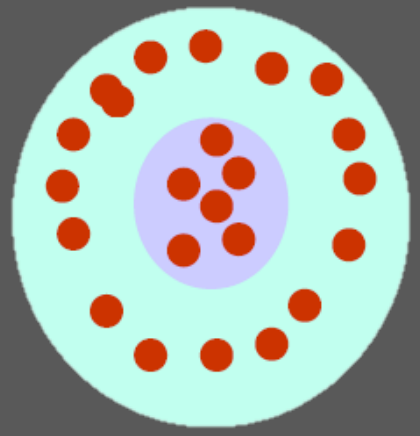

Data Mining
Cluster Analysis: Basic Concepts and Algorithms

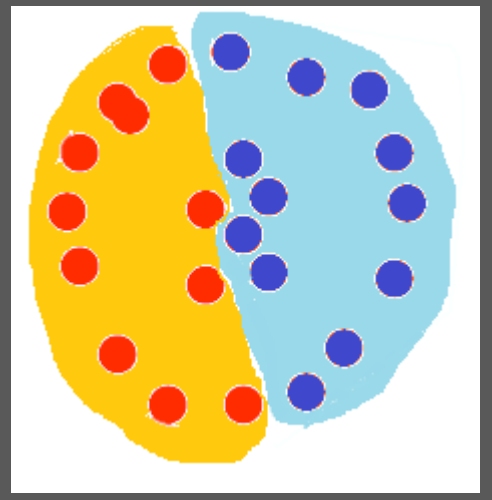
Supervised learning vs. unsupervised learning

- Supervised learning: discover patterns in the data that relate data attributes with a target (class) attribute.
  - These patterns are then utilized to predict the values of the target attribute in future data instances.
- Unsupervised learning: The data have no target attribute.
  - We want to explore the data to find some intrinsic structures in them.

What is clustering?

- The organization of unlabeled data into similarity groups called clusters.
- A cluster is a collection of data items which are “similar” between them, and “dissimilar” to data items in other clusters.





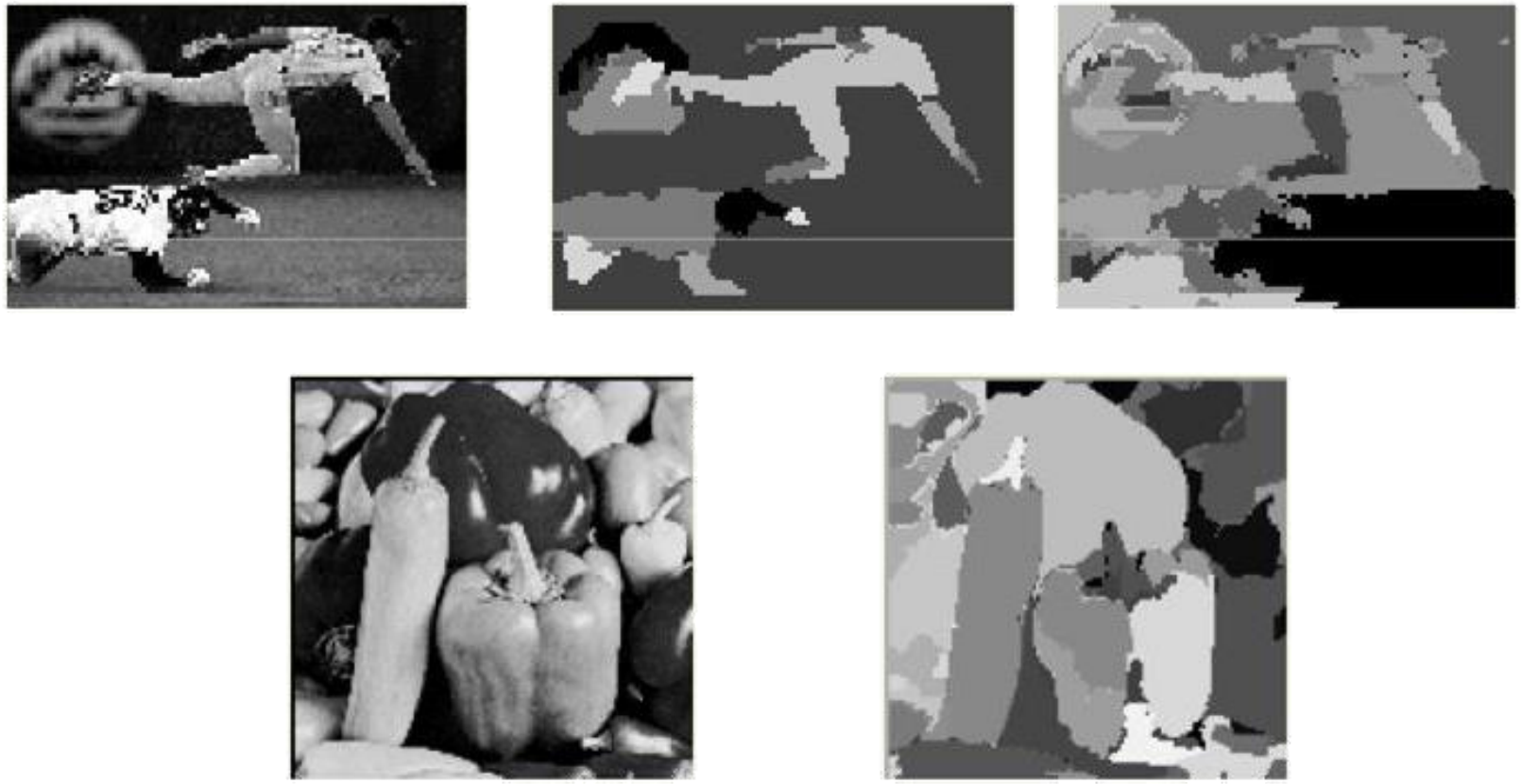
What is clustering for?

- Let us see some real-life examples
- Example 1: groups people of similar sizes together to make “small”, “medium” and “large” T-Shirts.
  - Tailor-made for each person: too expensive
  - One-size-fits-all: does not fit all.
- Example 2: In marketing, segment customers according to their similarities
  - To do targeted marketing.

What is clustering for? (cont…)

- Example 3: Given a collection of text documents, we want to organize them according to their content similarities,
  - To produce a topic hierarchy
- In fact, clustering is one of the most utilized data mining techniques.
  - It has a long history, and used in almost every field, e.g., medicine, psychology, botany, sociology, biology, archeology, marketing, insurance, libraries, etc.
  - In recent years, due to the rapid increase of online documents, text clustering becomes important.

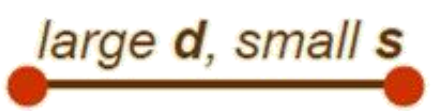
Computer vision application: Image segmentation

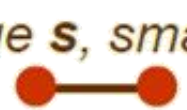


From: Image Segmentation by Nested Cuts, O. Veksler, CVPR2000

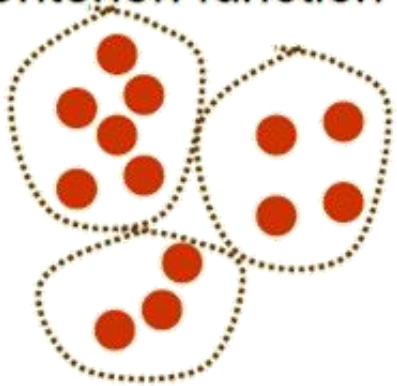
What do we need for clustering?

1. Proximity measure, *either*
   - similarity measure $s(x_i,x_k)$: large if x_i,x_k are similar
   - dissimilarity(or distance) measure $d(x_i,x_k)$: small if x_i,x_k are similar

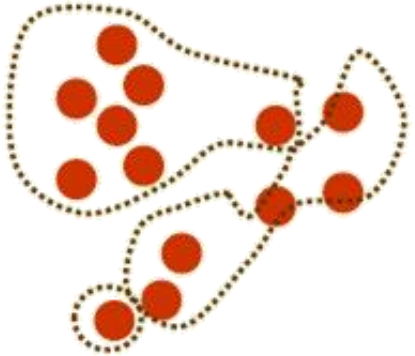




2. Criterion function to evaluate a clustering



good clustering



bad clustering

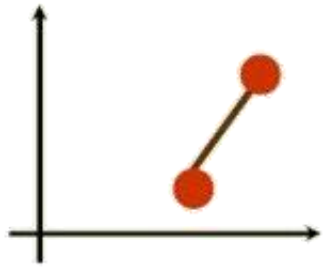
3. Algorithm to compute clustering
   - For example, by optimizing the criterion function

Distance (dissimilarity) measures

- Euclidean distance

$$d(x_i, x_j) = \sqrt{\sum_{k=1}^{d} (x_i^{(k)} - x_j^{(k)})^2}$$

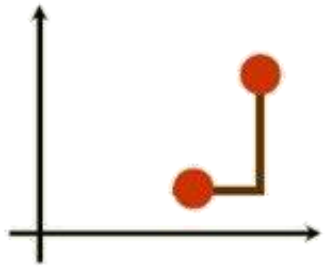
  - translation invariant



- Manhattan (city block) distance

$$d(x_i, x_j) = \sum_{k=1}^{d} |x_i^{(k)} - x_j^{(k)}|$$

  - approximation to Euclidean distance, cheaper to compute



- They are special cases of **Minkowski distance**:

$$d_p(\mathbf{x}_i, \mathbf{x}_j) = \left(\sum_{k=1}^{m} |x_{ik} - x_{jk}|^p \right)^{\frac{1}{p}}$$

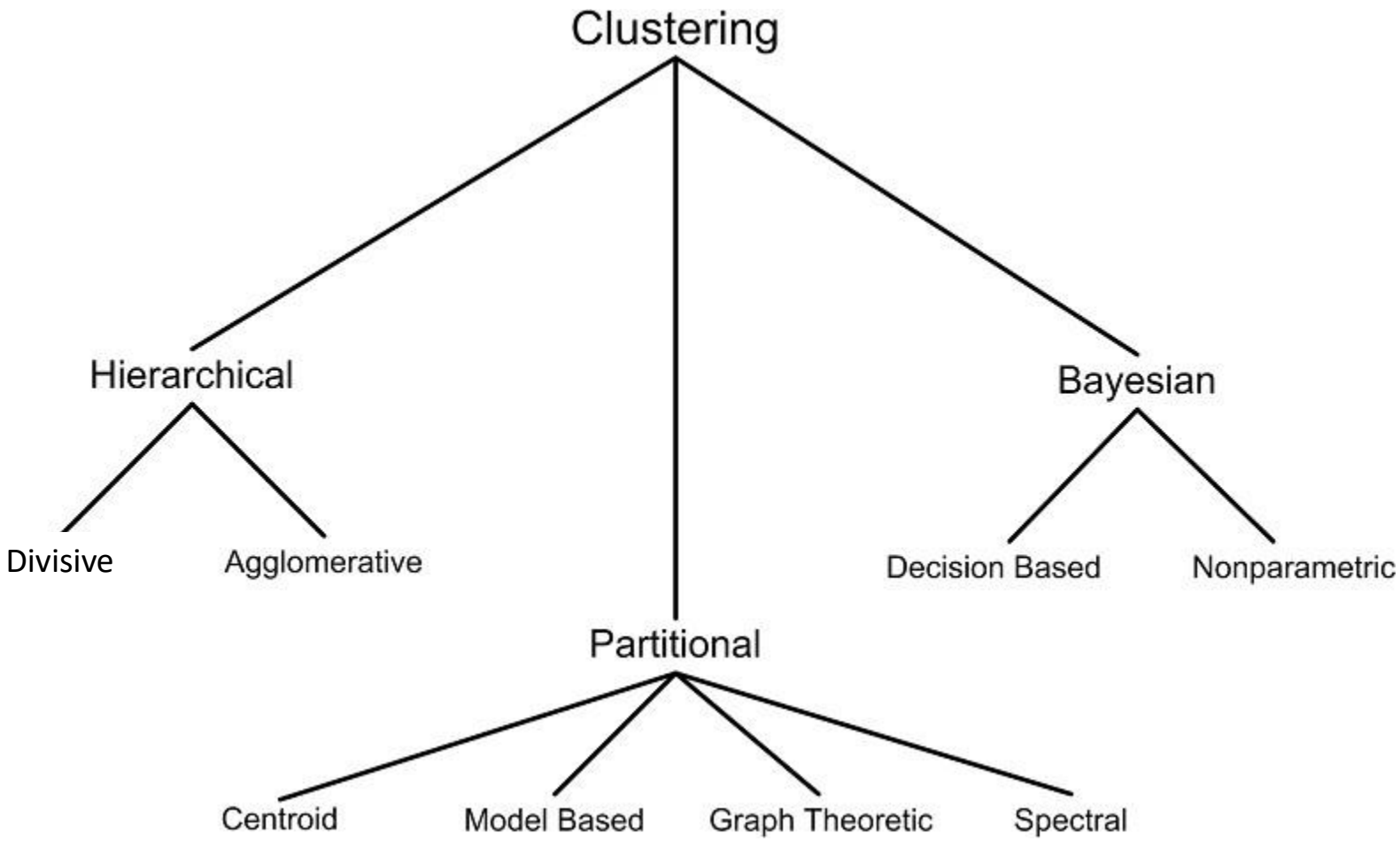
(p is a positive integer)

How many clusters?

3 clusters or 2 clusters?

- Possible approaches
  1. fix the number of clusters to ***k***
  2. find the best clustering according to the criterion function (number of clusters may vary)

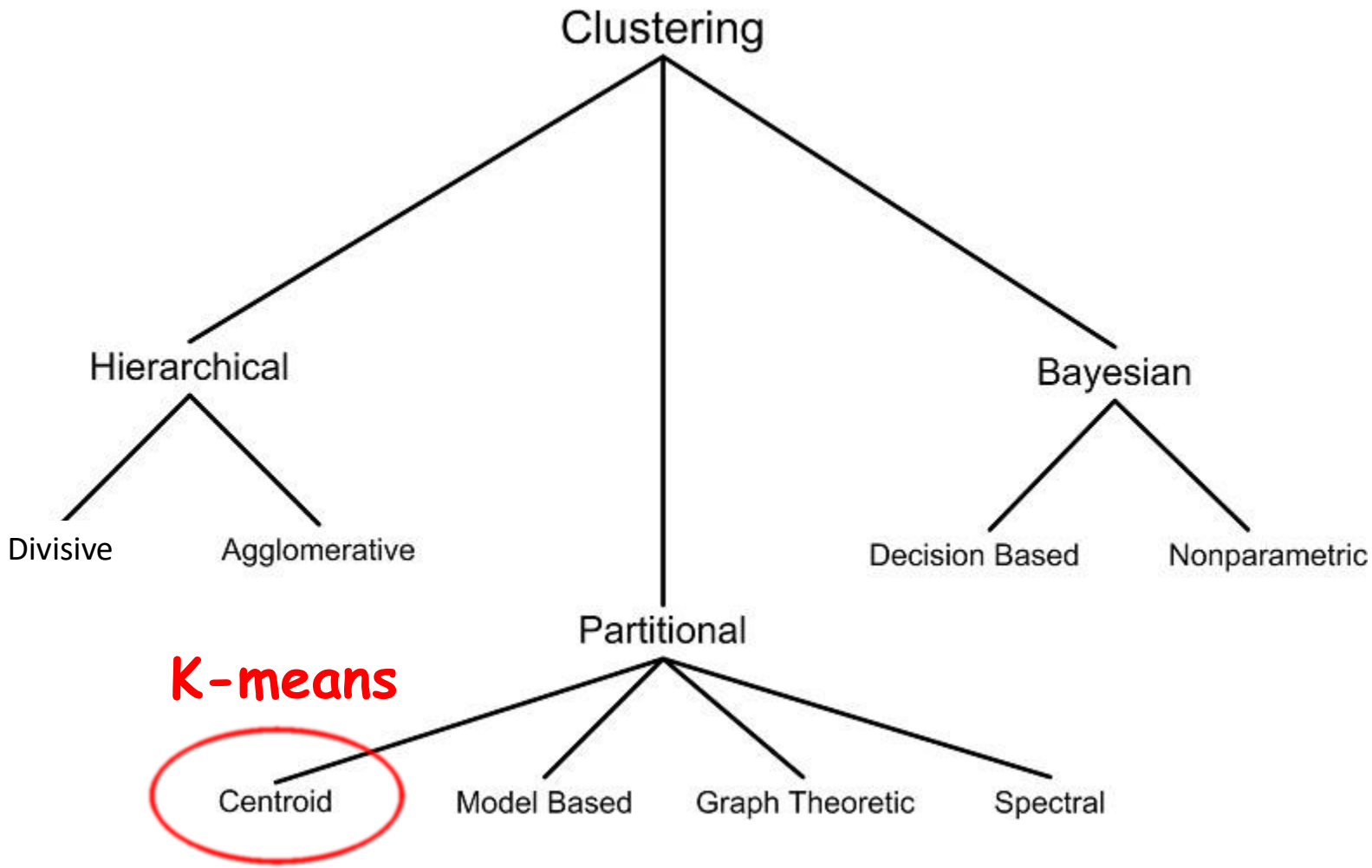
Clustering techniques



Clustering techniques

- **Hierarchical** algorithms find successive clusters using previously established clusters. These algorithms can be either agglomerative ("*bottom-up*") or divisive ("*top-down*"):
  1. Agglomerative algorithms begin with each element as a separate cluster and merge them into successively larger clusters;
  2. Divisive algorithms begin with the whole set and proceed to divide it into successively smaller clusters.
- **Partitional** algorithms typically determine all clusters at once, but can also be used as divisive algorithms in the hierarchical clustering.
- **Bayesian** algorithms try to generate a *posteriori distribution* over the collection of all partitions of the data.

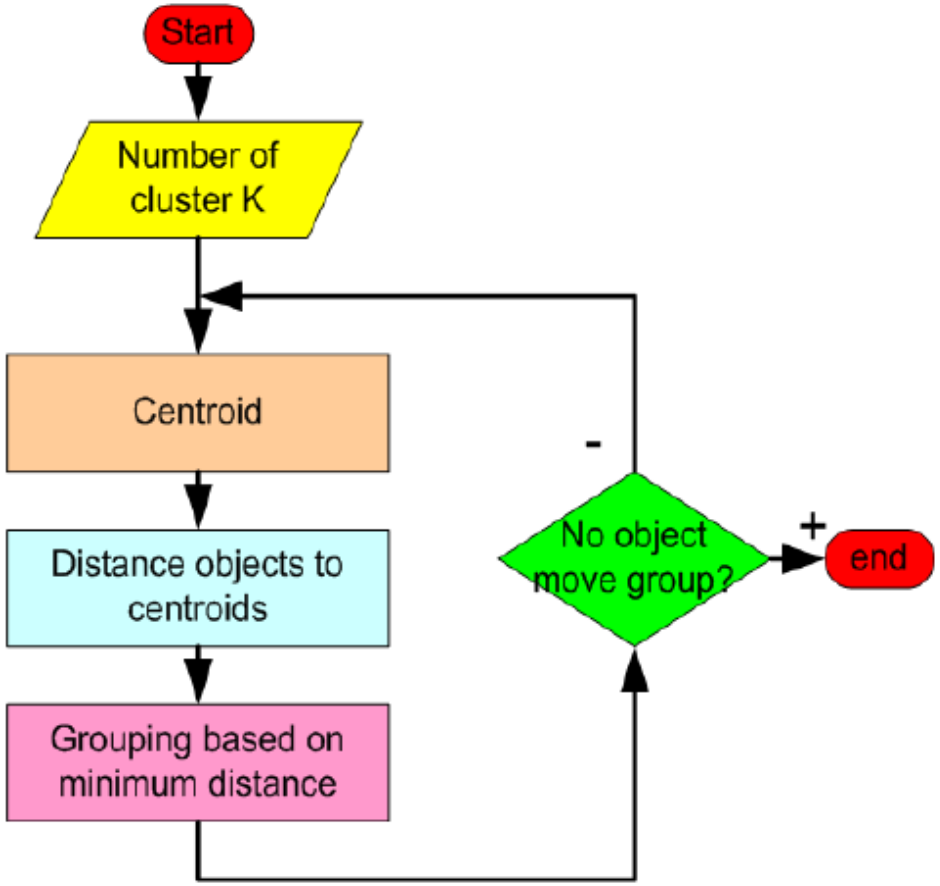
Clustering techniques



K-Means clustering

- K-means (MacQueen, 1967) is a partitional clustering algorithm
- Let the set of data points D be $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$,
  - where $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{ir})$ is a vector in $X \subseteq R^r$, and r is the
  - number of dimensions.
- The *k*-means algorithm partitions the given data into
- *k* clusters:
  - Each cluster has a cluster **center**, called **centroid**.
  - *k* is specified by the user

K-means algorithm



K-means convergence (stopping) criterion

- no (or minimum) re-assignments of data points to different clusters, *or*
- no (or minimum) change of centroids, *or*
- minimum decrease in the **sum of squared error** (SSE),

$$SSE = \sum_{j=1}^{k} \sum_{\mathbf{x} \in C_j} d(\mathbf{x}, \mathbf{m}_j)^2$$

  - C_j is the jth cluster,
  - $\mathbf{m}_j$ is the centroid of cluster C_j (the mean vector of all the data points in C_j),
  - $d(\mathbf{x}, \mathbf{m}_j)$ is the (Eucledian) distance between data point $\mathbf{x}$ and centroid $\mathbf{m}_j$.

K-means clustering example: start

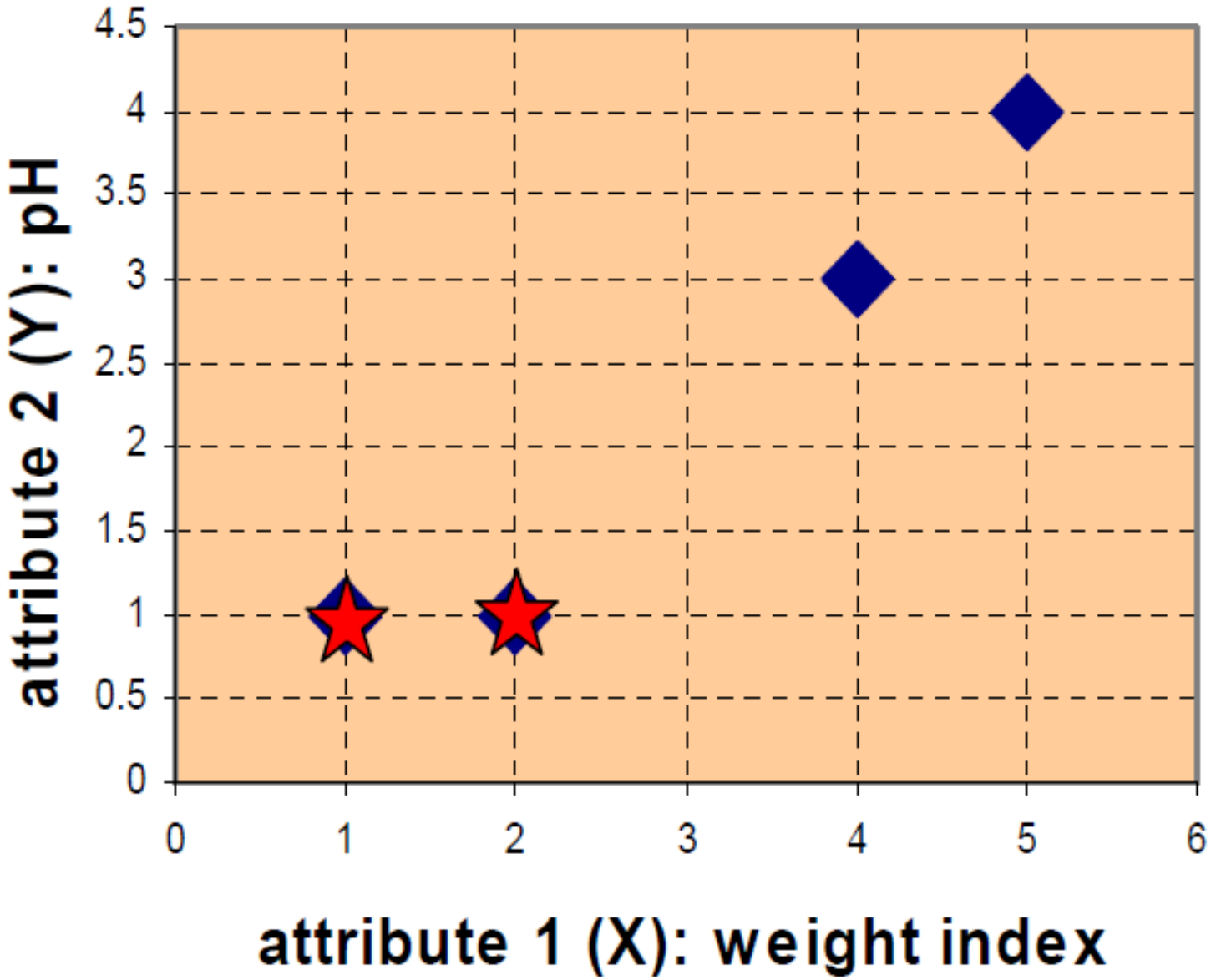
Suppose we have several objects (4 types of medicines) and each object have attributes or

features as shown in table below. Our goal is to group these objects into K=2 group of medicine

based on the two features (pH and weight index). These 4 objects are called training data points.

Object	Attribute 1 (X): weight index	Attribute 2 (Y): pH
Medicine A	1	1
Medicine B	2	1
Medicine C	4	3
Medicine D	5	4

Each object in our example has 2 attributes. We can represent each attribute as a coordinate in 2-dimensional chart. Each medicine represents one point with two components coordinate. We call the two components of the coordinate as (X, Y). Thus, we can represent objects as points in a feature space as shown in the figure below.


attribute 2 (Y): pH
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
0
1
2
3
4
5
6
attribute 1 (X): weight index

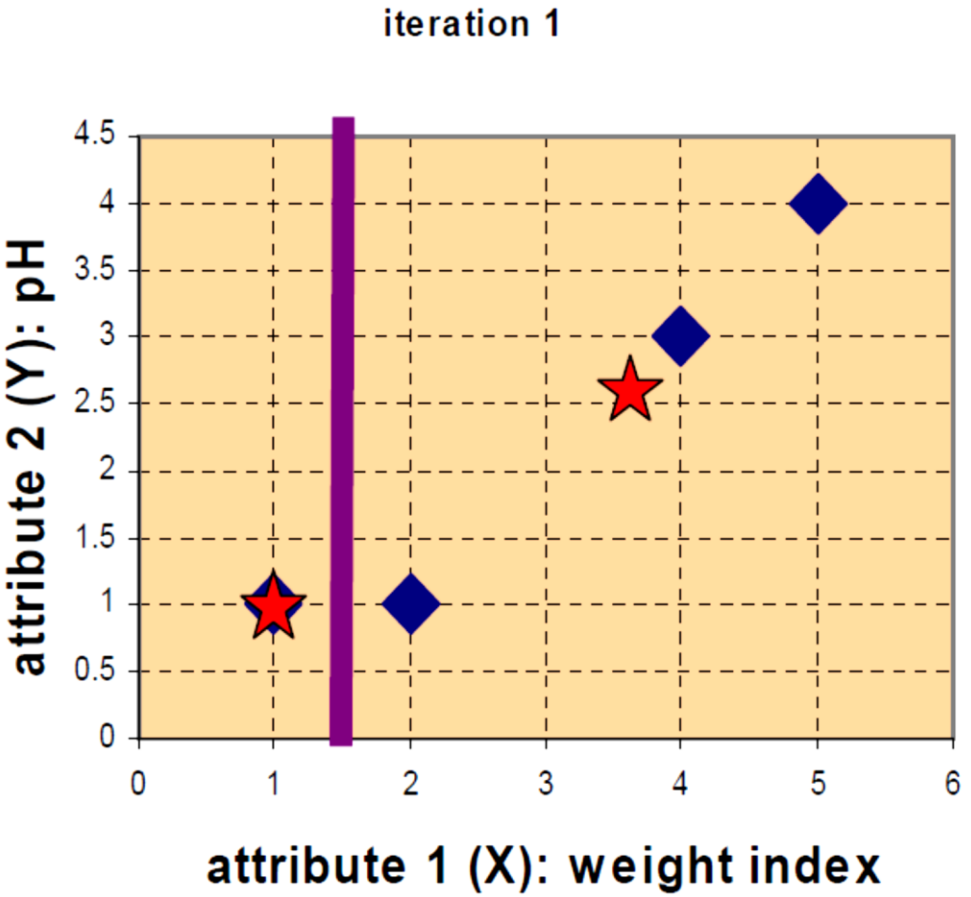
K-means clustering example – step 1

To use k means algorithm, we must *know beforehand* that these objects belong to two groups of medicine (let name these two groups as cluster 1 and cluster 2).

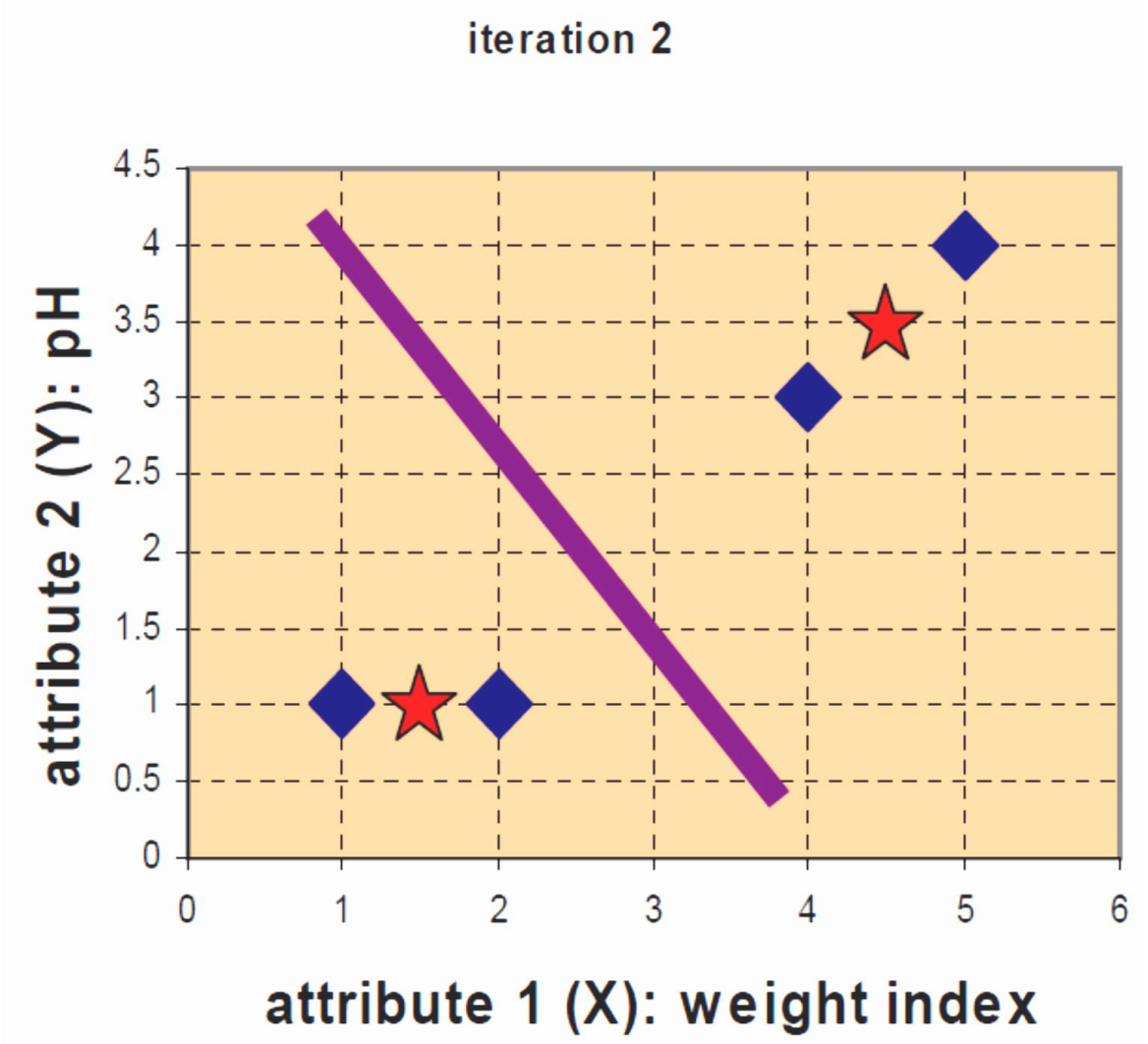
The stars represent the initial location of the two centroids.

The problem now is to determine which medicines belong to cluster 1 and which medicines belong to the other cluster.

K-means clustering example – step 2



K-means clustering example – step 3



K-means clustering example – step 4

- Compute one more iteration.
- Comparing the grouping of last iteration and this iteration reveals that the objects does not move group anymore.
- Thus, the computation of the k-mean clustering has reached its stability and no more iteration is needed.
- We get the final grouping as the results

Object	Feature 1 (X): weight index	Feature 2 (Y): pH	Group (result)
Medicine A	1	1	**1**
Medicine B	2	1	**1**
Medicine C	4	3	**2**
Medicine D	5	4	**2**

Why use K-means?

- Strengths:
  - Simple: easy to understand and to implement
  - Efficient: Time complexity: $O(tkn)$,

    where n is the number of data points,

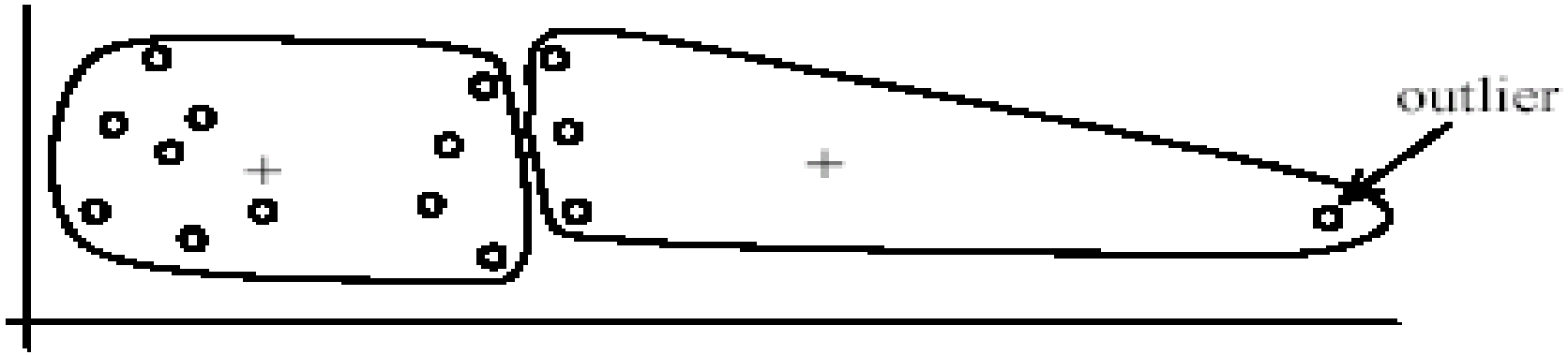
    k is the number of clusters, and

    t is the number of iterations.
  - Since both k and t are small. k-means is considered a linear algorithm.
- K-means is the most popular clustering algorithm.
- Note that: it terminates at a local optimum if SSE is used. The global optimum is hard to find due to complexity.

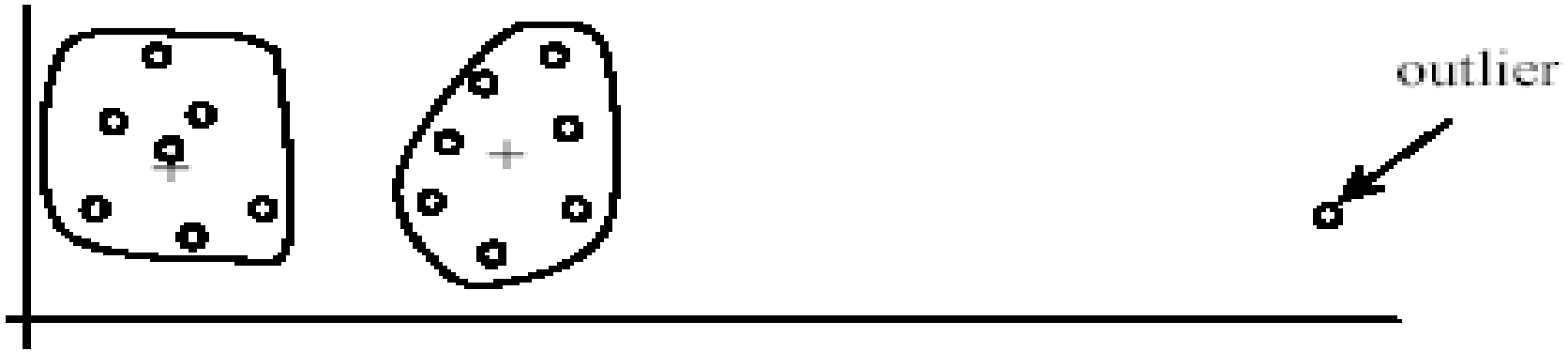
Weaknesses of K-means

- The algorithm is only applicable if the mean is defined.
  - For categorical data, *k*-mode - the centroid is represented by most frequent values.
- The user needs to specify *k*.
- The algorithm is sensitive to **outliers**
  - Outliers are data points that are very far away from other data points.
  - Outliers could be errors in the data recording or some special data points with very different values.

Outliers



(A): Undesirable clusters

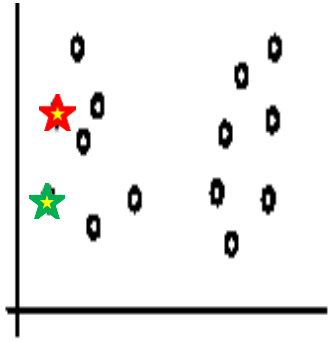


(B): Ideal clusters

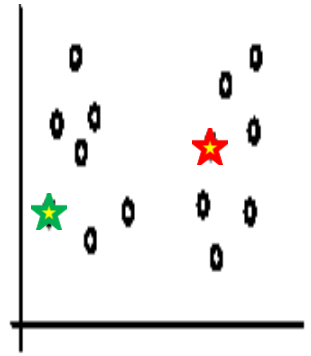
Dealing with outliers

- Remove some data points that are much further away from the centroids than other data points
  - To be safe, we may want to monitor these possible outliers over a few iterations and then decide to remove them.
- Perform random sampling: by choosing a small subset of the data points, the chance of selecting an outlier is much smaller
  - Assign the rest of the data points to the clusters by distance or
  - similarity comparison, or classification

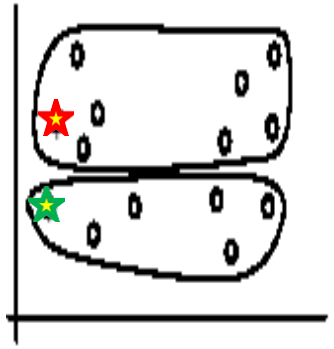
Sensitivity to initial seeds



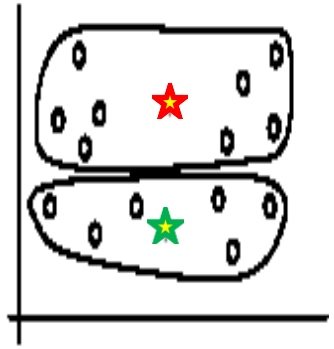
Random selection of seeds (centroids)



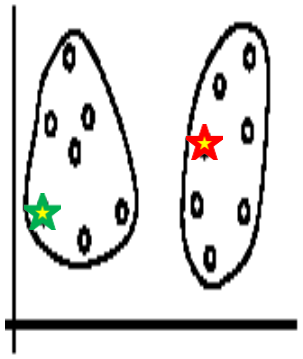
Random selection of seeds (centroids)



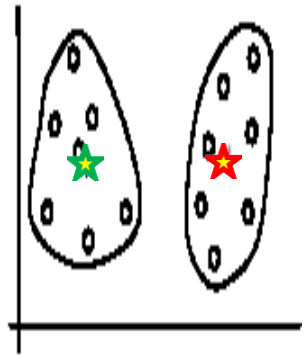
Iteration 1



Iteration 2



Iteration 1



Iteration 2

Elbow Method – Inertia Calculation

- Within-cluster sum of squares

$$\mathrm{WSS}(C) = \sum_{i=1}^{k} \sum_{x_j \in C_i} ||x_j - \mu_i||^2$$

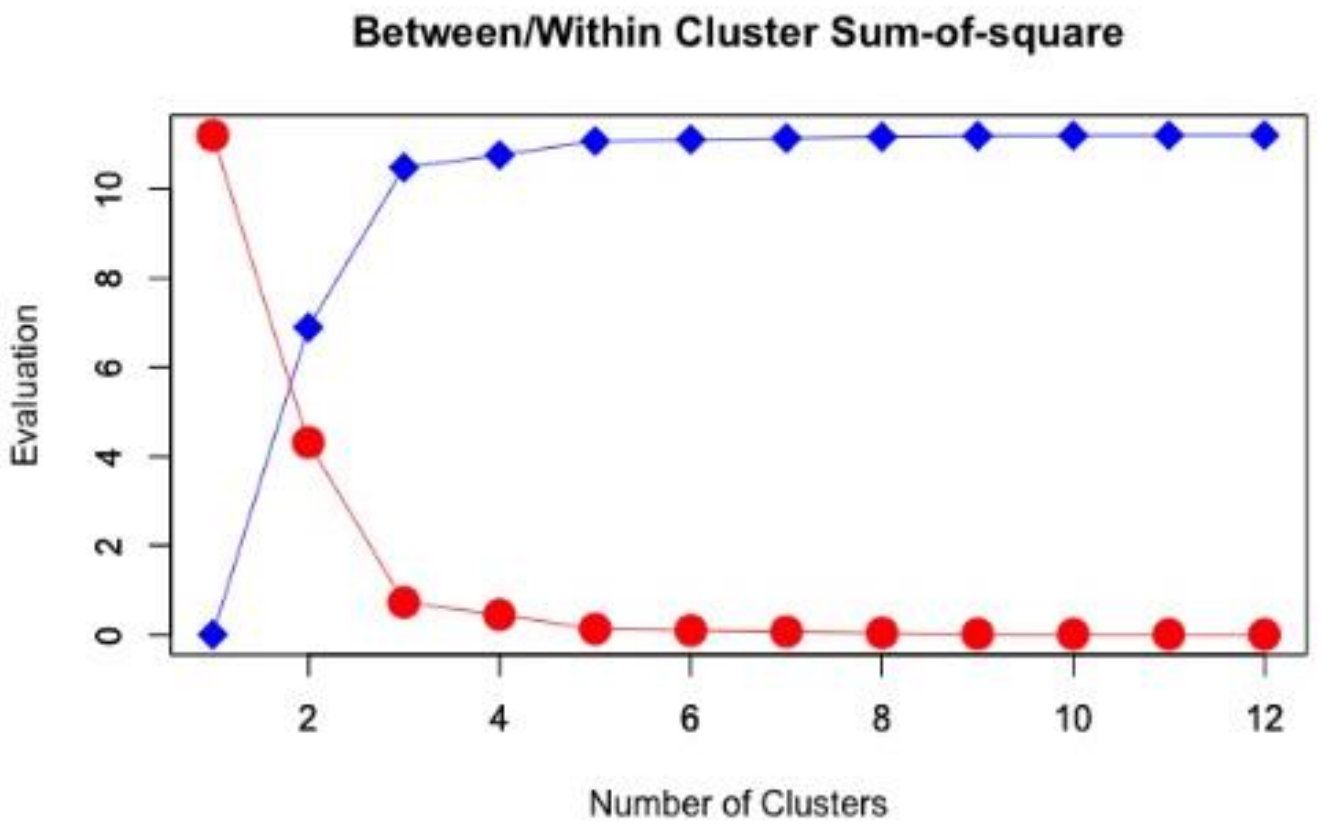
where μ_i is the centroid of cluster C_i (in case of Euclidean spaces)

- Between-cluster sum of squares

$$\mathrm{BSS}(C) = \sum_{i=1}^{k} |C_i| \cdot ||\mu - \mu_i||^2$$

where μ is the centroid of the whole dataset

Elbow Method



Framework for Cluster Validity

Need a framework to interpret any measure.

- For example, if our measure of evaluation has the value, 10, is that good, fair, or poor?

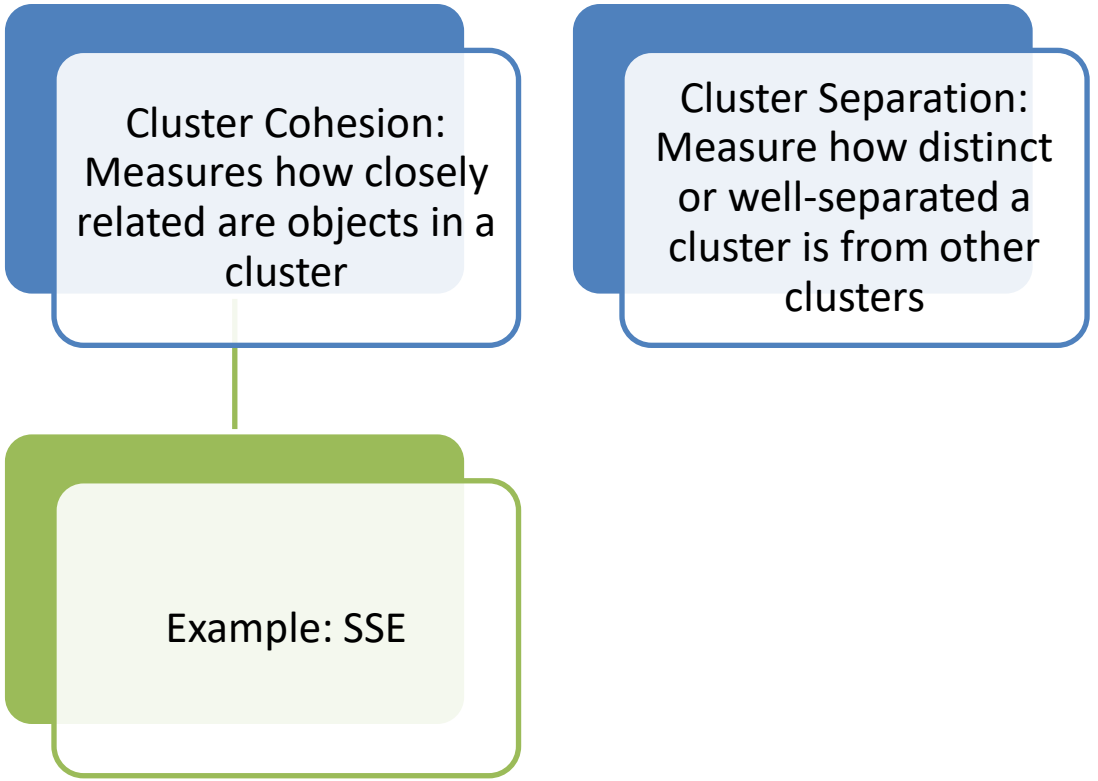
Statistics provide a framework for cluster validity

- The more “atypical” a clustering result is, the more likely it represents valid structure in the data
- Can compare the values of an index that result from random data or clusterings to those of a clustering result.
  - If the value of the index is unlikely, then the cluster results are valid
- These approaches are more complicated and harder to understand.

For comparing the results of two different sets of cluster analyses, a framework is less necessary.

- However, there is the question of whether the difference between two index values is significant

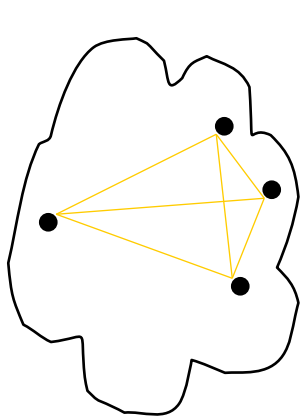
Internal Measures: Cohesion and Separation



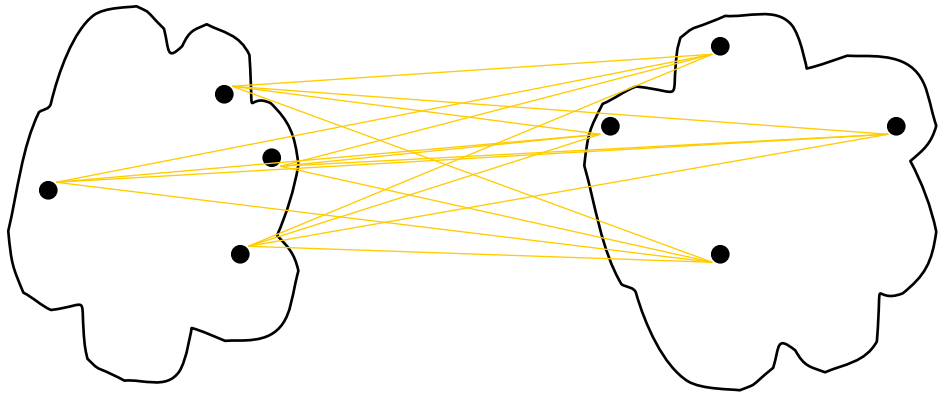
Internal Measures: Cohesion and Separation

A proximity graph based approach can also be used for cohesion and separation.

- Cluster cohesion is the sum of the weight of all links within a cluster.
- Cluster separation is the sum of the weights between nodes in the cluster and nodes outside the cluster.



cohesion



separation

Internal Measures: Silhouette Coefficient

Silhouette Coefficient combine ideas of both cohesion and separation, but for individual points, as well as clusters and clusterings

For an individual point, *i*

Calculate ***a*** = average distance of *i* to the points in its cluster

Calculate ***b*** = min (average distance of *i* to points in another cluster)

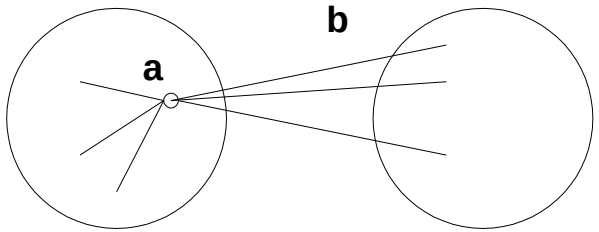
The silhouette coefficient for a point is then given by:

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

Typically, between 0 and 1.

The closer to 1 the better

Negative values are pathological



Can calculate the Average Silhouette width for a cluster or for the whole clustering

K-means summary

- Despite weaknesses, *k*-means is still the most popular algorithm due to its simplicity and efficiency
- No clear evidence that any other clustering algorithm performs better in general
- Comparing different clustering algorithms is a difficult task. No one knows the correct clusters!

Hierarchical Clustering

Let’s say we have the points and we want to cluster them into groups:

Hierarchical Clustering

1

We can assign each of the points to a separate cluster.

2

Now, based on the similarity of these clusters, we can combine the most similar clusters together and repeat this process until only a single cluster is left.

3

We are essentially building a hierarchy of clusters. That's why this algorithm is called hierarchical clustering.

Types of hierarchical clustering

Divisive (top down) clustering

- Starts with all data points in one cluster, the root, then
  - Splits the root into a set of child clusters. Each child cluster is
  - recursively divided further
  - stops when only singleton clusters of individual data points remain, i.e., each cluster with only a single point

Agglomerative (bottom up) clustering

- The dendrogram is built from the bottom level by
  - merging the most similar (or nearest) pair of clusters
  - stopping when all the data points are merged into a single cluster (i.e., the root cluster).

Agglomerative Clustering - Example

Student_ID	Marks
1	10
2	7
3	28
4	20
5	35

Suppose a teacher wants to divide her students into different groups. She has the marks scored by each student in an assignment and based on these marks, she wants to segment them into groups.

Agglomerative Clustering - Creating a Proximity Matrix

ID	1	2	3	4	5
1	0	3	18	10	25
2	3	0	21	13	28
3	18	21	0	8	7
4	10	13	8	0	15
5	25	28	7	15	0

Steps to Perform Hierarchical Clustering

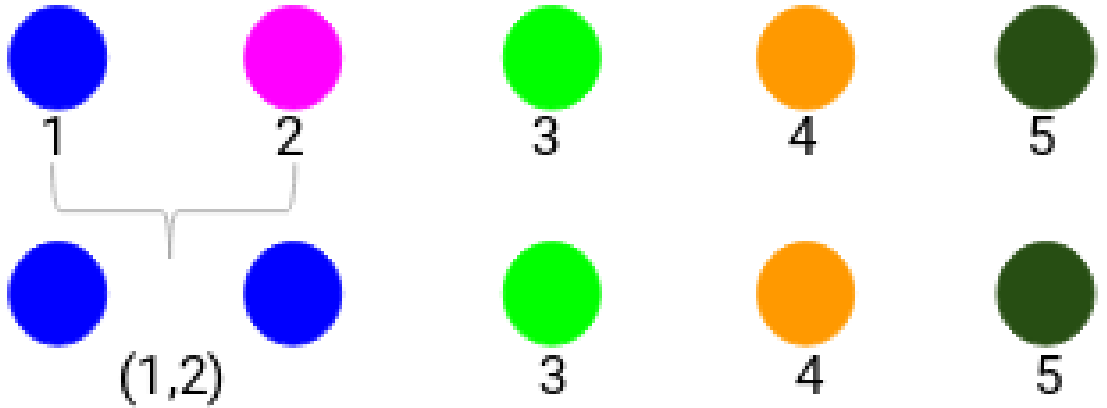
Step 1: First, we assign all the points to an individual cluster.

Step 2: Next, we will look at the smallest distance in the proximity matrix and merge the points with the smallest distance.

ID	1	2	3	4	5
1	0	3	18	10	25
2	3	0	21	13	28
3	18	21	0	8	7
4	10	13	8	0	15
5	25	28	7	15	0

Steps to Perform Hierarchical Clustering

Step 3: Update Proximity Matrix

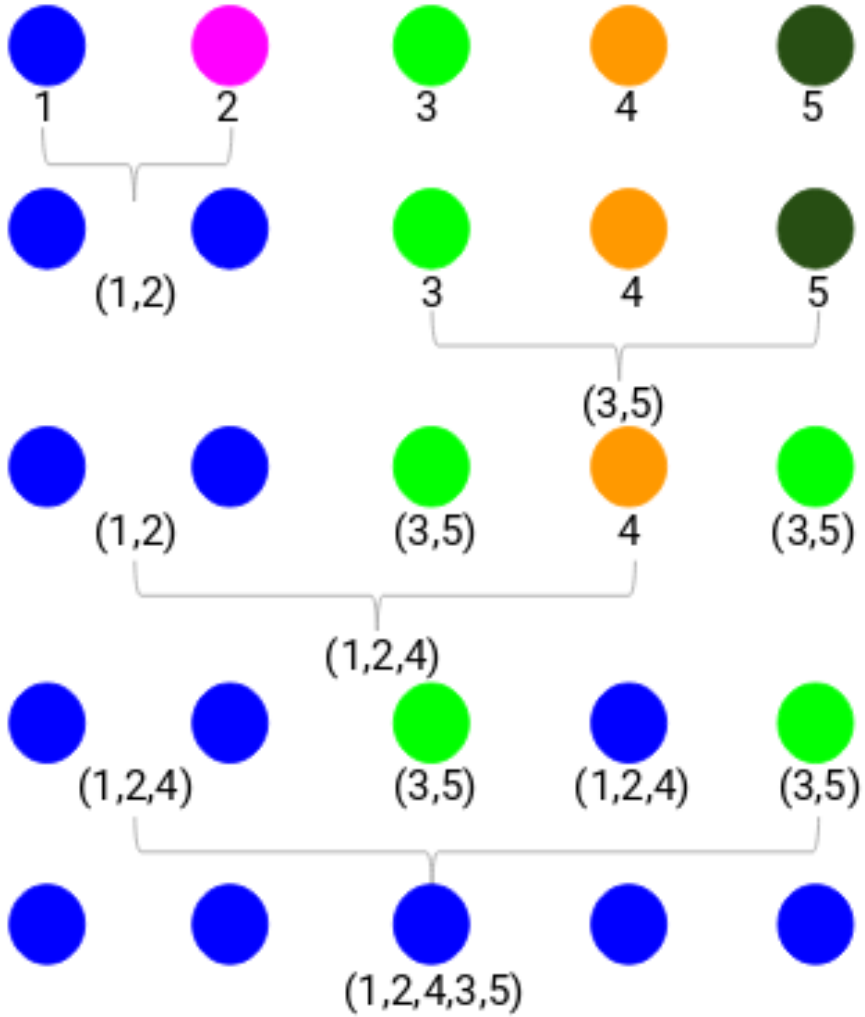


Student_ID	Marks
(1,2)	10
3	28
4	20
5	35

Steps to Perform Hierarchical Clustering

ID	(1,2)	3	4	5
(1,2)	0	18	10	25
3	18	0	8	7
4	10	8	0	15
5	25	7	15	0

We will repeat step 2 until only a single cluster is left.



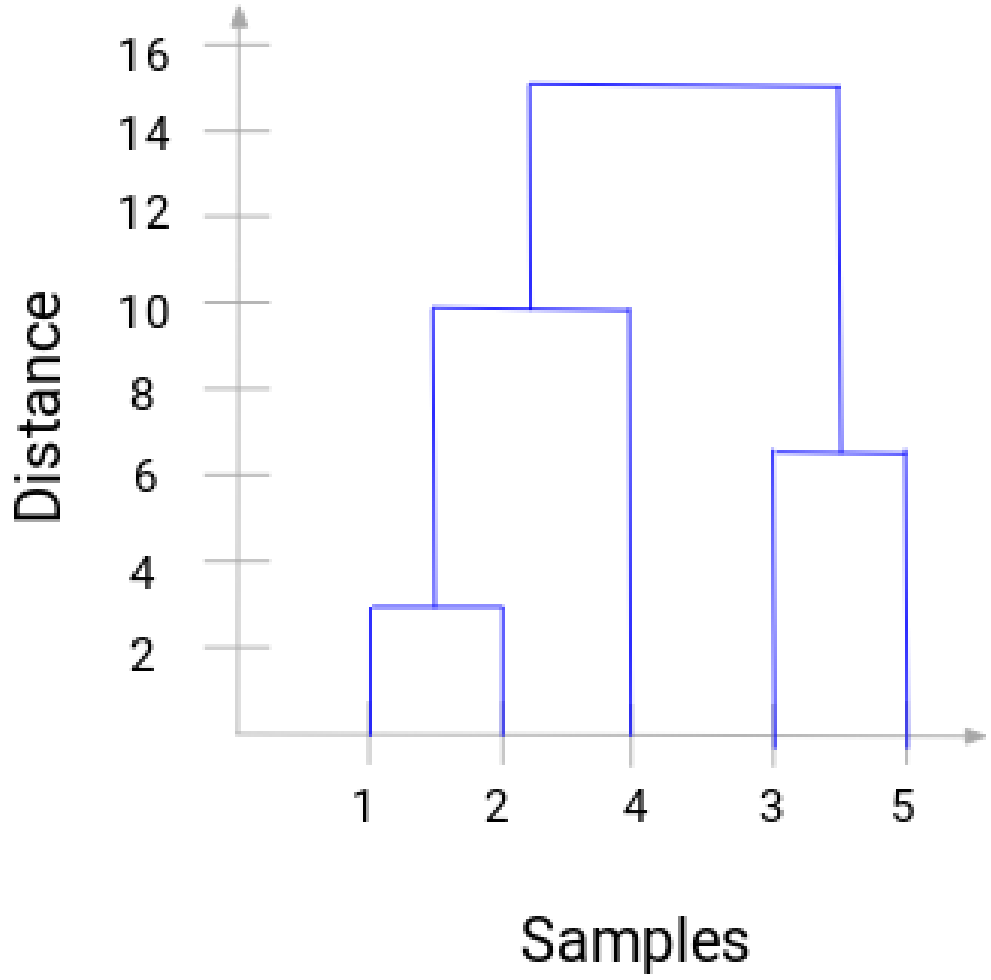
How should we Choose the Number of Clusters in Hierarchical Clustering?

To get the number of clusters for hierarchical clustering, we make use of an awesome concept called a Dendrogram.

A dendrogram is a tree-like diagram that records the sequences of merges or splits.

For our Teacher-Student Example, Whenever two clusters are merged, we will join them in this dendrogram and the height of the join will be the distance between these points.

Dendogram



Clusters