Lesson 10 Clustering

Outline

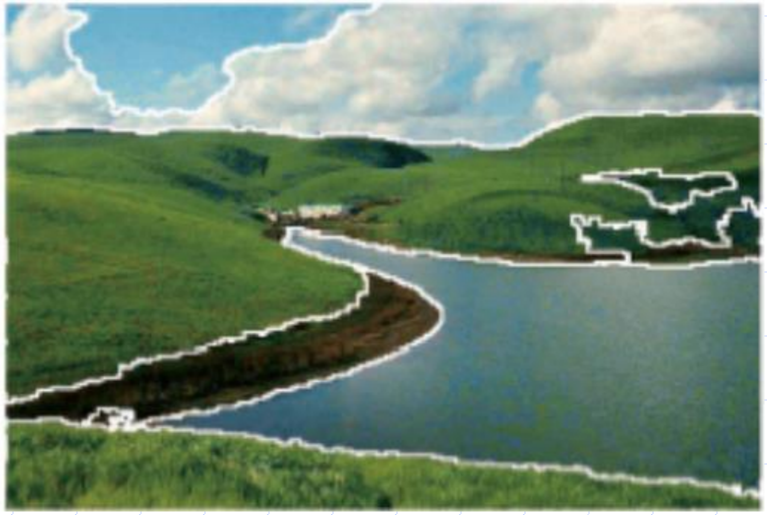
- Cluster Analysis and techniques
- Partitioning Algorithms: K-Means
- Determine the number of clusters
- Hierarchical Algorithms: Divisive and Agglomerative

Cluster Analysis

- Clustering is a technique used for automatic identification of natural groupings of things
  - data instances that are similar to each other are categorized into one cluster
  - data instances that are very different from each other into different clusters.
  - Learns the clusters of things from past data, then assigns new instances to their cluster homes
- Unsupervised learning
  - There is no class label
- Also known as data segmentation

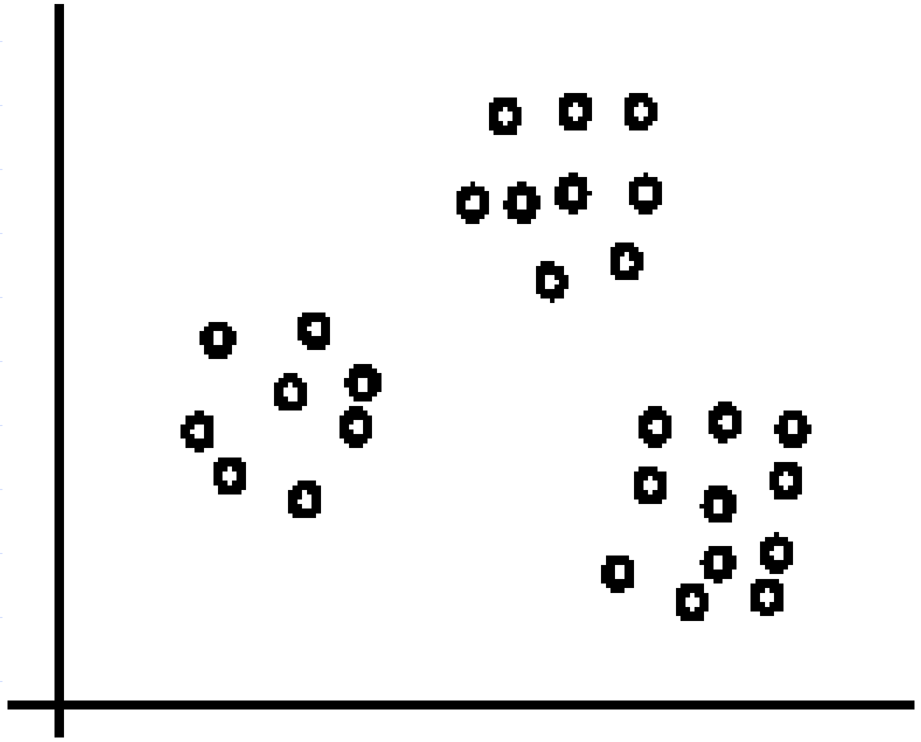
Clustering Applications

- Group people of similar sizes together to make small, medium and large T-Shirt sizes
  - To provide fit clothes at mass production rates
- Segment customers according to their similarities
  - To do targeted marketing.
- Image Segmentation
  - Goal: Break up the image into meaningful or perceptually similar regions



An illustration

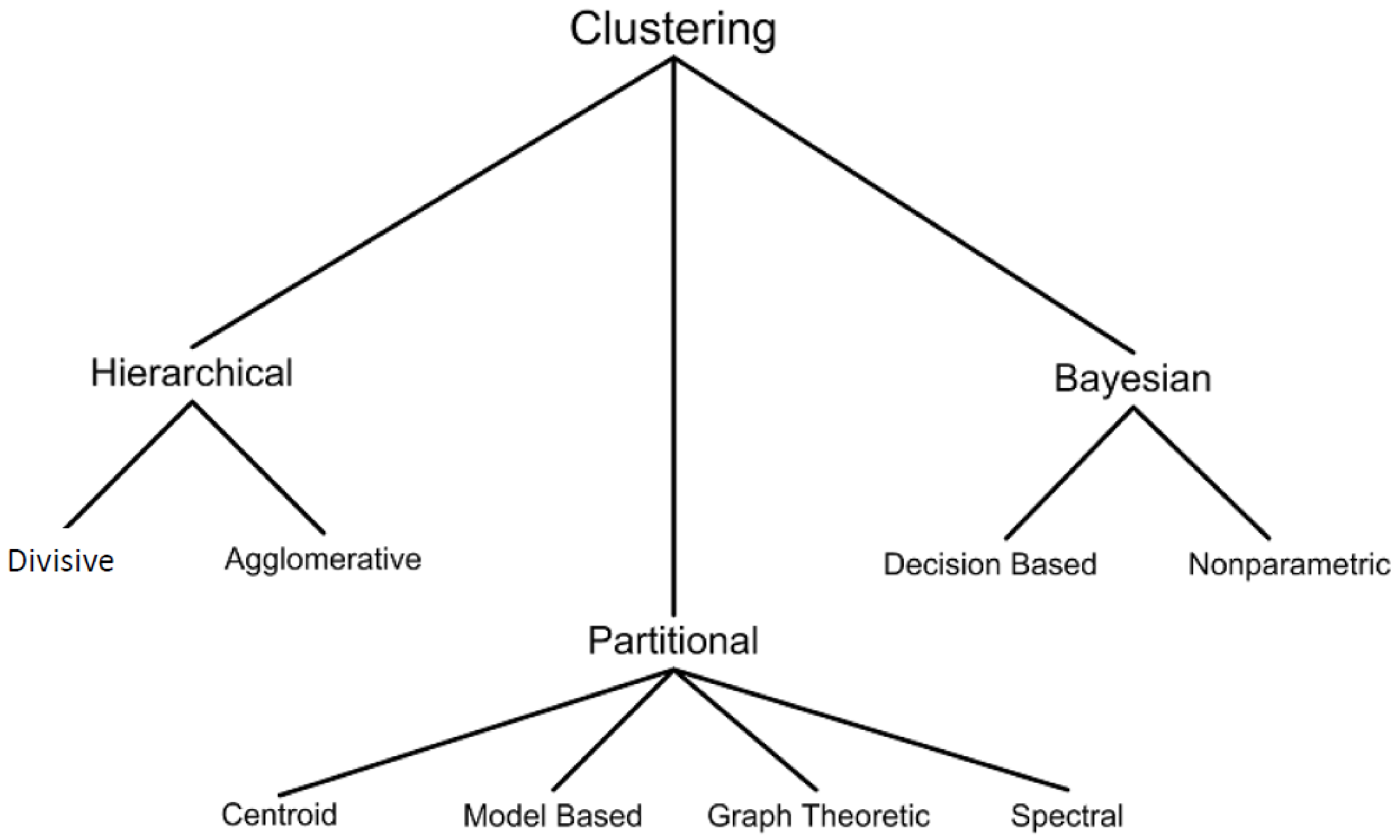
- This data set has three natural groups of data points, i.e., 3 natural clusters.



Cluster Analysis for Data Mining

- A measure of similarity: most cluster analysis methods use a distance measure to calculate the similarity between pairs of items
  - Euclidean distance, Manhattan distance etc.
- A good clustering method will produce high quality clusters
  - high intra-class similarity: cohesive within clusters
  - low inter-class similarity: distinctive between clusters

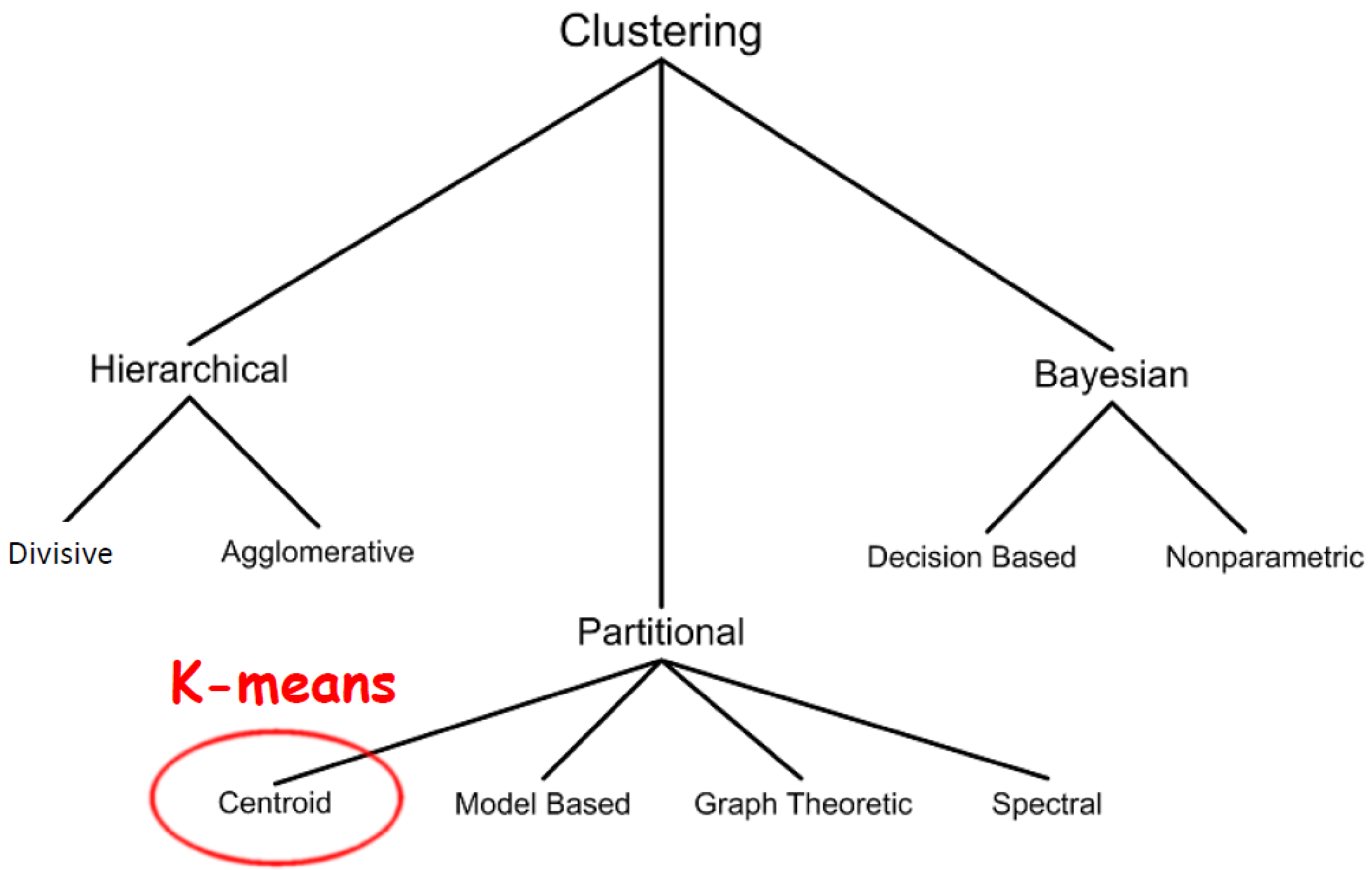
Clustering techniques



Clustering techniques

- Hierarchical algorithms find successive clusters using previously established clusters. These algorithms can be either agglomerative (“bottom-up”) or divisive (“top-down”):
  - Agglomerative algorithms begin with each element as a separate cluster and merge them into successively larger clusters;
  - Divisive algorithms begin with the whole set and proceed to divide it into successively smaller clusters.
- Partitional algorithms: Given a set of n objects, a partitioning method constructs k partitions of the data, where each partition represents a cluster and $k \leq n$.

Clustering techniques



Partitioning Algorithms: Basic Concept

- Partitioning method: Partitions a database ***D*** of ***n*** objects into a set of ***k*** clusters, such that the sum of squared distances is minimized (where *p* is a data point and c_i is the centroid of cluster C_i).

$$E = \Sigma_{i=1}^{k} \Sigma_{p \in C_i} (p - c_i)^2$$

- This is also called *SSE* (*sum of squared errors*).

Partitioning Algorithms: Basic Concept

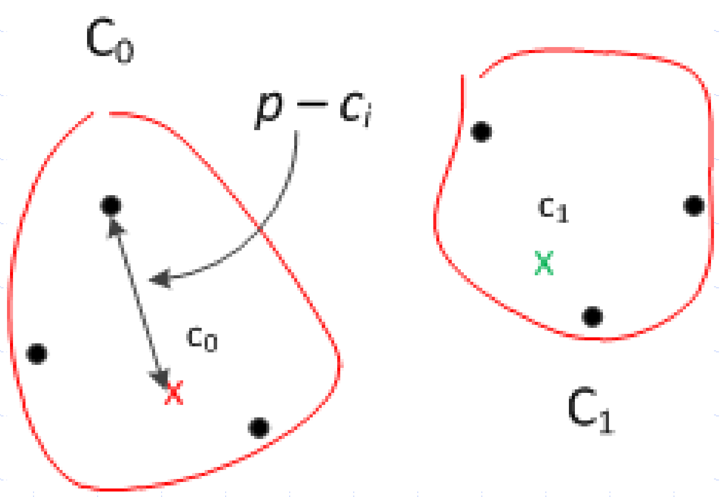
- More about SSE:

$$E = \Sigma_{i=1}^{k} \Sigma_{p \in C_i} (\underline{p - c_i})^2$$

sum for all clusters

squared distance between an object and the centroid of its cluster

sum for all objects in a cluster



k-Means Clustering Algorithm: Pseudo-code

Algorithm K-Means(k, D)

Input database D of n objects, k (# of clusters)

Output n objects and their cluster assignments

```
Choose k data points as the initial centroids (cluster centers)
repeat
    for each data point x in D do
        compute the distance from x to each centroid
        assign x to the closest centroid // a centroid represents a cluster
    re-compute the centroids using the current cluster memberships
until the stopping criterion is met
return all objects and their cluster assignments
```

Stopping Criteria

- The membership assignment does not change. (We will use this criterion for demo.)
- After each reassignment, E is computed and if E falls below a predefined threshold.
- If the decrease in E, between two consecutive iterations, falls below a predefined threshold.
- Run for a predetermined number of iterations (e.g., run 20 iterations).

Distance (Dissimilarity) Measures

- Manhattan (city block) distance
  - consider two data items, A and B, each with two attribute values, (x1, y1) and (x2, y2), respectively.

$$d(A,B) = |x1 - x2| + |y1 - y2|$$

- Euclidean distance
  - consider two data items, A and B, each with two attribute values, (x1, y1) and (x2, y2), respectively.

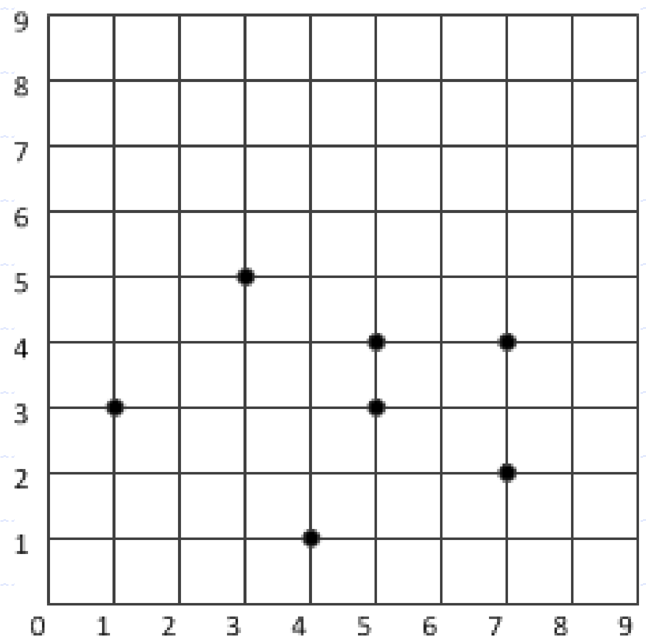
$$d(A,B) = \sqrt{|x1 - x2|^2 + |y1 - y2|^2}$$

- They are special cases of Minkowski distance
  - In general, the Minkowski distance between the two p-dimensional data items is given by the following formula, where *q* is a positive integer. (x1, y1, z1 … are the attribute values.)

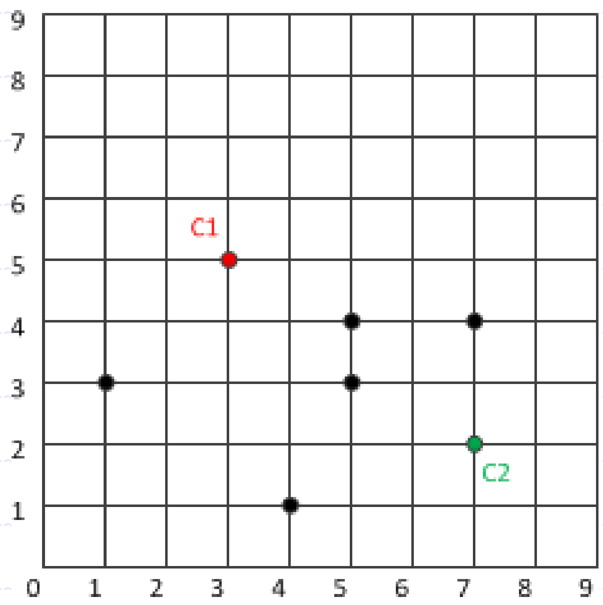
$$d(A, B) = \sqrt[q]{|x1 - x2|^q + |y1 - y2|^q + |z1 - z2|^q + \cdots}$$

Worked Example

Initial dataset, D = {(1,3), (3,5), (4,1), (5, 3), (5, 4), (7, 2), (7, 4)}

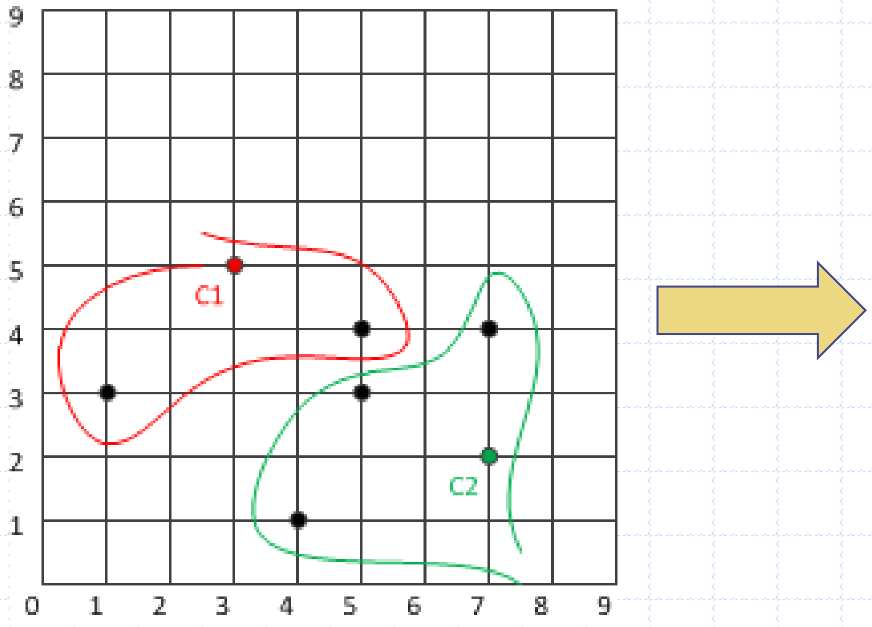


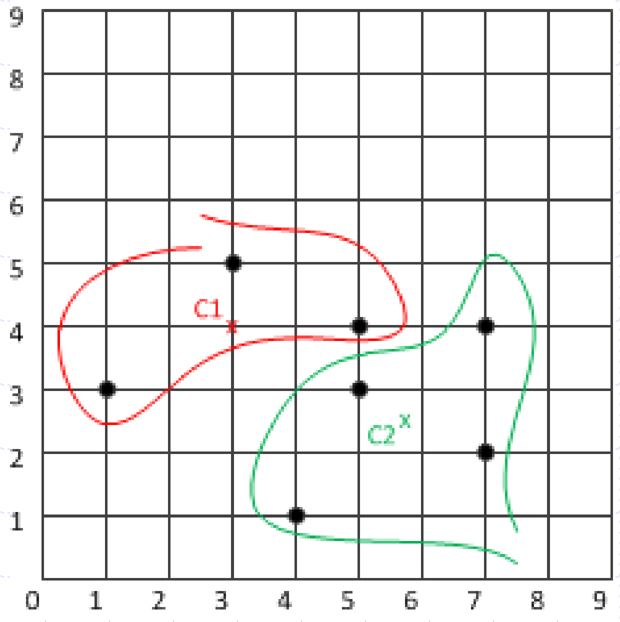
Initial dataset



Two objects (3,5),(7,2) are randomly chosen as initial centroids

Worked Example

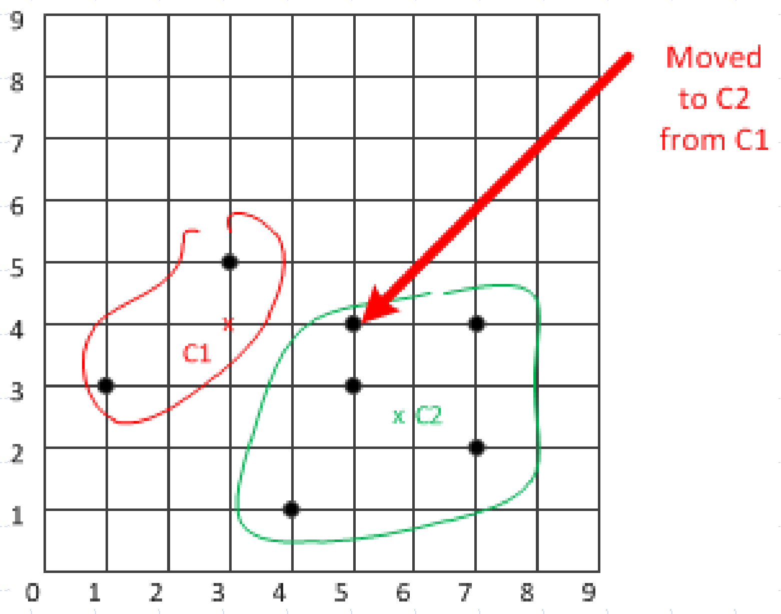


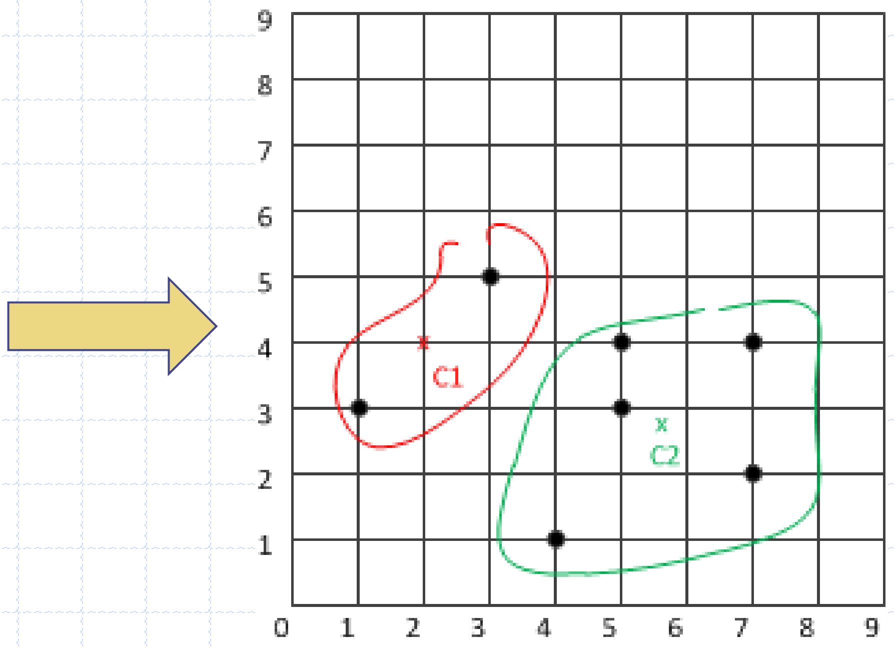


Objects are assigned to the cluster with the closest centroid using Euclidean distance. For example: D((1,3),C1)=2.83< D((1,3),C2)=6.08 so (1,3) will be in cluster with C1.

New centroids are computed.
C1.x = (1+3+5)/3 = 3
C1.y = (3+4+5)/3 = 4
C2.x = (4+5+7+7)/4= 5.75
C2.y = (1+2+3+4)/4 = 2.5

Worked Example

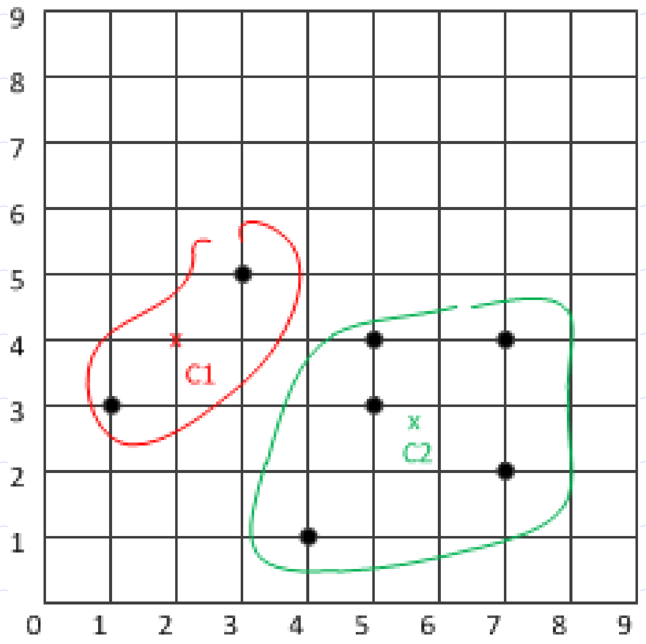




Objects are reassigned based on the distances to new centroids. Note that object (5,4) moved to cluster of C2.
D((5,4),C1)=2< D((5,4),C2)=1.68

New centroids are computed.
C1.x = (1+3)/2 = 2
C1.y = (3+5)/2 = 4
C2.x = (4+5+5+7+7)/5= 5.6
C2.y = (1+2+3+4+4)/5 = 2.8

Worked Example



Objects are reassigned based on the distances to new centroids.

There is no membership change.

So, stop here.

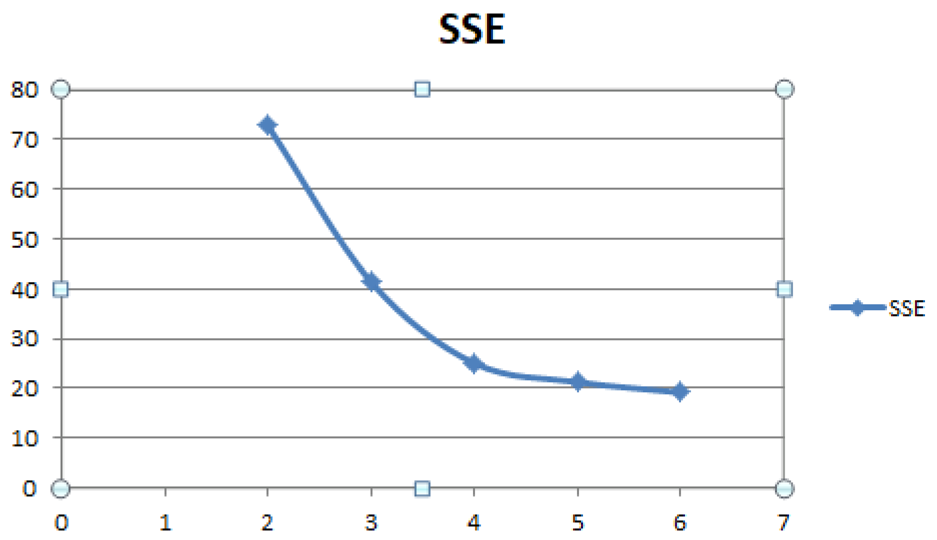
R code

```
#this command initializes R's random number generator to a specific
#sequence, so set.seed to ensure reproducibility.
set.seed(20)
irisCluster <- kmeans(iris[, 3:4], 3)
irisCluster

#plot the data to see the clusters
irisCluster$cluster <- as.factor(irisCluster$cluster)
ggplot(iris, aes(Petal.Length, Petal.Width, color = irisCluster$cluster))
+ geom_point()
```

Determine the number of clusters

- In practice, we don’t know “right” number of clusters.
- A simple method: k = $\sqrt{n/2}$
- Elbow method: as we increase k, find the point where the marginal benefit in regard to SSE does not increase significantly (typically a turning point in a graph). As the graph shows, reduce in SSE is significant at first, but becomes less significant after the "turning point" at around k = 3.5. In this approach, you would use an integer close to 3.5 – either 3 or 4 – as the value for the number of clusters.



Determine the number of clusters

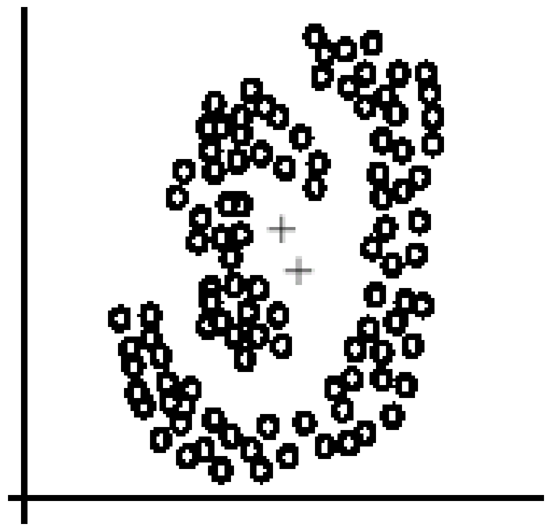
- Notes about determining the number of clusters in real work:
  - There are numerous statistics to measure "SSE" that can be used with the elbow method. Still, in practice, it is not always feasible to iteratively test a large number of k values because clustering large datasets can be fairly time consuming; clustering the data repeatedly is even worse. Regardless, applications requiring the exact optimal set of clusters are fairly rare. In most clustering applications., it suffices to choose a k value based on convenience rather than strict performance requirements.

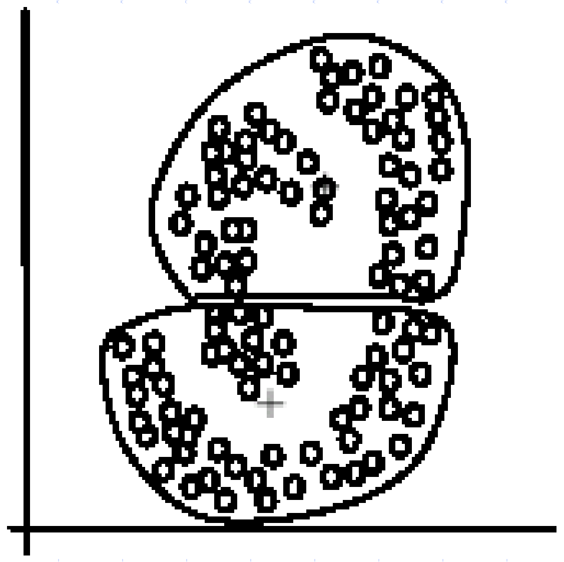
k-Means algorithm analysis

- K-Means is the most popular clustering algorithm.
- Strengths:
  - Simple: easy to understand and to implement
  - Efficient: K-Means is considered a linear algorithm.
- Weaknesses:
  - Initial random selection of centroids affects the results
    - Run K-Means multiple times with different initial centroids
  - Applicable only when the mean of objects can be defined
    - Use the k-modes method for categorical data
  - Sensitive to outliers
  - Not suitable to discover clusters with arbitrary shapes

Weaknesses of K-Means: Clusters of arbitrary shapes

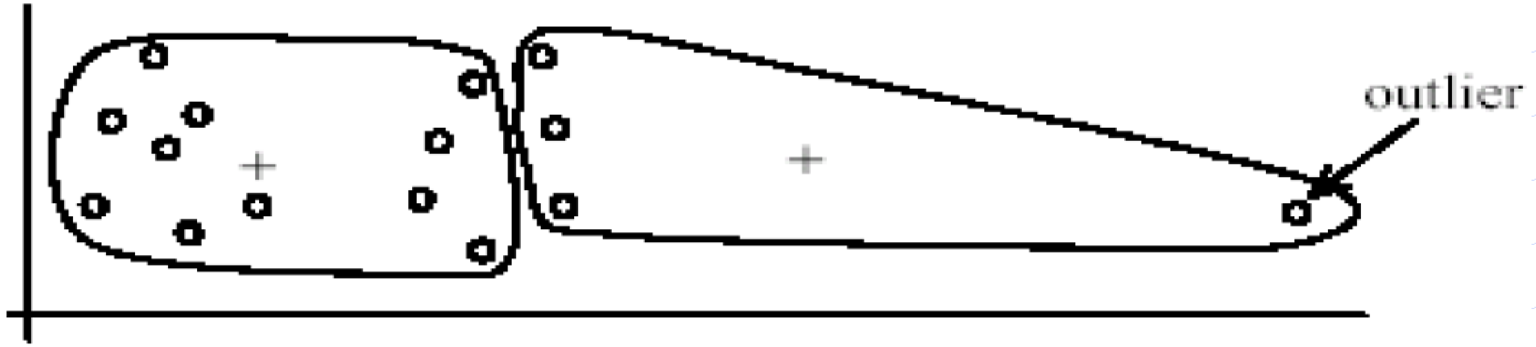
- The K-Means algorithms is not suitable for discovering clusters that are not hyper-ellipsoids (or hyper-spheres).


Two natural clusters

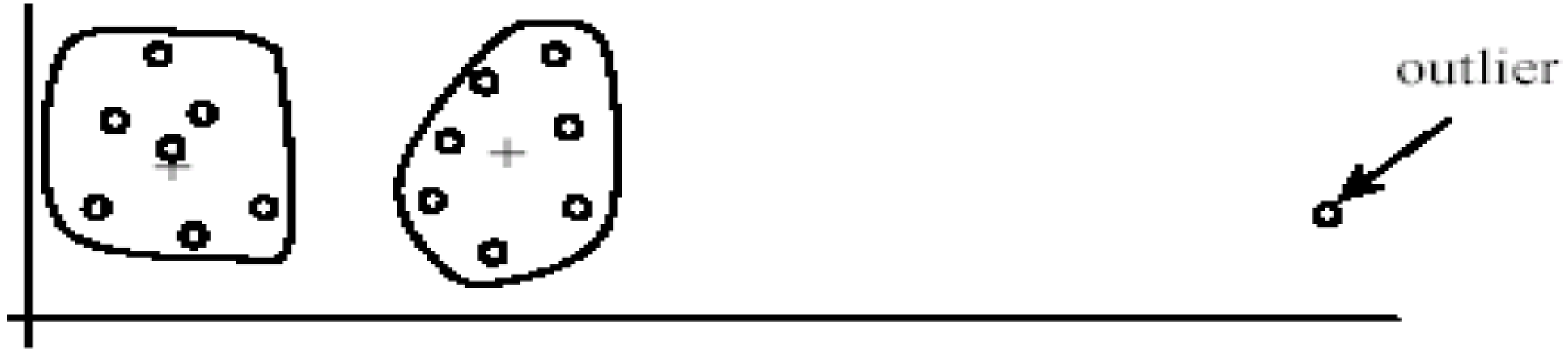

K-Means clusters

Weaknesses of K-Means: Outliers

- Undesirable clusters



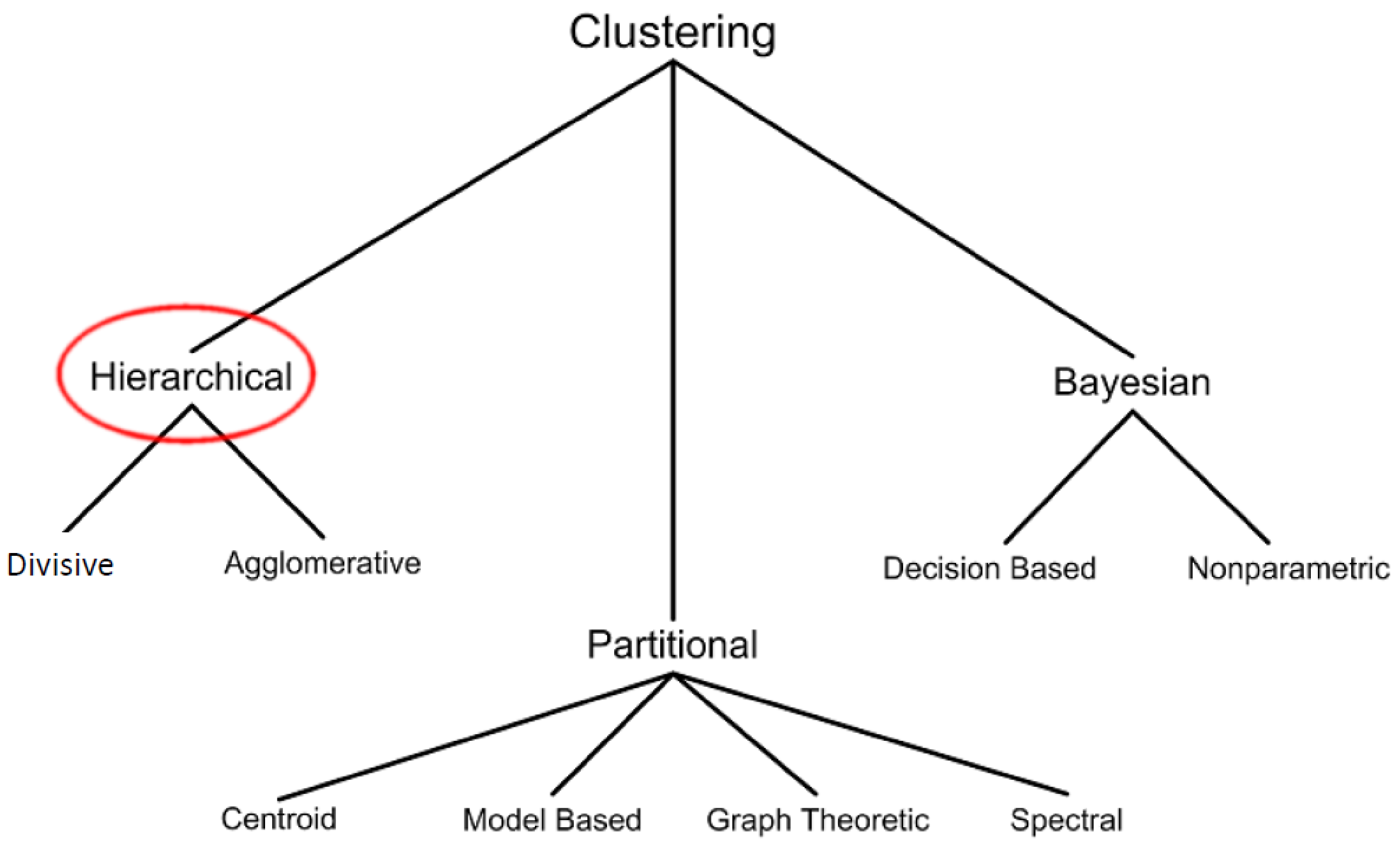
- Ideal clusters



K-Means variations

- K-Medoids – instead of mean, use medians of each cluster
  - Mean of 1, 3, 5, 7, 1009 is 205
  - Median of 1, 3, 5, 7, 1009 is 5
  - Median advantage: not affected by extreme values.

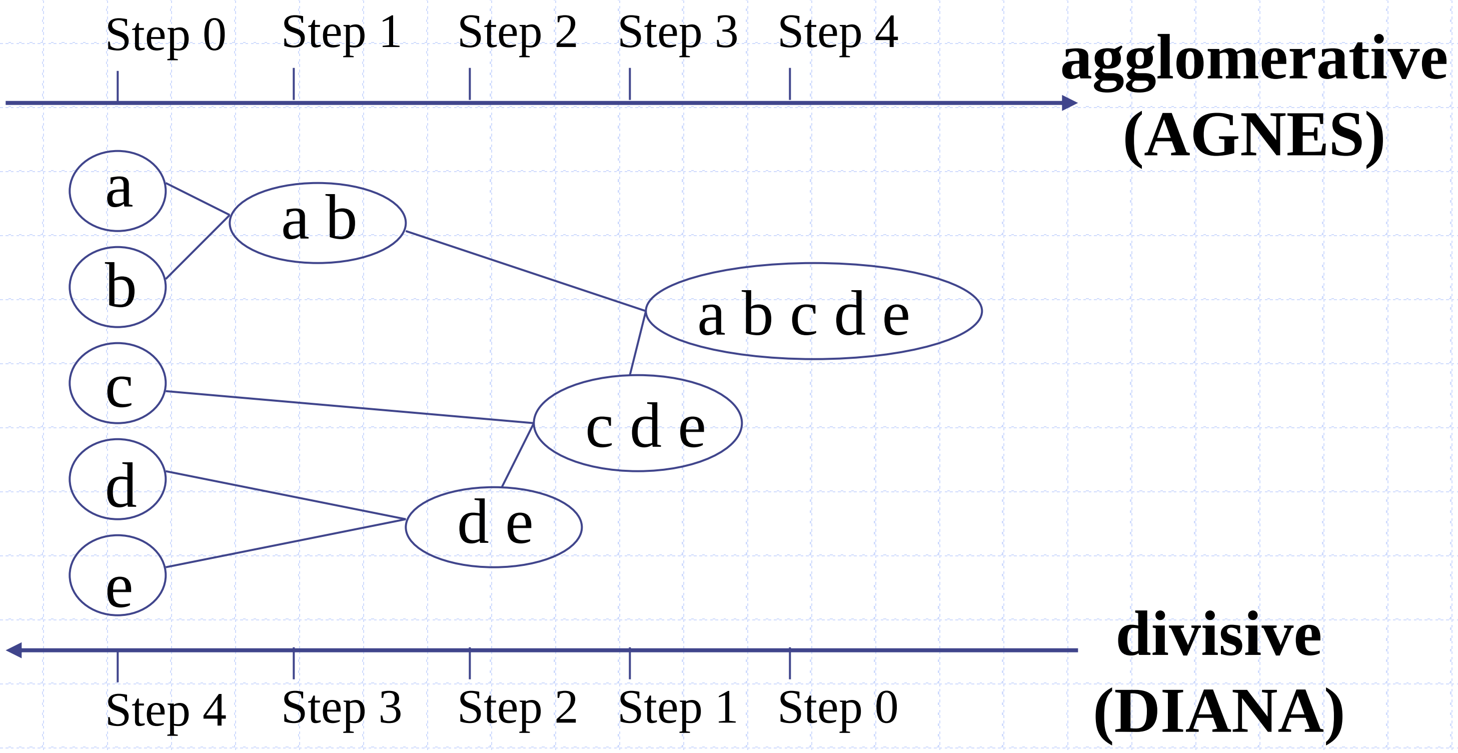
Clustering techniques



Hierarchical Clustering

- Agglomerative (bottom up) clustering
  - Starting with each data point as a separate cluster
  - Merging the most similar (or nearest) pair of clusters
  - Stopping when all the data points are merged into a single cluster or certain termination conditions are satisfied.
- Divisive (top down) clustering
  - Starting with all data points in one cluster, the root, then
  - Splitting the root into a set of child cluster (according to some principle). Each child cluster is recursively divided further.
  - Stopping when each cluster at the lower level is coherent enough – either containing only one object, or the objects within a cluster are sufficiently similar to each other.
- In either agglomerative or divisive hierarchical clustering, a user can specify the desired umber of clusters as a termination condition

Hierarchical Clustering



Distance between Clusters

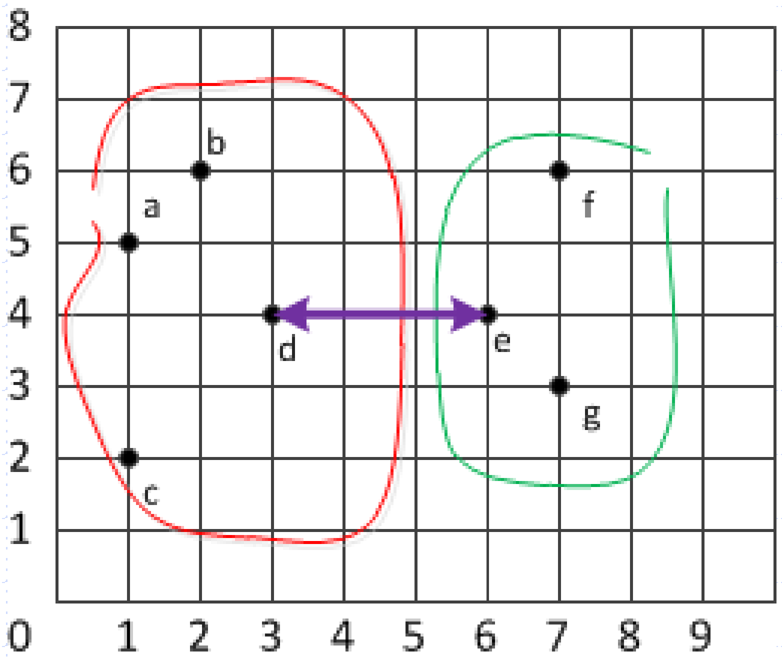
- Minimum distance (single link)
  - $minDis(Ci, Cj) = min||x - y|| \ (x \in Ci, y \in Cj)$
- Maximum distance (complete link)
  - $maxDis(Ci, Cj) = max||x - y|| \ (x \in Ci, y \in Cj)$
- Average distance
  - $avgDis(Ci, Cj) = \frac{1}{n_i n_j} \sum_{x \in Ci} \sum_{y \in Cj} ||x - y||$ ((n_i, n_j are the number of data points)
- Mean distance
  - $meanDis(Ci, Cj) = ||c_i - c_j||$

Ci, Cj represent clusters, *c_i, c_j* represent centroids.

Distance between Clusters

- **Minimum distance (single link):** smallest distance between an element in one cluster and an element in the other.

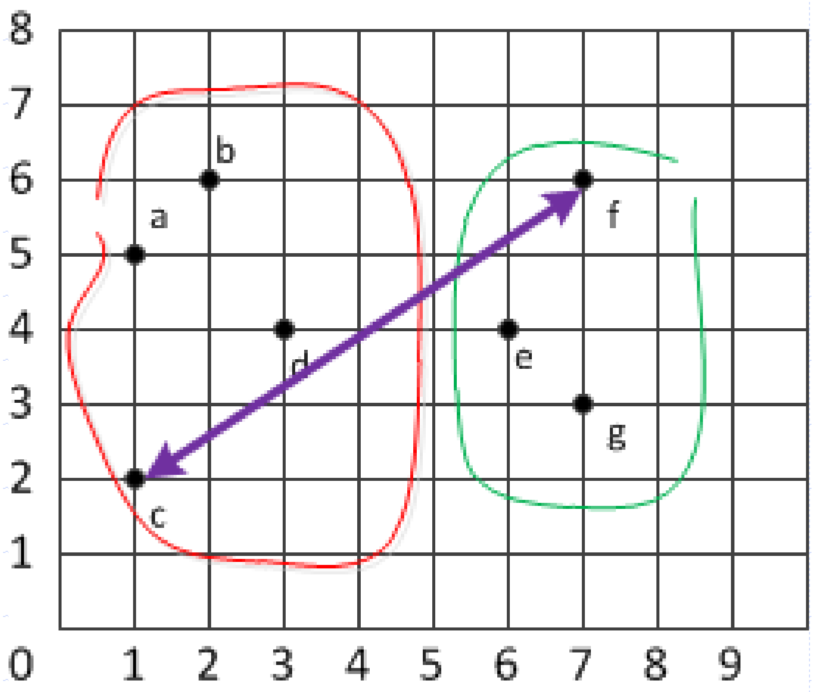
minimum distance = 3
(using Manhattan distance)



Distance between Clusters

- Maximum distance (complete link): largest distance between an element in one cluster and an element in the other.

maximum distance = 10
(using Manhattan distance)



Distance between Clusters

- **Average distance:** average of distances between all pairs of elements.

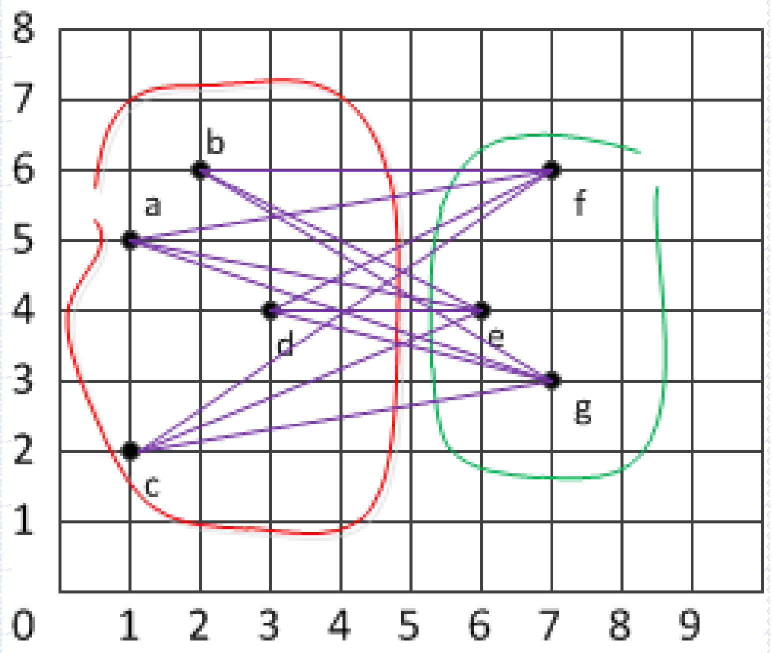
d(a,e) = 6
d(a,f) = 7
d(a,g) = 8

d(b,e) = 6
d(b,f) = 5
d(b,g) = 8

d(c,e) = 7
d(c,f) = 10
d(c,g) = 7

d(d,e) = 3
d(d,f) = 6
d(d,g) = 5

average distance
  = average of all the above
  = 78 / 12
  = 6.5

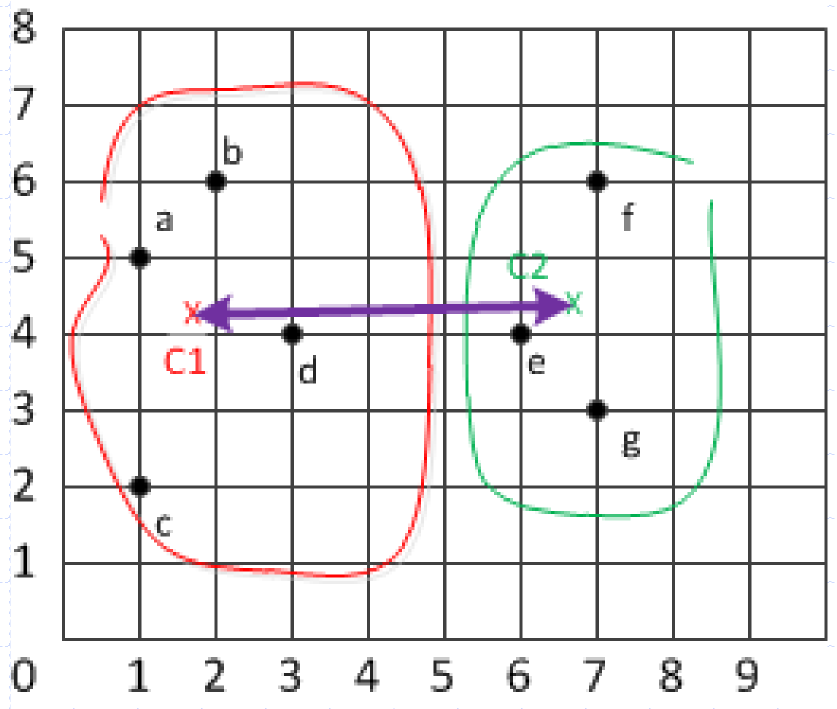


Distance between Clusters

- **Mean distance:** distance between the centroids of two clusters

c1 = (1.75, 4.25)
c2 = (6.67, 4.33)

mean distance = 5.0
(using Manhattan distance)



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!
- Classification and Clustering
  - Both are pattern recognition mechanisms
  - Classification is supervised learning
  - Clustering is unsupervised learning