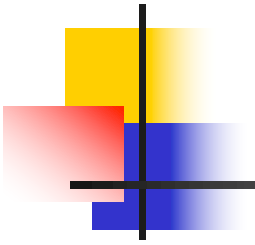
CSCI 4520 – Introduction to Machine Learning

Spring 2020

Mehdi Allahyari
Georgia Southern University

Clustering

(slides borrowed from Tom Mitchell, Maria Florina Balcan, Ali Borji, Ke Chen)

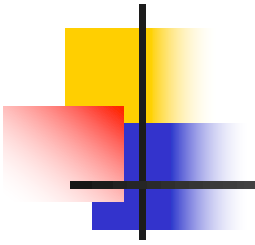


Clustering, Informal Goals

Goal: Automatically partition unlabeled data into groups of similar datapoints.

Question: When and why would we want to do this?

Useful for:



Clustering, Informal Goals

Goal: Automatically partition unlabeled data into groups of similar datapoints.

Question: When and why would we want to do this?

Useful for:

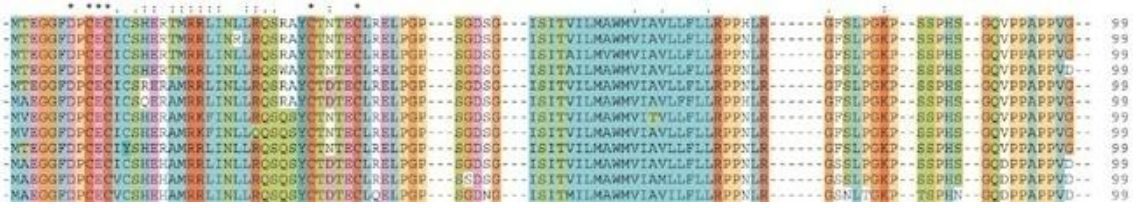
- Automatically organizing data.
- Understanding hidden structure in data.
- Preprocessing for further analysis.
  - Representing high-dimensional data in a low-dimensional space (e.g., for visualization purposes).

Applications

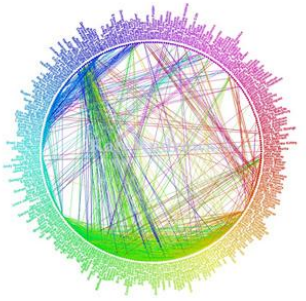
- Cluster news articles or web pages or search results by topic.



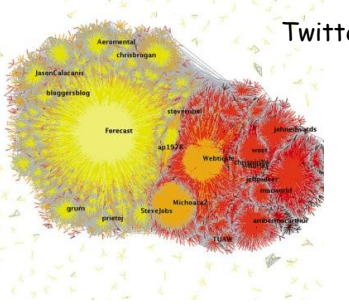
- Cluster protein sequences by function or genes according to expression profile.

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



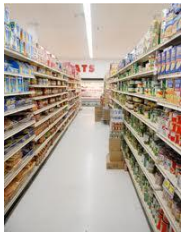
Facebook network



Twitter Network

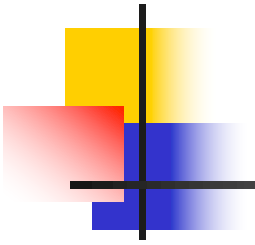
Applications

- Cluster customers according to purchase history.



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

- And many many more applications….



Clustering

Groups together "similar" instances in the data sample

Basic clustering problem:

- distribute data into *k* different groups such that data points similar to each other are in the same group
- Similarity between data points is defined in terms of some distance metric (can be chosen)

Clustering is useful for:

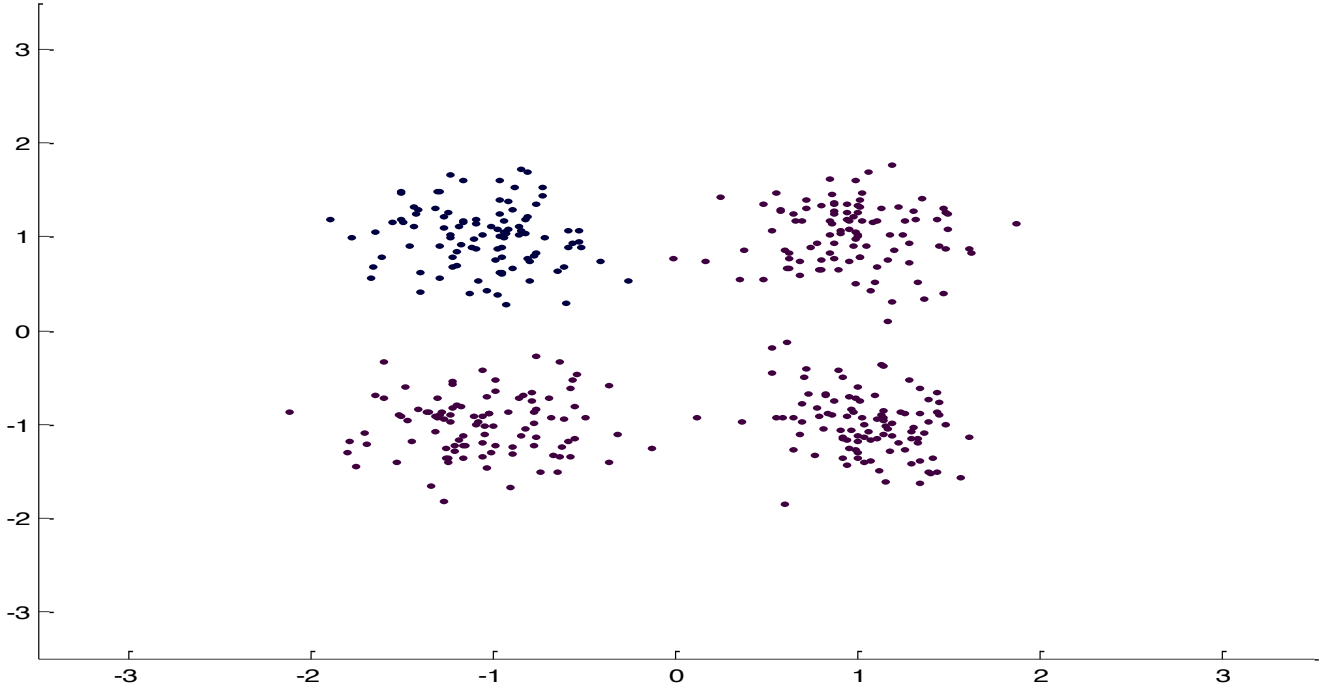
- **Similarity/Dissimilarity analysis**

  Analyze what data points in the sample are close to each other
- **Dimensionality reduction**

  High dimensional data replaced with a group (cluster) label

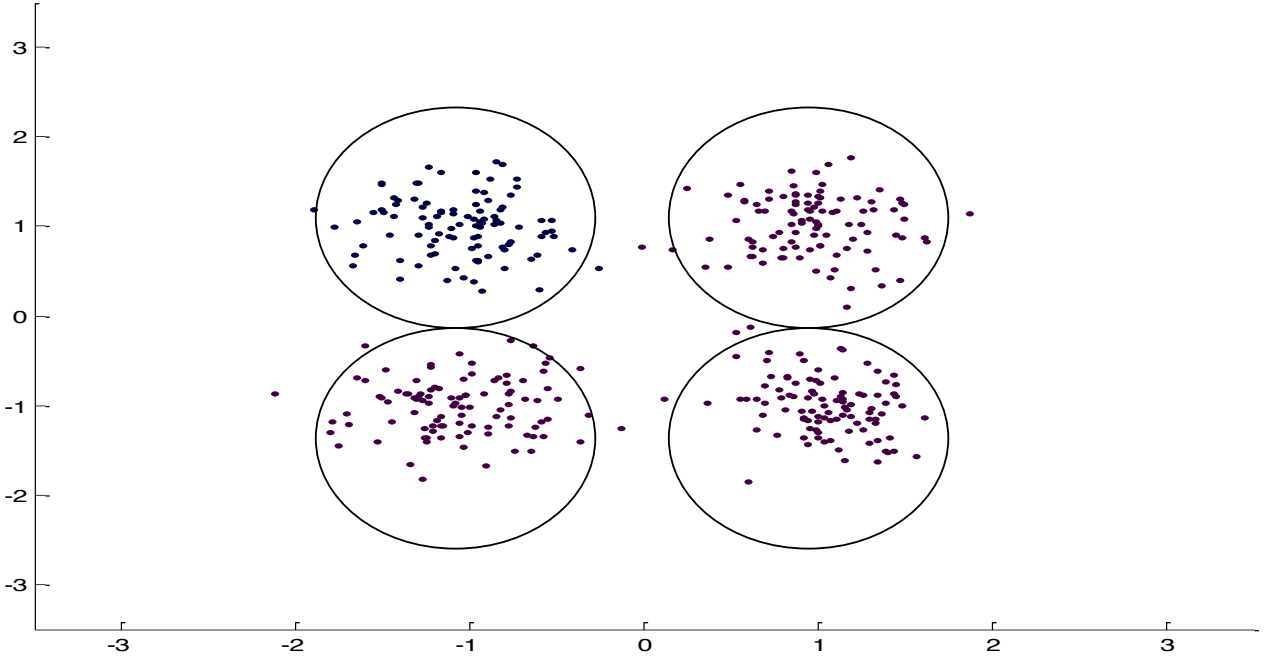
Example

- We see data points and want to partition them into groups
- Which data points belong together?



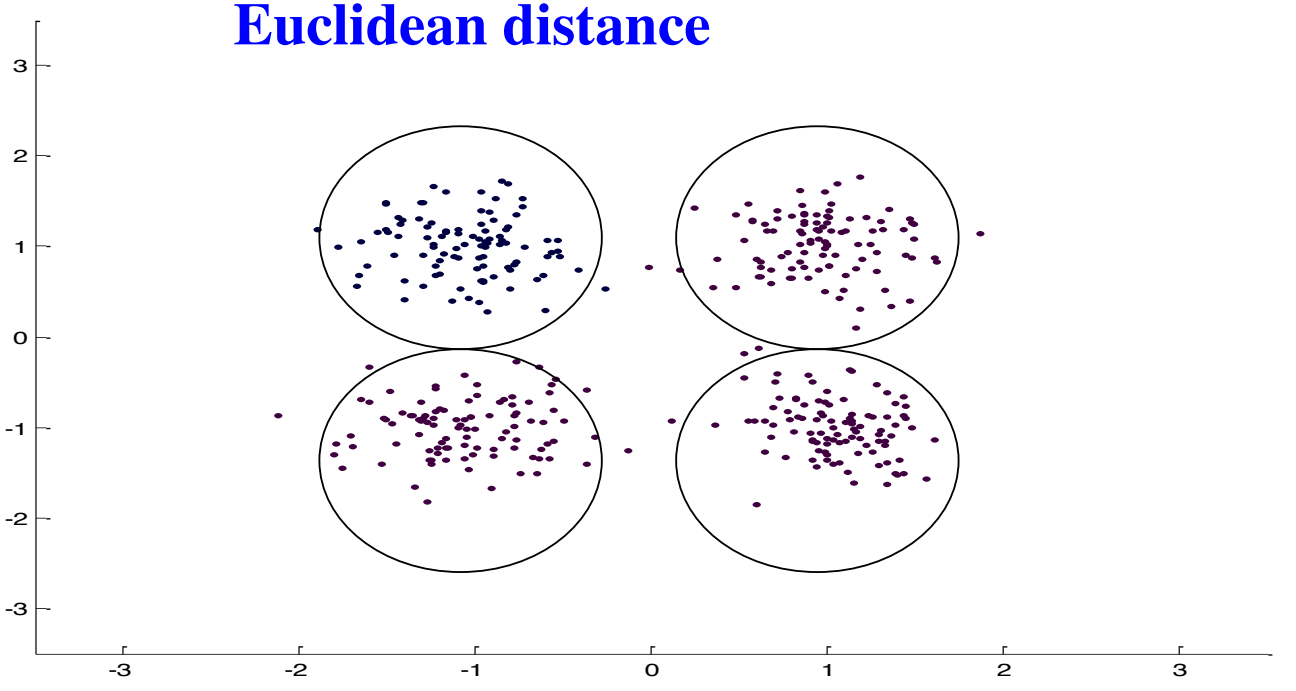
Example

- We see data points and want to partition them into the groups
- Which data points belong together?



Example

- We see data points and want to partition them into the groups
- Requires a distance metric to tell us what points are close to each other and are in the same group



Example

- A set of patient cases
- We want to partition them into groups based on similarities

Patient #	Age	Sex	Heart Rate	Blood pressure …
Patient 1	55	M	85	125/80
Patient 2	62	M	87	130/85
Patient 3	67	F	80	126/86
Patient 4	65	F	90	130/90
Patient 5	70	M	84	135/85

Example

- A set of patient cases
- We want to partition them into the groups based on similarities

Patient #	Age	Sex	Heart Rate	Blood pressure …
Patient 1	55	M	85	125/80
Patient 2	62	M	87	130/85
Patient 3	67	F	80	126/86
Patient 4	65	F	90	130/90
Patient 5	70	M	84	135/85

How to design the distance metric to quantify similarities?

Clustering Example. Distance Measures

In general, one can choose an arbitrary distance measure.

Properties of distance metrics:

Assume 2 data entries *a*, *b*

Positiveness: $d(a,b) \geq 0$

Symmetry: $d(a,b) = d(b,a)$

Identity: $d(a,a) = 0$

Triangle inequality: $d(a,c) \leq d(a,b) + d(b,c)$

Distance Measures

Assume pure real-valued data-points:

12	34.5	78.5	89.2	19.2
23.5	41.4	66.3	78.8	8.9
33.6	36.7	78.3	90.3	21.4
17.2	30.1	71.6	88.5	12.5
…				

What distance metric to use?

Distance Measures

Assume pure real-valued data-points:

12	34.5	78.5	89.2	19.2
23.5	41.4	66.3	78.8	8.9
33.6	36.7	78.3	90.3	21.4
17.2	30.1	71.6	88.5	12.5
…				

What distance metric to use?

Euclidian: works for an arbitrary k-dimensional space

$$d(a,b) = \sqrt{\sum_{i=1}^{k} (a_i - b_i)^2}$$

Distance Measures

Assume pure real-valued data-points:

12	34.5	78.5	89.2	19.2
23.5	41.4	66.3	78.8	8.9
33.6	36.7	78.3	90.3	21.4
17.2	30.1	71.6	88.5	12.5

What distance metric to use?

Squared Euclidian: works for an arbitrary k-dimensional space

$$d^2(a,b) = \sum_{i=1}^{k} (a_i - b_i)^2$$

Distance Measures

Assume pure real-valued data-points:

12	34.5	78.5	89.2	19.2
23.5	41.4	66.3	78.8	8.9
33.6	36.7	78.3	90.3	21.4
17.2	30.1	71.6	88.5	12.5

Manhattan distance:

works for an arbitrary k-dimensional space

$$d(a,b) = \sum_{i=1}^{k} | a_i - b_i |$$

Etc. ..

Clustering Algorithms

- **K-means algorithm**
  - **suitable** only when data points have continuous values; groups are defined in terms of cluster centers (also called **means**). Refinement of the method to categorical values: **K-medoids**
- **Probabilistic methods (with EM)**
  - **Latent variable models**: class (cluster) is represented by a latent (hidden) variable value
  - Every point goes to the class with the highest posterior
  - **Examples:** mixture of Gaussians, Naïve Bayes with a hidden class
- **Hierarchical methods**
  - **Agglomerative**
  - **Divisive**

Introduction

- Partitioning Clustering Approach
  - a typical clustering analysis approach via iteratively partitioning training data set to learn a partition of the given data space
  - learning a partition on a data set to produce several non-empty clusters (usually, the number of clusters given in advance)
  - in principle, optimal partition achieved via minimizing the sum of squared distance to its "representative object" in each cluster

$$E = \Sigma_{k=1}^{K} \Sigma_{\mathbf{x} \in C_k} d^2(\mathbf{x}, \mathbf{m}_k)$$

e.g., Euclidean distance $d^2(\mathbf{x}, \mathbf{m}_k) = \sum_{n=1}^{N} (x_n - m_{kn})^2$

Introduction

- Given a *K*, find a partition of *K clusters* to optimize the chosen partitioning criterion (cost function)
  - global optimum: exhaustively search all partitions

- The *K-means* algorithm: a heuristic method
  - K-means algorithm (MacQueen'67): each cluster is represented by the center of the cluster and the algorithm converges to stable centriods of clusters.
  - K-means algorithm is the simplest partitioning method for clustering analysis and widely used in data mining applications.

K-means Algorithm

- Given the cluster number *K*, the *K-means* algorithm is carried out in three steps after initialization:
- Initialisation: set seed points (randomly)
  - Assign each object to the cluster of the nearest seed point measured with a specific distance metric
  - Compute new seed points as the centroids of the clusters of the current partition (the centroid is the center, i.e., *mean point*, of the cluster)
  - Go back to Step 1), stop when no more new assignment (i.e., membership in each cluster no longer changes)

K-means Clustering

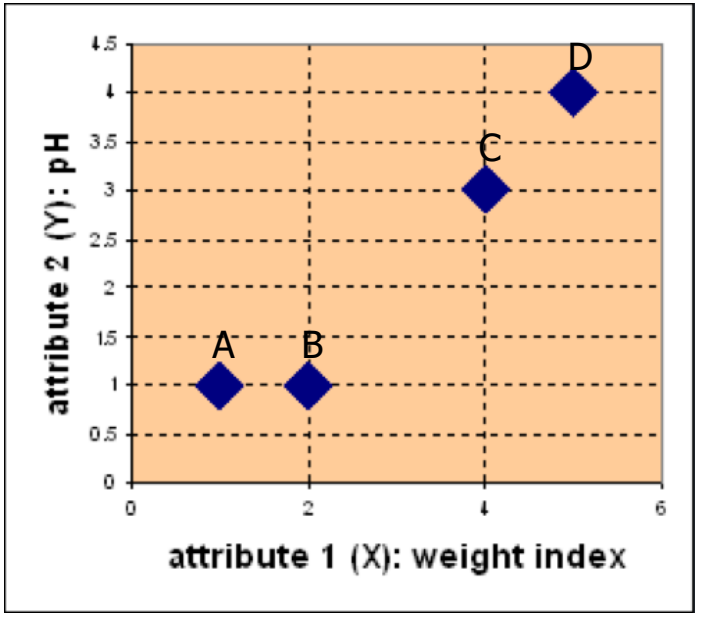
- Choose a number of clusters *k*
- Initialize cluster centers $\mu_1, \ldots \mu_k$
  - Could pick *k* data points and set cluster centers to these points
  - Or could randomly assign points to clusters and take means of clusters
- For each data point, compute the cluster center it is closest to (using 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

Example

- **Problem**

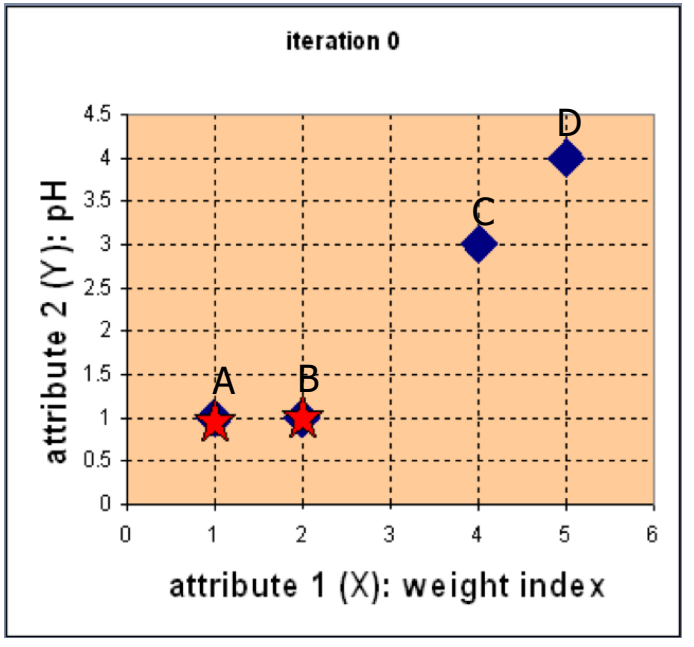
  Suppose we have 4 types of medicines and each has two attributes (pH and weight index). Our goal is to group these objects into *K=2* group of medicine.

Medicine	Weight	pH-Index
A	1	1
B	2	1
C	4	3
D	5	4



Example

- Step 1: Use initial seed points for partitioning



$$c_1 = A, c_2 = B$$

$$D^0 = \begin{bmatrix} 0 & 1 & 3.61 & 5 \\ 1 & 0 & 2.83 & 4.24 \end{bmatrix} \quad \begin{matrix} c_1 = (1,1) & group-1 \\ c_2 = (2,1) & group-2 \end{matrix}$$

Euclidean distance

$$\begin{matrix} A & B & C & D \end{matrix}$$

$$\begin{bmatrix} 1 & 2 & 4 & 5 \\ 1 & 1 & 3 & 4 \end{bmatrix} \quad \begin{matrix} X \\ Y \end{matrix}$$

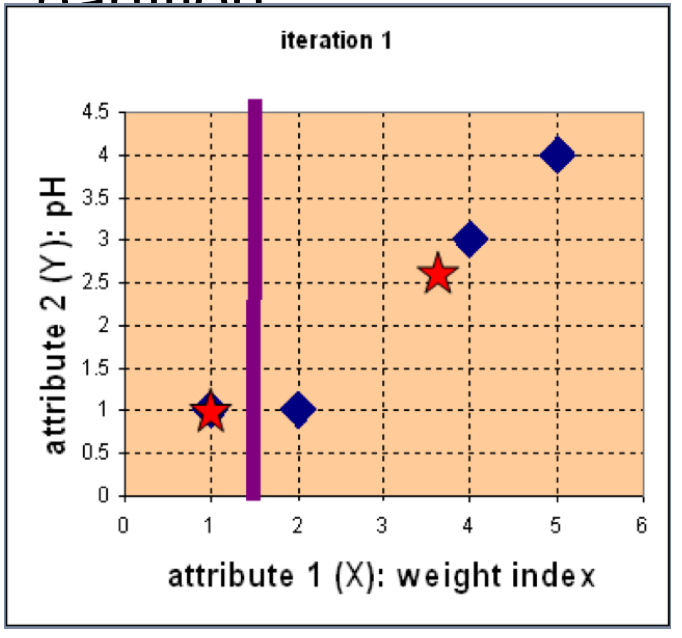
$$d(D, c_1) = \sqrt{(5-1)^2 + (4-1)^2} = 5$$

$$d(D, c_2) = \sqrt{(5-2)^2 + (4-1)^2} = 4.24$$

Assign each object to the cluster with the nearest seed point

Example

- Step 2: Compute new centroids of the current partition



Knowing the members of each cluster, now we compute the new centroid of each group based on these new memberships.

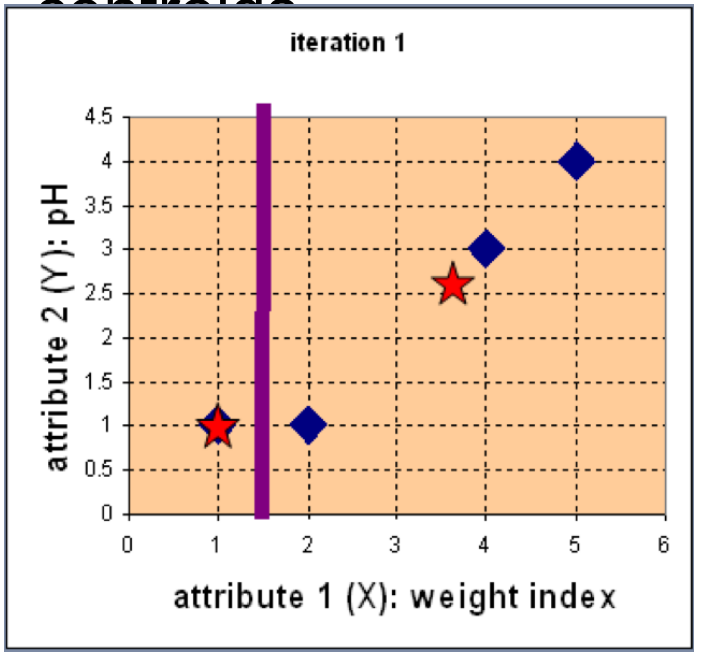
$$c_1 = (1,\ 1)$$

$$c_2 = \left(\frac{2+4+5}{3},\ \frac{1+3+4}{3}\right)$$

$$= \left(\frac{11}{3},\ \frac{8}{3}\right)$$

Example

- Step 2: Renew membership based on new centroids



Compute the distance of all objects to the new centroids

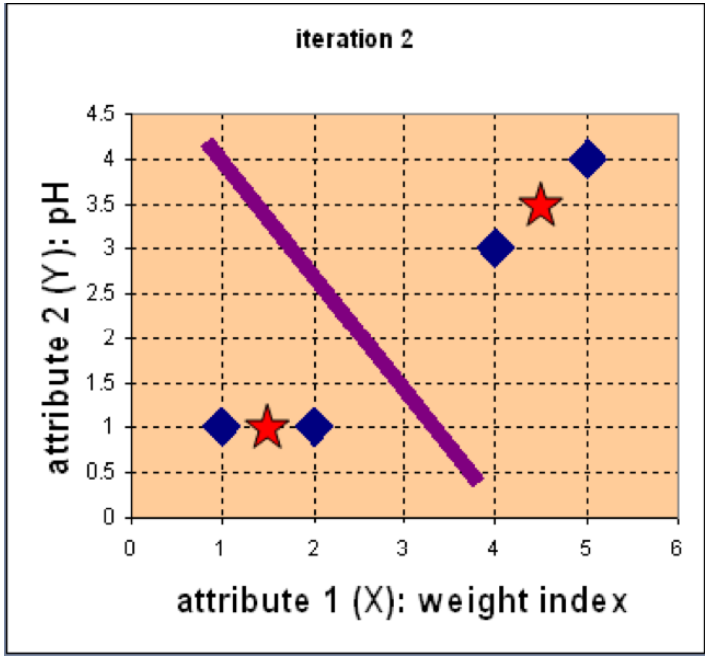
$$D^1 = \begin{bmatrix} 0 & 1 & 3.61 & 5 \\ 3.14 & 2.36 & 0.47 & 1.89 \end{bmatrix} \quad \begin{matrix} c_1 = (1,1) & group-1 \\ c_2 = (\frac{11}{3}, \frac{8}{3}) & group-2 \end{matrix}$$

$$\begin{matrix} A & B & C & D \\ \begin{bmatrix} 1 & 2 & 4 & 5 \\ 1 & 1 & 3 & 4 \end{bmatrix} & \begin{matrix} X \\ Y \end{matrix} \end{matrix}$$

Assign the membership to objects

Example

- Step 3: Repeat the first two steps until its



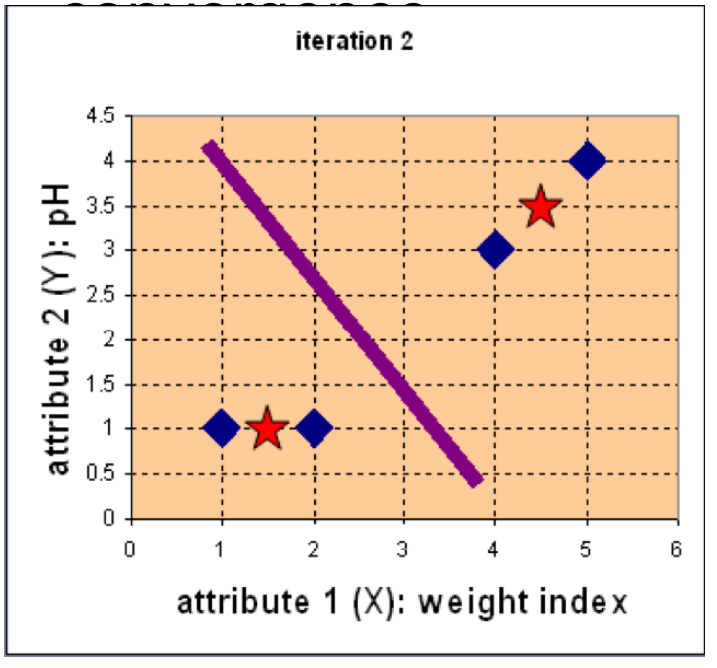
Knowing the members of each cluster, now we compute the new centroid of each group based on these new memberships.

$$c_1 = \left(\frac{1+2}{2}, \frac{1+1}{2}\right) = (1\frac{1}{2}, 1)$$

$$c_2 = \left(\frac{4+5}{2}, \frac{3+4}{2}\right) = (4\frac{1}{2}, 3\frac{1}{2})$$

Example

- Step 3: Repeat the first two steps until its



Compute the distance of all objects to the new centroids

$$D^2 = \begin{bmatrix} 0.5 & 0.5 & 3.20 & 4.61 \\ 4.30 & 3.54 & 0.71 & 0.71 \end{bmatrix} \quad \begin{matrix} c_1 = (1\frac{1}{2}, 1) & group-1 \\ c_2 = (4\frac{1}{2}, 3\frac{1}{2}) & group-2 \end{matrix}$$

$$\begin{matrix} A & B & C & D \end{matrix}$$

$$\begin{bmatrix} 1 & 2 & 4 & 5 \\ 1 & 1 & 3 & 4 \end{bmatrix} \begin{matrix} X \\ Y \end{matrix}$$

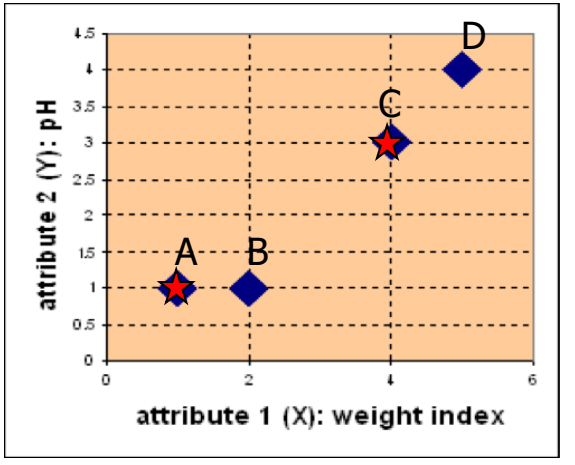
Stop due to no new assignment
Membership in each cluster no longer change

Exercise

For the medicine data set, use K-means with the Manhattan distance metric for clustering analysis by setting $K=2$ and initialising seeds as C_1 = A and C_2 = C. Answer three questions as follows:

1. How many steps are required for convergence?
2. What are memberships of two clusters after convergence?
3. What are centroids of two clusters after convergence?

Medicine	Weight	pH-Index
A	1	1
B	2	1
C	4	3
D	5	4



Euclidean k-means Clustering

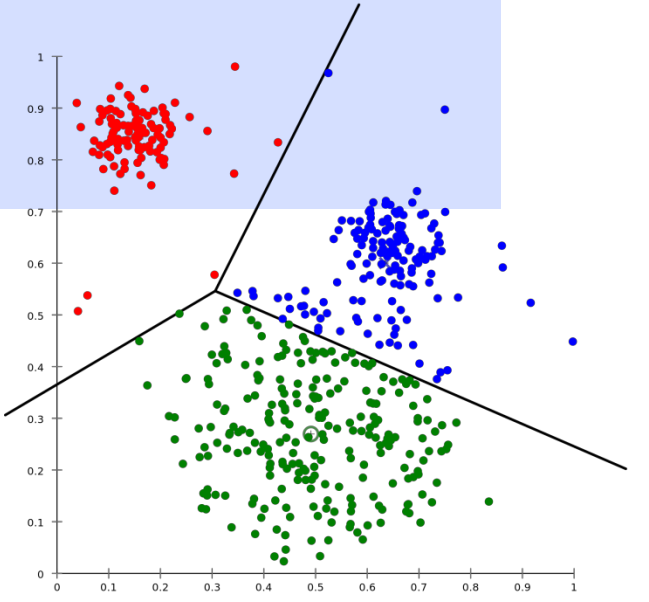
Input: A set of n datapoints $\mathbf{x^1}, \mathbf{x^2}, \ldots, \mathbf{x^n}$ in $\mathrm{R^d}$

target #clusters k

Output: k representatives $\boldsymbol{c_1}, \mathbf{c_2}, \ldots, \boldsymbol{c_k} \in \mathrm{R^d}$

Objective: choose $\boldsymbol{c_1}, \mathbf{c_2}, \ldots, \boldsymbol{c_k} \in \mathrm{R^d}$ to minimize

$$\sum_{i=1}^{n} \min_{j \in \{1,\ldots,k\}} \left|\left|\mathbf{x^i} - \mathbf{c_j}\right|\right|^2$$



Euclidean k-means Clustering

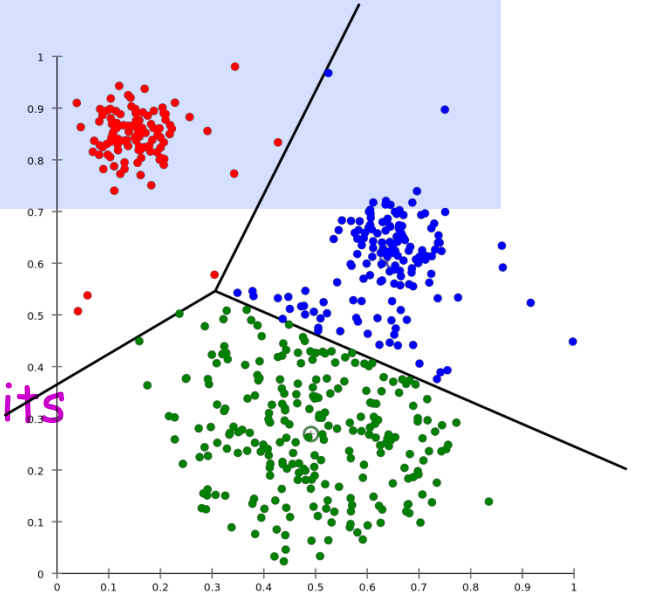
Input: A set of n datapoints $\mathbf{x^1}, \mathbf{x^2}, \ldots, \mathbf{x^n}$ in $\mathrm{R^d}$
target #clusters k

Output: k representatives $\boldsymbol{c_1}, \mathbf{c_2}, \ldots, \boldsymbol{c_k} \in \mathrm{R^d}$

Objective: choose $\boldsymbol{c_1}, \mathbf{c_2}, \ldots, \boldsymbol{c_k} \in \mathrm{R^d}$ to minimize

$$\sum_{i=1}^{n} \min_{j \in \{1,\ldots,k\}} \left|\left| \mathbf{x^i} - \mathbf{c_j} \right|\right|^2$$

Natural assignment: each point assigned to its closest center, leads to a Voronoi partition.



Euclidean k-means Clustering

Input: A set of n datapoints $\mathbf{x^1}, \mathbf{x^2}, \ldots, \mathbf{x^n}$ in R^d

target #clusters k

Output: k representatives $\boldsymbol{c_1}, \mathbf{c_2}, \ldots, \boldsymbol{c_k} \in \mathrm{R}^d$

Objective: choose $\boldsymbol{c_1}, \mathbf{c_2}, \ldots, \boldsymbol{c_k} \in \mathrm{R}^d$ to minimize

$$\sum_{i=1}^{n} \min_{j \in \{1,\ldots,k\}} \left|\left|\mathbf{x^i} - \mathbf{c_j}\right|\right|^2$$

Computational complexity:

NP hard: even for $k = 2$ [Dagupta'08] or

$d = 2$ [Mahajan-Nimbhorkar-Varadarajan09]

There are a couple of easy cases…

An Easy Case for k-means: k=1

Input: A set of n datapoints $\mathbf{x^1}, \mathbf{x^2}, \ldots, \mathbf{x^n}$ in R^d

Output: $\boldsymbol{c} \in R^d$ to minimize $\sum_{i=1}^{n} \left|\left|\mathbf{x^i} - \mathbf{c}\right|\right|^2$

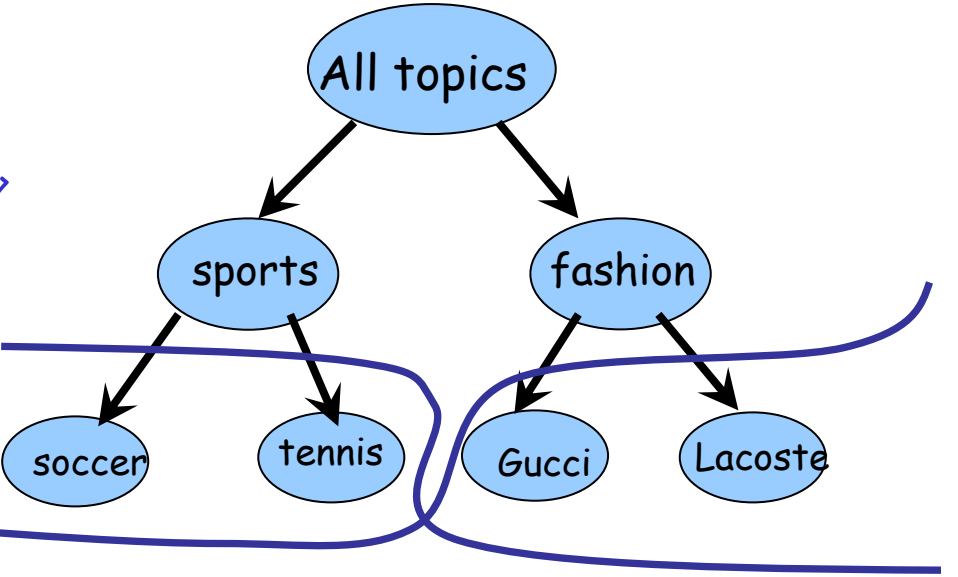
Solution: The optimal choice is $\boldsymbol{\mu} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{x^i}$

So, the optimal choice for $\mathbf{c}$ is $\boldsymbol{\mu}$.

k-means Clustering Issues

- Computational complexity
  - O(*tKn*), where *n* is number of objects, *K* is number of clusters, and *t* is number of iterations. Normally, *K*, *t* << *n*.
- Local optimum
  - sensitive to initial seed points
  - converge to a local optimum: maybe an unwanted solution
- Other problems
  - Need to specify *K,* the *number* of clusters, in advance
  - Unable to handle noisy data and outliers (*K-Medoids* algorithm)
  - Not suitable for discovering clusters with non-convex shapes
  - Applicable only when mean is defined, then what about categorical data? (*K-mode* algorithm)
  - how to evaluate the *K*-mean performance?

Hierarchical Clustering

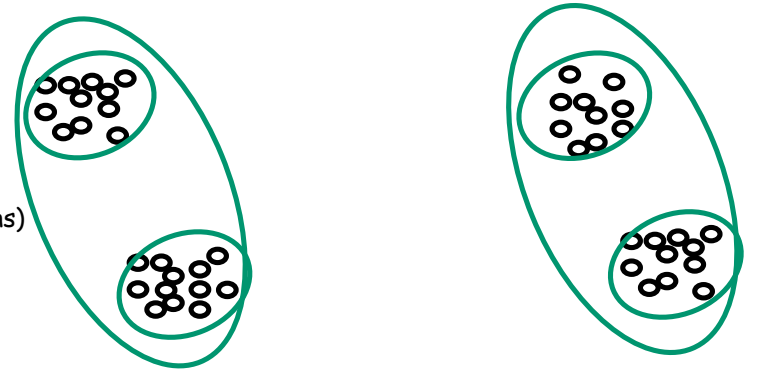


- A hierarchy might be more natural.
- Different users might care about different levels of granularity or even prunings.

Hierarchical Clustering

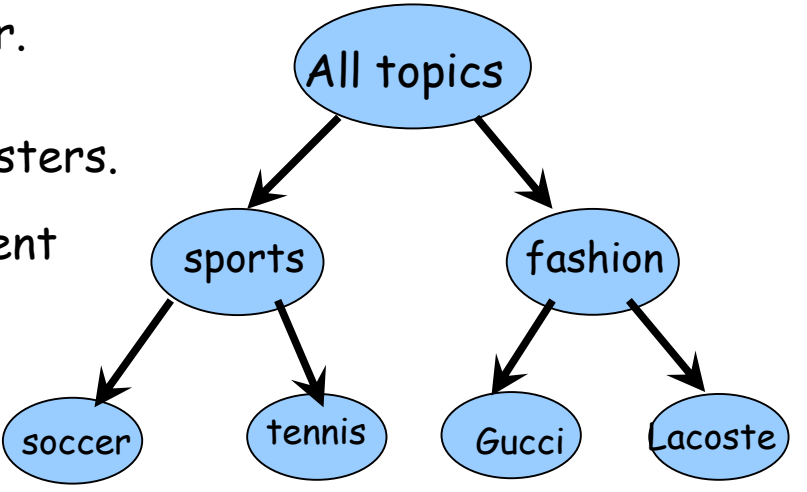
Top-down (divisive)

- Partition data into 2-groups (e.g., 2-means)
- Recursively cluster each group.



Bottom-Up (agglomerative)

- Start with every point in its own cluster.
- Repeatedly merge the "closest" two clusters.
- Different defs of "closest" give different algorithms.

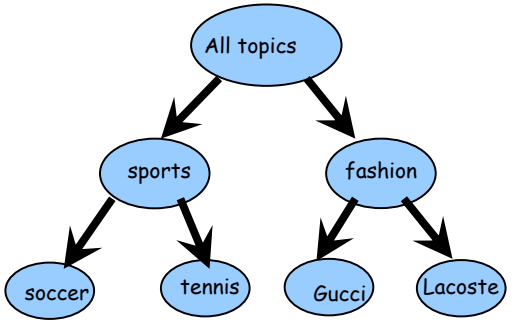


Bottom-Up (agglomerative)

Have a distance measure on pairs of objects.

d(x,y) – distance between x and y

E.g., # keywords in common, edit distance, etc



- Single linkage: $\text{dist}(A, B) = \min_{x \in A, x' \in B'} \text{dist}(x, x')$
- Complete linkage: $\text{dist}(A, B) = \max_{x \in A, x' \in B'} \text{dist}(x, x')$
- Average linkage: $\text{dist}(A, B) = \text{avg}_{x \in A, x' \in B'} \text{dist}(x, x')$
- Wards' method

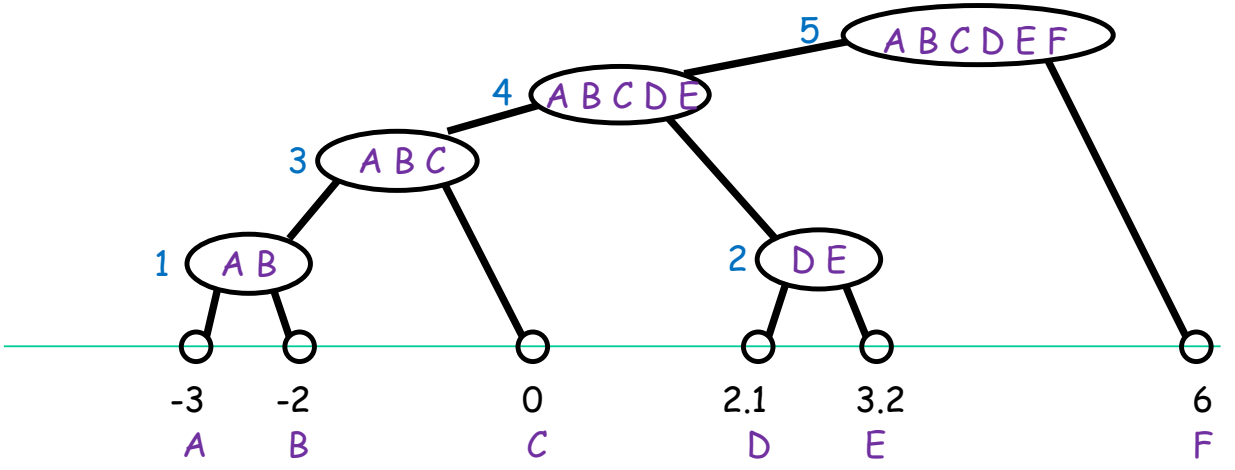
Single Linkage

Bottom-up (agglomerative)

- Start with every point in its own cluster.
- Repeatedly merge the "closest" two clusters.

Single linkage: $\text{dist}(A, B) = \min_{x \in A, x' \in B} \text{dist}(x, x')$

Dendogram



Single Linkage

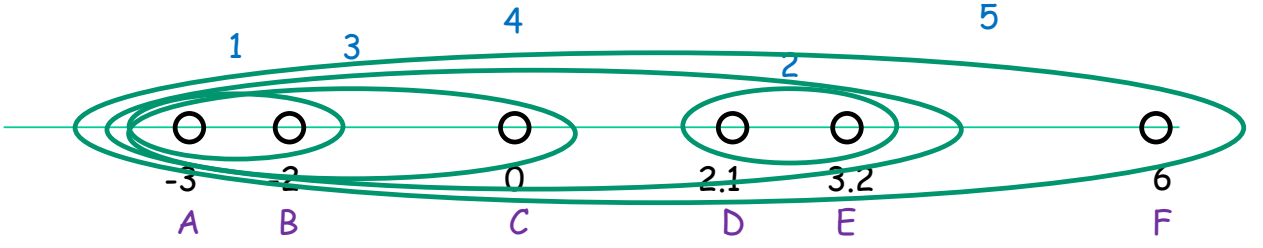
Bottom-up (agglomerative)

- Start with every point in its own cluster.
- Repeatedly merge the "closest" two clusters.

Single linkage: $\text{dist}(A, B) = \min_{x \in A, x' \in B} \text{dist}(x, x')$

One way to think of it: at any moment, we see connected components of the graph where connect any two pts of distance < r.

Watch as r grows (only n-1 relevant values because we only we merge at value of r corresponding to values of r in different clusters).



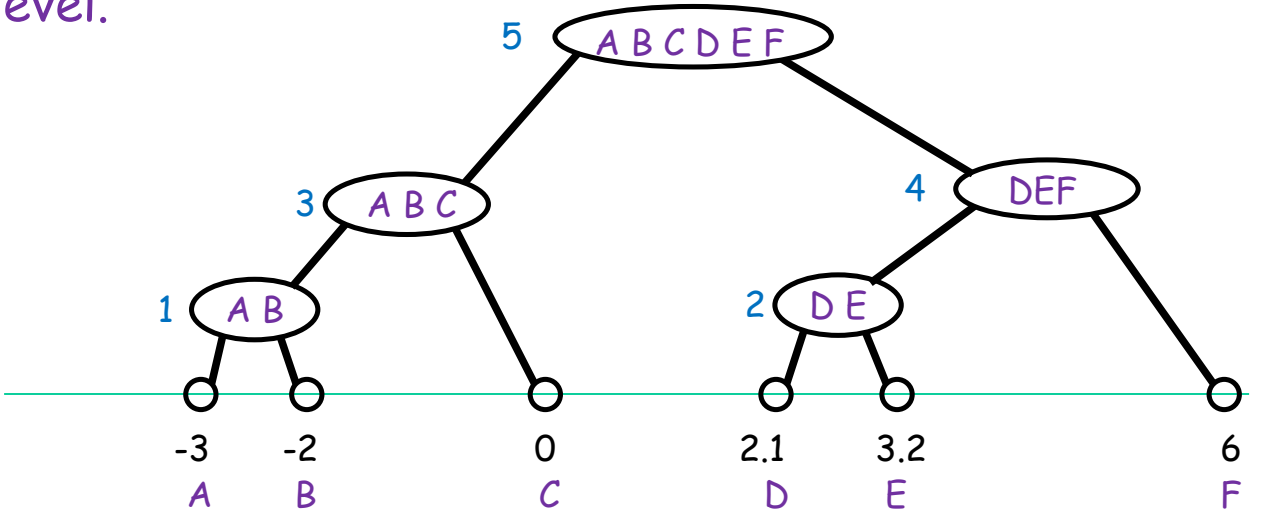
Complete Linkage

Bottom-up (agglomerative)

- Start with every point in its own cluster.
- Repeatedly merge the "closest" two clusters.

Complete linkage: $\text{dist}(A, B) = \max_{x \in A, x' \in B} \text{dist}(x, x')$

One way to think of it: keep max diameter as small as possible at any level.



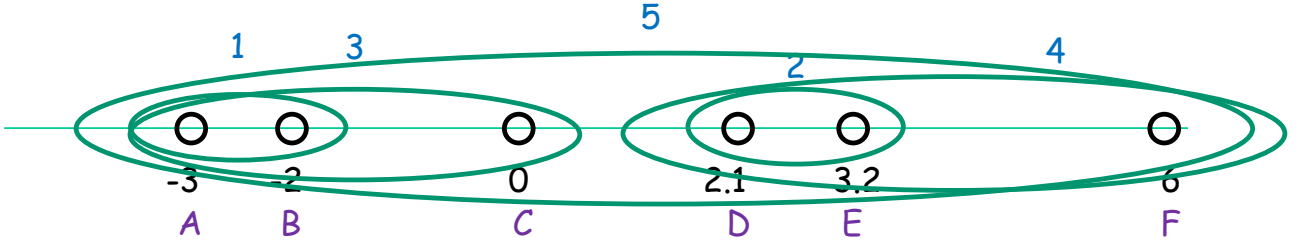
Complete Linkage

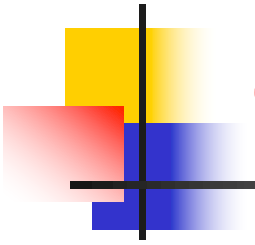
Bottom-up (agglomerative)

- Start with every point in its own cluster.
- Repeatedly merge the "closest" two clusters.

Complete linkage: $\text{dist}(A, B) = \max_{x \in A, x' \in B} \text{dist}(x, x')$

One way to think of it: keep max diameter as small as possible.





Other Clustering Algorithms

- **Spectral clustering**
  - Uses similarity matrix and its spectral decomposition (eigenvalues and eigenvectors)

- **Multidimensional scaling**
  - techniques often used in data visualization for exploring similarities or dissimilarities in data.