Clustering

CE-477: Machine Learning - CS-828: Theory of Machine Learning
Sharif University of Technology
Fall 2024

Fatemeh Seyyedsalehi

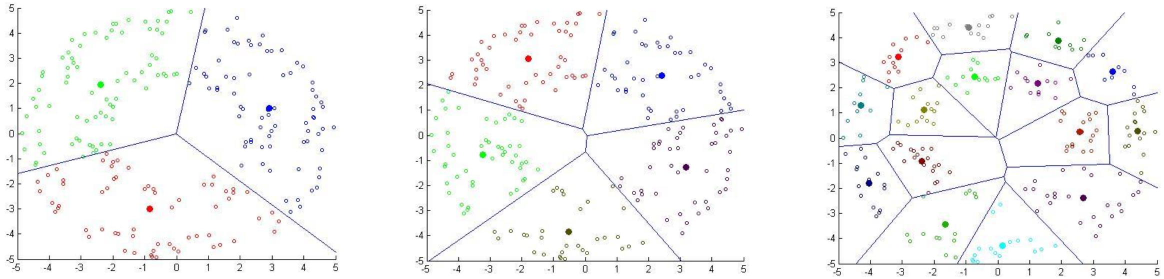
References of the lecture are mentioned in the last slide

Two views on clustering

- We have a set of unlabeled data points $\{\boldsymbol{x}^{(i)}\}_{i=1}^{N}$
- Clusters:
  - Similarity or distance based definition
    - high intra-cluster similarity
    - low inter-cluster similarity
  - Density-based definition:
    - Clusters are regions of high density that are separated from one another by regions of low density

Clustering Purpose

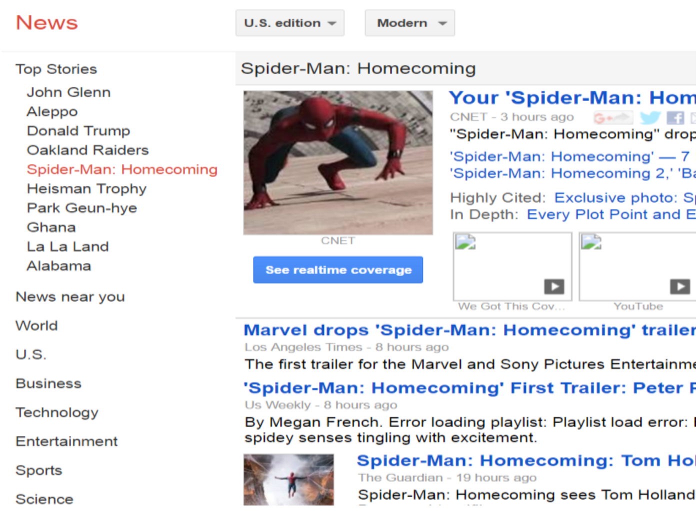
- Automatically organizing data. Ex. Compression, quantization, visualization



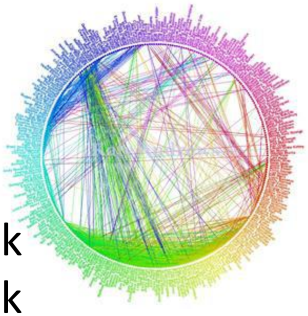
- Knowledge discovery from data: As a tool to **understand the hidden structure** in data or to **group** them
  - To gain insight into the structure of the data (prior to classifier design)
  - Provides information about the internal structure of the data

Clustering Applications

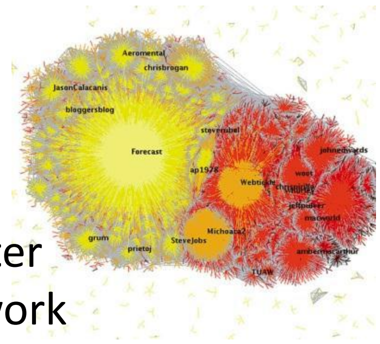
- Cluster news articles or web pages or search results by topic.
  - Google news



- Cluster users of social networks by interest (community detection).



Facebook network

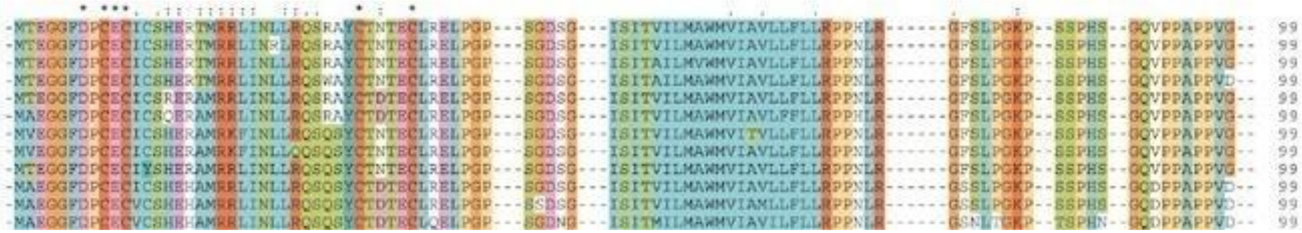


Twitter network

Clustering Applications

- Cluster galaxies or nearby stars (e.g. Sloan Digital Sky Survey)

- Cluster protein sequences by function or genes according to expression profile.



- Market segmentation
  - Clustering customers based on the their purchase history and their characteristics

Clustering methods we will discuss

- Objective based clustering
  - Construct various partitions and then evaluate them by some criterion
    - K-means
    - EM-style algorithm for clustering for mixture of Gaussians (in the next lecture)
- Hierarchical clustering
  - Create a hierarchical decomposition of the set of objects using some criterion

Partitioning Algorithms: Basic Concept

- Construct a partition of a set of N objects into a set of K clusters
  - The number of clusters K is given in advance
  - Each object belongs to **exactly one** cluster in hard clustering methods

- K-means is the most popular partitioning algorithm
  - K-means was proposed near 60 years ago
  - Thousands of clustering algorithms have been published since then
  - However, K-means is still widely used.

K-means Clustering

- **Input**: a set $\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(N)}$ of data points (in a d-dim feature space) and an integer K

- **Output**: a set of K representatives $\boldsymbol{c}_1, \boldsymbol{c}_2, \dots, \boldsymbol{c}_K \in \mathbb{R}^d$ as the cluster representatives
  - data points are assigned to the clusters according to their distances to $\boldsymbol{c}_1, \boldsymbol{c}_2, \dots, \boldsymbol{c}_K$
    - Each data is assigned to the cluster whose representative is nearest to it

- **Objective**: choose $\boldsymbol{c}_1, \boldsymbol{c}_2, \dots, \boldsymbol{c}_K$ to minimize:

$$\sum_{i=1}^{N} \min_{j \in 1,\dots,K} d(\boldsymbol{x}^{(i)}, \boldsymbol{c}_j)$$

Euclidean k-means Clustering

- **Input**: a set $\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(N)}$ of data points (in a d-dim feature space) and an integer K

- **Output**: a set of K representatives $\boldsymbol{c}_1, \boldsymbol{c}_2, \dots, \boldsymbol{c}_K \in \mathbb{R}^d$ as the cluster representatives
  - data points are assigned to the clusters according to their distances to $\boldsymbol{c}_1, \boldsymbol{c}_2, \dots, \boldsymbol{c}_K$
    - Each data is assigned to the cluster whose representative is nearest to it

- **Objective**: choose $\boldsymbol{c}_1, \boldsymbol{c}_2, \dots, \boldsymbol{c}_K$ to minimize:

$$\sum_{i=1}^{N} \min_{j \in 1,\dots,K} \left\| \boldsymbol{x}^{(i)} - \boldsymbol{c}_j \right\|^2$$

each point assigned to its closest cluster representative

Euclidean k-means Clustering: Computational Complexity

- NP hard: even for $k = 2$ or $d = 2$

- For k=1: $\min_{\boldsymbol{c}} \sum_{i=1}^{N} \left\| \boldsymbol{x}^{(i)} - \boldsymbol{c} \right\|^2$
  - $\boldsymbol{c} = \boldsymbol{\mu} = \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{x}^{(i)}$

- For $d = 1$, dynamic programming in time $O(N^2 K)$.

Common Heuristic in Practice: The Lloyd's method

- Input: A set $\mathcal{X}$ of N data points $\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(N)}$ in $\mathbb{R}^d$

- **Initialize** centers $\boldsymbol{c}_1, \boldsymbol{c}_2, \dots, \boldsymbol{c}_K \in \mathbb{R}^d$ in any way.
- **Repeat** until there is no further change in the cost.
  - For each j: $\mathcal{C}_j \leftarrow \{\boldsymbol{x} \in \mathcal{X} \,|\, \text{where } \boldsymbol{c}_j \text{ is the closest center to } \boldsymbol{x}\}$
  - For each j: $\boldsymbol{c}_j \leftarrow$mean of members of $\mathcal{C}_j$

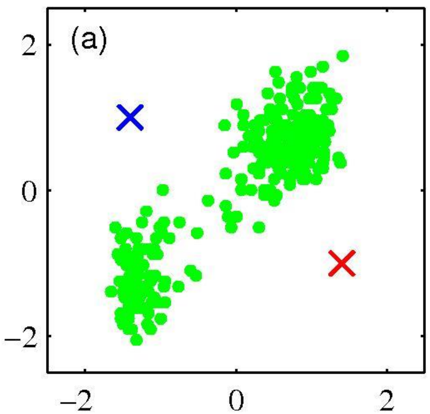
Holding centers $\boldsymbol{c}_1, \boldsymbol{c}_2, \dots, \boldsymbol{c}_K$ fixed
Find optimal assignments $\mathcal{C}_1, \dots, \mathcal{C}_K$ of data points to clusters

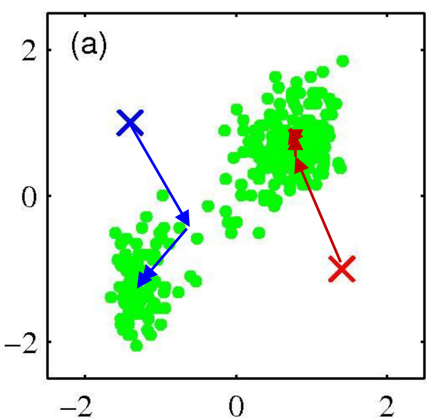
Holding cluster assignments $\mathcal{C}_1, \dots, \mathcal{C}_K$ fixed
Find optimal centers $\boldsymbol{c}_1, \boldsymbol{c}_2, \dots, \boldsymbol{c}_K$

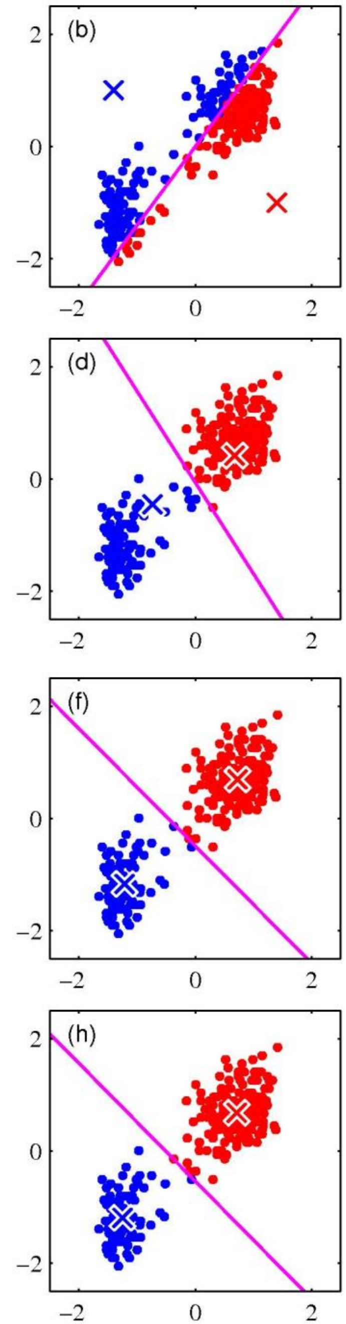
$$\boldsymbol{c}_j = \frac{1}{|\mathcal{C}_j|} \sum_{\boldsymbol{x}^{(i)} \in \mathcal{C}_j} \boldsymbol{x}^{(i)}$$

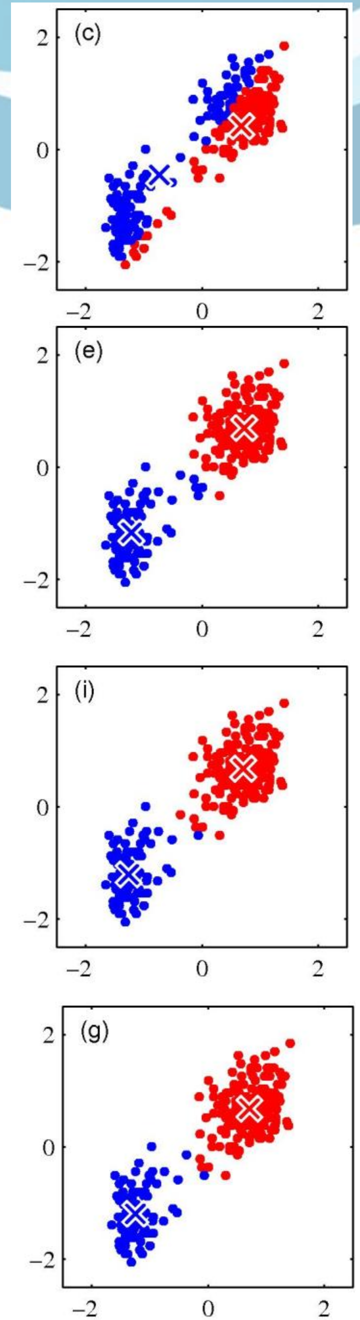
Assigning data to clusters

Updating means









[Bishop]

Intra-cluster similarity view

- k-means optimizes intra-cluster similarity:

$$J(\mathcal{C}) = \sum_{j=1}^{K} \sum_{\boldsymbol{x}^{(i)} \in \mathcal{C}_j} \left\| \boldsymbol{x}^{(i)} - \boldsymbol{c}_j \right\|^2$$

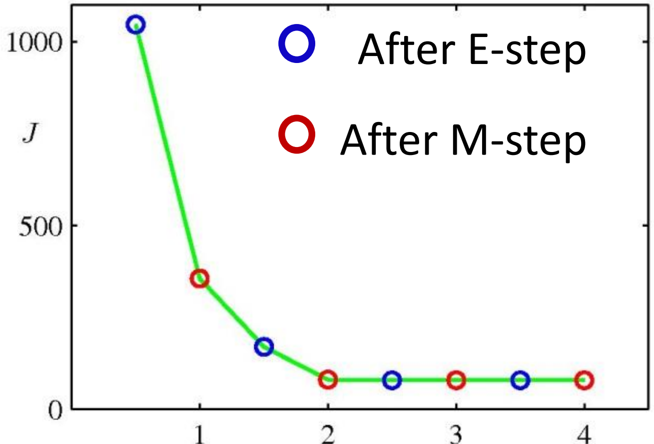
$$\boldsymbol{c}_j = \frac{1}{|\mathcal{C}_j|} \sum_{\boldsymbol{x}^{(i)} \in \mathcal{C}_j} \boldsymbol{x}^{(i)}$$

$$\sum_{\boldsymbol{x}^{(i)} \in \mathcal{C}_j} \left\| \boldsymbol{x}^{(i)} - \boldsymbol{c}_j \right\|^2 = \frac{1}{2|\mathcal{C}_j|} \sum_{\boldsymbol{x}^{(i)} \in \mathcal{C}_j} \sum_{\boldsymbol{x}^{(i')} \in \mathcal{C}_j} \left\| \boldsymbol{x}^{(i)} - \boldsymbol{x}^{(i')} \right\|^2$$

the average distance to members of the same cluster

K-means: Convergence

- It always converges.
  - The cost always drop
  - There is only a finite number of partitioning
- *K*-means algorithm reaches a state in which clustering doesn't change.
  - Reassignment stage monotonically decreases J since each vector is assigned to the closest centroid.
  - Centroid update stage also for each cluster minimizes the sum of squared distances of the assigned points to the cluster from its center.

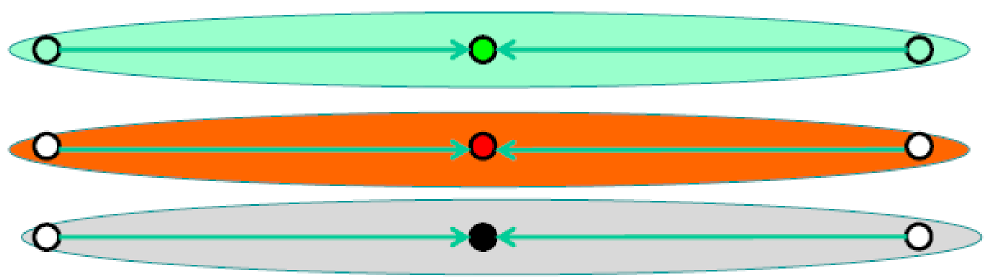


[Bishop]

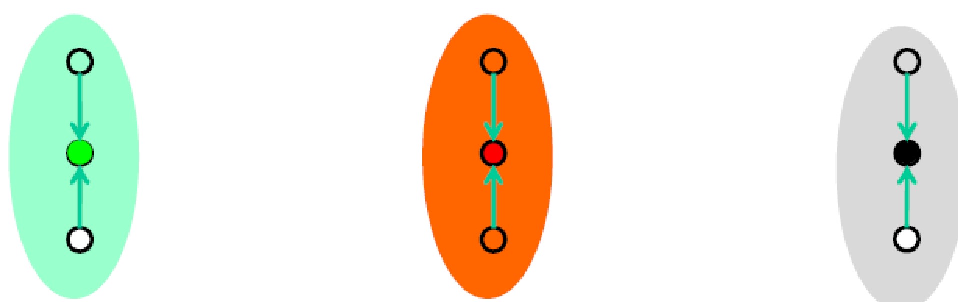
Local optimum problem

- It always converges
  - But it may converge at a local optimum
  - Colored points show initial centroids, two resulted partitioning for two different initial centroids

Local optimum



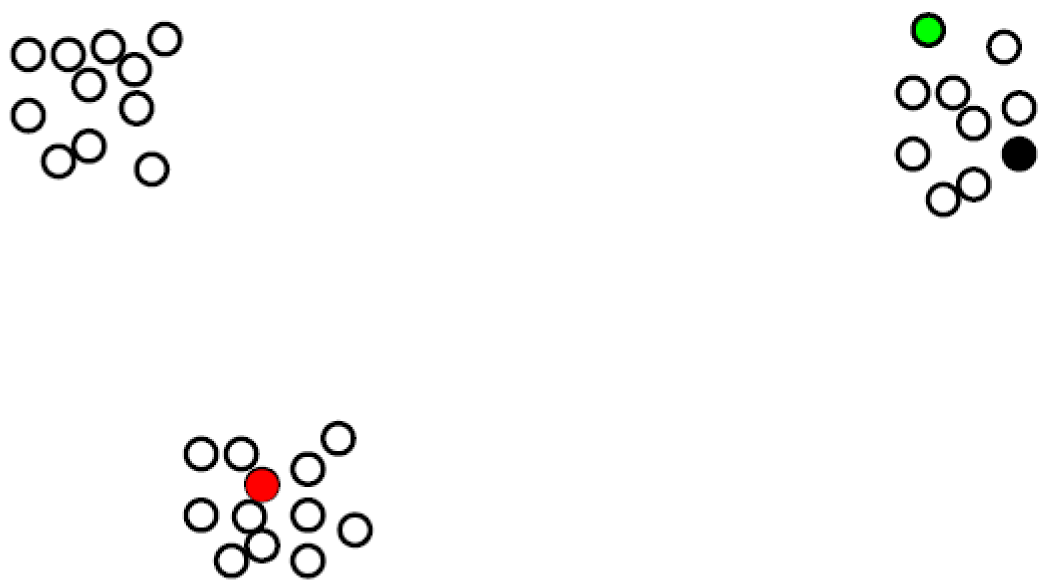
Best partitioning or global optimum



Local optimum problem

- It always converges
  - But it may converge at a local optimum

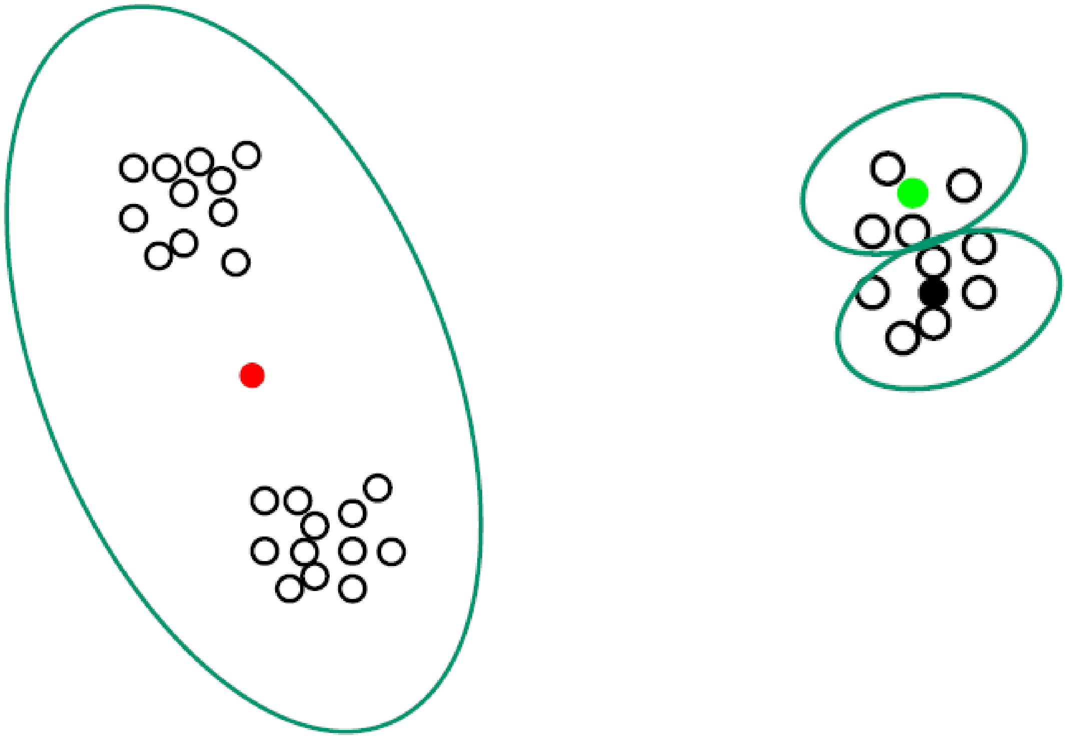
  - This bad performance, can happen even with well separated Gaussian clusters.



Local optimum problem

- It always converges
  - But it may converge at a local optimum

  - This bad performance, can happen even with well separated Gaussian clusters.

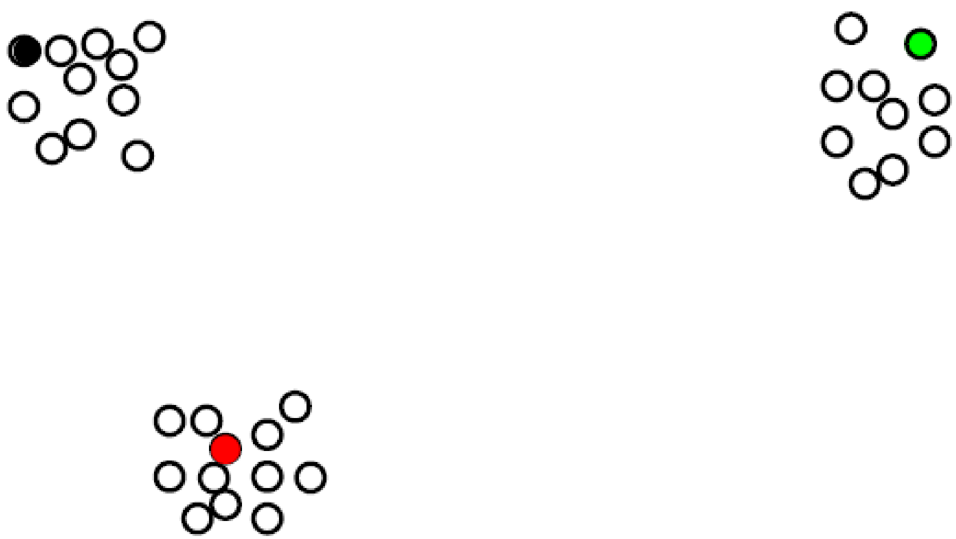


The Lloyd's method: Initialization

- Initialization is crucial (how fast it converges, quality of clustering)
  - Random centers from the data points
    - Multiple runs and select the best ones
  - Initialize with the results of another method
  - Select good initial centers using a heuristic
    - Furthest traversal
    - K-means ++ (works well and has provable gaurantees)

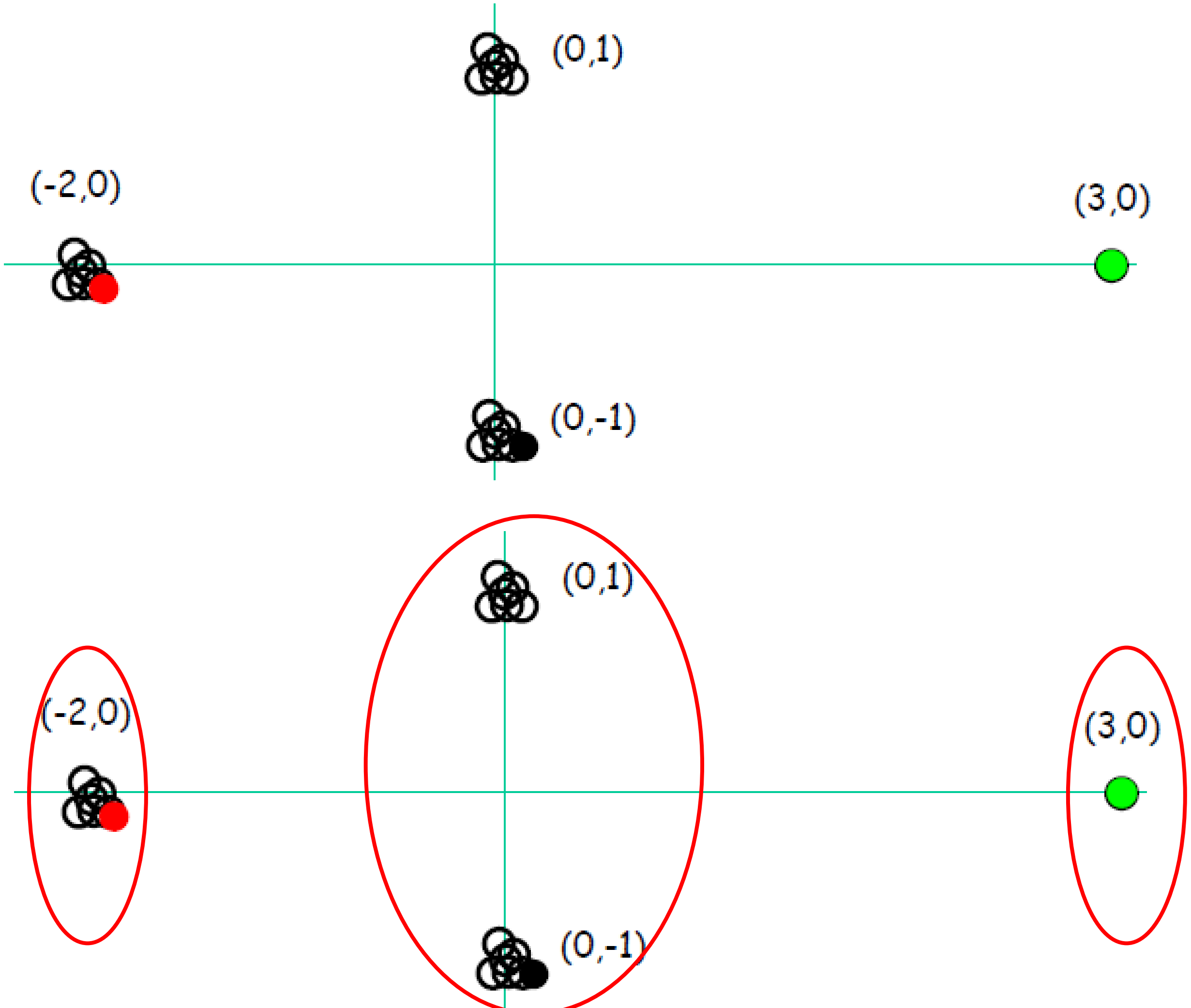
Initialization Idea: Furthest Point Heuristic

- Choose $\boldsymbol{c}_1$ arbitrarily (or at random).
- For $j = 2, \dots, K$
  - Select $\boldsymbol{c}_j$ among datapoints $\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(N)}$ that is farthest from previously chosen $\boldsymbol{c}_1, \dots, \boldsymbol{c}_{j-1}$
- Good for our previous example



Initialization Idea: Furthest Point Heuristic

- However, it is sensitive to outliers



K-means++ Initialization: D2 sampling

[D. Arthur and S. Vassilvitskii, 2007]

- Combine random initialization and furthest point initialization ideas

- Let the probability of selection of the point be proportional to the distance between this point and its nearest center.
  - probability of selecting of $\boldsymbol{x}$ is proportional to $D^2(\boldsymbol{x}) = \min_{k<j} \|\boldsymbol{x} - \boldsymbol{c}_k\|^2$.

- Choose $\boldsymbol{c}_1$ arbitrarily (or at random).

- For $j = 2, \dots, K$
  - Select $\boldsymbol{c}_j$ among data points $\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(N)}$ according to the distribution:

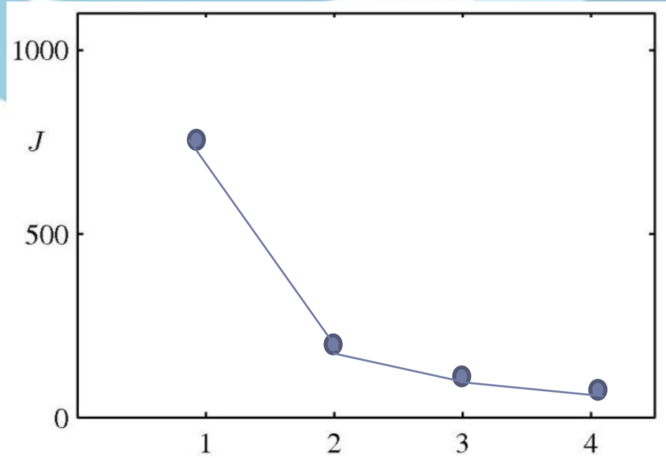
$$\Pr(\boldsymbol{c}_j = \boldsymbol{x}^{(i)}) \propto \min_{k<j} \left\|\boldsymbol{x}^{(i)} - \boldsymbol{c}_k\right\|^2$$

How Many Clusters?

- Number of clusters k is given in advance in the k-means algorithm
  - However, finding the “right” number of clusters is a part of the problem

- Tradeoff between having better focus within each cluster and having too many clusters

How Many Clusters?



- Heuristic:
  - Find large gap between $k-1$-means cost and k-means cost.
  - "knee finding" or "elbow finding".
- Hold-out validation/cross-validation on auxiliary task (e.g., supervised learning task).
- Optimization problem: penalize having lots of clusters
  - some criteria can be used to automatically estimate k
    - Penalize the number of bits you need to describe the extra parameter

$$J'(\mathcal{C}) = J(\mathcal{C}) + |\mathcal{C}| \times \log N$$

- Hierarchical clustering

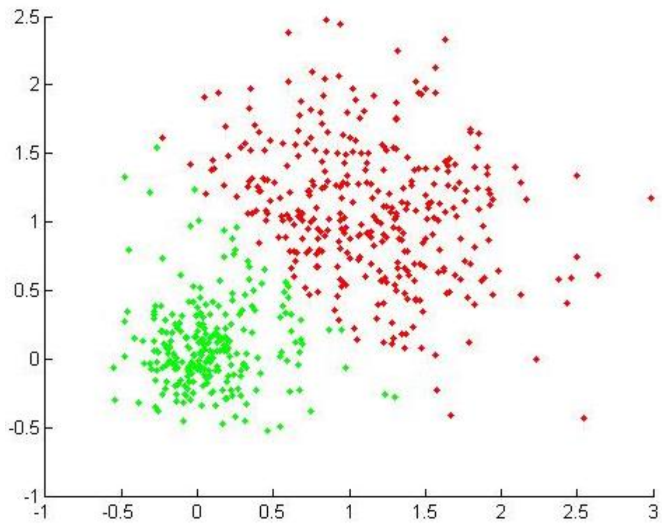
K-means summary

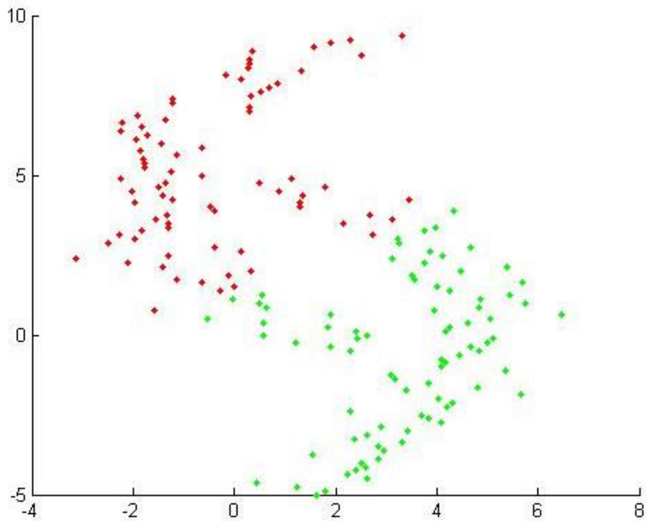
- Relatively efficient: $O(tKNd)$, where t is the number of iterations.
  - *K*-means typically converges quickly
    - Usually $t \ll n$.
  - Exponential # of rounds in the worst case [Andrea Vattani 2009].

- Limitations
  - Need to specify *K,* the *number* of clusters, in advance
  - Often terminates at a *local optimum*.
    - Initialization is important.
  - Not suitable to discover clusters with arbitrary shapes
  - Works for numerical data. What about categorical data?
  - Noise and outliers can be considerable trouble to *K*-means

k-means summary

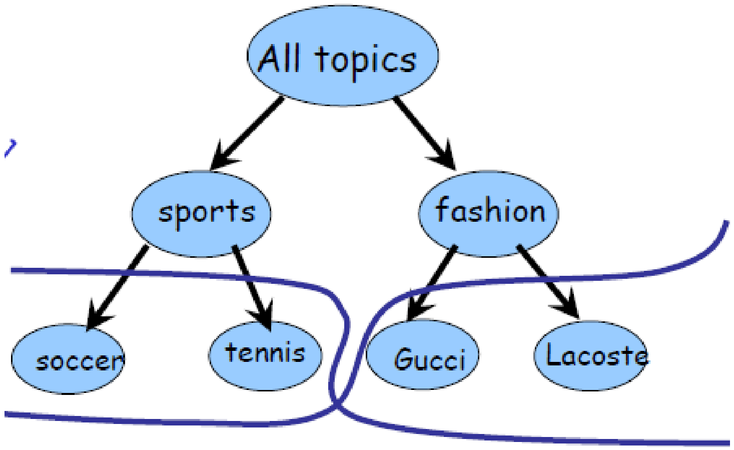
- In general, k-means is unable to find clusters of arbitrary shapes, sizes, and densities
  - Except to very distant clusters

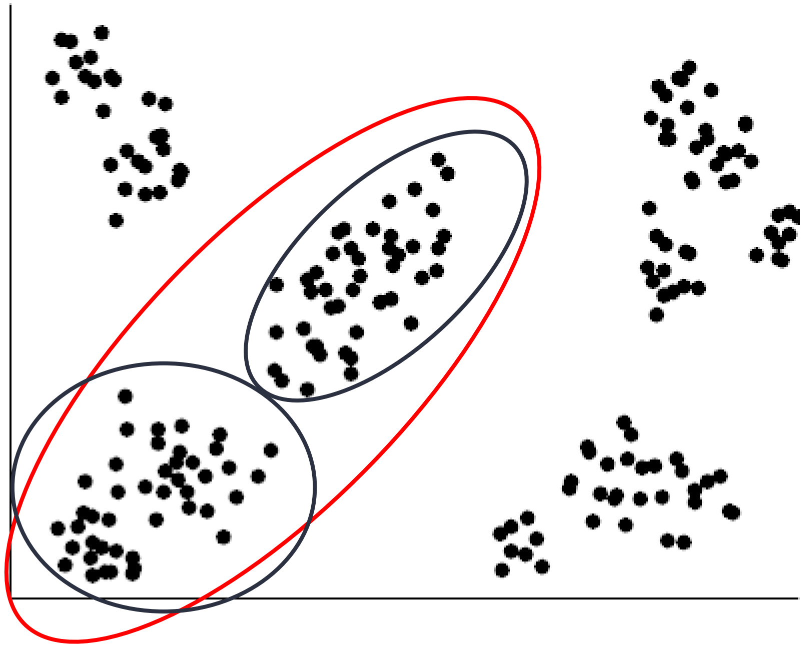




Hierarchical Clustering

- Hierarchical Clustering: Clusters contain sub-clusters and sub-clusters themselves can have sub-sub-clusters, and so on
  - Several levels of details in clustering
- A hierarchy might be more natural.
  - Different levels of granularity





Hierarchical Clustering

- Agglomerative (bottom up):
  - Starts with each data in a separate cluster
  - Repeatedly joins the closest pair of clusters, until there is only one cluster (or other stopping criteria).
    - Different definition for distance between clusters

- Divisive (top down):
  - Starts with the whole data as a cluster
  - Repeatedly divide data in one of the clusters until there is only one data in each cluster (or other stopping criteria).

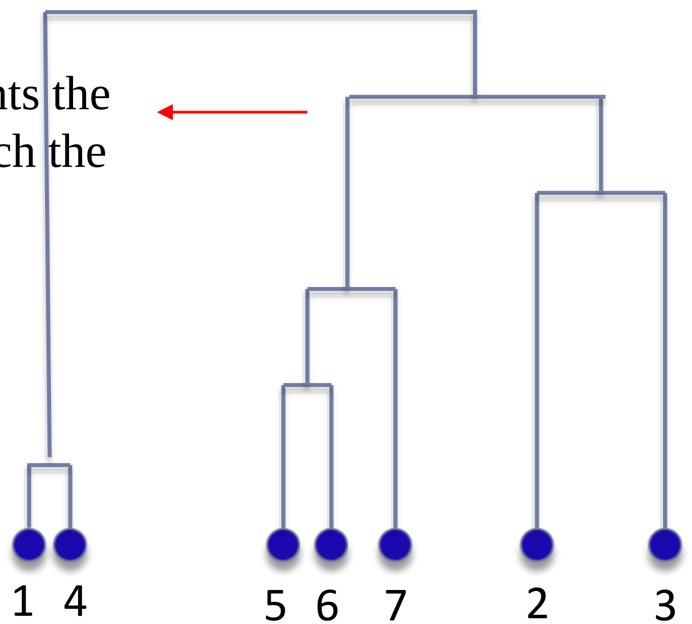
Agglomerative (bottom up):

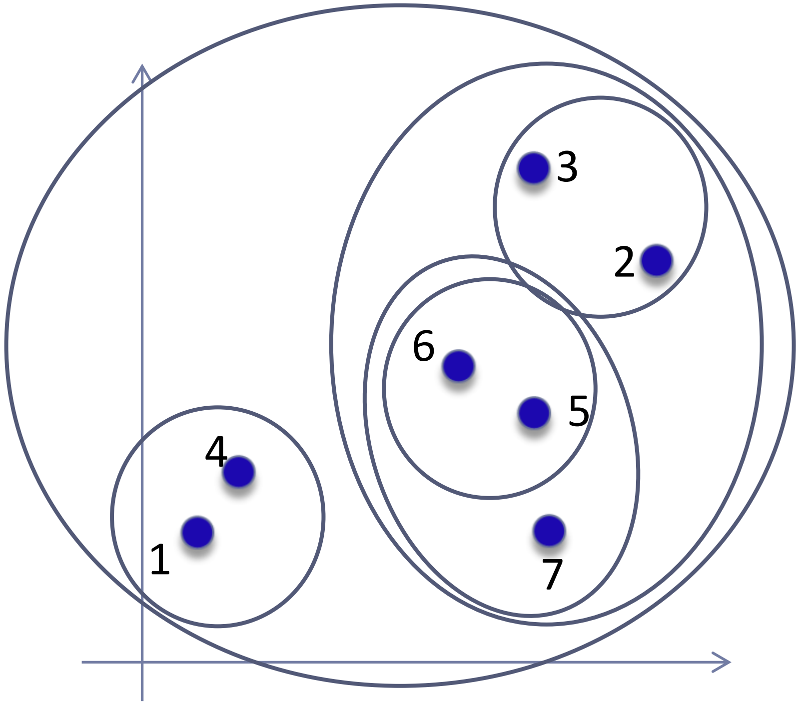
1. Initially, each instance forms a cluster
2. While there are more than one cluster

   Pick the two closest one

   Merge them into a new cluster

Height represents the distance at which the merge occurs



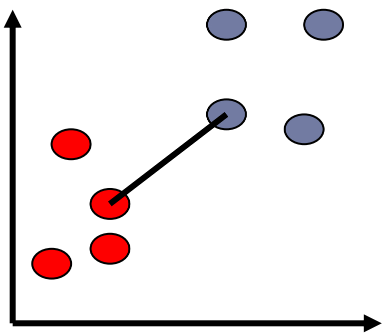


Agglomerative (bottom up):

- Having a distance measure $dist_{SL}(x, y)$ on a pair of objects, many variants exist to define distances between pair of clusters

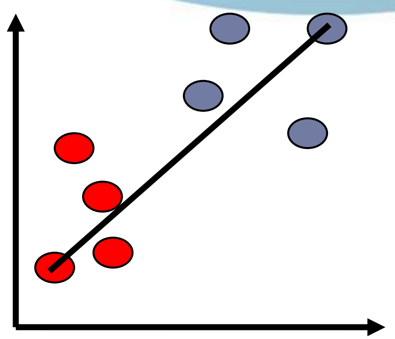
  - **Single-link**
    - Minimum distance between different pairs of data
  - **Complete-link**
    - Maximum distance between different pairs of data
  - **Centroid (Ward's)**
    - Distance between centroids (centers of gravity)
  - **Average-link**
    - Average distance between pairs of elements

Distances between Cluster Pairs



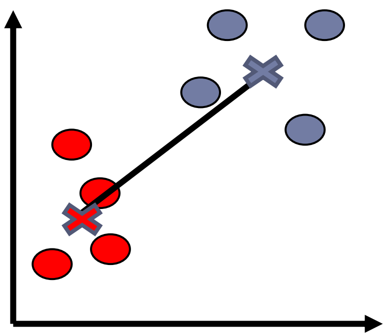
Single-link

$$dist_{SL}(\mathcal{C}_i, \mathcal{C}_j) = \min_{\boldsymbol{x} \in \mathcal{C}_i,\, \boldsymbol{x}' \in \mathcal{C}_j} dist(\boldsymbol{x}, \boldsymbol{x}')$$



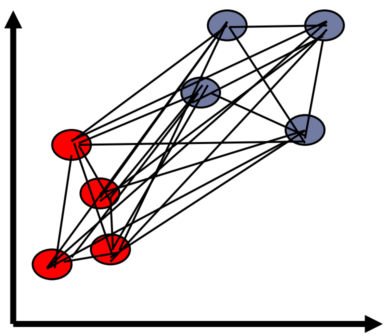
Complete-link

$$dist_{CL}(\mathcal{C}_i, \mathcal{C}_j) = \max_{\boldsymbol{x} \in \mathcal{C}_i,\, \boldsymbol{x}' \in \mathcal{C}_j} dist(\boldsymbol{x}, \boldsymbol{x}')$$



Ward's

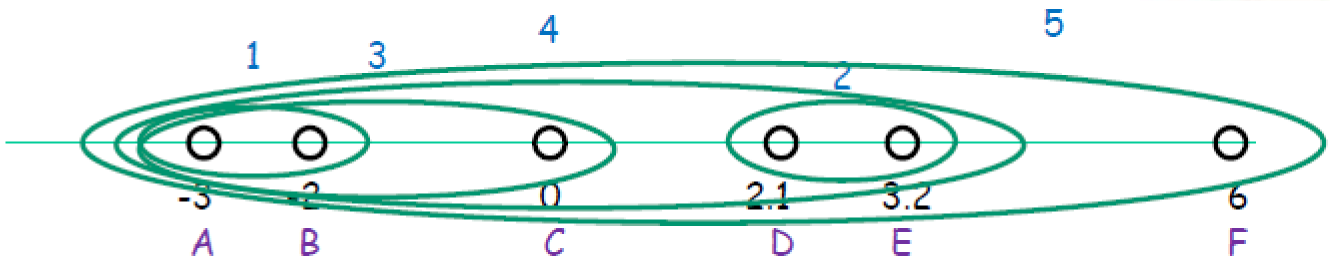
$$dist_{Ward}(\mathcal{C}_i, \mathcal{C}_j) = \frac{|\mathcal{C}_i||\mathcal{C}_j|}{|\mathcal{C}_i| + |\mathcal{C}_j|} dist(\boldsymbol{c}_i, \boldsymbol{c}_j)$$



Average-link

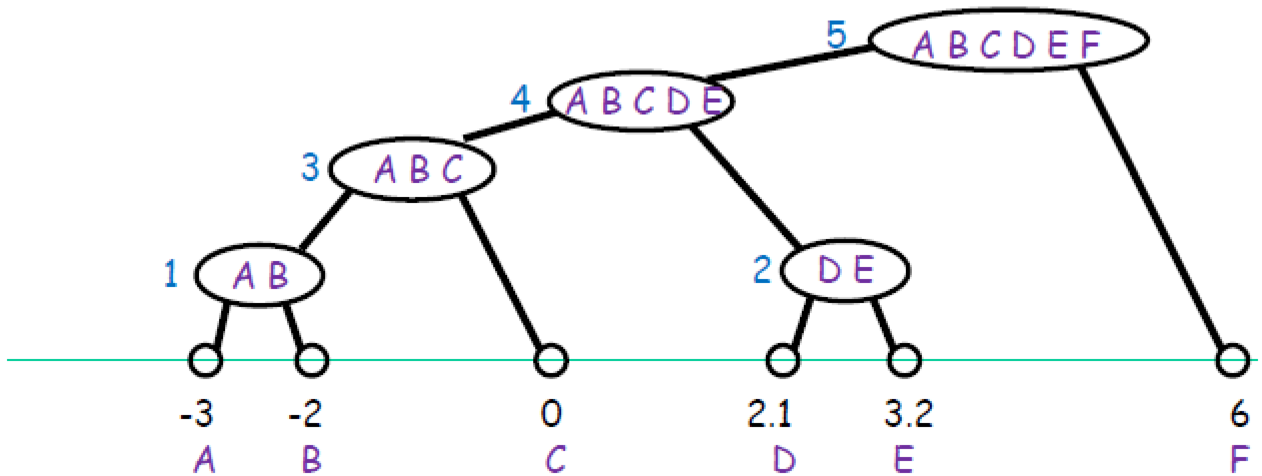
$$dist_{AL}(\mathcal{C}_i, \mathcal{C}_j) = \frac{1}{|\mathcal{C}_i \cup \mathcal{C}_j|} \sum_{\boldsymbol{x} \in \mathcal{C}_i \cup \mathcal{C}_j} \sum_{\boldsymbol{x}' \in \mathcal{C}_i \cup \mathcal{C}_j} dist(\boldsymbol{x}, \boldsymbol{x}')$$

Single-Link

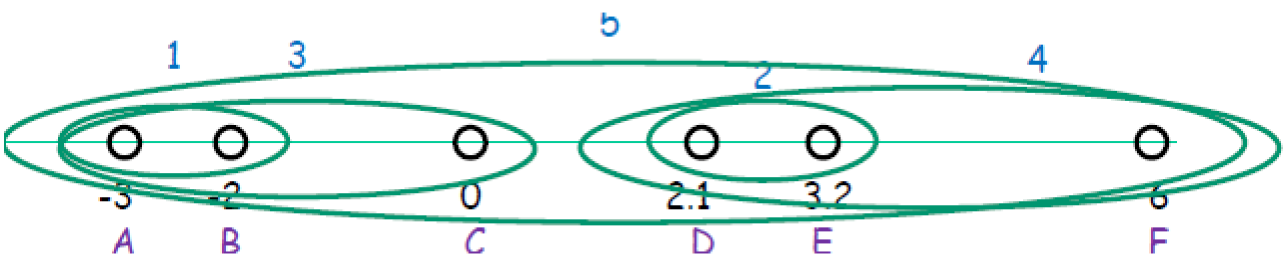


Single linkage can produce long stretched clusters.

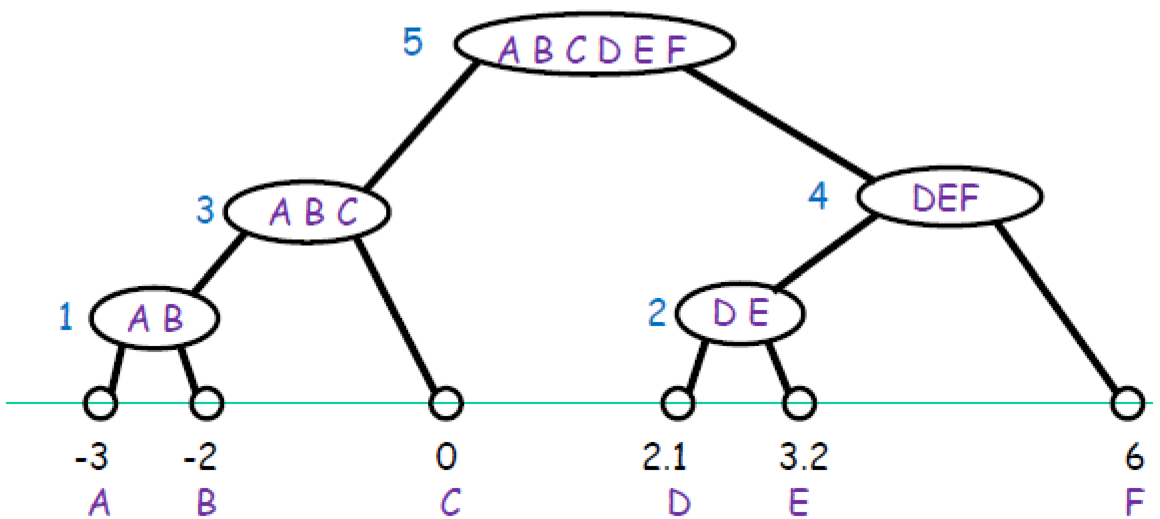
Dendogram



Complete Link



One way to think of it: keep max diameter as small as possible at any level.
Complete linkage prefers compact clusters.

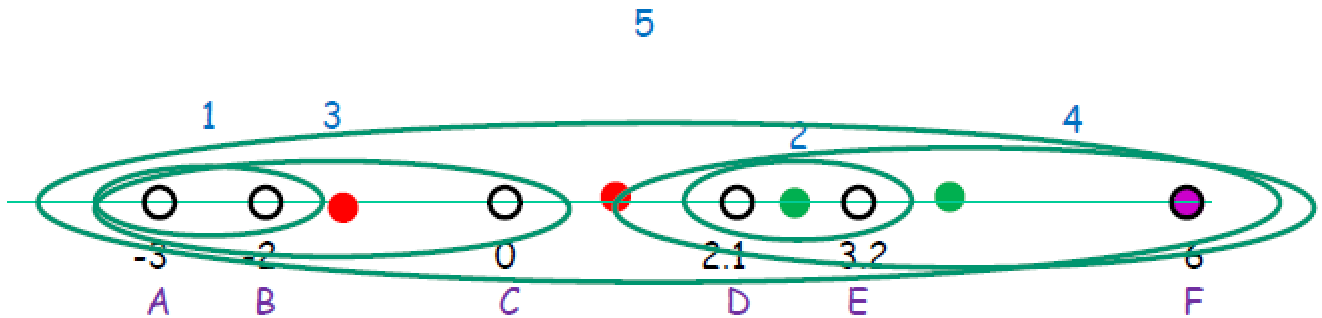


Ward's method

- The distances between centers of the two clusters (weighted to consider sizes of clusters too):

$$dist_{Ward}(\mathcal{C}_i, \mathcal{C}_j) = \frac{|\mathcal{C}_i||\mathcal{C}_j|}{|\mathcal{C}_i| + |\mathcal{C}_j|} dist(\boldsymbol{c}_i, \boldsymbol{c}_j)$$

- Merge the two clusters such that the increase in k-means cost is as small as possible.
- Works well in practice.



Computational Complexity

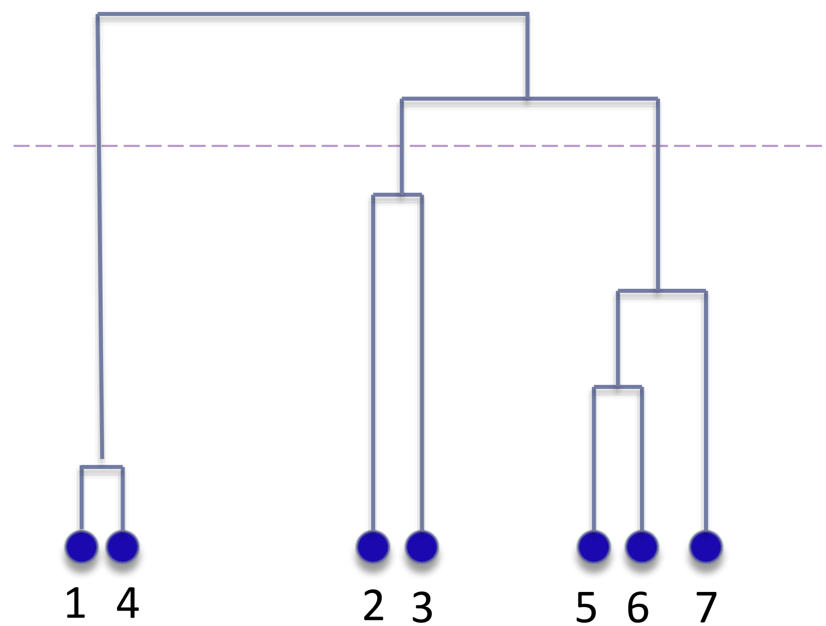
- In the first iteration, all HAC methods compute similarity of all pairs of N individual instances which is $O(N^2)$ similarity computation.

- In each $N-1$ merging iterations, compute the distance between the most recently created cluster and all other existing clusters.

- If done naively $O(N^3)$ but if done more cleverly $O(N^2 \log N)$

Dendrogram: Hierarchical Clustering

- Clustering obtained by cutting the dendrogram at a desired level
  - Cut at a pre-specified level of similarity
  - where the gap between two successive combination similarities is largest
  - select the cutting point that produces *K* clusters



K-means vs. Hierarchical

- Time cost:
  - K-means is usually fast while hierarchical methods do not scale well
- Human intuition
  - Hierarchical structure provides more natural output compatible with human intuition in some domains
- Local minimum problem
  - It is very common for k-means
  - Hierarchical methods like any heuristic search algorithms also suffer from local optima problem.
    - Since they can never undo what was done previously and greedily merge clusters
- Choosing of the number of clusters
  - There is no need to specify the number of clusters in advance for hierarchical methods