Machine Learning and Data Mining
(IT4242E)

Quang Nhat NGUYEN

quang.nguyennhat@hust.edu.vn

Hanoi University of Science and Technology

School of Information and Communication Technology

Academic year 2021-2022

The course's content:

- Introduction
- Performance evaluation of the ML/DM system
- Regression problem
- Classification problem
- **Clustering problem**
  - **Evaluation metrics**
  - **Partition-based (k-means)**
  - **Hierarchical agglomerative clustering (HAC)**
- Association rule mining problem

Supervised vs. Unsupervised learning

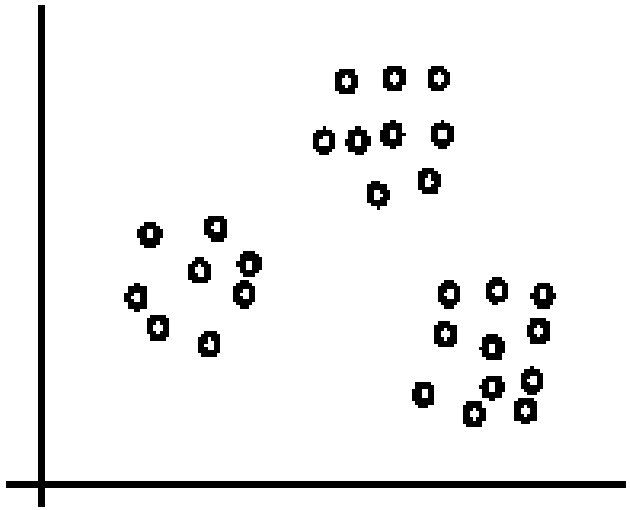
- **Supervised** learning
  - The training set is a set of examples, each associated **with a class/output value**
  - The goal is to learn (approximate) a hypothesis (e.g., a classification function, or a regression function) that fits the given **labelled** dataset
  - The learned hypothesis will then be used to classify/predict future (unseen) examples
- **Unsupervised** learning
  - The training set is a set of instances **with no class/output value**
  - The goal is to find some intrinsic groups/structures/relations

Clustering

- The most popular and important *unsupervised* learning method
  - There exist other unsupervised learning methods, such as collaborative filtering, association rules mining, etc.
- Clustering
  - Take as input an *unlabeled* dataset (i.e., a set of instances with no class/output value)
  - Group the instances in clusters
- A cluster is a set of instances that are
  - similar together (i.e., by some measure/meaning), and
  - dissimilar to the instances in other clusters

Clustering – Example

A clustering example, where the instances are grouped into three clusters



[Liu, 2006]

Clustering methods – Main components

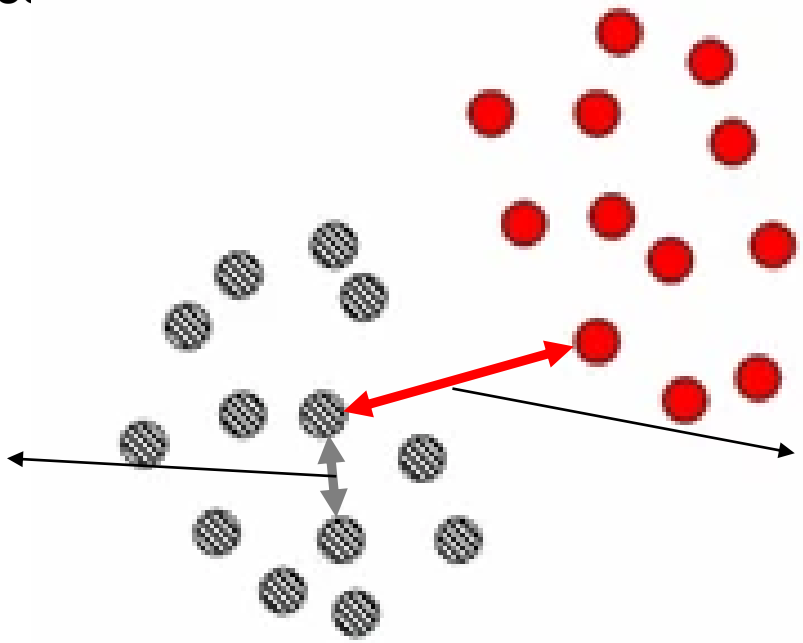
- A distance (or similarity, or dissimilarity) function
- A clustering algorithm
  - **Partition-based clustering**
  - **Hierarchical clustering**
  - Self-organizing map (SOM)
  - Mixture models
  - …
- Clustering quality measure
  - *Inter-cluster* distance/dissimilarity → To be maximized
  - *Intra-cluster* distance/dissimilarity → To be minimized

Clustering problem: Performance evaluation

- How to evaluate clustering efficiency?
  - *External evaluation*: Use additional external information (e.g., the class label of each example)
    - Example: Accuracy, Precision,…
  - *Internal evaluation*: Based on clustered examples only (without additional external information)
    - Very challenging!
    - Is the focus to be presented next

Internal evaluation: Principle

- **Compactness (coherence)**
  - Distance between examples in the same cluster (**intra-cluster distance**)
- **Separation**
  - The distance between the examples belongs to 2 different clusters (**inter-cluster distance**)



Distance between examples in the same cluster (**intra-cluster distance**)

The distance between the examples belongs to 2 different clusters (**inter-cluster distance**)

Internal evaluation: Metrics (1)

- **RMSSTD** (Root-mean-square standard deviation)
  - Evaluate the cohesion (compactness) of the obtained clusters
  - Expected that the **RMSSTD** value is *as small as possible*!

$$RMSSTD = \sqrt{\frac{\sum_{i=1}^{k} \sum_{x \in C_i} \|x - m_i\|^2}{P \sum_{i=1}^{k} (n_i - 1)}}$$

  - k: The number of clusters
  - C_i: Cluster i
  - m_i: The center (centroid) of cluster C_i
  - P: The number of dimensions (i.e., the number of attributes) used to represent examples
  - n_i: The number of examples in cluster C_i

Internal evaluation: Metrics (2)

- **R-squared**
  - Evaluate the separation between the obtained clusters
  - Expected that the **R-squared** value is *as large as possible*!

$$R\text{-}squared = \frac{\sum_{x \in D} \|x - g\|^2 - \sum_{i=1}^{k} \sum_{x \in C_i} \|x - m_i\|^2}{\sum_{x \in D} \|x - g\|^2}$$

  - k: The number of clusters
  - C_i: Cluster i
  - m_i: The center (centroid) of cluster C_i
  - D: The entire set of examples
  - g: The center (centroid) of the entire set of examples

Internal evaluation: Metrics (3)

- **Dunn index**
  - ~ (Separation/Compactness): The ratio between the minimum inter-cluster distance and the maximum intra-cluster distance
  - Expected that the **Dunn index** is *as large as possible*!

$$Dunn - index = \frac{min_{1 \le i < j \le k} inter - distance(i,j)}{max_{1 \le h \le k} intra - distance(h)}$$

  - *k*: The number of clusters
  - *inter-distance(i,j)*: The distance between the 2 clusters *i* and *j*
  - *intra-distance(h)*: The distance (dissimilarity) between the examples of cluster *h*

Internal evaluation: Metrics (4)

- **Davies-Bouldin index**
  - ~ (Compactness/Separation): The ratio of the average intra-cluster distance and the inter-cluster distance
  - Expected that the **Davies-Bouldin index** is *as small as possible*!

$$DB - index = \frac{1}{k}\sum_{i=1}^{k} max_{j \neq i} \frac{\frac{1}{n_i}\sum_{x \in C_i} d(x, m_i) + \frac{1}{n_j}\sum_{x \in C_j} d(x, m_j)}{d(m_i, m_j)}$$

  - k: The number of clusters
  - n_i, m_i: The number of examples and the centroid of cluster i
  - n_j, m_j: The number of examples and the centroid of cluster j
  - $d(m_i,m_j)$: The distance between the 2 cluster centroids m_i and m_j

k-means clustering

- The most popular method of partition-based clustering
- Let's call $D=\{x_1,x_2,\ldots,x_r\}$ the dataset
  - Where x_i is an instance (i.e., a vector in an *n*-dimensional vector space X)
- The k-means algorithm partitions the given dataset into k clusters
  - Each cluster has a cluster center, called **centroid**
  - k (i.e., the number of clusters) is pre-defined (i.e., decided by the system designer)

k-means algorithm – Main steps

Given a pre-defined value of k

- Step 1. Randomly choose k instances (i.e., **seeds**) to be the *initial centroids* (i.e., the *k* initial clusters)
- Step 2. For each instance, *assign it to the cluster* (among the k clusters) whose centroid is closest to the instance
- Step 3. For each cluster, *re-compute its centroid* based on the instances in that cluster
- Step 4. If the *convergence criterion* is satisfied, then stop; otherwise, go to Step 2

k-means(D, k)

D: The dataset

k: The number of clusters

Randomly select k instances in D as the initial centroids

while not CONVERGENCE

  for each instance x∈D

    Compute the distance from x to each centroid

    Assign x to the cluster whose centroid is closest to x

  end for

  for each cluster

    Re-compute its centroid based on its own instances

end while

return {The k clusters}

Convergence criterion

The clustering process stops if:

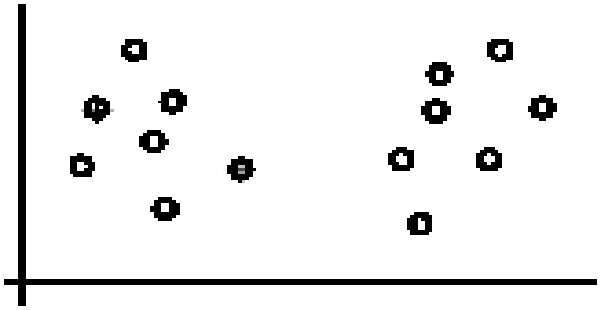
- no (or insignificant) re-assignment of instances to different clusters, or
- no (or insignificant) change of centroids, or
- insignificant decrease in the sum of squared error:

$$Error = \sum_{i=1}^{k} \sum_{\mathbf{x} \in C_i} d(\mathbf{x}, \mathbf{m_i})^2$$

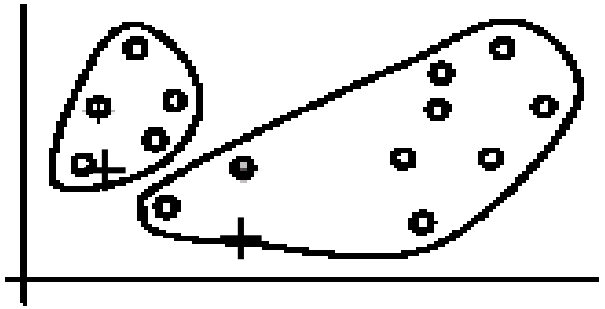
where

- C_i: The *i*-th cluster
- $\mathbf{m_i}$: The centroid of cluster C_i, and
- $d(\mathbf{x}, \mathbf{m_i})$: The distance between instance $\mathbf{x}$ and centroid $\mathbf{m_i}$

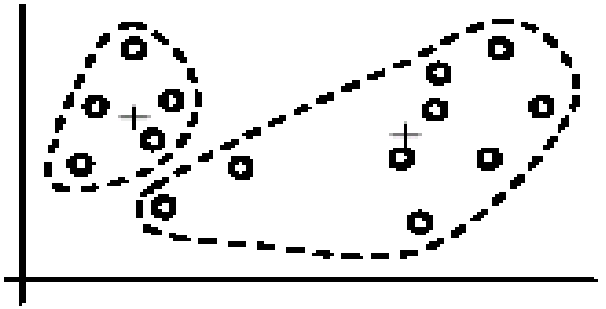
k-means algorithm – Illustration (1)



(A). Random selection of *k* centers



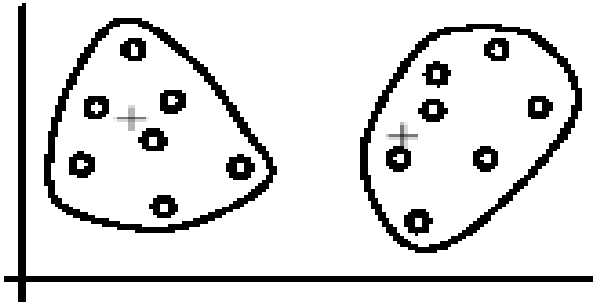
Iteration 1: (B). Cluster assignment



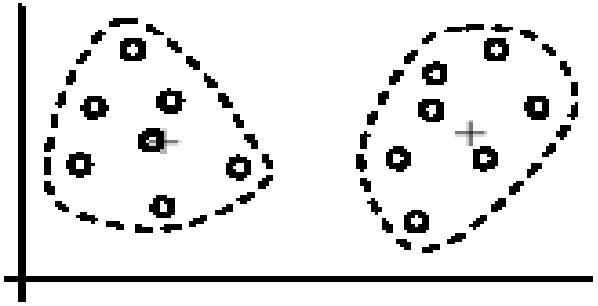
(C). Re-compute centroids

[Liu, 2006]

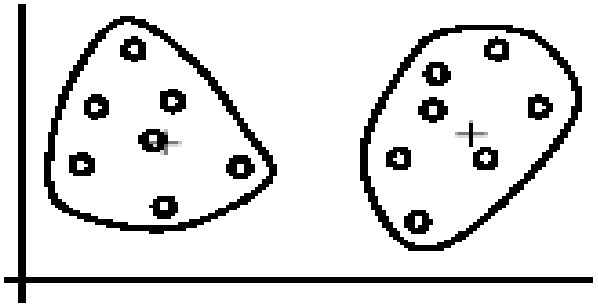
k-means algorithm – Illustration (2)



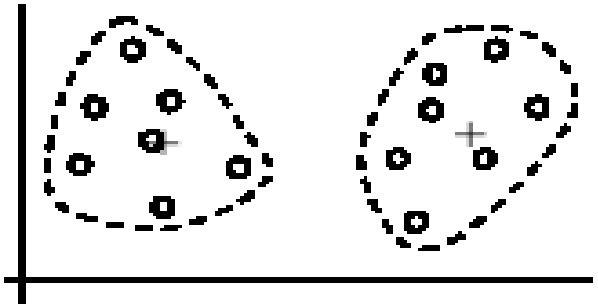
Iteration 2: (D). Cluster assignment



(E). Re-compute centroids



Iteration 3: (F). Cluster assignment



(G). Re-compute centroids

[Liu, 2006]

Centroid computation and Distance function

- Example of the centroid computation: *Mean centroid*

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

  - (vector) $\boldsymbol{m_i}$ is the centroid of cluster C_i
  - $|C_i|$ is the size of cluster C_i (i.e., the number of instances in C_i)

- Example of the distance function: *Euclidean distance*

$$d(\mathbf{x}, \mathbf{m_i}) = \|\mathbf{x} - \mathbf{m_i}\| = \sqrt{(x_1 - m_{i1})^2 + (x_2 - m_{i2})^2 + \ldots + (x_n - m_{in})^2}$$

  - (vector) $\boldsymbol{m_i}$ is the centroid of cluster C_i
  - $d(\boldsymbol{x}, \boldsymbol{m_i})$ is the distance between instance $\boldsymbol{x}$ and centroid $\boldsymbol{m_i}$

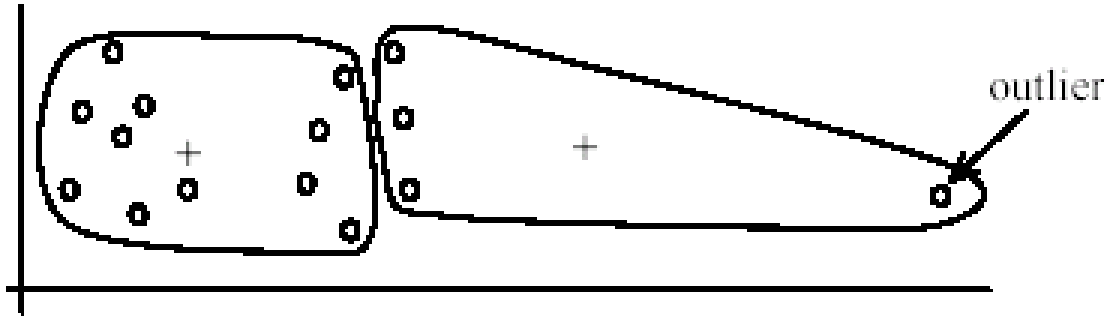
k-means algorithm – Strengths

- Simple
  - Easy to implement
  - Easy to understand
- Efficient
  - The time complexity ~ $O(rkt)$
    - r: The number of instances (i.e., the size of the dataset)
    - k: The number of clusters
    - t: The number of iterations
  - If both k and t are small, then k-means is considered as a linear algorithm
- k-means is the most popular clustering algorithm

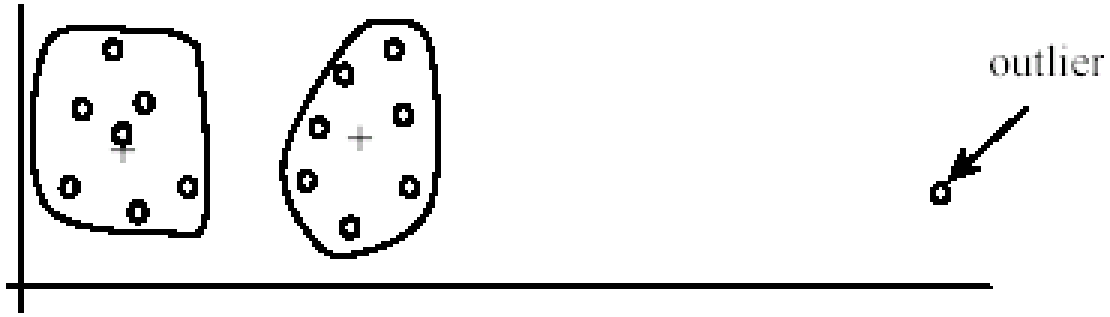
k-means algorithm – Weaknesses (1)

- The value of k (i.e., # of clusters) must be pre-defined
- The k-means algorithm needs the mean definition (in order to compute a cluster's centroid)
  - For nominal attributes, the centroid can be represented by the most frequent values of those attributes
- The k-means algorithm is sensitive to ***outliers***
  - Outliers are such instances that are (very) far away (dissimilar) from all the other instances
  - Outliers may be resulted by errors in the data recording/collection
  - Outliers may be special/abnormal instances with very different values

k-means algorithm – Outliers problem



(A): Undesirable clusters



(B): Ideal clusters

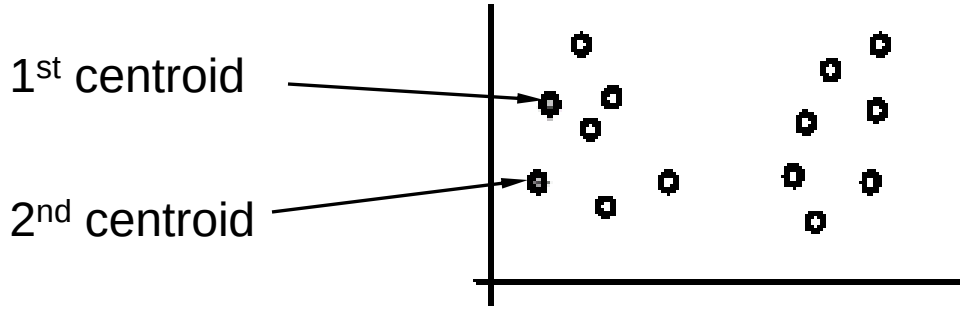
[Liu, 2006]

Solving the outliers problem

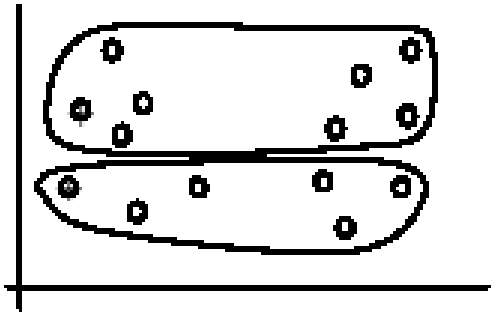
- Solution 1. To remove some instances in the clustering process that are much further away from the centroids than other instances
  - To be safe, track the outliers over a few (instead of only one) iterations
- Solution 2. To perform a random sampling
  - Since a sampling process selects only a small subset of the dataset, the chance of selecting an outlier is very small
  - Assign the rest of the dataset to the clusters by distance (or similarity) comparison

k-means algorithm – Weaknesses (2)

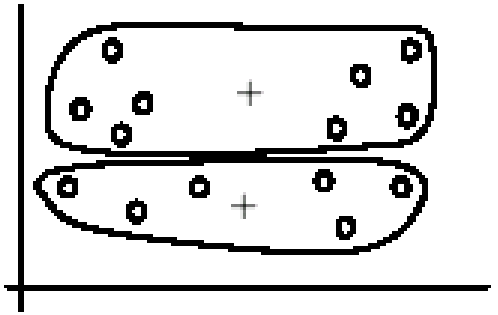
- The *k*-means algorithm is sensitive to the initial centroids



(A). Random selection of seeds (centroids)



(B). Iteration 1

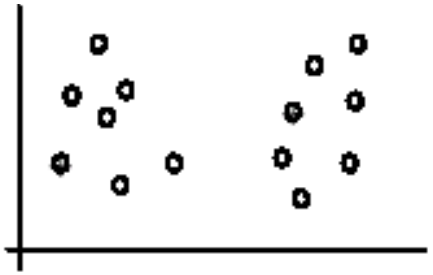


(C). Iteration 2

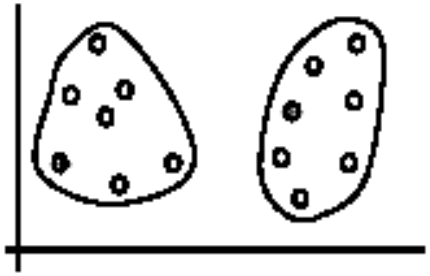
[Liu, 2006]

k-means algorithm – The initial seeds (1)

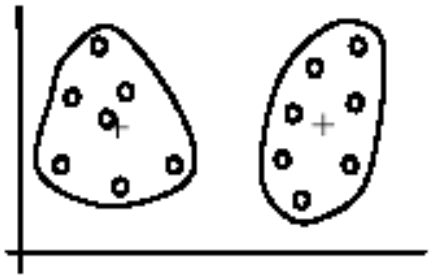
- To use different seeds → A better result!
  - Do many runs of k-means, each starting with different random initial seeds



(A). Random selection of k seeds (centroids)



(B). Iteration 1



(C). Iteration 2

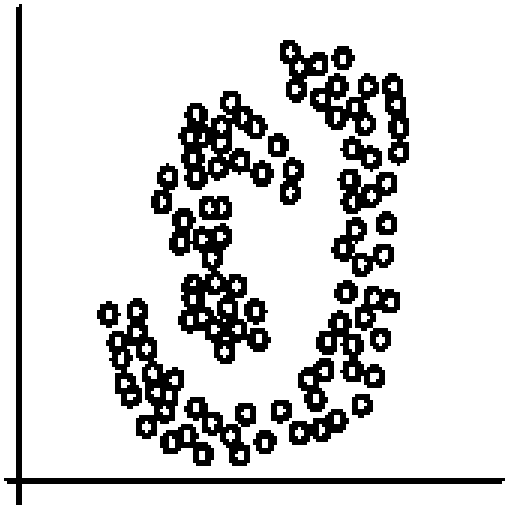
[Liu, 2006]

k-means algorithm – The initial seeds (2)

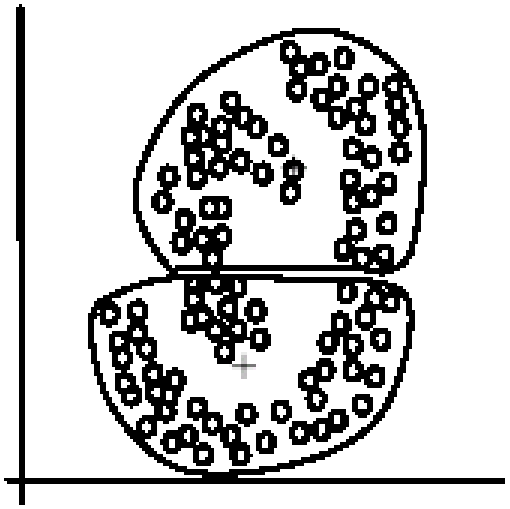
- Randomly select the first centroid ($\boldsymbol{m_1}$)
- Select a second centroid ($\boldsymbol{m_2}$) that is *as far away as possible* from the first one
- …
- Select the i-th centroid ($\boldsymbol{m_i}$) that is *as far away as possible* from the closest of $\{\boldsymbol{m_1}, \boldsymbol{m_2}, \ldots, \boldsymbol{m_{i-1}}\}$
- ...

k-means algorithm – Weaknesses (3)

- The k-means algorithm is not suitable for discovering clusters that are not hyper-ellipsoids (or hyper-spheres)



(A): Two natural clusters



(B): k-means clusters

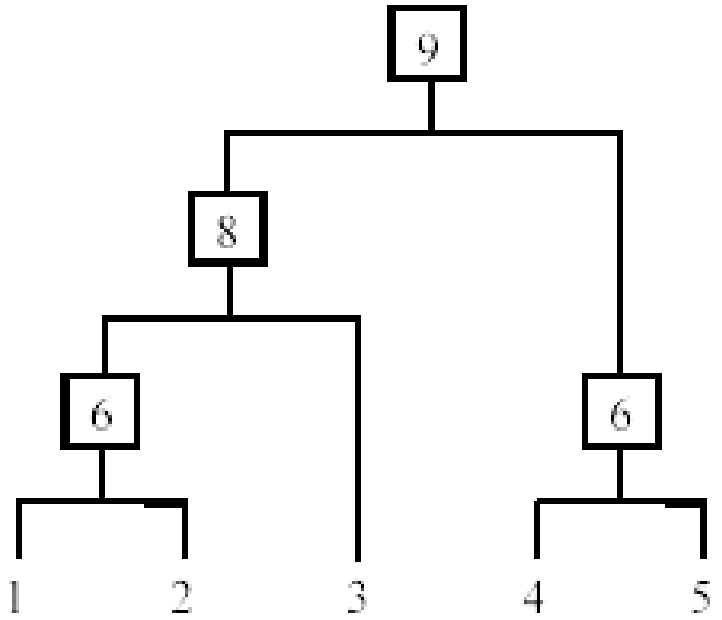
[Liu, 2006]

k-means algorithm – Summary

- Despite its weaknesses, k-means is still the most popular algorithm due to its simplicity and efficiency
  - Other clustering algorithms have also their own weaknesses
- No clear evidence that any other clustering algorithm performs better than k-means in general
  - Some clustering algorithms may be more suitable for some specific types of dataset, or for some specific application problems, than the others
- Comparing the performance of different clustering algorithms is a difficult task
  - No one knows the correct clusters!

Hierarchical agglomerative clustering (1)

- Produce a nested sequence of clusters - called **dendrogram**
  - Also called *taxonomy/hierarchy/tree* of instances

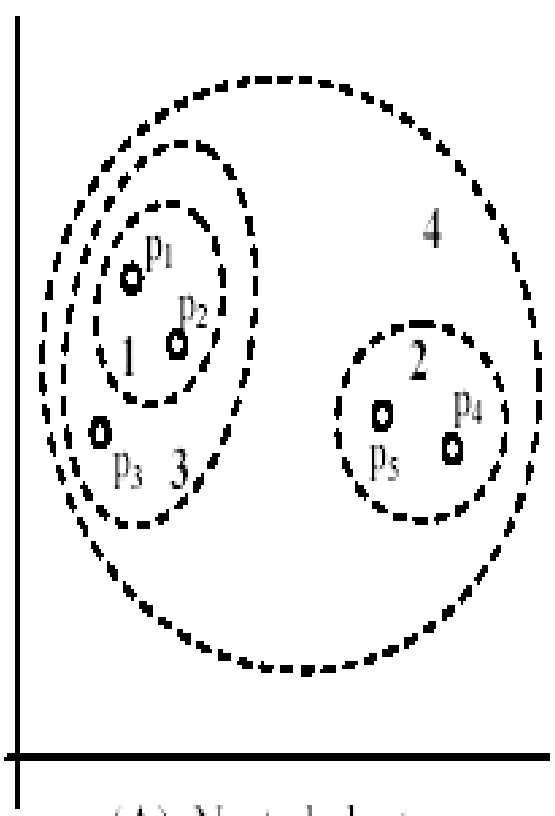


[Liu, 2006]

Hierarchical agglomerative clustering (2)

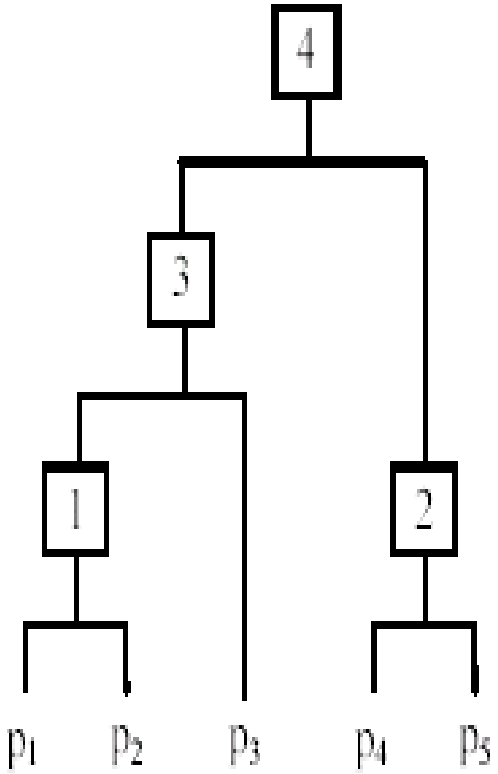
- Hierarchical agglomerative (bottom-up) clustering builds the dendrogram from the bottom level
- The algorithm:
  - At the beginning, each instance forms a cluster (also called a node)
  - Merge *the most similar* (nearest) pair of clusters
    - i.e., The pair of clusters that have *the least distance* among all the possible pairs
  - Continue the merging process
  - Stop when all the instances are merged into a single cluster (i.e., the *root* cluster)

HAC algorithm – Example



(A). Nested clusters

(Venn diagram)



(B) Dendrogram

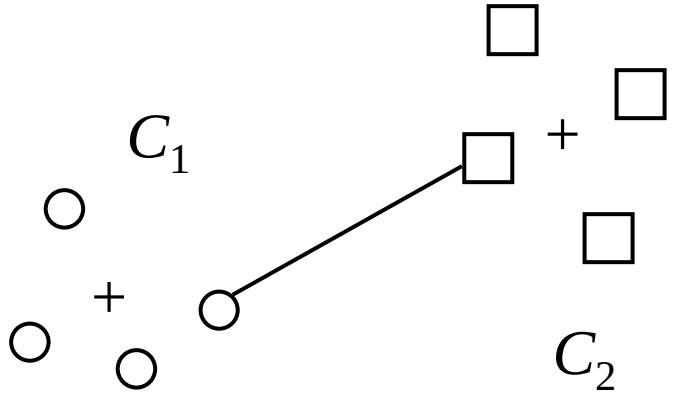
[Liu, 2006]

Distance of two clusters

- The HAC algorithm requires the computation of the distance between two clusters
  - Before the merging, for every possible pairs of clusters the distance between the two clusters is computed
- Different methods to measure the distances of two clusters (i.e., resulting in variations of the HAC algorithm)
  - Single link
  - Complete link
  - Average link
  - Centroid link
  - …

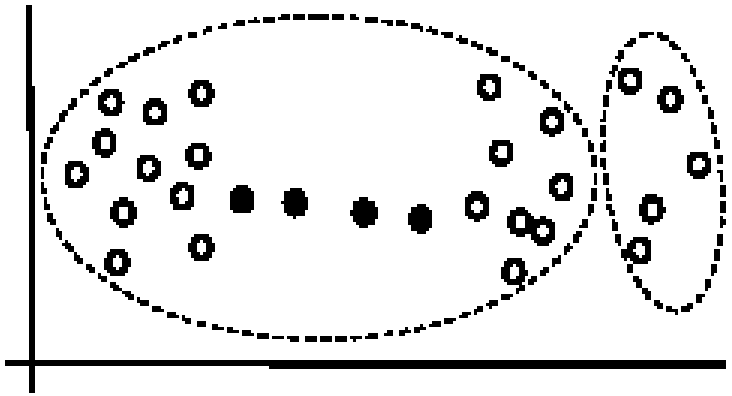
HAC – Single link

- The distance between two clusters is the **minimum distance** between the instances (members) of the two clusters



- Tend to generate "long chains"

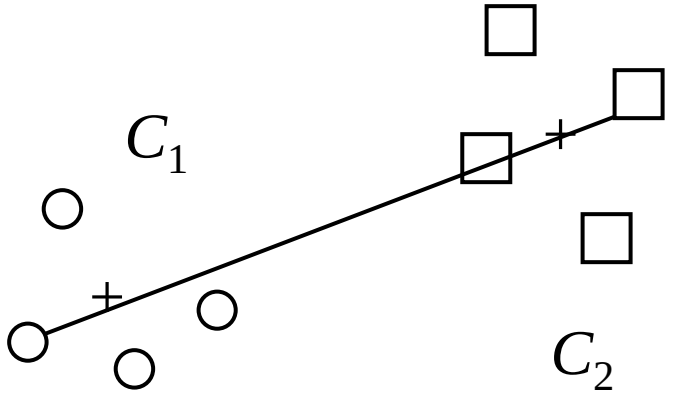
Two natural clusters are split into two



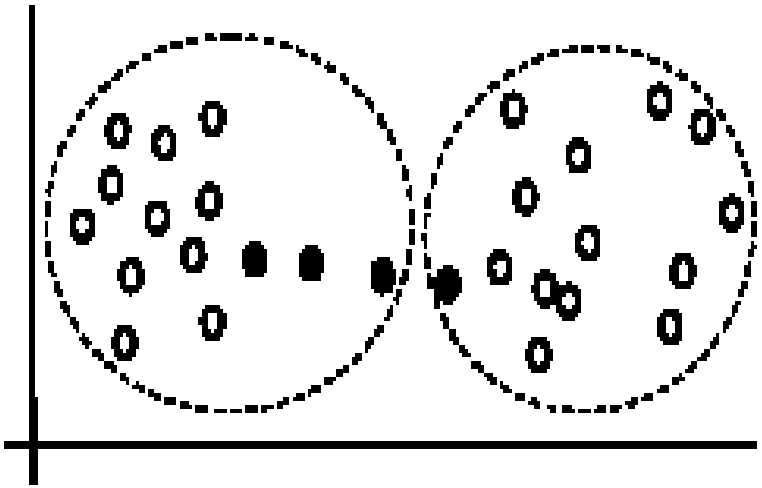
[Liu, 2006]

HAC – Complete link

- The distance between two clusters is the **maximum distance** between the instances (members) of the two clusters



- Sensitive to outliers



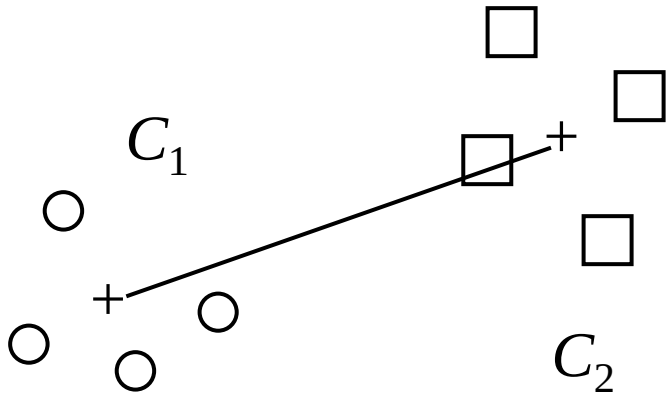
[Liu, 2006]

HAC – Average link

- Average-link distance is a compromise between complete-link and single-link distances
  - To reduce the sensitivity of complete-link clustering to outliers
  - To reduce the tendency of single-link clustering to form long chains (that do not correspond to the intuitive notion of clusters)
- The distance between two clusters is the average distance of all pairs of instances (one from each cluster)

HAC – Centroid link

- The distance between two clusters is the distance between their centroids



HAC algorithm – Complexity

- All the variations of the HAC algorithm have the complexity of at least $O(r^2)$
  - r: The number of instances (i.e., the size of the dataset)
- Single-link can be done in $O(r^2)$
- Complete-link and average-link can be done in $O(r^2 \log r)$
- Because of the complexity, the HAC algorithm is hard to use for large datasets

Clustering – Distance functions

- A key component to clustering
  - "similarity functions" and "dissimilarity functions" are also commonly used terms
- There are different distance functions for
  - Different types of data
    - Numeric data
    - Nominal data
  - Specific application problems

Distance functions for numeric attributes

- The family of geometry distance functions (Minkowski distance)
- Most commonly used functions
  - Euclidean distance and
  - Manhattan (a.k.a. city-block) distance
- Let's denote $d(\mathbf{x}_i, \mathbf{x}_j)$ the distance between the two instances (vectors) $\mathbf{x}_i$ and $\mathbf{x}_j$
- The general Minkowski distance (p is a positive integer)

$$d(\mathbf{x_i}, \mathbf{x_j}) = [(x_{i1} - x_{j1})^p + (x_{i2} - x_{j2})^p + ... + (x_{in} - x_{jn})^p]^{1/p}$$

Distance functions for binary attributes

- We use a confusion matrix to introduce the distance function
  - *a*: The number of attributes with value of 1 for both $\mathbf{x}_i$ and $\mathbf{x}_j$
  - *d*: The number of attributes with value of 0 for both $\mathbf{x}_i$ and $\mathbf{x}_j$
  - *b*: The number of attributes for which the value in $\mathbf{x}_i$ is 1 whereas the value in $\mathbf{x}_j$ is 0
  - *c*: The number of attributes for which the value in $\mathbf{x}_i$ is 0 whereas the value in $\mathbf{x}_j$ is 1

instance $\mathbf{x}_i$ \ instance $\mathbf{x}_j$	1	0
1	a	b
0	c	d

- **Simple matching coefficient**: The proportion of mismatches of the attribute values between the two examples $\mathbf{x}_i$ and $\mathbf{x}_j$

$$d(\mathbf{x}_i, \mathbf{x}_j) = \frac{b+c}{a+b+c+d}$$

Distance functions for nominal attributes

- The distance function is also based on the simple matching method
- Given two examples $\mathbf{x}_i$ and $\mathbf{x}_j$, let's denote p the number of attributes and q the number of attributes whose values are identical in $\mathbf{x}_i$ and $\mathbf{x}_j$

$$d(\mathbf{x_i}, \mathbf{x_j}) = \frac{p - q}{p}$$

References

- B. Liu. *Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data*. Springer, 2006.