Data Management

Supervised learning

Malka Guillot

HEC Liège | ECON2306

Table of Contents

Reference:

- JWHT, chap 6.3, 10

Unsupervised Learning

- So far, supervised learning methods such as regression
- Unlike for supervised learning, there is no known goal
  - **No labeled data**,
  - Not a prediction exercise
- The algorithm **discovers** patterns in the data
- We (human) **interpret** the results
  - More subjective than supervised learning
- Hard to assess the results

Setting

- Features $X_1, X_2, \ldots X_p$ measured on n observations, but no associated labeled variable Y
- **Dimensionality reduction** methods are needed
  - ML pbs often involve thousands of features.
  - Especially in the case of **text data**,
  - Not just for computational tractability, but also to help find a good solution.
- Can be use as a **descriptive tool**
  - Can we extract information from the data and visualize it?
  - Can we discover subgroups among the variables or among the observations?

Examples

- Dimension reduction for pre-processing
- Costumer segmentation in marketing

Dimensionality Reduction

Principal Component Analysis (PCA)

- Popular **dimension reduction** technique.
- Identifies the axis that accounts for the **largest amount of variance** in the data.
- Finds a second axis, orthogonal to the first, that accounts for the largest amount of the remaining variance, and so on
- The unit vector defining the i^{th} axis is called the i^{th} **principal component**.

Objectives

- Summarize a large set of feature variables with a smaller number of representative variables
  - collectively explain most of the variability in the original dataset
- Find a **low-dimensional representation** of the data that captures as much of the information possible
- If we can obtain a 2-dimensional representation, then we can plot the observations in 2D.

Principal Components

- What we are after
- Each of the dimensions found by the PCA is a linear combination of the p features.
- The **First Principal Component** of a set of features $X_1, X_2, \ldots, X_p$
  - $=$ the normalized linear combination of the features:

$$Z_1 = \phi_{11} X_1 + \phi_{21} X_2 + \ldots + \phi_{p1} X_p$$

    that has the largest variance
- Normalized means that $\sum_{j=1}^{p} \phi_{j1}^2 = 1$
- $\phi_1 = (\phi_{11}, \phi_{21}, \ldots \phi_{p1}),^T$ is the **loading vector** of the first principal component

Computing the First Principal Component

- We solve:

$$\max_{\phi_{11},\phi_{21},\ldots,\phi_{p1}} \underbrace{\frac{1}{n}\sum_{i=1}^{n}\left(\sum_{j=1}^{p}\phi_{j1}x_{ij}\right)^2}_{\text{Sample variance of } Z_1}$$

$$\text{subject to } \sum_{j=1}^{p}\phi_{j1}^2 = 1$$

- Re-writen :

$$\max_{\phi_{11},\phi_{21},\ldots,\phi_{p1}} \frac{1}{n}\sum_{i=1}^{n} z_{i1}^2 \text{ subject to } \sum_{j=1}^{p}\phi_{j1}^2 = 1$$

  - as $\frac{1}{n}\sum_{i=1}^{n} x_{ij} = 0$ (mean zero property)

Computing the First Principal Component

- Using eigen decomposition (outside the scope of the class)
- $z_{11}, \ldots, z_{n1}$ are the *scores *of the first principal component
- Solved using Singular Value Decomposition (SVD) [a standard linear algebra tool]

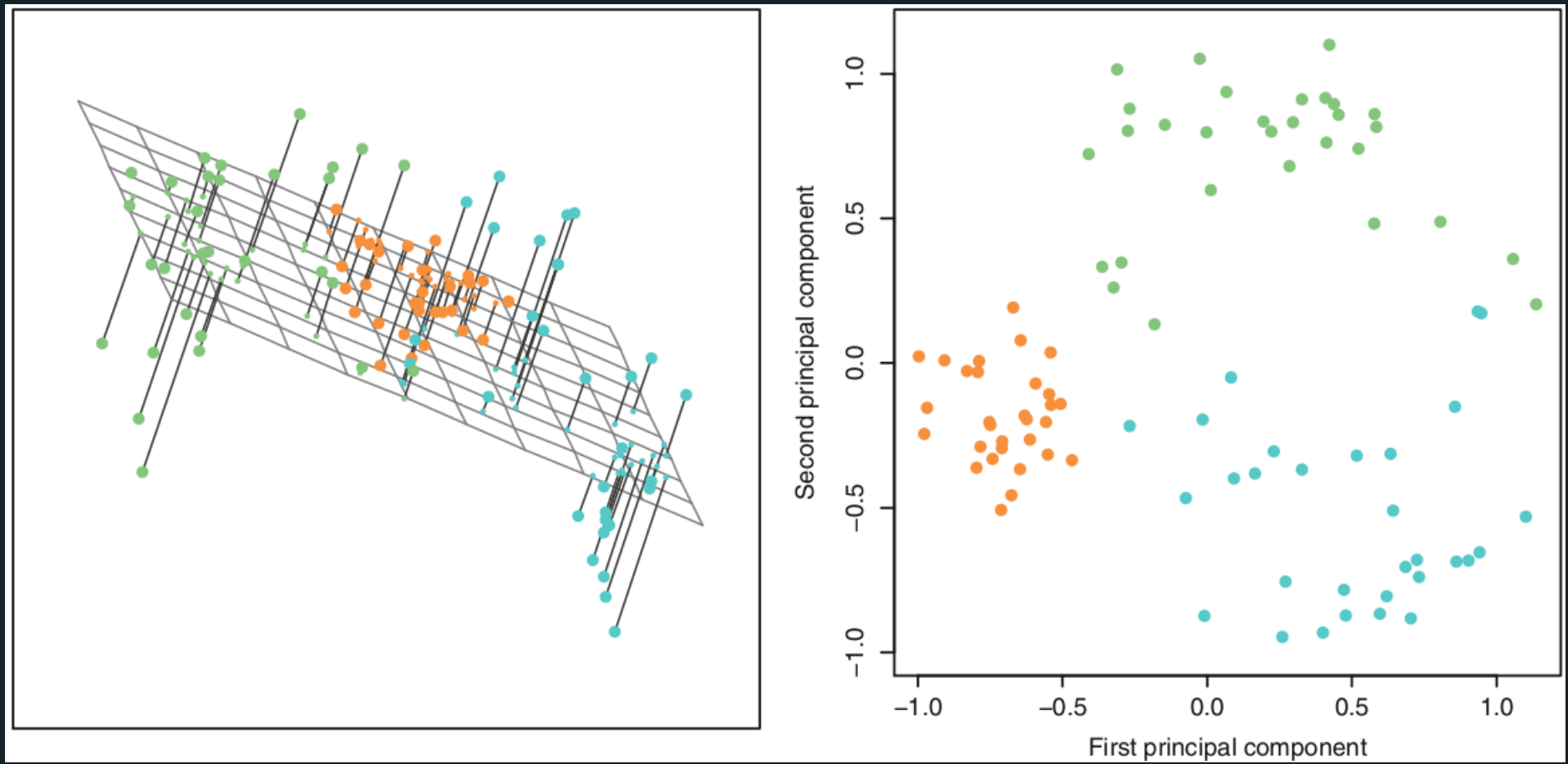
Finding the second principal component Z_2

- Z_2 is the linear combination of $X_1, X_2, \ldots, X_p$:

$$Z_2 = \phi_{12} X_1 + \phi_{22} X_2 + \ldots + \phi_{p2} X_p$$

- With maximal variance out of all linear combinations that are uncorrelated with $Z1$
- Z_2 uncorrelated with $Z_1 \Leftrightarrow \phi_2$ is orthgonal to ϕ_1

Illustration in 3D, projected on a 2D space



Left: simulated data in 3 dimensions
Right: projection on the first 2 principal components (plan represented on the left)

Alternative Interpretation

- The M principal component score vectors + M principal component loading vectors:
  - can give a good approximation to the data when M is **sufficiently large**.
- When $M = min(n - 1, p)$, then the representation is exact

Pre-processing the variables

- Variables should
  - be centered to have mean zero
  - have the same variance 1
- the results obtained depend on whether the variables have been individually scaled
- Step done by default in python

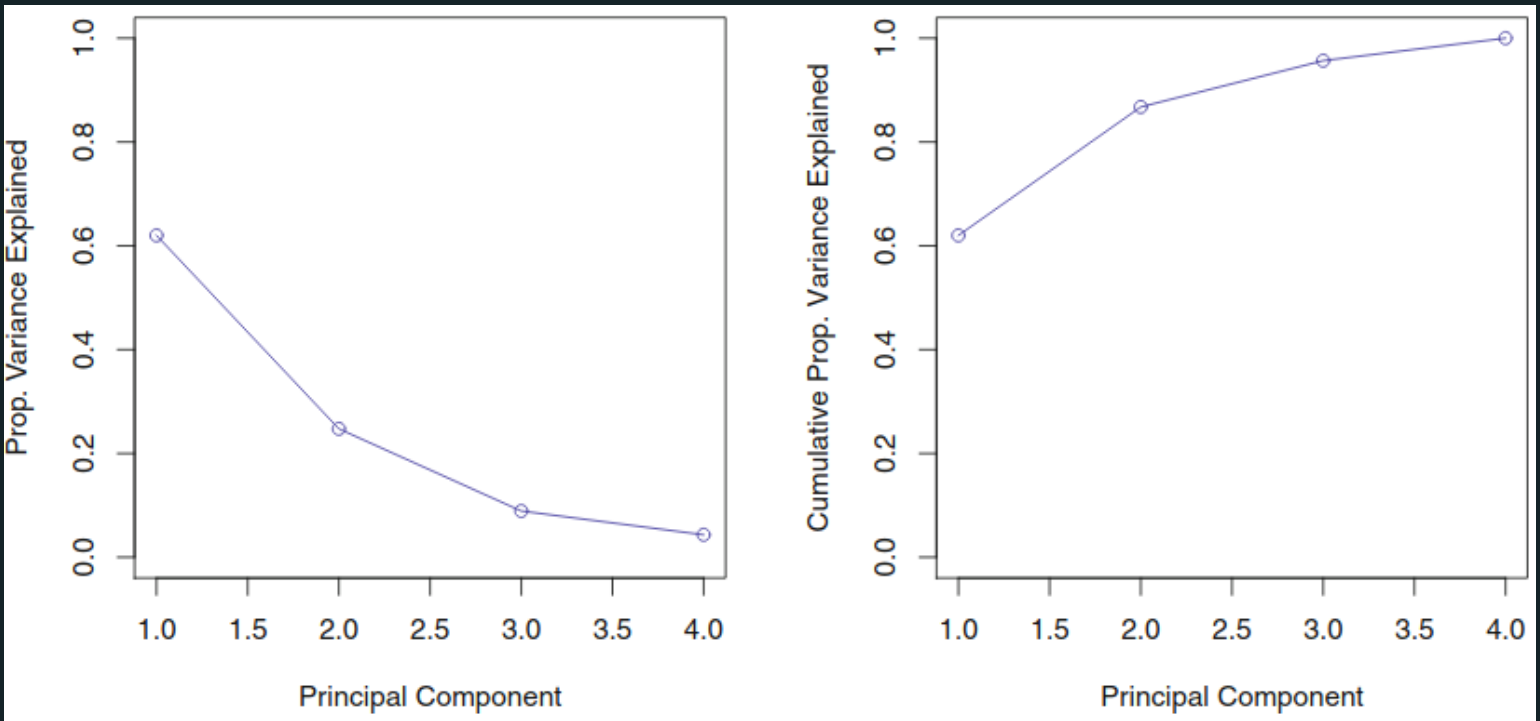
Proportion of the Variance Explained (PVE)

How much of the information in a given data set is lost by projecting the observations onto the first few PC?

→ plot the proportion of the variance explained by each PC

$$PVE_m = \frac{\text{Variance exeplained by the m}^{th}\text{component}}{\text{Total variance}}$$

PVE


Prop. Variance Explained
0.0
0.2
0.4
0.6
0.8
1.0
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Principal Component
Cumulative Prop. Variance Explained
0.0
0.2
0.4
0.6
0.8
1.0
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Principal Component

Choosing the number of dimensions

No criteria for deciding how many PC are required, but some **rules of thumb**:

- Choose the **smallest number of PC required to explain a sizable amount** of the variation in the data
- For dimensionality reduction
  - Explaining 95% of the variance is a good objective
- For data visualization:
  - Focus on a small number of axis that you can interpret
  - Do not interpret the components explaining less than 10%

Clustering Methods

Objective

When performing clustering, the aim is to group data into subsets so that:

- the objects grouped in each subset are similar, close to one another, **homogeneous**
- and different from the objects in other groups.

⇒ find some structure in the data.

2 possibilities

We can

- **Cluster observations** on the basis of the features in order to identify subgroups among the observations,
- or we can **cluster features** on the basis of the observations in order to discover subgroups among the features

Equivalent: simply **transpose** the data matrix

K-means clustering

Partitioning the data into a **pre-specified** number (k) of clusters

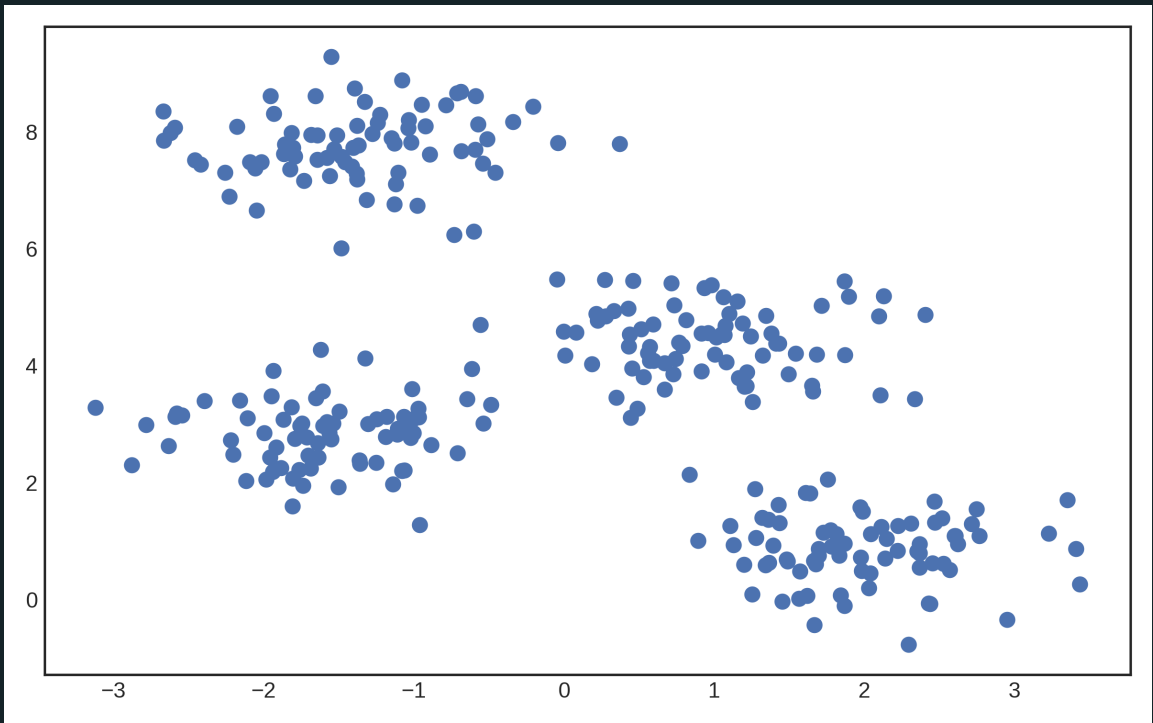
The partitioning corresponds to an **optimization problem** which consists in:

- partitioning the data into k clusters of equal variance
- so that it minimizes the within-cluster sum-of-squares [**inertia**]:

$$\sum_{i=0}^{k} \min_{\mu_j}(||x_i - \mu_j||^2)$$

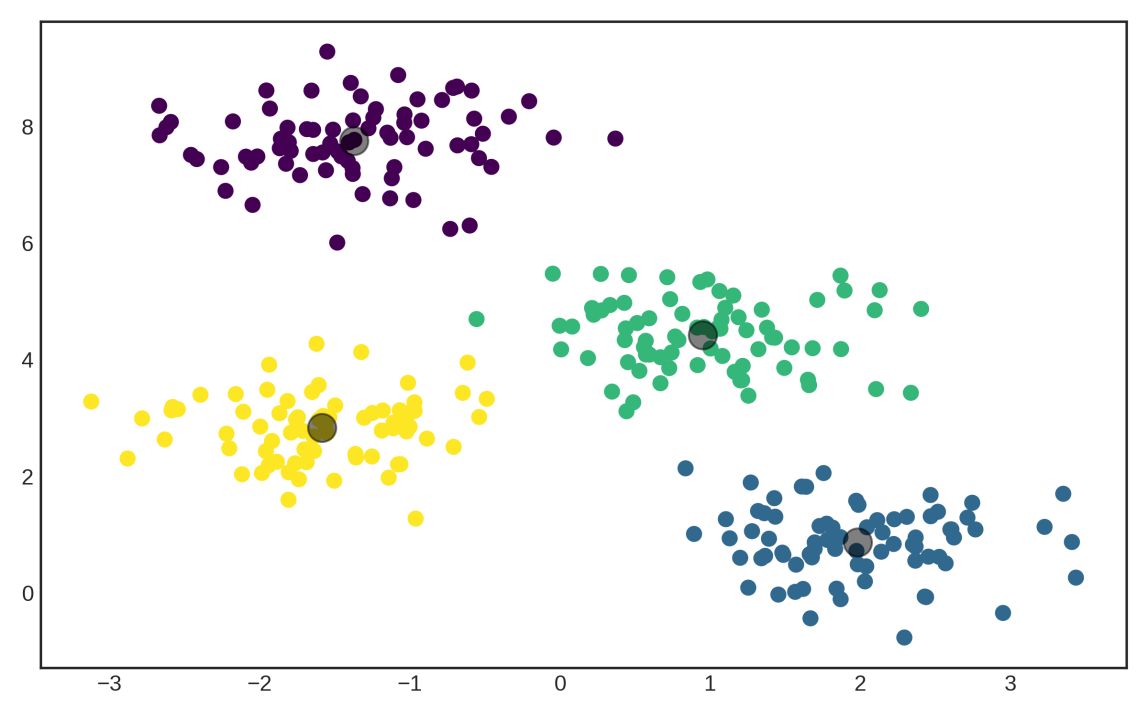
- Each cluster is represented by the **central vector** or **centroïd** μ_j

K-means clustering



Simulated data

K-means clustering



4 clusters and their centroïds

K-means algorithm

1. **Randomly assign a number** (1 to k) to each of the observations.

   = initial cluster assignments

2. **Iterate until the cluster assignments stop changing**:

   1. For each of the k clusters,

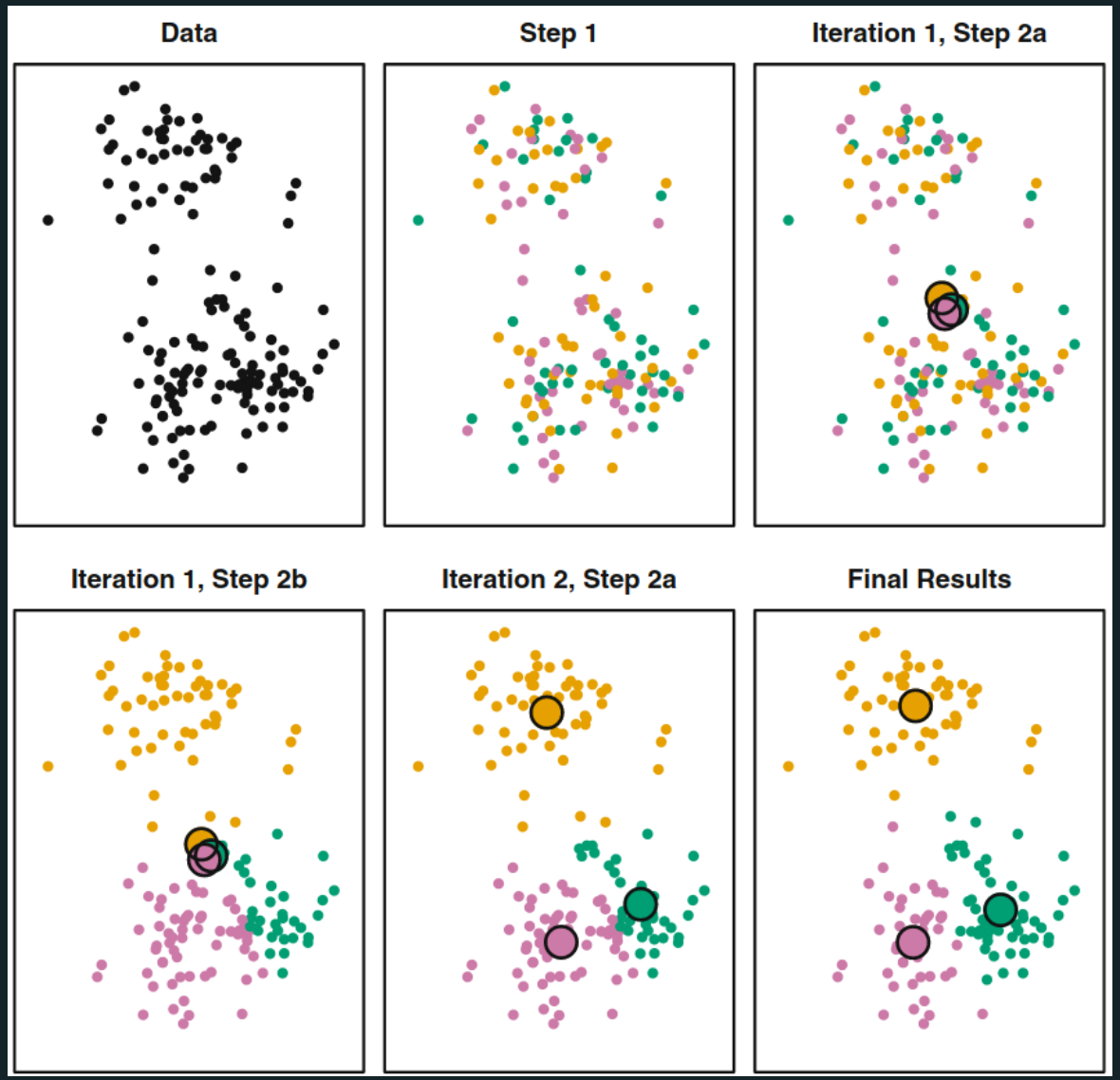
      - compute the cluster **centroid**.
      - The k^{th} cluster centroid is the vector of the p feature means for the observations in the k^{th} cluster.

   2. Assign each observation to the cluster whose centroid is closest

      - where **closest** is defined using Euclidean distance

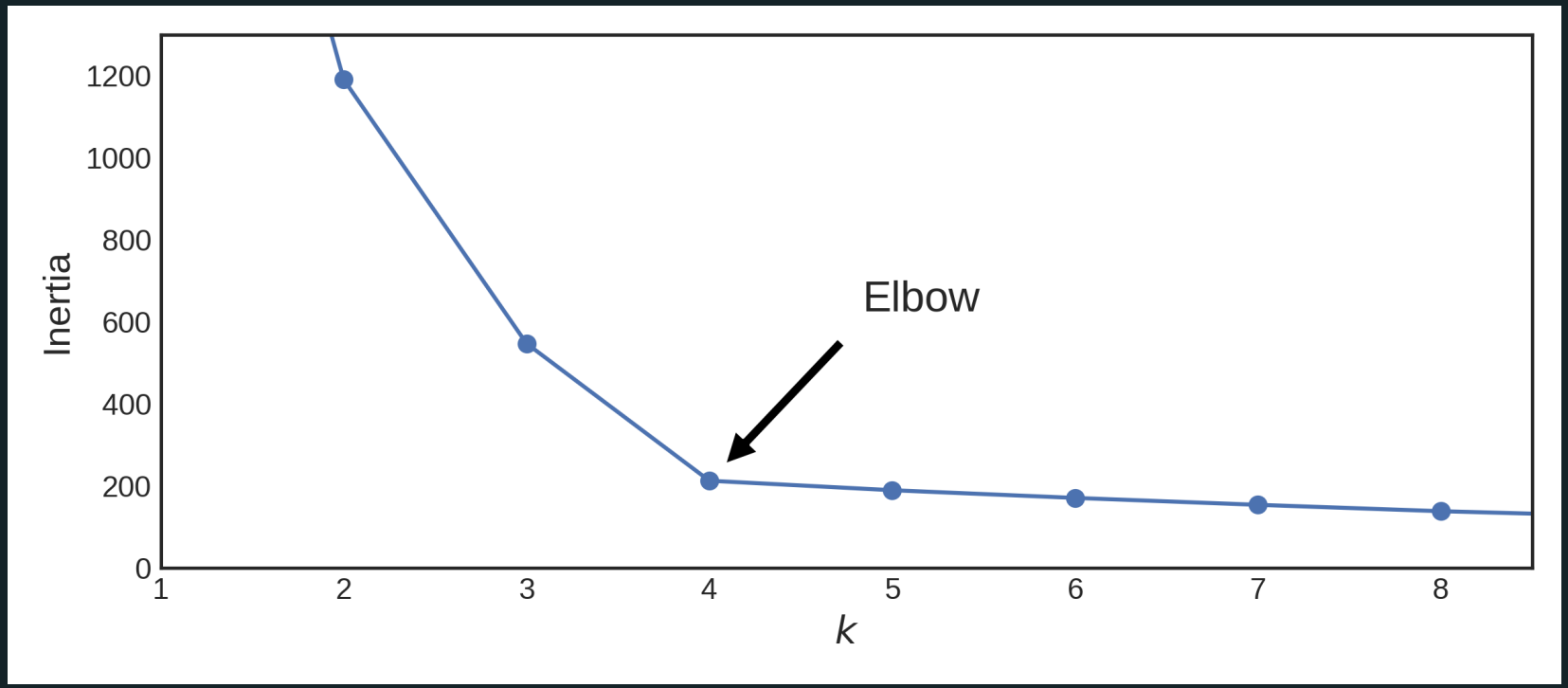
→ The algorithm aims to choose **centroids that minimise the inertia** (=within-cluster sum-of-squares criterion)

K-means algorithm: example



Finding the optimal number of clusters

- Most of the time, the number of clusters does not stand out from looking at the data
- Why not picking the model with the lowest inertia?
- Because inertia decreases with the number of clusters
- **Rule of thumb**: choose the number of clusters at the “elbow”

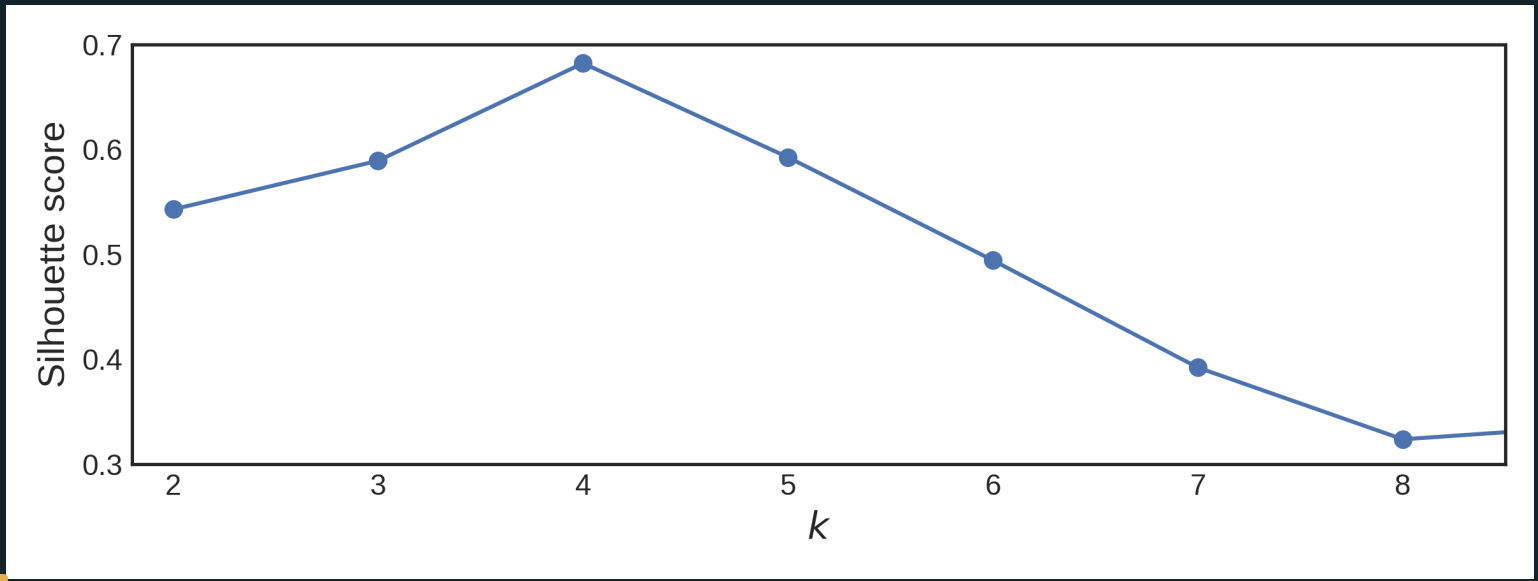

Inertia
1200
1000
800
600
400
200
0
1
2
3
4
5
6
7
8
k
Elbow

Finding the optimal number of clusters

- Can pick the optimal number of clusters with the **silhouette score**:

$$\frac{b_i - a_i}{max(a_i, b_i)}$$

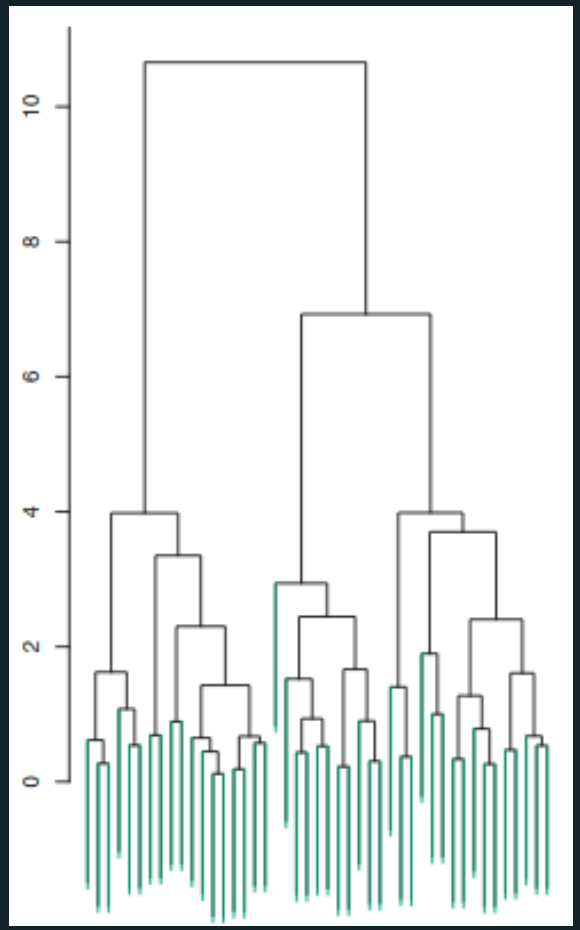
  - a_i mean distance to members of i's cluster
  - b_i mean distance to members of i's second-closest cluster



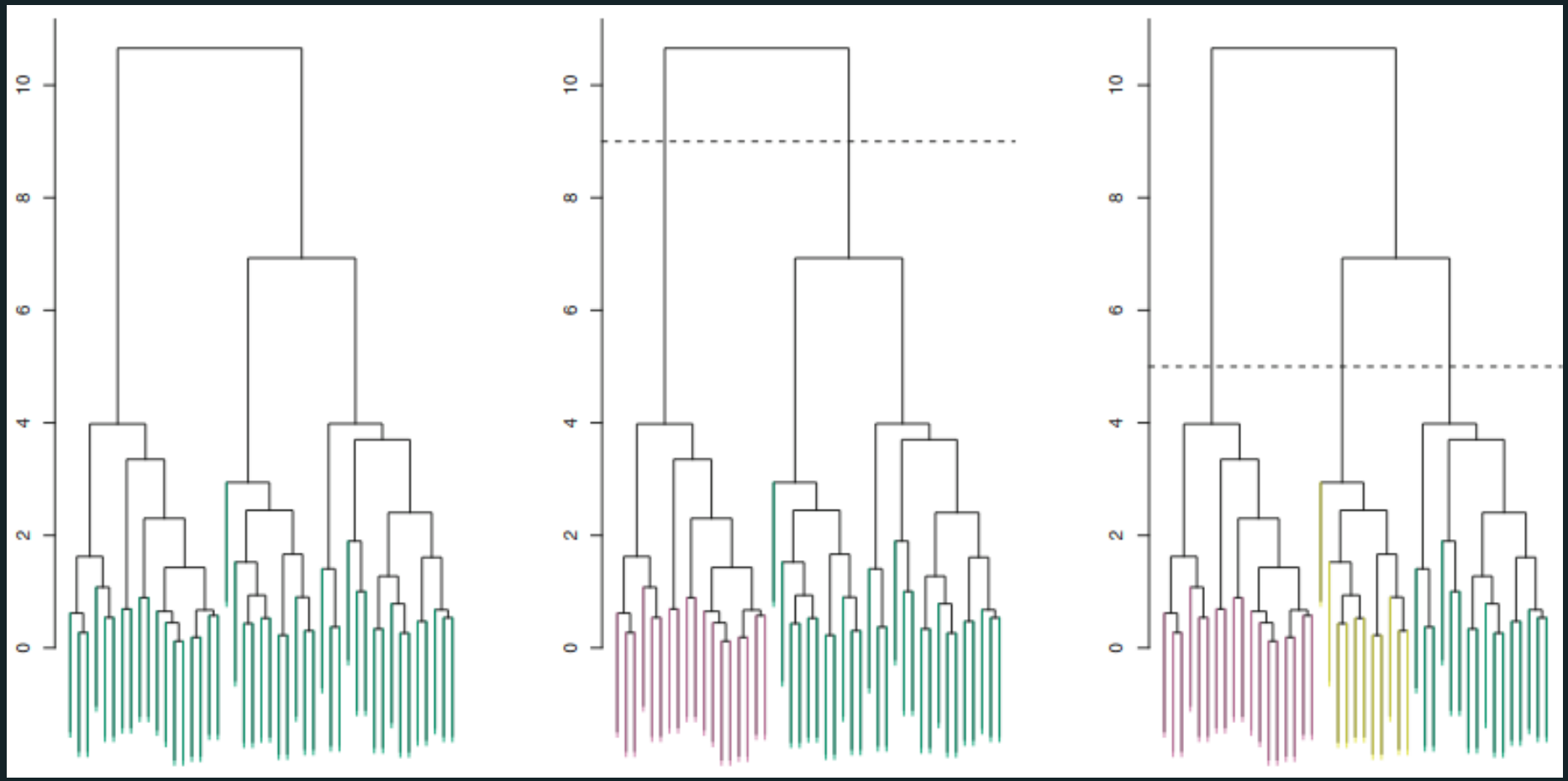
Hierarchical Clustering

- Alternative to k-means
- No pre-existing choice of k
- Tree-based representation of the observation = **dendogram**
- Methodology:
  - **agglomerative** clustering (*bottom-up*)
  - Euclidean distance

Dendrogram



Hierarchical Clustering



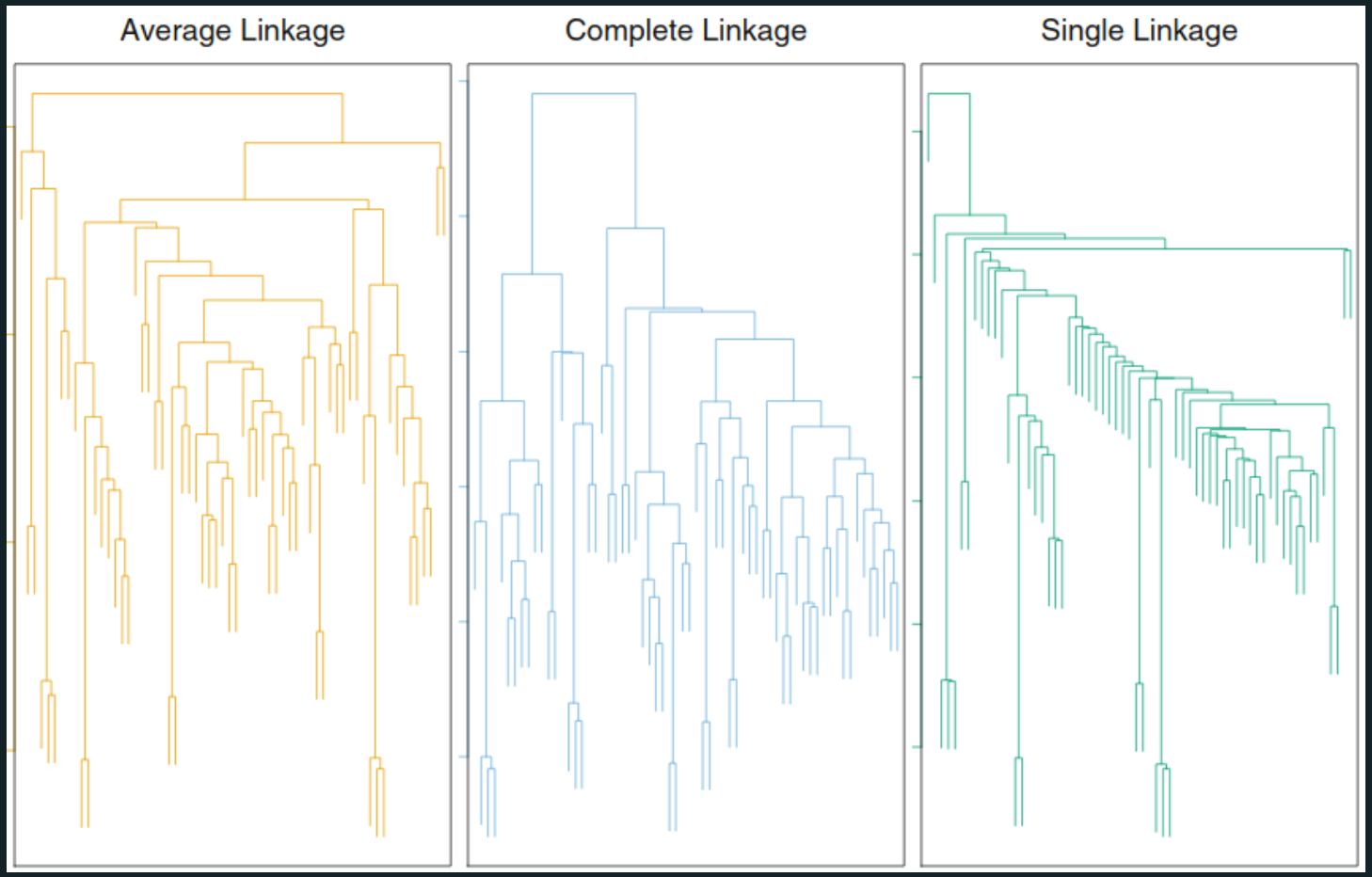
Identifying Clusters

- One single dendrogram can be used to obtain any number of clusters
- *Common practice*: select by eye a sensible number of clusters
- **Dissimilarity measure** between each pair of observations
  - E.g. Euclidean distance
  - Concept extended to a pair of *groups of observations*
    - *Commonly-used linkages*: Complete/single/average/centroïd

Hierarchical Clustering Algorithm

1. **Bottom**: each observation = 1 clusters
    1. Measure pairwise dissimilarities
    2. Fuse the most similar observations $\rightarrow n-1$ **clusters**
2. For $i = n, n-1, \cdots, 2$:
    - Examine all pairwise inter-cluster dissimilarities among their clusters and
    - Identify the pair of clusters that are **least dissimilar** $\rightarrow$ \$ fuse them
    - Compute the **new pairwise** inter-cluster dissimilarities among the $i-1$ remaining clusters

Results Depend on the Type of Linkage



Other Clustering Algorithms

- **DBSCAN** defines clusters as continuous regions of high density
  - Works well if all the clusters are dense enough and if they are very well separated by low-density regions
  - Detects and excludes outliers automatically