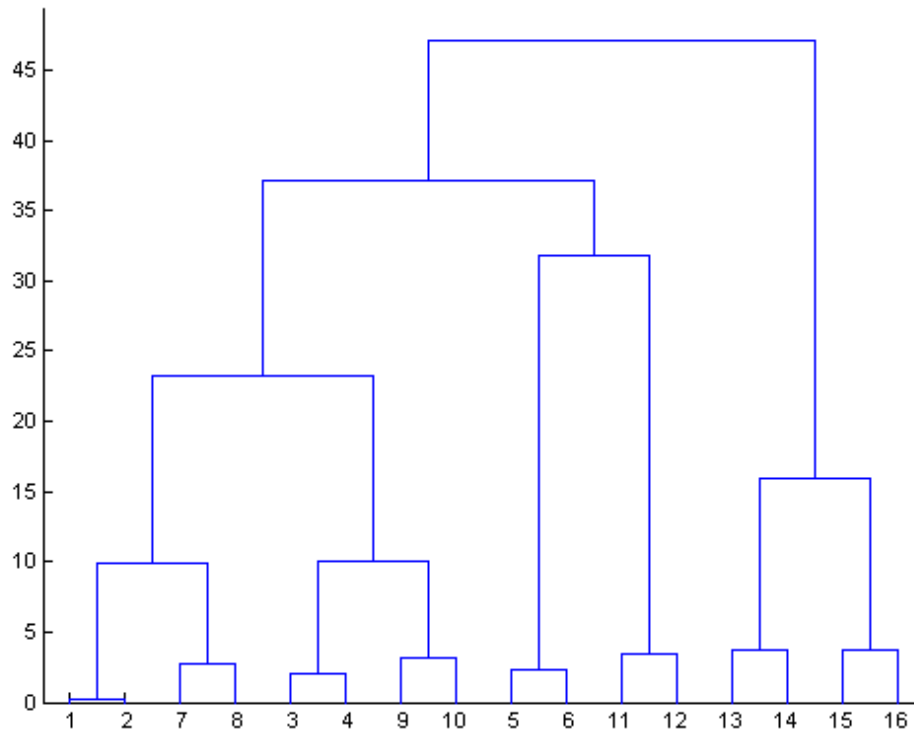


# Hierarchical Agglomerative Clustering

- We start with every data point in a separate cluster
- We keep merging the most similar pairs of data points/clusters until we have one big cluster left
- This is called a bottom-up or agglomerative method



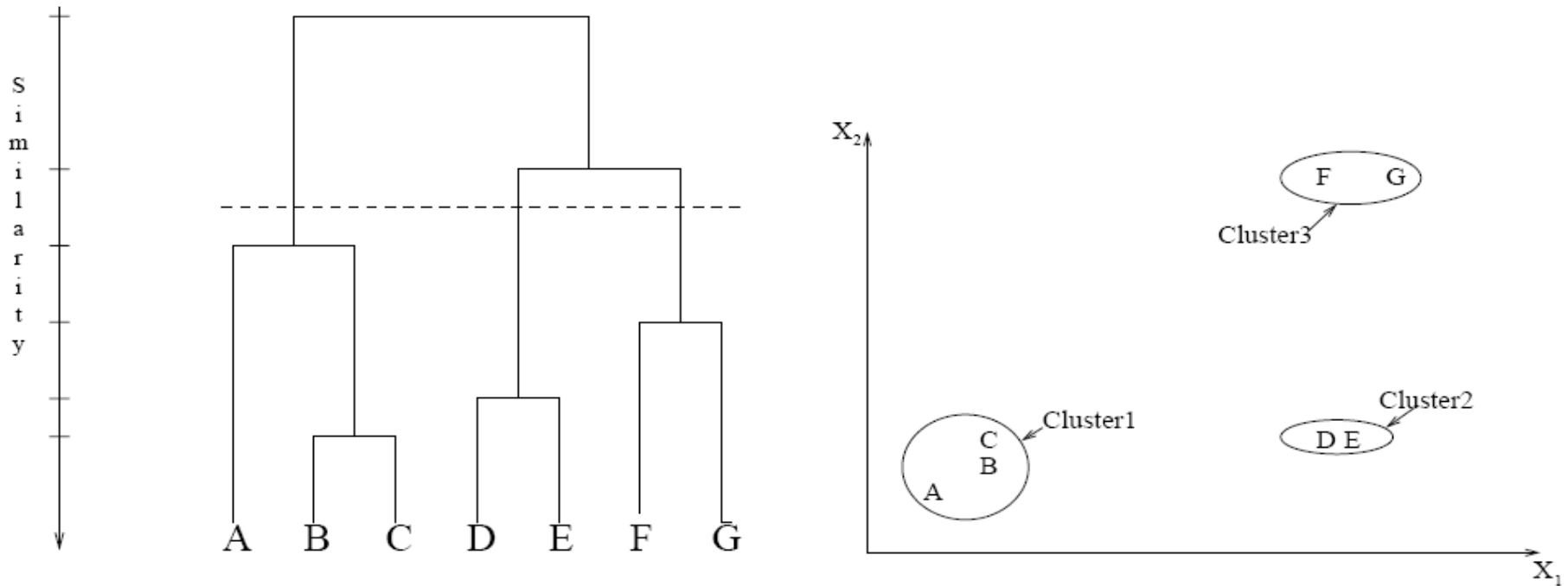
# Hierarchical Clustering (cont.)



- This produces a binary tree or ***dendrogram***
- The final cluster is the root and each data item is a leaf
- The height of the bars indicate how close the items are

# Hierarchical clustering: forming clusters

## ■ Forming clusters from dendograms





# Hierarchical clustering

- There are two styles of hierarchical clustering algorithms to build a tree from the input set  $S$ :
  - **Agglomerative (bottom-up):**
    - Beginning with singletons (sets with 1 element)
    - Merging them until  $S$  is achieved as the root.
    - It is the most common approach.
  - **Divisive (top-down):**
    - Recursively partitioning  $S$  until singleton sets are reached.



# Linkage in Hierarchical Clustering

- We already know about distance measures between data items, but what about between a data item and a cluster or between two clusters?
- We just treat a data point as a cluster with a single item, so our only problem is to define a ***linkage*** method between clusters
- As usual, there are lots of choices...



# Centroid Linkage & Average Linkage

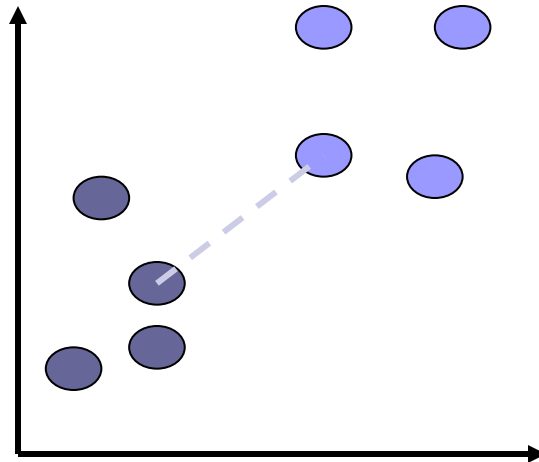
## ■ Centroid Linkage

- Each cluster  $c_i$  is associated with a mean vector  $\mu_i$  which is the mean of all the data items in the cluster
- The distance between two clusters  $c_i$  and  $c_j$  is then just  $d(\mu_i, \mu_j)$

## ■ Average linkage is defined as the average of all pairwise distances between points in the two clusters

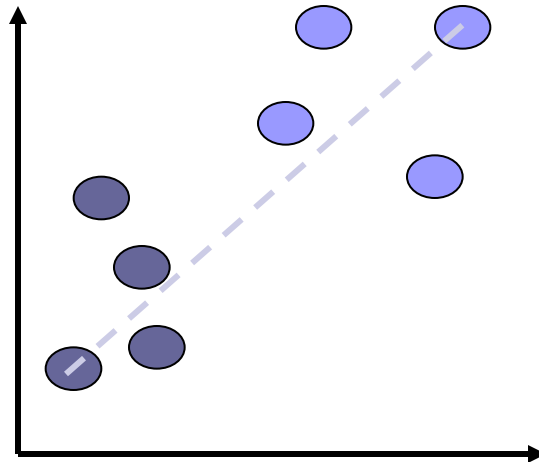
# Single Linkage

- The minimum of all pairwise distances between points in the two clusters
- Tends to produce long, “loose” clusters



# Complete Linkage

- The maximum of all pairwise distances between points in the two clusters
- Tends to produce very tight clusters





# Hierarchical Clustering Issues

- Distinct clusters are not produced – sometimes this can be good, if the data has a hierarchical structure w/o clear boundaries
- There are methods for producing distinct clusters, but these usually involve specifying somewhat arbitrary cutoff values
- What if data doesn't have a hierarchical structure? Is HC appropriate?





# Hierarchical clustering

## ■ Advantages

- ☐ Dendograms are great for visualization
- ☐ Provides hierarchical relations between clusters
- ☐ Shown to be able to capture concentric clusters

## ■ Disadvantages

- ☐ Not easy to define levels for clusters
- ☐ Experiments showed that other clustering techniques outperform hierarchical clustering



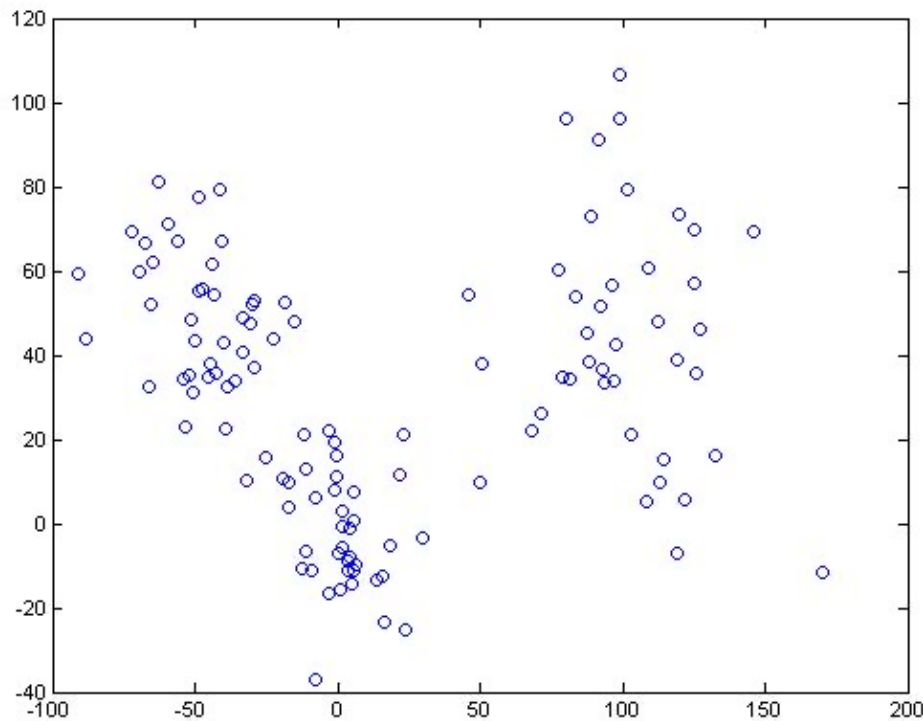
# K-means



# K-means Clustering

- Choose a number of clusters  $k$
- Initialize cluster centers  $\mu_1, \dots, \mu_k$ 
  - Could pick  $k$  data points and set cluster centers to these points
  - Or could randomly assign points to clusters and take means of clusters
- For each data point, compute the cluster center it is closest to (using some distance measure) and assign the data point to this cluster
- Re-compute cluster centers (mean of data points in cluster)
- Stop when there are no new re-assignments

# K-means Clustering (cont.)



How many clusters do you think there are in this data? How might it have been generated?



# K-means Clustering Demo

<https://www.naftaliharris.com/blog/visualizing-k-means-clustering/>



# K-means Clustering Issues

- Random initialization means that you may get different clusters each time
- Data points are assigned to only one cluster (hard assignment)
- You have to pick the number of clusters...

# K-means

- Stopping criteria:

- ☐ No change in the members of all clusters
- ☐ when the squared error is less than some small threshold value  $\alpha$

- Squared error  $se$

$$se = \sum_{i=1}^k \sum_{p \in c_i} \|p - m_i\|^2$$

- ☐ where  $m_i$  is the mean of all instances in cluster  $c_i$

- $se^{(j)} < \alpha$

- Properties of k-means

- ☐ Guaranteed to converge
- ☐ Guaranteed to achieve local optimal, not necessarily global optimal.

# K-means

## ■ Pros:

- Low complexity

- *complexity is  $O(nkt)$ , where  $t = \text{\#iterations}$*

## ■ Cons:

- Necessity of specifying  $k$

- Sensitive to noise and outlier data points

- Outliers: a small number of such data can substantially influence the mean value)

- Clusters are sensitive to initial assignment of centroids

- K-means is not a deterministic algorithm

- Clusters can be inconsistent from one run to another





# K-Means notes

- K-means uses **centroids**, average (mean) of samples in the cluster
- Instead we can use **medoid** – a representative object within the cluster
- Less sensitive to outliers



# Validity of Clusters



# Validity of clusters

- Why validity of clusters?
  - Given *some* data, any clustering algorithm generates clusters
  - So we need to make sure the clustering results are valid and meaningful.
- Measuring the validity of clustering results usually involve
  - Optimality of clusters
  - Verification of meaning of clusters

# Determining the “correct” number of clusters

- We'd like to have a measure of **cluster quality**  $Q$  and then try different values of  $k$  until we get an optimal value for  $Q$
- But, since clustering is an unsupervised learning method, we can't really expect to find a “correct” measure  $Q$ ...
- So, once again there are different choices of  $Q$  and our decision will depend on what dissimilarity measure we're using and what types of clusters we want

# Cluster Quality Measures

- Jagota (p.36) suggests a measure that emphasizes **cluster tightness** or **homogeneity**:

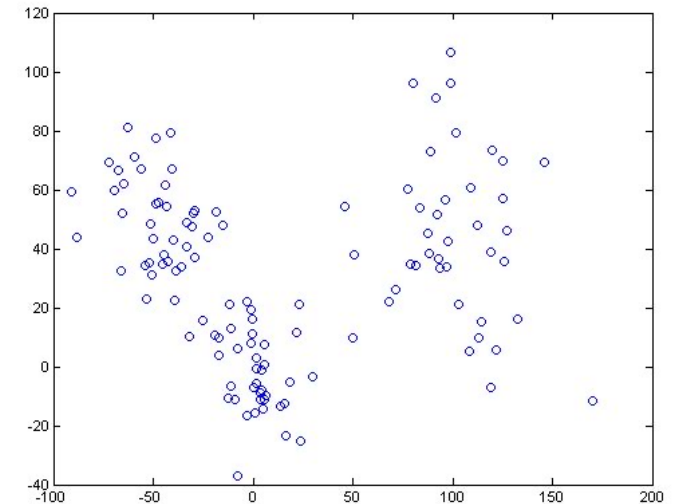
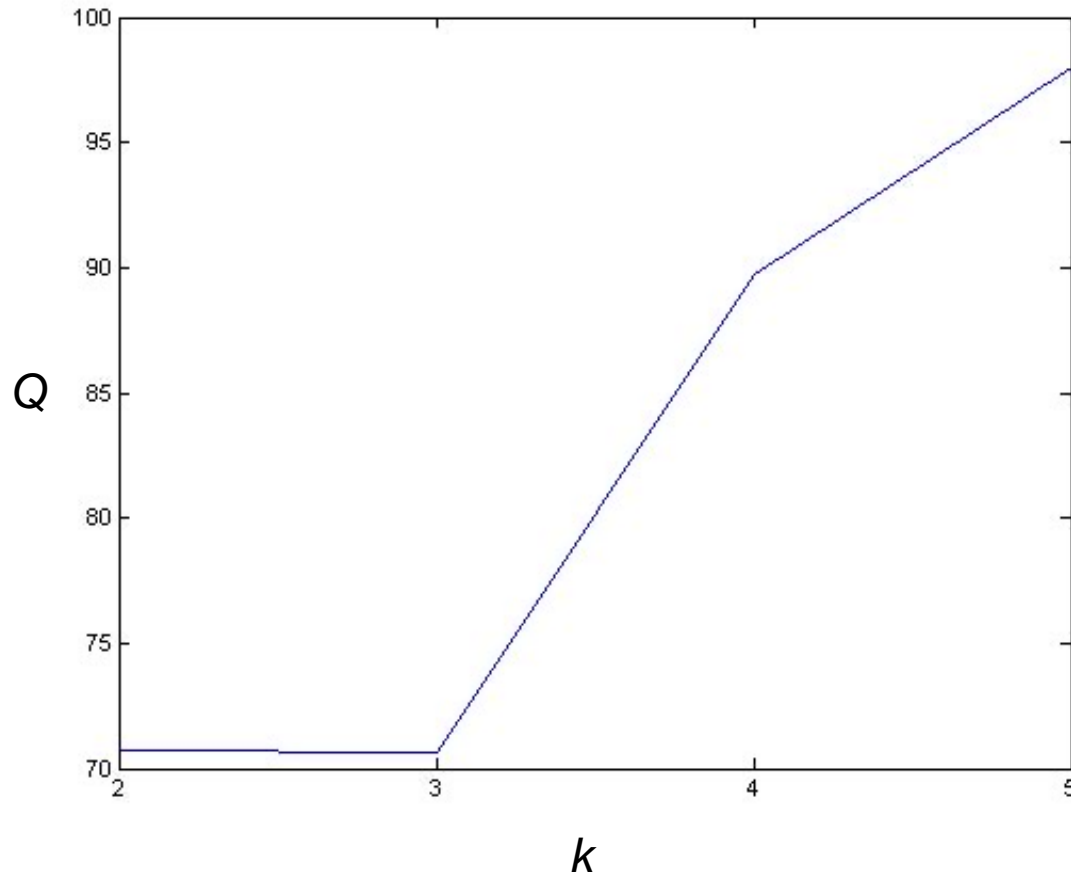
$$Q = \sum_{i=1}^k \frac{1}{|C_i|} \sum_{\mathbf{x} \in C_i} d(\mathbf{x}, \mu_i)$$

- $|C_i|$  is the number of data points in cluster  $i$
- $Q$  will be **small** if (on average) the data points in each cluster are close

# Cluster Quality (cont.)

This is a plot of the  $Q$  measure as given in Jagota for  $k$ -means clustering on the data shown earlier

How many clusters do you think there actually are?



# Cluster Quality (cont.)

- The Q measure given in Jagota takes into account homogeneity within clusters, but not separation between clusters
- Optimal clusters should
  - minimize distance **within** clusters (intracluster)
  - maximize distance **between** clusters (intercluster)
- Other measures try to combine these two characteristics (i.e., the Davies-Bouldin measure)
- An alternate approach is to look at cluster stability:
  - Add random noise to the data many times and count how many pairs of data points no longer cluster together
  - How much noise to add? Should reflect estimated variance in the data
- ScikitLearn: <https://scikit-learn.org/stable/modules/clustering.html#clustering-performance-evaluation>