MEME19403: Topic 4 Predictive Modelling

Dr Liew How Hui

May 2021

Revision

What have we learn:

- Week 7: Data Frames for structured data
- Week 8: Try to find simple statistical patterns in data.
  - For each column, determine whether the data is categorical or numeric.
  - For categorical: Counting, bar chart, pie chart, etc.
  - For numerical: mean, stdev, median, histogram, normality, skewness etc.
  - Interaction between data: pair plot, ANOVA, etc.
- Week 9: Data encoding, pre-processing and dimensional reduction.

Revision (cont)

If **no simple interaction** or there inputs cannot be reduced by dimensional reduction, more sophisticated statistical models are required:

- The user knows about the **target variable**:
  - It is changing with time $\rightarrow$ time series? stochastic process?
  - It is time-independent $\rightarrow$ classifiable or not?
- We may need to scale (Week 9 pre-processing) the data so that the training of the model works better(?).

Introduction

This Topic is about Classification Problems with or without targer variable (AKA label) for time-independent data:

- Unsupervised learning classification: Unknown labels. E.g. unknown viruses, species, etc.
- Supervised learning classification: Known labels E.g. tagged images, type of cancers, ...

Outline

1. Unsupervised Learning
   - Data Similarity
   - k-Means Clustering
   - Hierarchical Clustering (Unsupervised Tree?)
2. Supervised Learning
   - Logistic Regression

Unsupervised Learning

This is a huge topic which covers:

- Data visualisation (Week 8)
- Dimensional Reduction: PCA (Week 9)
- Various clustering models (based on the distance/similarity measure):
  - k-Means — partitioning method
  - Hierarchical clustering — "inclusion" tree
  - ...
- Association rules: Use in supermarket

Outline

Distance

The kNN algorithm fundamentally relies on a "distance" metric.

It is mentioned in hierarchical clustering. Here, we state the mathematical definition.

A *distance* $d(x_i, x_j)$ is a function which satisfies the following conditions: For any x_i, x_j, x_k,

(i) $d(x_i, x_j) \geq 0$ and $d(x_i, x_j) = 0$ iff $x_i = x_j$;

(ii) $d(x_i, x_j) = d(x_j, x_i)$; and

(iii) $d(x_i, x_j) \leq d(x_i, x_k) + d(x_k, x_j)$ (triangle inequality).

Distance (cont)

The most common choice is the *Minkowski distance*:

$$d(\mathrm{x}, \mathrm{z}) = \|\mathrm{x} - \mathrm{z}\|_r = \left(\sum_{i=1}^{p} |x_i - z_i|^r \right)^{\frac{1}{r}}, \quad \mathrm{x},\ \mathrm{z} \in \mathbb{R}^p. \tag{1}$$

Note that $\|\cdot\|^r$ is called the ℓ^r norm.

When $r = 1$, we have the *Manhattan distance*:

$$\|\mathrm{x} - \mathrm{z}\|_1 = |x_1 - z_1| + |x_2 - z_2| + \cdots + |x_p - z_p|.$$

When $r = 2$, we have the *Euclidean distance*:

$$\|\mathrm{x} - \mathrm{z}\|_2 = \sqrt{(x_1 - z_1)^2 + (x_2 - z_2)^2 + \cdots + (x_p - z_p)^2}.$$

Distance (cont)

When $r = \infty$, we have the *Chebyshev distance*:

$$\|\mathsf{x} - \mathsf{z}\|_\infty = \max\{|x_1 - z_1|, |x_2 - z_2|, \cdots, |x_p - z_p|\}.$$

The Euclidean distance is more sensitive to outliers than the Manhattan distance. When outliers are rare, the Euclidean distance performs very well and is generally preferred. When the outliers are significant, the Manhattan distance is more stable.

Distance (cont)

The **Mahalanobis distance** between two objects is defined as:

$$d_{Mahalanobis}(X, Y) = \sqrt{(X - Y)^T C^{-1}(X - Y)}$$

where X and Y is a pair of objects, and C is the sample covariance matrix.

Another version of the formula, which uses distances from each observation to the central mean:

$$d_i = \sqrt{(\mathrm{x}_i - \bar{\mathrm{x}})^T C^{-1}(\mathrm{x}_i - \bar{\mathrm{x}})}$$

where x_i is an object vector, $\bar{\mathrm{x}}$ is an arithmetic mean vector.

Distance (cont)

The Jaccard index, also known as Intersection over Union and the Jaccard similarity coefficient (originally given the French name coefficient de communautÃ©© by Paul Jaccard), is a statistic used for gauging the similarity and diversity of sample sets. The Jaccard coefficient measures similarity between finite sample sets, and is defined as the size of the intersection divided by the size of the union of the sample sets:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}.$$

(If A and B are both empty, define $J(A,B) = 1$.)

$$0 \leq J(A,B) \leq 1.$$

Distance (cont)

The **Jaccard distance**, which measures dissimilarity between sample sets, is complementary to the Jaccard coefficient and is obtained by subtracting the Jaccard coefficient from 1, or, equivalently, by dividing the difference of the sizes of the union and the intersection of two sets by the size of the union:

$$d_J(A,B) = 1 - J(A,B) = \frac{|A \cup B| - |A \cap B|}{|A \cup B|}.$$

Distance (cont)

The **Tanimoto similarity index** (which is not a true distance measure) is defined by

$$d(x, y) = \frac{\mathrm{x} \cdot \mathrm{y}}{(|\mathrm{x}| * |\mathrm{x}|) + (|\mathrm{y}| * |\mathrm{y}|) - \mathrm{x} \cdot \mathrm{y}}$$

for bit vectors x and y.

The **cosine similarity index**,

$$d(x, y) = \frac{\mathrm{x} \cdot \mathrm{y}}{|\mathrm{x}| * |\mathrm{y}|}$$

The two similarity coefficient differs by the denominators. The Tanimoto and cosine similarity coefficients would be the same if $\mathrm{x} \cdot \mathrm{y}$ is zero.

Distance (cont)

According to `https://jcheminf.biomedcentral.com/articles/10.1186/s13321-015-0069-3`

> *The Tanimoto index, Dice index, Cosine coefficient and Soergel distance were identified to be the best (and in some sense equivalent) metrics for similarity calculations. The similarity metrics derived from Euclidean and Manhattan distances are not recommended on their own, although their variability and diversity from other similarity metrics might be advantageous in certain cases (e.g. for data fusion).*

Outline

k-Means Clustering

- Most popular clustering method!
- It partitions a data set of n observations into k distinct, non-overlapping clusters (groups) $C_1, \cdots, C_k$ with the nearest mean, serving as a prototype for the cluster. It tries to find a "cluster" to optimise

$$\min_{C_1,\cdots,C_K} \left\{ \sum_{m=1}^{k} \frac{1}{|C_m|} \sum_{i,j \in C_m} d(x_i, x_j)^2 \right\}.$$

k-Means Clustering (cont)

Note that

- $WSS_m = \sum_{i,j \in C_m} d(\mathrm{x}_i, \mathrm{x}_j)^2$ is called the *within sum of squares*.
- d can be any distance / dissimilarity function but the default is

$$d(\mathrm{x}, \mathrm{y}) = \sqrt{(x_1 - y_1)^2 + \cdots + (x_p - y_p)^2}.$$

k-Means Clustering (cont)

After the desired number of clusters k is specified, the *(naïve) k-means algorithm* below can be applied to perform the k-means clustering:

1. Randomly pick k observations as the initial centroid(s) c_m.
2. Iterate until the centroid(s) stop changing:
   1. Assign each observation to the cluster with the nearest centroid:

$$C_m^{(t)} = \left\{ \mathrm{x}_p : d(\mathrm{x}_p, \mathrm{c}_m^{(t)}) \leq d(\mathrm{x}_p, \mathrm{c}_j^{(t)}), \quad \forall j, 1 \leq j \leq k \right\},$$

   2. Recalculate centroids for observations assigned to each cluster:

$$\mathrm{c}_m^{(t+1)} = \frac{1}{|C_m^{(t)}|} \sum_{x_j \in C_m^{(t)}} \mathrm{x}_j.$$

k-Means Clustering (cont)

Software Implementations:

```
sklearn.cluster.KMeans(n_clusters=8, *,
  init='k-means++', n_init=10, max_iter=300,
  tol=0.0001, verbose=0, random_state=None)
```

where 'k-means++' is the method for initialisation which selects initial cluster centres for k-mean clustering in a smart way to speed up convergence. Alternative are 'random', which chooses n_clusters observations (rows) at random from data for the initial centroids; and an ndarray in the shape of (n_clusters, n_features) which gives the initial centres.

k-Means Clustering (cont)

Example:

You are given the following observations with two variables, x_1 and x_2.

Obs	x_1	x_2
A	1	1
B	1	2
C	2	1
D	2	2
E	8	8
F	8	9
G	9	8
H	9	9

k-Means Clustering (cont)

Example (cont):

Perform k-means clustering to group the observations into two groups using Euclidean distance. Given that the random initial centroids be D and A.

k-Means Clustering (cont)

Example (cont):

$t = 0$: $c_1^{(0)} = (2, 2)$ (point D), $c_2^{(0)} = (1, 1)$ (point A)

Obs	x	y	distance from c_1	distance from c_2	closest
A	1	1	1.414214	0.000000	2
B	1	2	1.000000	1.000000	1
C	2	1	1.000000	1.000000	1
D	2	2	0.000000	1.414214	1
E	8	8	8.485281	9.899495	1
F	8	9	9.219544	10.630146	1
G	9	8	9.219544	10.630146	1
H	9	9	9.899495	11.313708	1

k-Means Clustering (cont)

Example (cont):

$t = 1$: $c_1^{(1)} = \frac{(1,2)+\ldots+(9,9)}{7} = (5.571429, 5.571429)$,
$c_2^{(1)} = (1, 1)$

Obs	x	y	distance from c_1	distance from c_2	closest
A	1	1	6.464976	0.000000	2
B	1	2	5.801126	1.000000	2
C	2	1	5.801126	1.000000	2
D	2	2	5.050763	1.414214	2
E	8	8	3.434519	9.899495	1
F	8	9	4.201555	10.630146	1
G	9	8	4.201555	10.630146	1
H	9	9	4.848732	11.313708	1

k-Means Clustering (cont)

Example (cont):

$t = 2$: $c_1^{(2)} = \frac{(8,8)+\ldots+(9,9)}{4} = (8.5, 8.5)$,
$c_2^{(2)} = \frac{(1,1)+\ldots+(2,2)}{4} = (1.5, 1.5)$

Obs	x	y	distance from c_1	distance from c_2	closest
A	1	1	10.6066017	0.7071068	2
B	1	2	9.9247166	0.7071068	2
C	2	1	9.9247166	0.7071068	2
D	2	2	9.1923882	0.7071068	2
E	8	8	0.7071068	9.1923882	1
F	8	9	0.7071068	9.9247166	1
G	9	8	0.7071068	9.9247166	1
H	9	9	0.7071068	10.6066017	1

Stop when clusters / centroids unchanged.

k-Means Clustering (cont)

Two types of data preprocessing:

- Min-max scaling transforms all variables into an interval [0,1] by using

$$M(X_{ij}) = \frac{X_{ij} - X_{\text{min,j}}}{X_{\text{max,j}} - X_{\text{min,j}}}.$$

- Standardisation transforms all variables to a standard scale with mean of zero and standard deviation of one by using

$$M(X_{ij}) = \frac{X_{ij} - \overline{X}_j}{s_{X,j}}$$

k-Means Clustering (cont)

Scaling Example: By using k-means clustering with E and D as initial centroids, find 2 clusters for the data:

Obs	x_1	x_2
A	0	0
B	0	2
C	20	0
D	20	2
E	80	8
F	80	10
G	100	8
H	100	10
I	10	7
J	30	2
K	40	9
L	60	1
M	70	8
N	90	3

k-Means Clustering (cont)

Scaling Example (cont):

(a) Rescale both variables using min-max normalisation and perform k-means clustering.

(b) Rescale both variables x_1 and x_2 using standardisation and perform k-means clustering.

k-Means Clustering (cont)

Scaling Example (cont): (a) After the min-max scaling and performing k-mean clustering, if we obtain the final centroids: $c_1 = (0.2, 0.2), \quad c_2 = (0.8, 0.8)$

	x_1'	x_2'	dist1	dist2	cluster
A	0.0	0.0	0.2828427	1.1313708	1
B	0.0	0.2	0.2000000	1.0000000	1
C	0.2	0.0	0.2000000	1.0000000	1
D	0.2	0.2	0.0000000	0.8485281	1
E	0.8	0.8	0.8485281	0.0000000	2
F	0.8	1.0	1.0000000	0.2000000	2
G	1.0	0.8	1.0000000	0.2000000	2
H	1.0	1.0	1.1313708	0.2828427	2
I	0.1	0.7	0.5099020	0.7071068	1
J	0.3	0.2	0.1000000	0.7810250	1
K	0.4	0.9	0.7280110	0.4123106	2
L	0.6	0.1	0.4123106	0.7280110	1
M	0.7	0.8	0.7810250	0.1000000	2
N	0.9	0.3	0.7071068	0.5099020	2

k-Means Clustering (cont)

Scaling Example (cont): (b) First calculate the mean

$$\overline{X}_1 = \frac{0 + \ldots + 90}{14} = 50; \quad \overline{X}_2 = \frac{0 + \ldots + 3}{14} = 5$$

and then the (sample) standard deviation

$$\begin{aligned}(s_X)_1 &= \sqrt{\frac{(0-50)^2 + \cdots + (90-50)^2}{14-1}} \\ &= \sqrt{1942.857143} = 37.00312\end{aligned}$$

$$(s_X)_2 = \sqrt{\frac{(0-5)^2 + \cdots + (3-5)^2}{14-1}} = 3.86304$$

k-Means Clustering (cont)

Scaling Example (cont): (b) If the cluster centroids are

$$c_1 = (-0.8107425, -0.7765905), \; c_2 = (0.8107425, 0.7765905)$$

Obs	$\tilde{x}_1$	$\tilde{x}_2$	$d(\tilde{x}_1, c_1)$	$d(\tilde{x}_1, c_2)$	cluster
A	-1.3512375	-1.2943175	0.7484491	2.9937964	1
B	-1.3512375	-0.7765905	0.5404950	2.6620534	1
C	-0.8107425	-1.2943175	0.5177270	2.6301850	1
D	-0.8107425	-0.7765905	0.0000000	2.2453473	1
E	0.8107425	0.7765905	2.2453473	0.0000000	2
F	0.8107425	1.2943175	2.6301850	0.5177270	2
G	1.3512375	0.7765905	2.6620534	0.5404950	2
H	1.3512375	1.2943175	2.9937964	0.7484491	2
I	-1.0809900	0.5177270	1.3222297	1.9093617	1
J	-0.5404950	-0.7765905	0.2702475	2.0586923	1
K	-0.2702475	1.0354540	1.8909363	1.1115528	2
L	0.2702475	-1.0354540	1.1115528	1.8909363	1
M	0.5404950	0.7765905	2.0586923	0.2702475	2
N	1.0809900	-0.5177270	1.9093617	1.3222297	2

k-Means Clustering (cont)

One of the requirements to perform k-means clustering is a pre-determined number of clusters.

Sometimes, appropriate k is known.

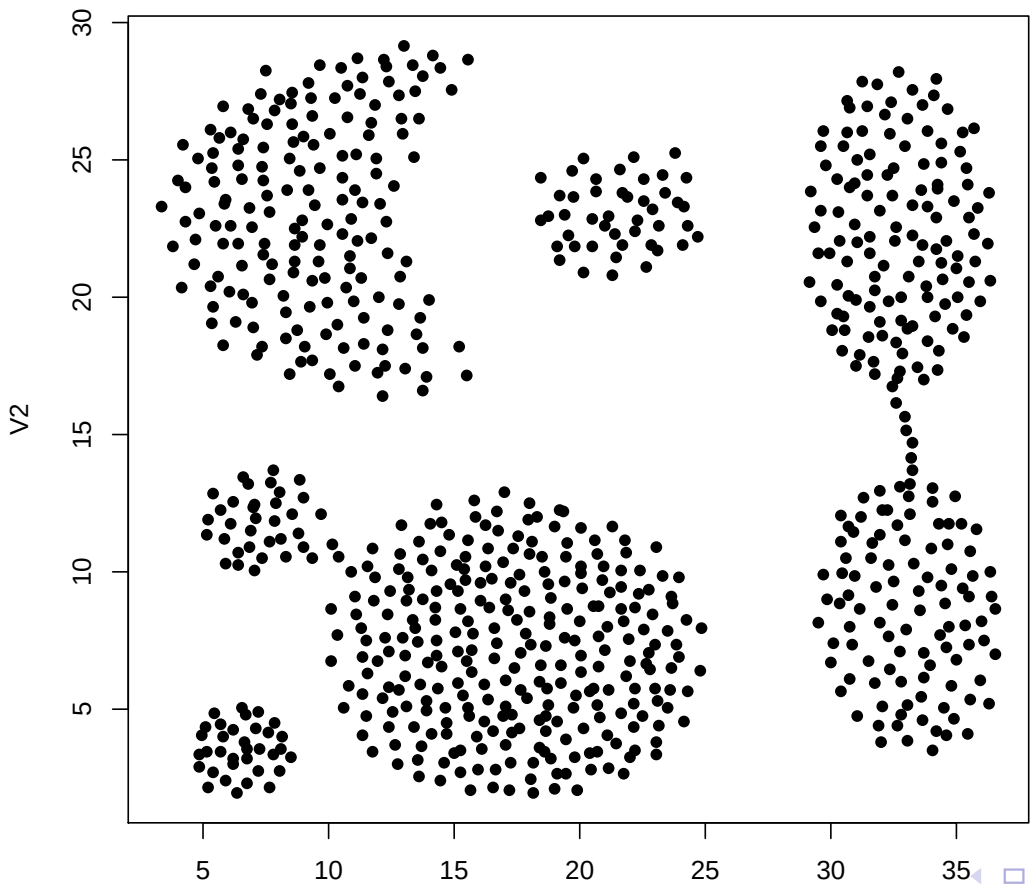
Most of the times, k is unknown. Use "elbow principle":

Observations within clusters will be close together when plotted geometrically. We can make use of this property to determine the optimum number of clusters to be formed. An optimum number of clusters is the one which *within groups sum of squares (wss)* decreases dramatically as compared to the previous number, and only decreased a little as compared to the after number.

k-Means Clustering (cont)

Elbow principle is not science.

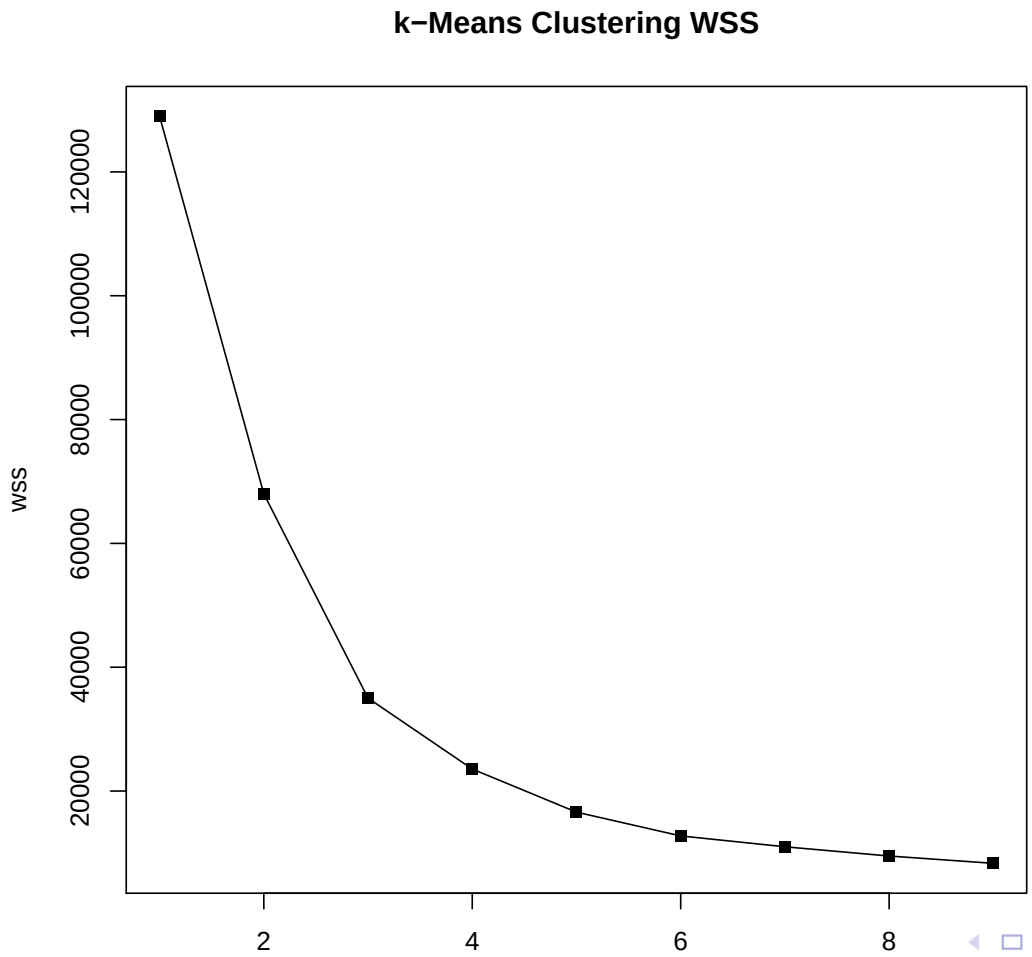
Example: Consider the following data:



k-Means Clustering (cont)

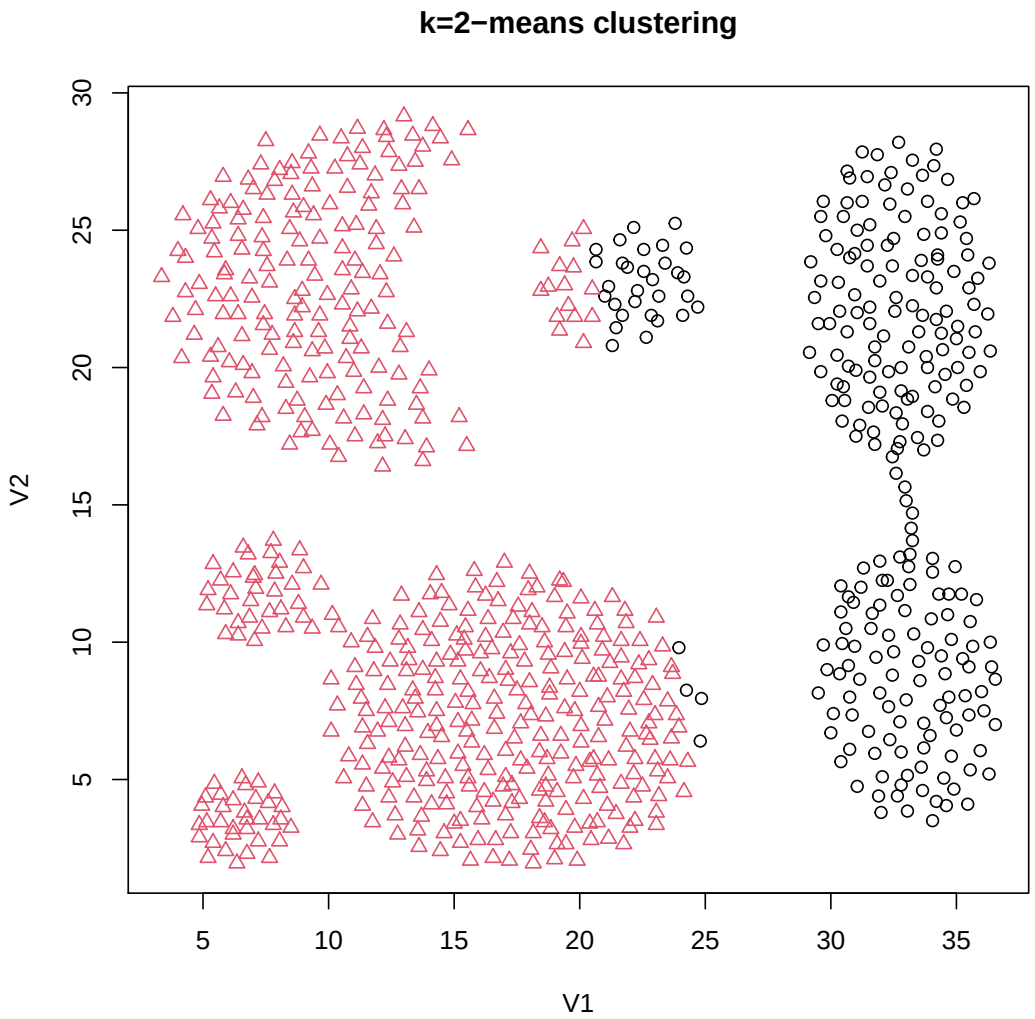
Elbow principle is not science.

Example (cont):



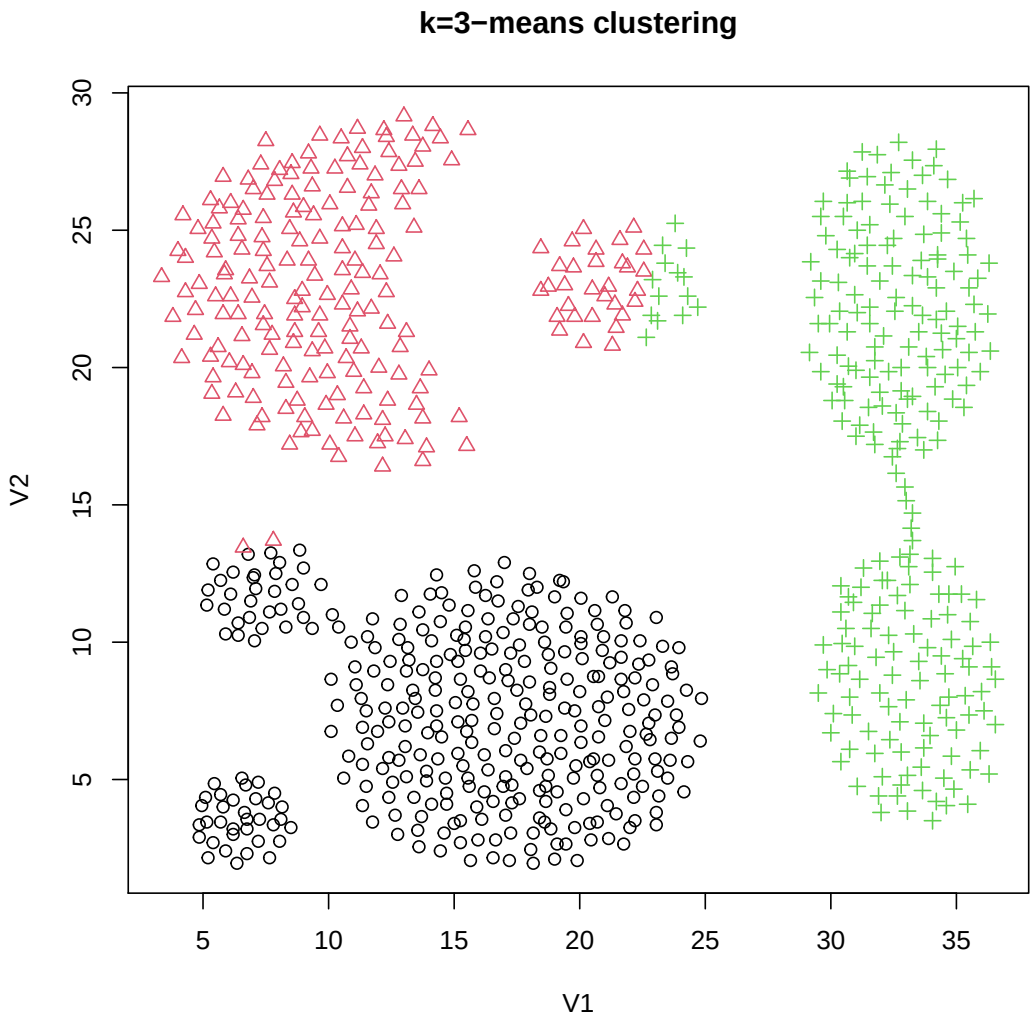
k-Means Clustering (cont)

Example (cont):



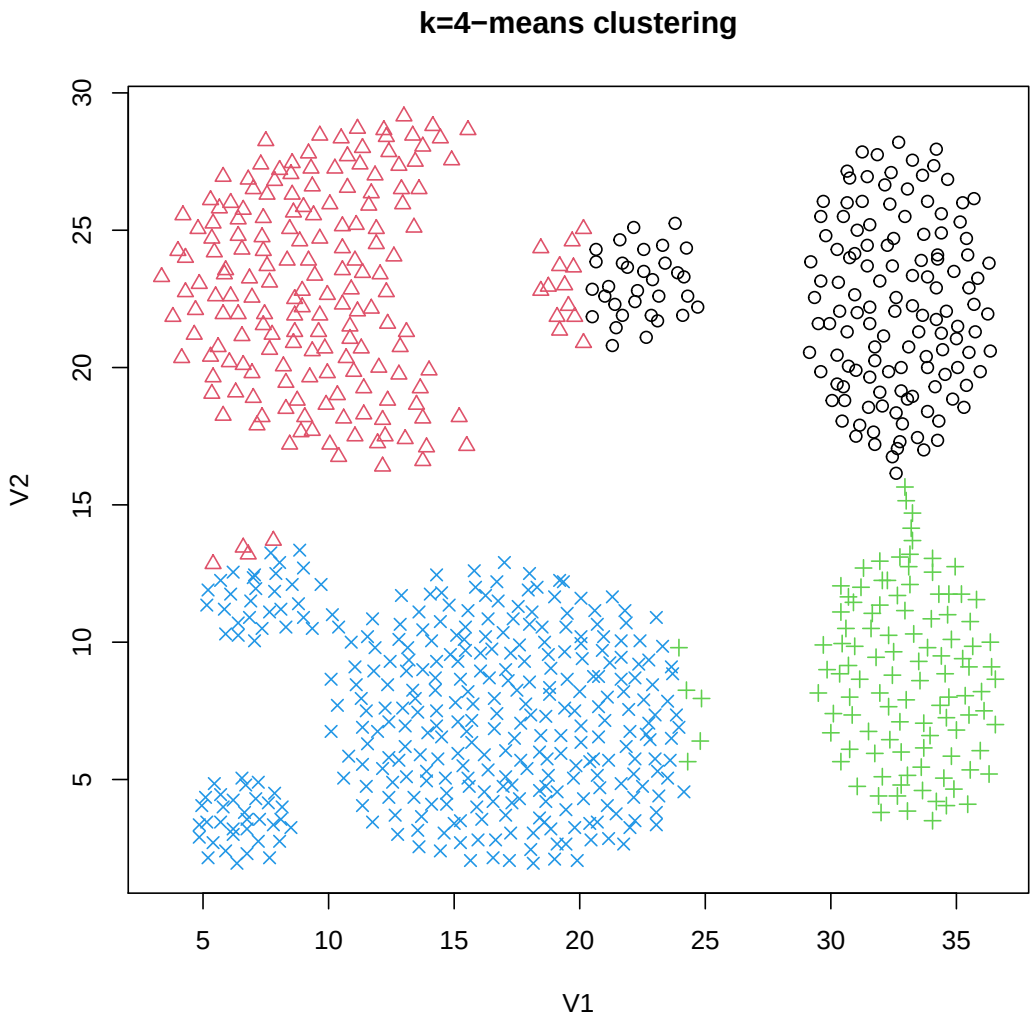
k-Means Clustering (cont)

Example (cont):



k-Means Clustering (cont)

Example (cont):



k-Means Clustering (cont)

Example (cont):

Outline

1. Unsupervised Learning
   - Data Similarity
   - k-Means Clustering
   - Hierarchical Clustering (Unsupervised Tree?)
2. Supervised Learning
   - Logistic Regression

Hierarchical Methods

Hierarchical methods construct a hierarchy of clusterings, with the number of clusters ranging from one to the number of observations.

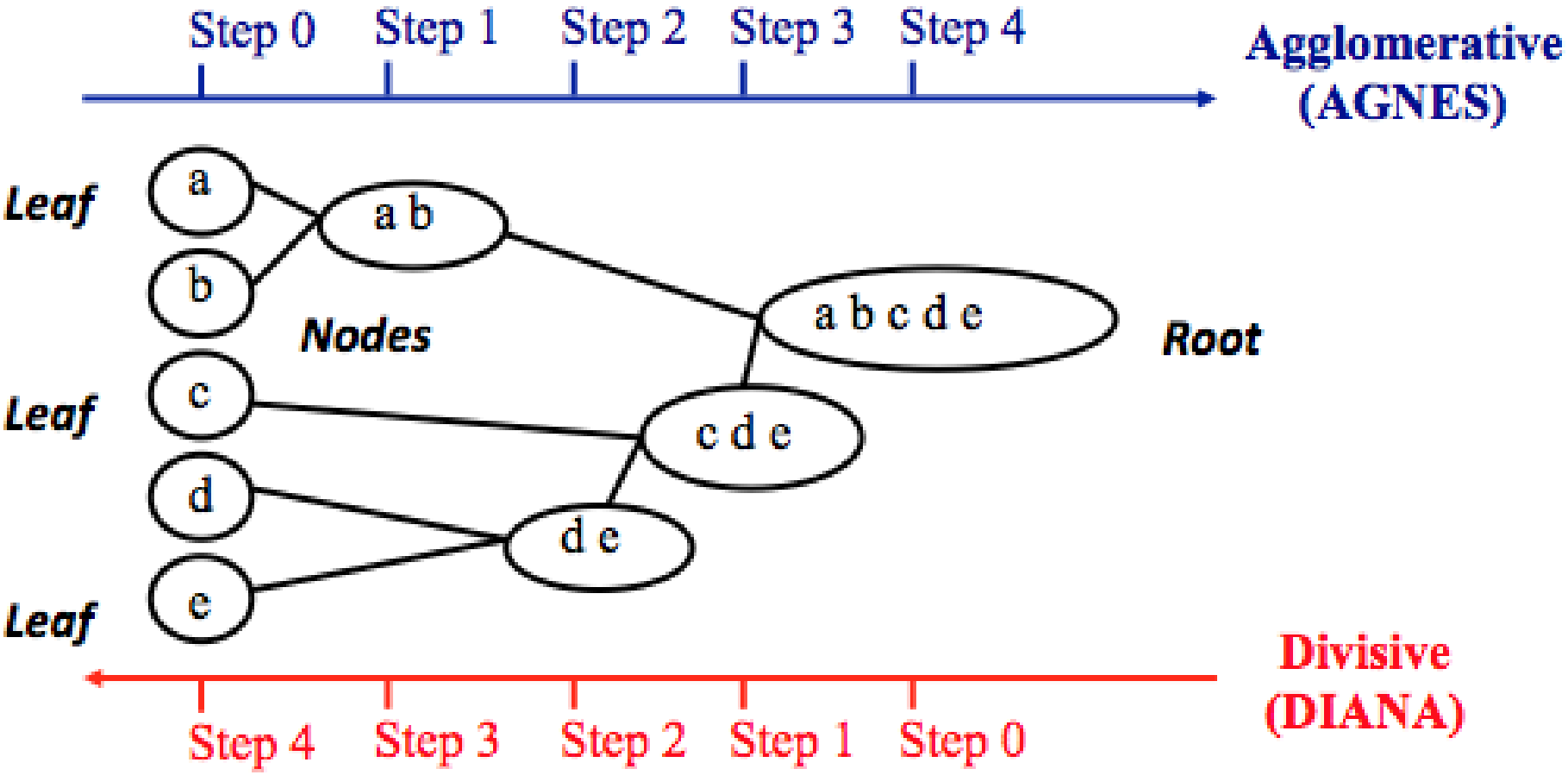
- Arrange or classify things according to inclusiveness
- A natural way of abstraction, summarisation, compression, and simplification for understanding
- Typical setting: organise a given set of objects to a hierarchy

Hierarchical Methods (cont)

Two possible types of hierarchical methods:

- *Agglomerative Nesting* (AGNES) or *Hierarchical (Agglomerative) Clustering* (HAC): builds a multilevel hierarchy of clusters from **points** to **clusters**.
- *Divisive Hierarchical Clustering* or *Divisia Analysis (DIANA)*: It breaks a **single cluster** to many **sub-clusters**.

Hierarchical Methods (cont)



Source: `https://uc-r.github.io/hc_clustering`

Dissimilarity / Distance Matrix (cont)

The *distance (or dissimilarity) matrix* between observations can be measured by a dissimilarity function d as

$$D = \begin{bmatrix} 0 & d_{12} & \cdots & d_{1n} \\ d_{21} & 0 & & d_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ d_{n1} & d_{n2} & \cdots & 0 \end{bmatrix}, \quad \mathrm{X} = \begin{bmatrix} \mathrm{x}_1 \\ \mathrm{x}_2 \\ \vdots \\ \mathrm{x}_n \end{bmatrix} \tag{2}$$

where $d_{ij} = d(\mathrm{x}_i, \mathrm{x}_j)$, $i, j = 1, \cdots, n$.

AGNES Clustering (cont)

Dissimilarity is defined for a **pair of observations**.

It needs to be extended to a pair of groups of observations, called **linkage**, which defines the dissimilarity between **two groups of observations** A and B.

Commonly used linkage criteria are:

(a) Minimum or single-linkage clustering: $\min \{ d(a, b) : a \in A, b \in B \}$.

(b) Maximum or complete-linkage clustering: $\max \{ d(a, b) : a \in A, b \in B \}$.

(c) (Unweighted) average linkage clustering (or UPGMA): $\frac{1}{|A| \cdot |B|} \sum_{a \in A} \sum_{b \in B} d(a, b)$

AGNES Clustering (cont)

(c) Weighted average linkage clustering (or WPGMA): $d(i \cup j, k) = \frac{d(i,k)+d(j,k)}{2}$.

(d) Centroid linkage clustering, or UPGMC: $\|c_s - c_t\|$ where c_s and c_t are the centroids of clusters s and t, respectively.

(e) Minimum energy clustering: $\frac{2}{nm}\sum_{i,j=1}^{n,m}\|a_i - b_j\|_2 - \frac{1}{n^2}\sum_{i,j=1}^{n}\|a_i - a_j\|_2 - \frac{1}{m^2}\sum_{i,j=1}^{m}\|b_i - b_j\|_2$

(f) Ward minimises the sum of squared differences within all clusters. It is a **variance-minimising** approach and in this sense is similar to the k-means objective function but tackled with an agglomerative hierarchical approach.

AGNES Clustering (cont)

Implementation in Python:

```
sklearn.cluster.AgglomerativeClustering(n_clusters=2,
  affinity='euclidean', memory=None, connectivity=None,
  compute_full_tree='auto', linkage='ward',
  distance_threshold=None)
```

Where:

- linkage = ward (default), complete, average, single
- affinity is the 'distance' function used to compute the linkage. Can be "euclidean", "l1", "l2", "manhattan", "cosine", or "precomputed". If linkage is "ward", only "euclidean" is accepted.

AGNES Clustering: Complete Linkage

In the *complete linkage method*, the dissimilarity between the groups of observations is defined as

$$d_{(hk)l} = \max\{d_{hl},\ d_{kl}\}, \quad l \neq h, k, \quad 1 \leq l \leq n. \qquad (3)$$

The method computes all pairwise dissimilarities between the elements in a cluster and the elements in another cluster, and considers the *largest value* of these dissimilarities, i.e. (3), as the distance between the two clusters.

Complete Linkage (cont)

Example (Wikipedia: Complete-Linkage Clustering):

Based on a JC69 genetic distance matrix computed from the 5S ribosomal RNA sequence alignment of five bacteria: Bacillus subtilis (A), Bacillus stearothermophilus (B), Lactobacillus viridescens (C), Acholeplasma modicum (D), and Micrococcus luteus (E).

	A	B	C	D	E
A	0	17	21	31	23
B	17	0	30	34	21
C	21	30	0	28	39
D	31	34	28	0	43
E	23	21	39	43	0

Complete Linkage (cont)

Perform complete-linkage hierarchical clustering.

Solution: Step 1: The (nonzero) smallest distance of the table is $d(A, B) = 17$. Therefore, we cluster A and B leading to the table below using the complete-linkage (3):

	A,B	C	D	E
A,B	0	**30**	**34**	**23**
C	**30**	0	28	39
D	**34**	28	0	43
E	**23**	39	43	0

A and B's branch length is $d(A, B) = 17$

Complete Linkage (cont)

Step 2: The smallest distance of the above table is $d((A, B), E) = 23$ and we cluster (A,B) and E leading to the table below using the complete-linkage:

	A,B,E	C	D
A,B,E	0	**39**	**43**
C	**39**	0	28
D	**43**	28	0

(A,B) and E's branch length is
$d((A, B), E) - d(A, B) = 23 - 17 = 6$.

Complete Linkage (cont)

Step 3: The smallest distance of the above table is $d(C, D) = 28$ and we cluster C and D leading to the table below using the complete-linkage:

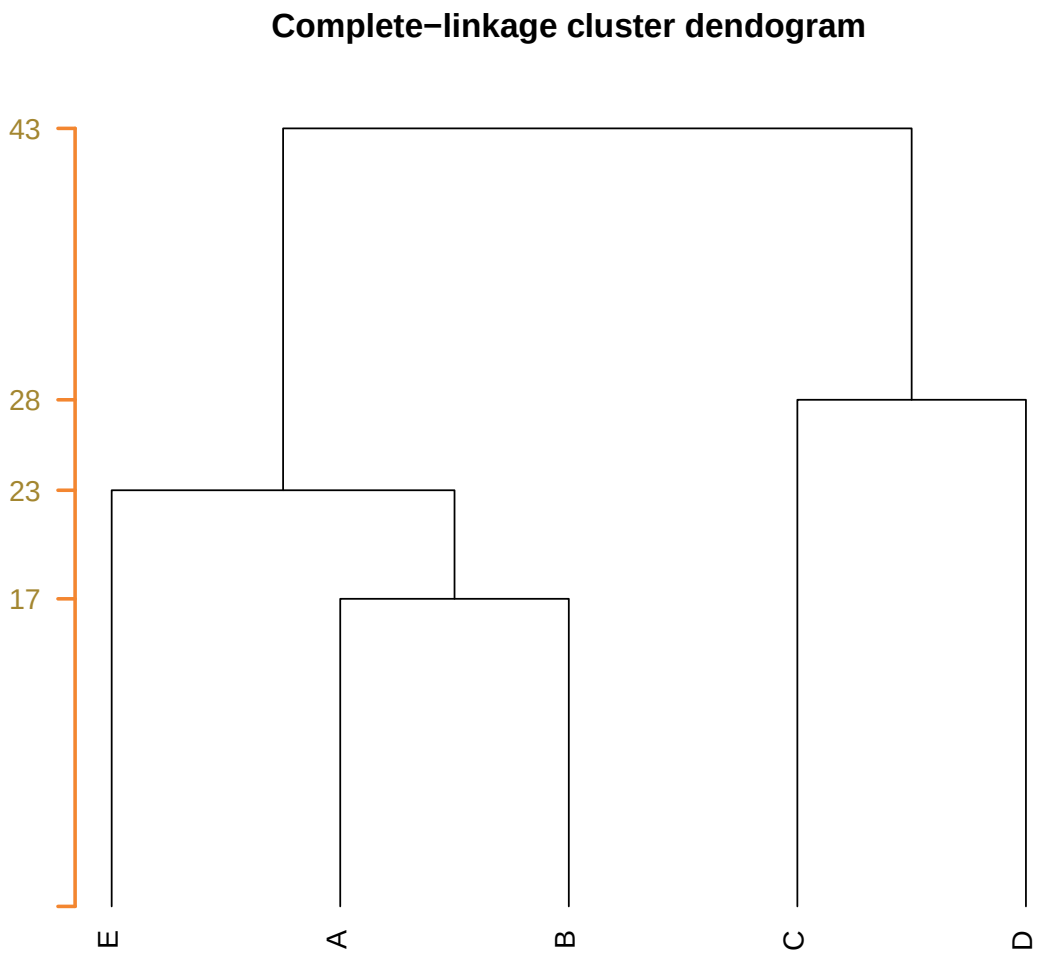
	A,B,E	C,D
A,B,E	0	**43**
C,D	**43**	0

Root r to (A,B,E)'s branch length is $d(r, (A, B, E)) = d((A, B, E), (C, D)) - d((A, B), E) = 43 - 23 = 20.$

Root r to (C,D)'s branch length is $d(r, (C, D)) = d((A, B, E), (C, D)) - d(C, D) = 43 - 28 = 15.$

Complete Linkage (cont)

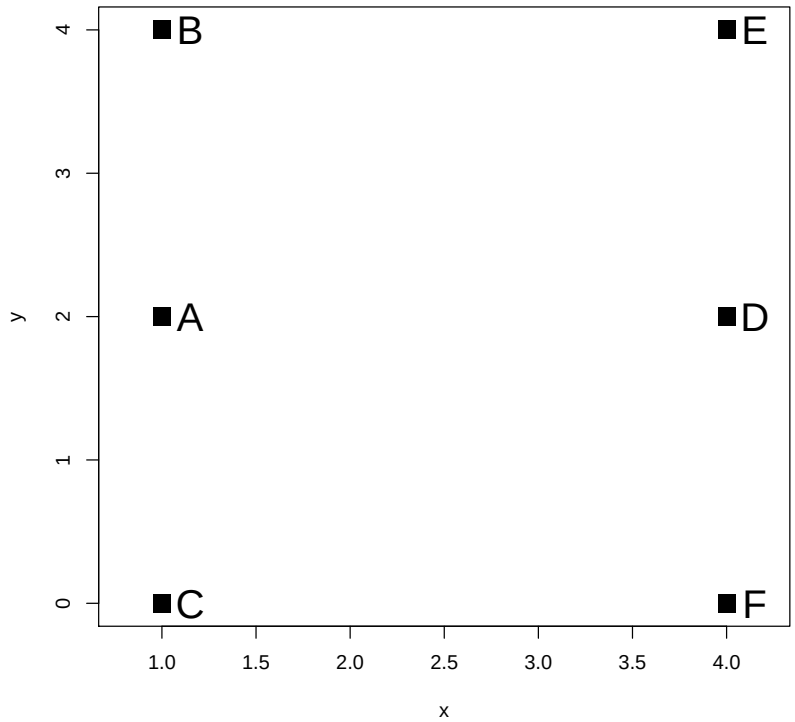
These data allows us to construct the dendrogram below:



Complete Linkage (cont)

Six-point Example: Given the points A(1, 2), B(1, 4), C(1, 0), D(4, 2), E(4, 4), F(4, 0). Form an AGNES clustering with complete linkage and show the dendrogram of the clustering.

The distribution of points:



Complete Linkage (cont)

Six-point Example (cont): Construct the distance matrix:

$$\begin{bmatrix} A & B & C & D & E & F \\ 0 & 2.000000 & 2.000000 & 3.000000 & 3.605551 & 3.605551 \\ 2.000000 & 0 & 4.000000 & 3.605551 & 3.000000 & 5.000000 \\ 2.000000 & 4.000000 & 0 & 3.605551 & 5.000000 & 3.000000 \\ 3.000000 & 3.605551 & 3.605551 & 0 & 2.000000 & 2.000000 \\ 3.605551 & 3.000000 & 5.000000 & 2.000000 & 0 & 4.000000 \\ 3.605551 & 5.000000 & 3.000000 & 2.000000 & 4.000000 & 0 \end{bmatrix}$$

Complete Linkage (cont)

Six-point Example (cont):

Find the "smallest distance", although there are many, we only need one, following order: $d(A, B) = 2$

$$\begin{bmatrix} (A,B) & C & D & E & F \\ 0 & 4.000000 & 3.605551 & 3.605551 & 5.000000 \\ 4.000000 & 0 & 3.605551 & 5.000000 & 3.000000 \\ 3.605551 & 3.605551 & 0 & 2.000000 & 2.000000 \\ 3.605551 & 5.000000 & 2.000000 & 0 & 4.000000 \\ 5.000000 & 3.000000 & 2.000000 & 4.000000 & 0 \end{bmatrix}$$

Complete Linkage (cont)

Six-point Example (cont):

The "smallest distance" in sequence is $d(E, F) = 2$

$$\begin{bmatrix} (A,B) & C & (D,E) & F \\ 0 & 4.000000 & 3.605551 & 5.000000 \\ 4.000000 & 0 & 5.000000 & 3.000000 \\ 3.605551 & 5.000000 & 0 & 4.000000 \\ 5.000000 & 3.000000 & 4.000000 & 0 \end{bmatrix}$$

Complete Linkage (cont)

Six-point Example (cont):

The "smallest distance" in sequence is $d(C, F) = 3$

$$\begin{bmatrix} (A,B) & (C,F) & (D,E) \\ 0 & 4.000000 & 3.605551 \\ 5.000000 & 0 & 5.000000 \\ 3.605551 & 5.000000 & 0 \end{bmatrix}$$

Complete Linkage (cont)

Six-point Example (cont):

The "smallest distance" in sequence is $d((A, B), (D, E)) = 3.605551$

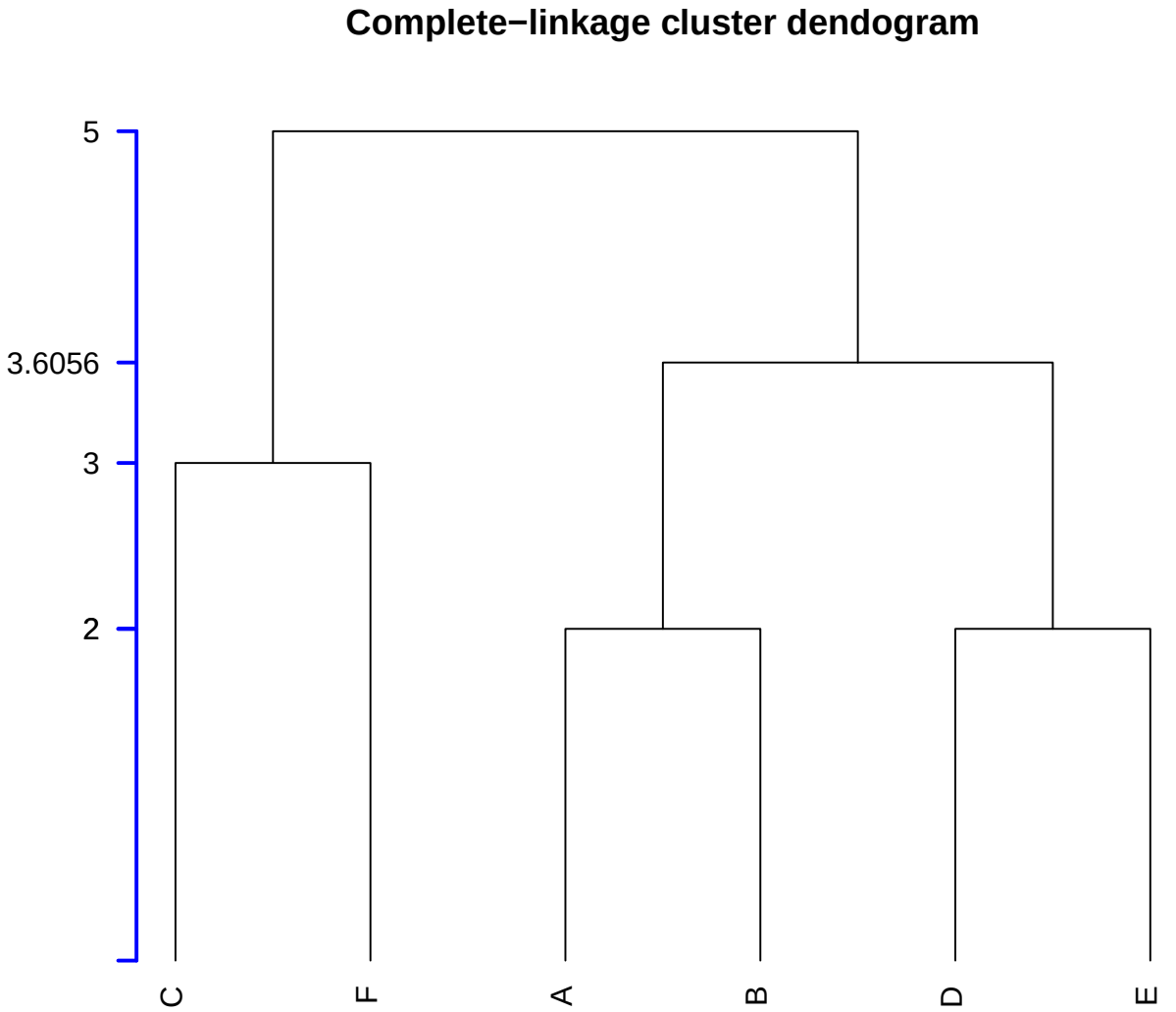
$$\begin{bmatrix} ((A, B), (D, E)) & (C, F) \\ 0 & 5.000000 \\ 5.000000 & 0 \end{bmatrix}$$

Finally, $d(((A, B), (D, E)), (C, F)) = 5$

Complete Linkage (cont)

Six-point Example (cont):

The dendrogram is



Agglomerative Clusterings

Six-point Example (cont): Not nice in Python???

```
1  from sklearn.cluster import AgglomerativeClustering
2  from scipy.cluster.hierarchy import linkage, dendrogram
3  import numpy as np, matplotlib.pylab as plt
4
5  X = np.array([[1, 2], [1, 4], [1, 0],
6                [4, 2], [4, 4], [4, 0]])
7
8  # setting distance_threshold=0 ensures we compute
9  # the full tree.
0  clust=AgglomerativeClustering(distance_threshold=0,
1    n_clusters=None,linkage='complete').fit(X)
2  print(clust.labels_)
3
4  dendrogram(linkage(X, method='complete'))
5  plt.title('Hierarchical Clustering Dendrogram')
6  plt.show()
```

AGNES Clustering: Other Linkage

- Single-linkage — easy to calculate by hand
- Average-linkage — a little bit difficult to calculate by hand
- Centroid-linkage — complex to calculate by hand
- Ward-linkage — complex to calculate by hand

Practical with Clustering

Practice 1: p_simul.py

Practice 2: p_nci60.py

Outline

1. Unsupervised Learning
   - Data Similarity
   - k-Means Clustering
   - Hierarchical Clustering (Unsupervised Tree?)

2. Supervised Learning
   - Logistic Regression

Supervised Learning Models

Linear Models:

- Logistic Regression
- Linear SVM
- Naive Bayes

Nonlinear Models:

- k-Nearest Neighbour (kNN)
- Decision Tree
- Neural Network

Classifier:

$$h_D(\mathrm{x}) = \underset{j=1,\ldots,K}{\operatorname{argmax}} \mathbb{P}(Y = j | \mathrm{X} = \mathrm{x}). \tag{4}$$

Outline

1. Unsupervised Learning
   - Data Similarity
   - k-Means Clustering
   - Hierarchical Clustering (Unsupervised Tree?)
2. Supervised Learning
   - Logistic Regression

Logisitic Regression Theory

The *Logistic Regression (LR)* algorithm is a parametric method used for **binary** classification. Its "strengths" compare to kNN are its ability to handle categorical features.

The assumption of LR is "the binary data are linearly separable with suitable parameters". Based on this assumption, a test input x would get a probability measure.

LR Theory (cont)

https://en.wikipedia.org/wiki/Logistic_function

$$S(x) = \frac{e^x}{1+e^x} = \frac{1}{1+e^{-x}}$$

Note that $0 < S(x) < 1$ for $-\infty < x < \infty$.

Cox (1958) proposed the "logistic regression" (LR) for binary classification problem:

$$\begin{aligned} &\mathbb{P}(Y = 1 | X_1 = x_1, \cdots, X_p = x_p) \\ =&\frac{1}{1+\exp(-(\beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p))}. \end{aligned} \quad (*)$$

LR Theory (cont)

Formula (*) can be written in vector form (using linear algebra)

$$\mathbb{P}(Y = 1|\mathrm{X} = \mathrm{x}) = S(\beta^T \tilde{\mathrm{x}})$$

where $\boldsymbol{\beta} = (\beta_0, \cdots, \beta_p)$ and $\widetilde{\mathrm{x}}_j = (1, \mathrm{x}_j)$.

Given an input x, the LR algorithm provides a prediction as follows (assuming the cut-off is 0.5):

$$h(\mathrm{x}) = \begin{cases} 1, & \mathbb{P}(Y = 1|X = \mathrm{x}) > 0.5 \\ 0, & \mathbb{P}(Y = 1|X = \mathrm{x}) \leq 0.5 \end{cases}$$

LR Theory (cont)

Estimating the parameters β_i using MLE: Given data (x_i, y_i), $i = 1, \cdots, n$, we want the parameters β_i so that the **likelihood function** of $\beta_0, \cdots, \beta_p$ is maximised:

$$\begin{aligned} & L(\beta_0, \cdots, \beta_p;\ y_1, \cdots, y_n | x_1, \cdots, x_n) \\ &= \prod_{i=1}^{n} \mathbb{P}(Y = y_i | X = x_i) \end{aligned} \tag{5}$$

Y is binary and follows a **Bernoulli distribution**

LR Theory (cont)

According to `https://en.wikipedia.org/wiki/Bernoulli_distribution`,

$Y \sim Bernoulli(p = \mathbb{P}(Y = 1|\mathrm{X}))$, then the probability mass function of observing $y \in \{0, 1\}$ is

$$\mathbb{P}(y) = p^y(1-p)^{1-y}.$$

$$\mathbb{P}(Y = y_i|\mathrm{X} = \mathrm{x}_i) = \left(\frac{e^{\widetilde{\mathrm{x}}_i^T\beta}}{1+e^{\widetilde{\mathrm{x}}_i^T\beta}}\right)^{y_i}\left(1-\frac{e^{\widetilde{\mathrm{x}}_i^T\beta}}{1+e^{\widetilde{\mathrm{x}}_i^T\beta}}\right)^{1-y_i}$$
$$= e^{y_i\widetilde{\mathrm{x}}_i^T\beta}\cdot(1+e^{\widetilde{\mathrm{x}}_i^T\beta})^{-y_i}\cdot(1+e^{\widetilde{\mathrm{x}}_i^T\beta})^{-(1-y_i)}$$

LR Theory (cont)

Where $\boldsymbol{\beta} = (\beta_0, \cdots, \beta_p)$ and $\widetilde{\mathrm{x}}_i = (1, \mathrm{x}_i)$.

Substituting it into (5), we have **loss function**

$$L(\beta_0, \cdots, \beta_p;\ y_1, \cdots, y_n | \mathrm{x}_1, \cdots, \mathrm{x}_n)$$
$$= \prod_{i=1}^{n} p_i^{y_i}(1-p_i)^{1-y_i} = \prod_{i=1}^{n} (e^{y_i \widetilde{\mathrm{x}}_i^T \boldsymbol{\beta}})(1 + e^{\widetilde{\mathrm{x}}_i^T \boldsymbol{\beta}})^{-1}.$$

Taking natural log leads to

$$\ln L = \sum_{i=1}^{n} y_i \widetilde{\mathrm{x}}_i^T \boldsymbol{\beta} - \sum_{i=1}^{n} \ln(1 + e^{\widetilde{\mathrm{x}}_i^T \boldsymbol{\beta}})$$
$$= \sum_{i=1}^{n} y_i \ln p_i + (1 - y_i) \ln(1 - p_i)$$

LR Theory (cont)

By Calculus,

$$\hat{\boldsymbol{\beta}} = \max_{\boldsymbol{\beta}} L = \max_{\boldsymbol{\beta}} \ln L \Rightarrow \frac{\partial}{\partial \boldsymbol{\beta}}(\ln L) = 0$$

i.e.

$$\frac{\partial}{\partial \boldsymbol{\beta}} \left(\sum_{i=1}^{n} y_i \tilde{\mathbf{x}}_i^T \boldsymbol{\beta} - \sum_{i=1}^{n} \ln(1 + e^{\tilde{\mathbf{x}}_i^T \boldsymbol{\beta}}) \right) = 0.$$

leading to

$$\sum_{i=1}^{n} y_i x_k^{(i)} - \sum_{i=1}^{n} \frac{x_k^{(i)} e^{\tilde{\mathbf{x}}_i^T \boldsymbol{\beta}}}{1 + e^{\tilde{\mathbf{x}}_i^T \boldsymbol{\beta}}} = 0, \quad k = 0, 1, \cdots, p$$

where $x_0^{(i)}$ is defined to be 1.

Hypothesis Testing and Inference

The *Z-statistic* tests the null hypothesis against the alternative hypothesis:

$$H_0 : \beta_i = 0 \quad \text{vs} \quad H_1 : \beta_i \neq 0.$$

`https://en.wikipedia.org/wiki/Wald_test`: With large "n",

$$\frac{\hat{\beta}_i - \beta_{i0}}{SE(\hat{\beta})} \sim Normal(0, 1),$$

The *standard error* $SE(\hat{\beta})$ is the inverse of the estimated information matrix with a shape $(p+1) \times (p+1)$:

$$SE(\hat{\beta}) = \left[\frac{\partial^2}{\partial \boldsymbol{\beta}^2} \left(\sum_{i=1}^{n} y_i \tilde{\mathbf{x}}_i^T \boldsymbol{\beta} - \sum_{i=1}^{n} \ln(1 + e^{\tilde{\mathbf{x}}_i^T \boldsymbol{\beta}}) \right) \right]^{-1}$$

Hypothesis Testing and Inference (cont)

- Z-statistic large $\Rightarrow$ *p-value* small,
  $\Rightarrow$ null hypothesis should be rejected (when p-value is less than some significance level, 5%, for example).
  $\Rightarrow$ X is associated with Y
  $\Rightarrow$ X is a significant factor.
- Z-statistic small $\Rightarrow$ *p-value* large
  $\Rightarrow$ null hypothesis should not be rejected (when (when p-value > 0.05).
  $\Rightarrow$ X and Y is most likely not related.
  $\Rightarrow$ X is an unimportant factor to Y.
- The interception $\hat{\beta}_0$ is typically not of interest and only for fitting data.

Hypothesis Testing and Inference (cont)

Logistic Regression in sklearn

```
sklearn.linear_model.LogisticRegression(penalty='l2', *, dual=False,
tol=0.0001, C=1.0, fit_intercept=True, intercept_scaling=1,
class_weight=None, random_state=None, solver='lbfgs', max_iter=100,
multi_class='auto', verbose=0, warm_start=False, n_jobs=None,
l1_ratio=None)
```

penalty = l1, l2, elasticnet, none

```
from sklearn.datasets import load_iris
from sklearn.linear_model import LogisticRegression
X, y = load_iris(return_X_y=True)
clf = LogisticRegression(random_state=0).fit(X, y)
clf.predict(X[:2, :])
clf.predict_proba(X[:2, :])
clf.score(X, y)
```

Practical with Logistic Regression

Practice 1: p_flame1.py

Practice 2: p_spiral.py