UNIVERSITÀ DEGLI STUDI DI BERGAMO
Dipartimento di Ingegneria Gestionale, dell'Informazione e della Produzione

Lesson 12.

Unsupervised learning

DATA SCIENCE AND AUTOMATION COURSE

MASTER DEGREE SMART TECHNOLOGY ENGINEERING

TEACHER
Mirko Mazzoleni

PLACE
University of Bergamo

Outline

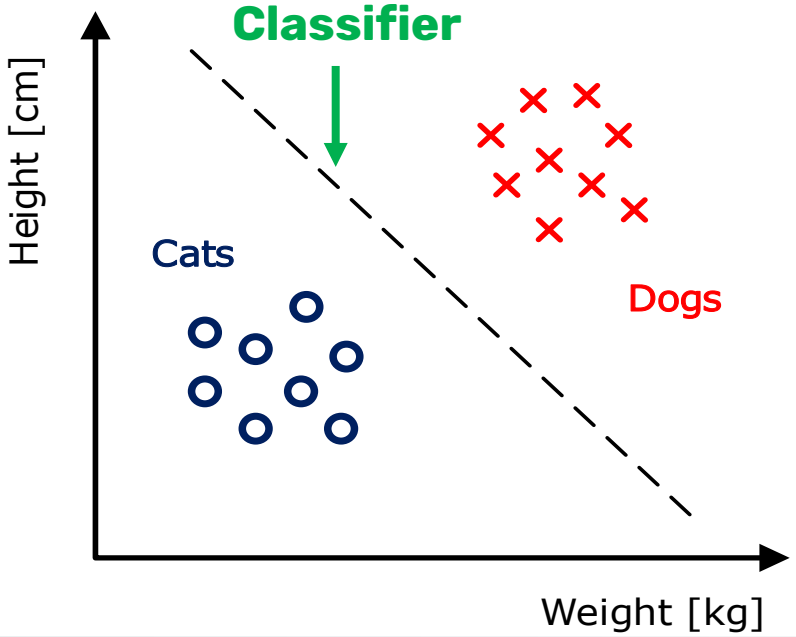
1. Introduction
2. K-means clustering
3. Hierarchial clustering
4. Principal Component Analysis (PCA)

Outline

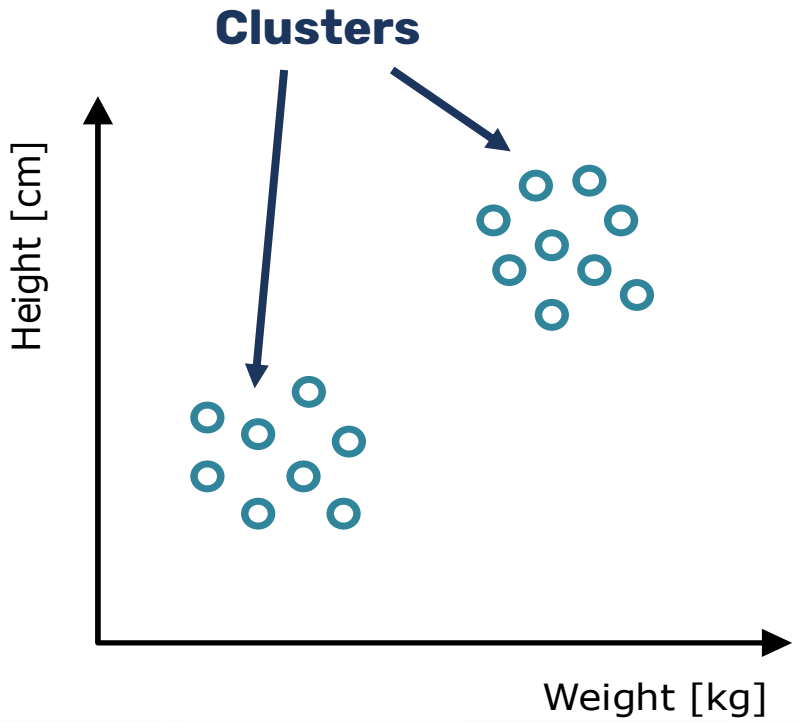
1. **Introduction**
2. K-means clustering
3. Hierarchial clustering
4. Principal Component Analysis (PCA)

Supervised vs. unsupervised

In **supervised** learning, we have a dataset $\mathcal{D} = \{(\boldsymbol{x}(1), y(1)), \ldots, (\boldsymbol{x}(N), y(N))\}$



In **unsupervised** learning, we have a dataset $\mathcal{D} = \{(\boldsymbol{x}(1), ?), \ldots, (\boldsymbol{x}(N), ?)\}$



Goals of unsupervised learning

The main questions that unsupervised learning can answer are:

1. Can we **discover subgroups** among the variables or among the observations?
   - ✓ Subgroups of patients with a disease grouped by their gene expression measurements
   - ✓ Movies grouped by their ratings, assigned by movie viewers
   - ✓ Groups of shoppers characterized by their browsing and purchase histories

2. Is there an **informative** way to **visualize** the data?
   - ✓ Find transformations which enhance certain data characteristics
   - ✓ Reduce the dimensionality of the data to visualize them in a 2-D plot

Motivation

Unsupervised learning presents some nice properties:

- It is often easier to obtain **unlabeled** data than **labeled** data
  - ✓ need to perform a dedicated experiment that can costly or disruptive
  - ✓ difficulty to assess the overall sentiment of a movie review
- If data pertain naturally into **different groups**, than observations in each group can have their own characteristic. In this case, a different supervised model can be **trained specifically** for the data in each group
- Clustering can be used to perform **collaborative filtering**, a technique used in recommendation systems

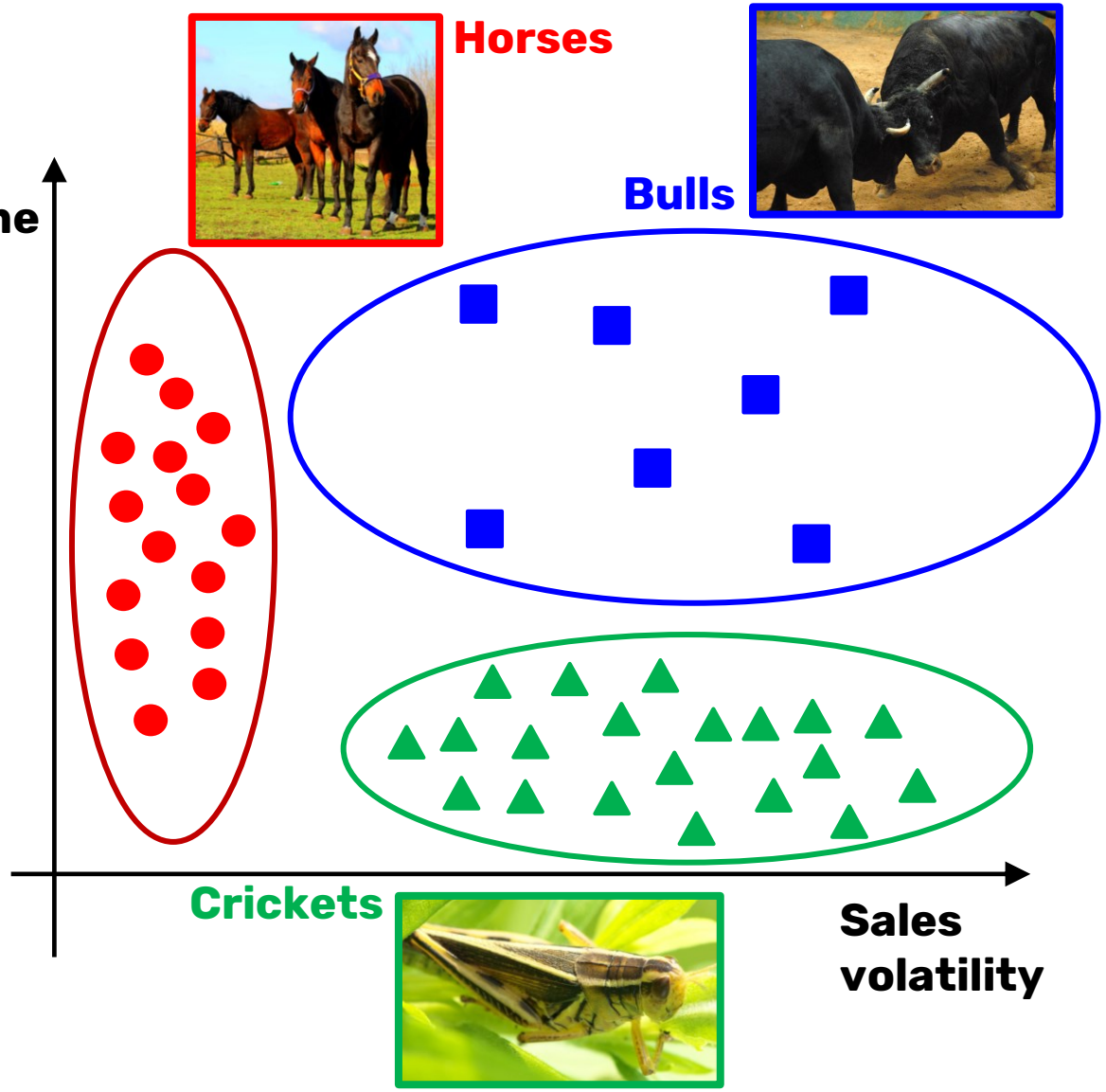
Motivation

Why we want to find groups (clusters):

- Treat **«similar»** objects in the same way
- Treat **«different»** objects differently

Suppose we have several SKUs in stock

- **Horses:** *make-to-stock* reorder strategy
- **Crickets:** *make-to-order* reorder strategy
- **Bulls:** reorder strategy case-by-case



Motivation

Text mining

- Cluster documents for related search
- Cluster words for query suggestion

Recommender systems and advertising

- Cluster users for item/advertising recommendation

Image search

- Cluster images for similar image search and duplication detection
- Image compression

Outline

1. Introduction

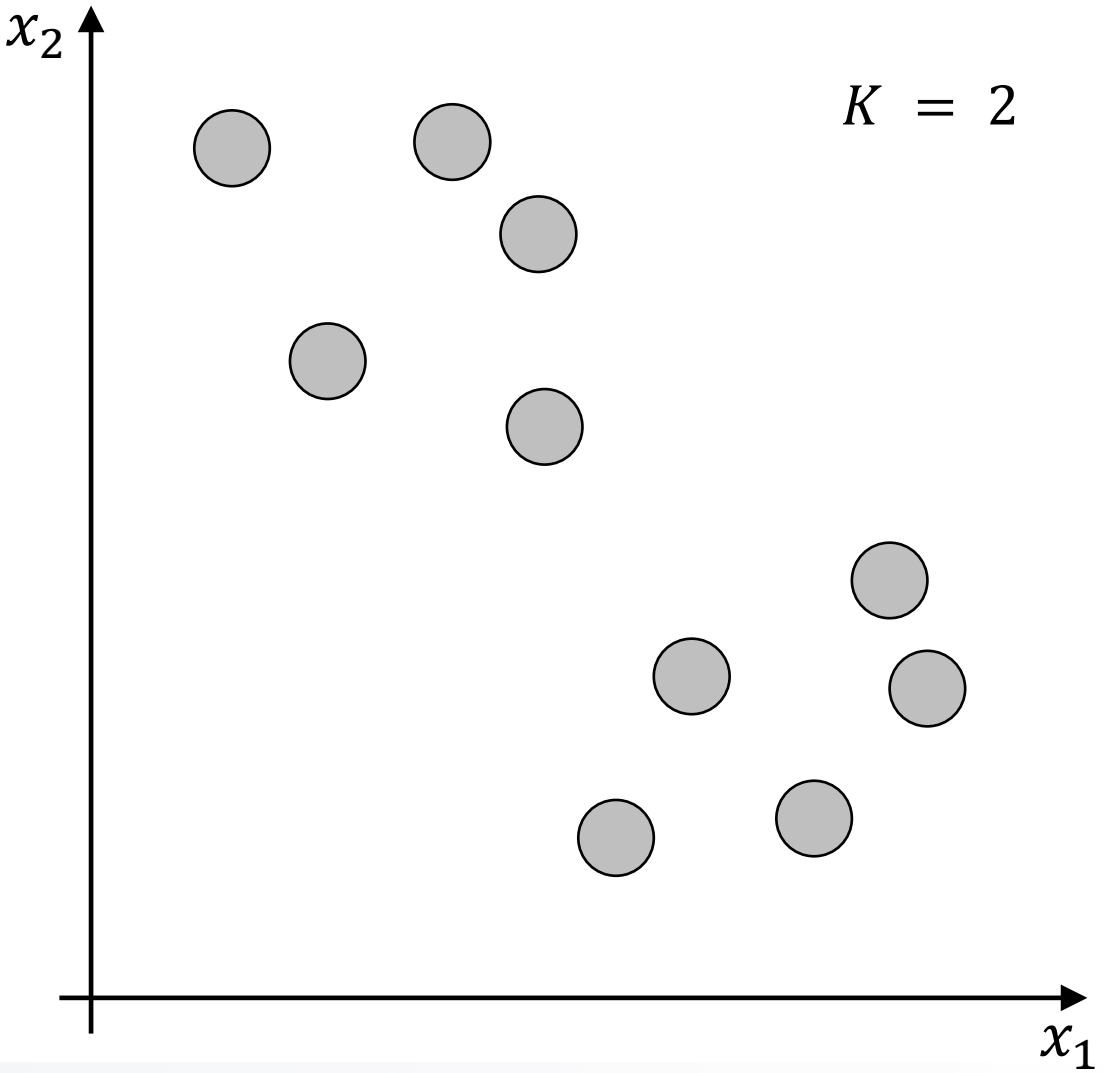
2. **K-means clustering**

3. Hierarchial clustering

4. Principal Component Analysis (PCA)

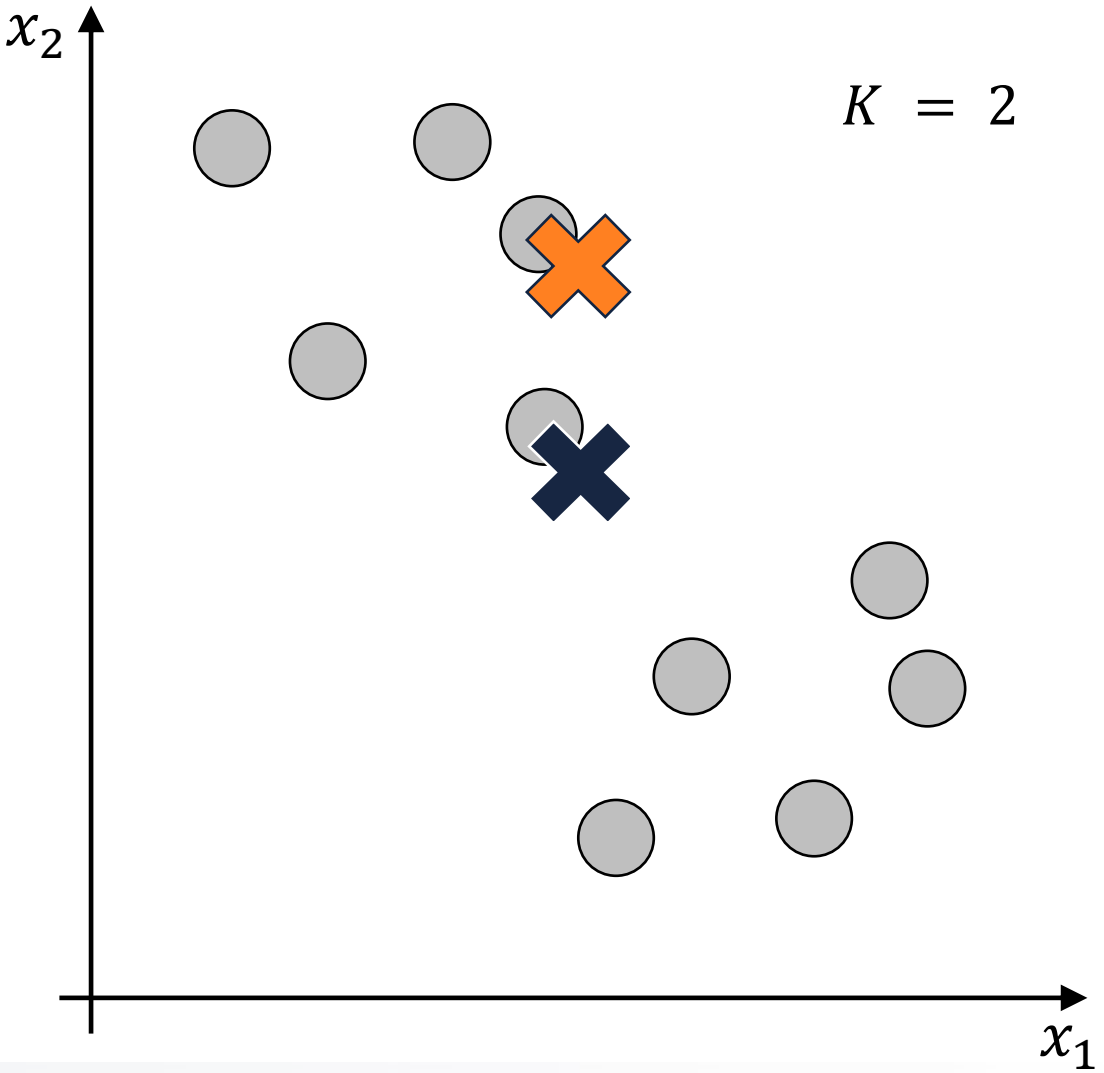
K-means in action

1. Choose the number of clusters K
2. Randomly initialize K cluster centroids
3. Assign each instance to the closest centroid
4. Re-estimate the centroid of each cluster
5. Iterate over 3. and 4. until there are no improvements



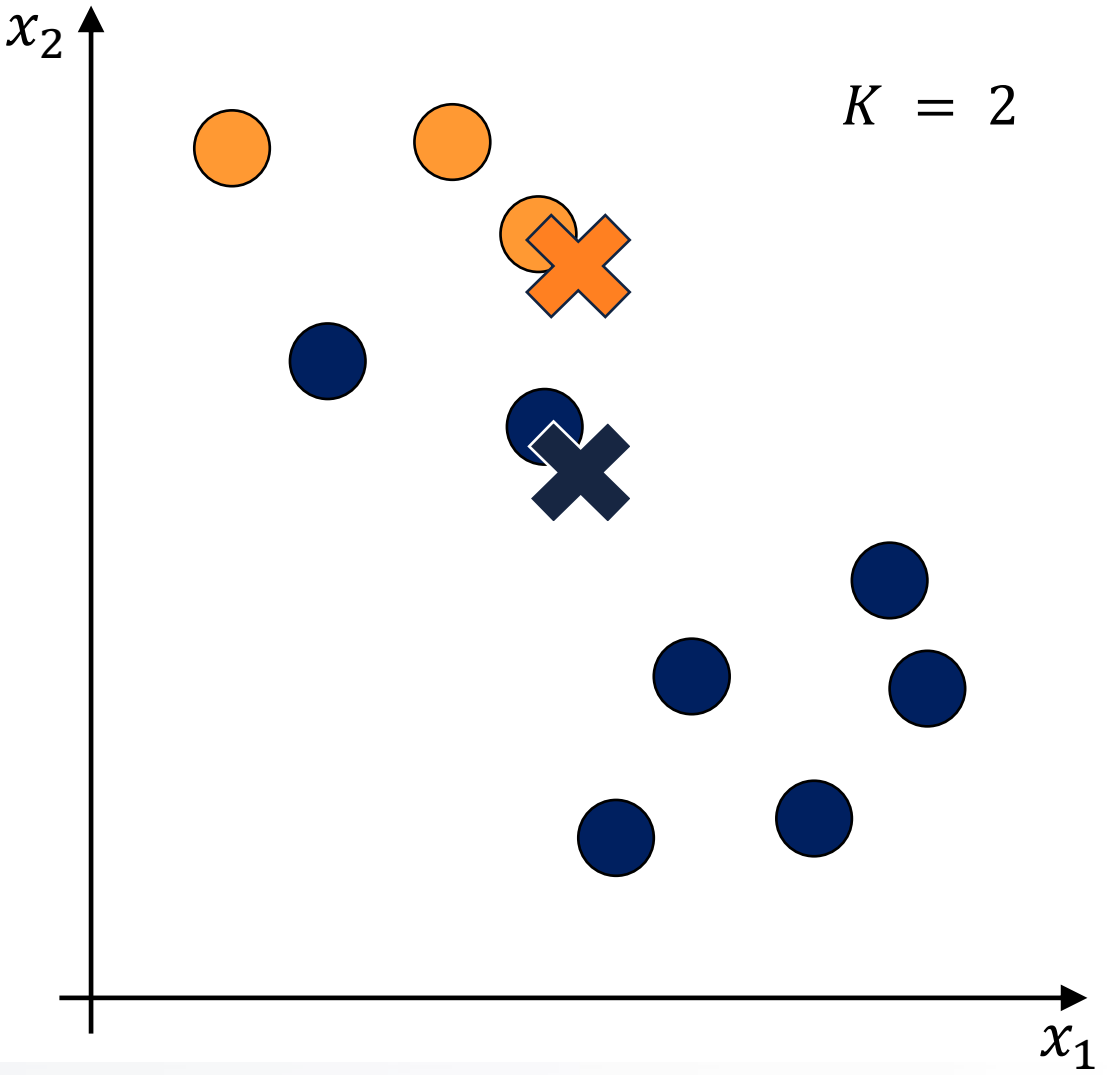
K-means in action

1. Choose the number of clusters K
2. Randomly initialize K cluster centroids
3. Assign each instance to the closest centroid
4. Re-estimate the centroid of each cluster
5. Iterate over 3. and 4. until there are no improvements



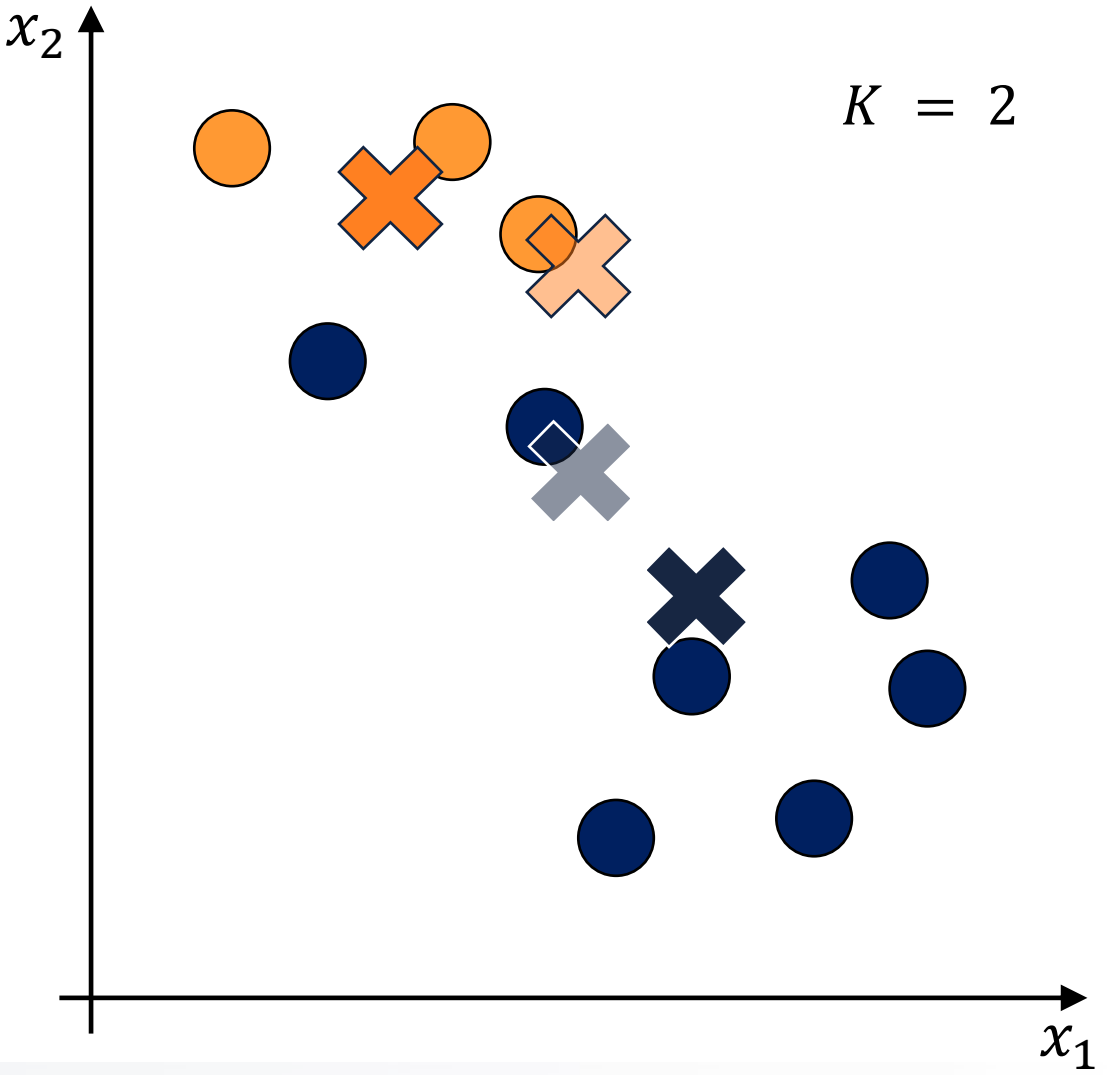
K-means in action

1. Choose the number of clusters K
2. Randomly initialize K cluster centroids
3. Assign each instance to the closest centroid
4. Re-estimate the centroid of each cluster
5. Iterate over 3. and 4. until there are no improvements



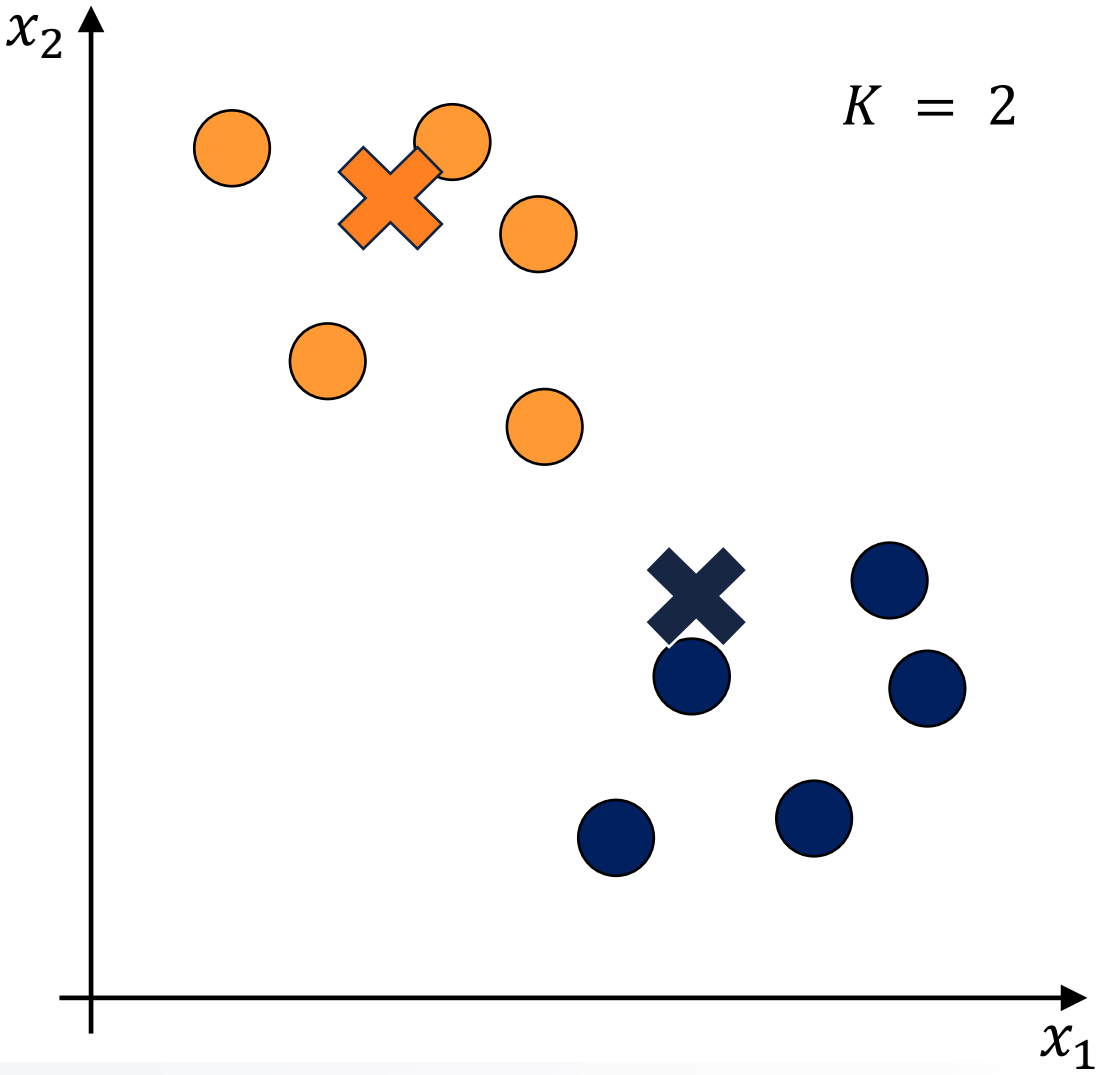
K-means in action

1. Choose the number of clusters K
2. Randomly initialize K cluster centroids
3. Assign each instance to the closest centroid
4. Re-estimate the centroid of each cluster
5. Iterate over 3. and 4. until there are no improvements



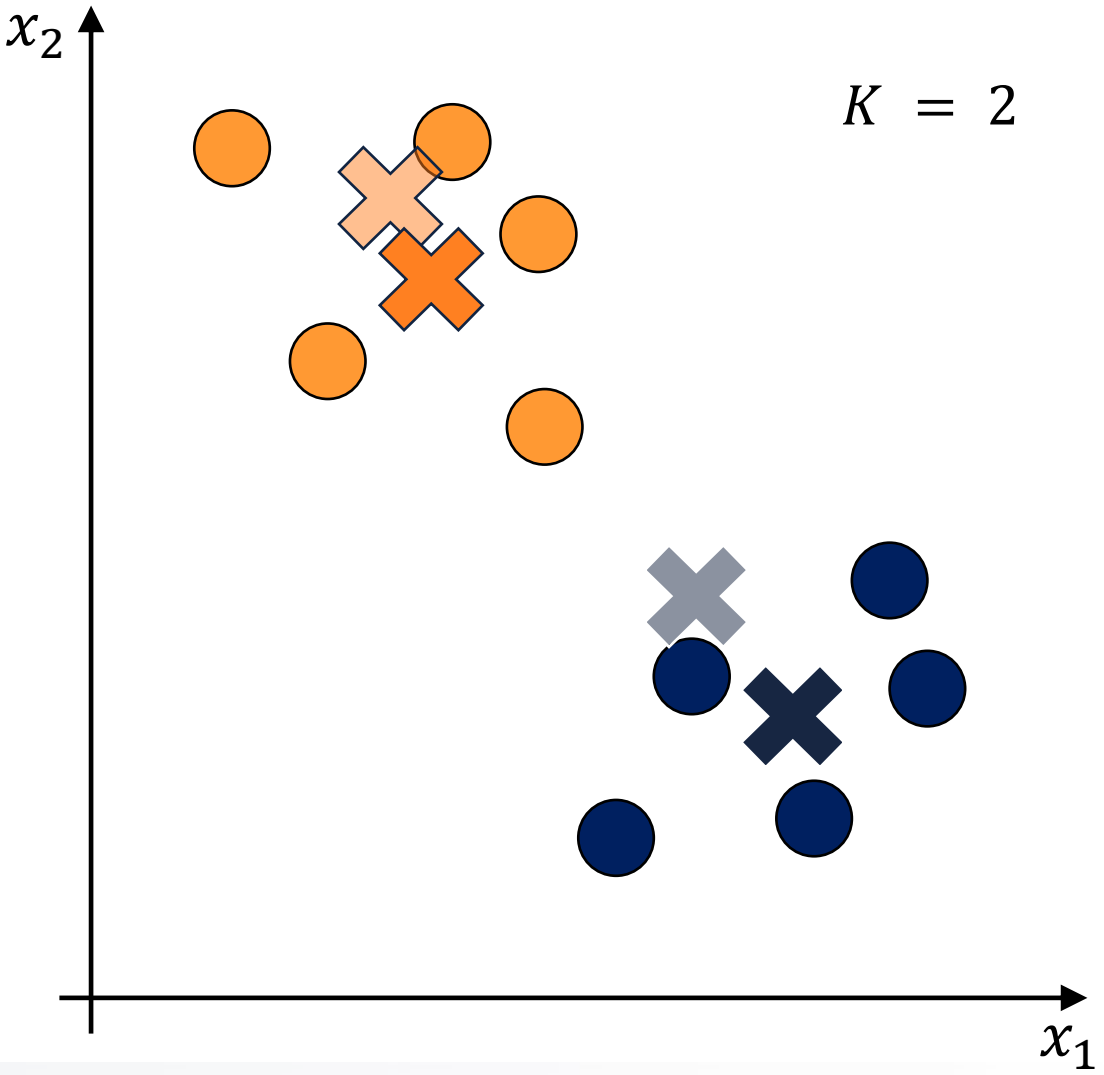
K-means in action

1. Choose the number of clusters K
2. Randomly initialize K cluster centroids
3. Assign each instance to the closest centroid
4. Re-estimate the centroid of each cluster
5. Iterate over 3. and 4. until there are no improvements



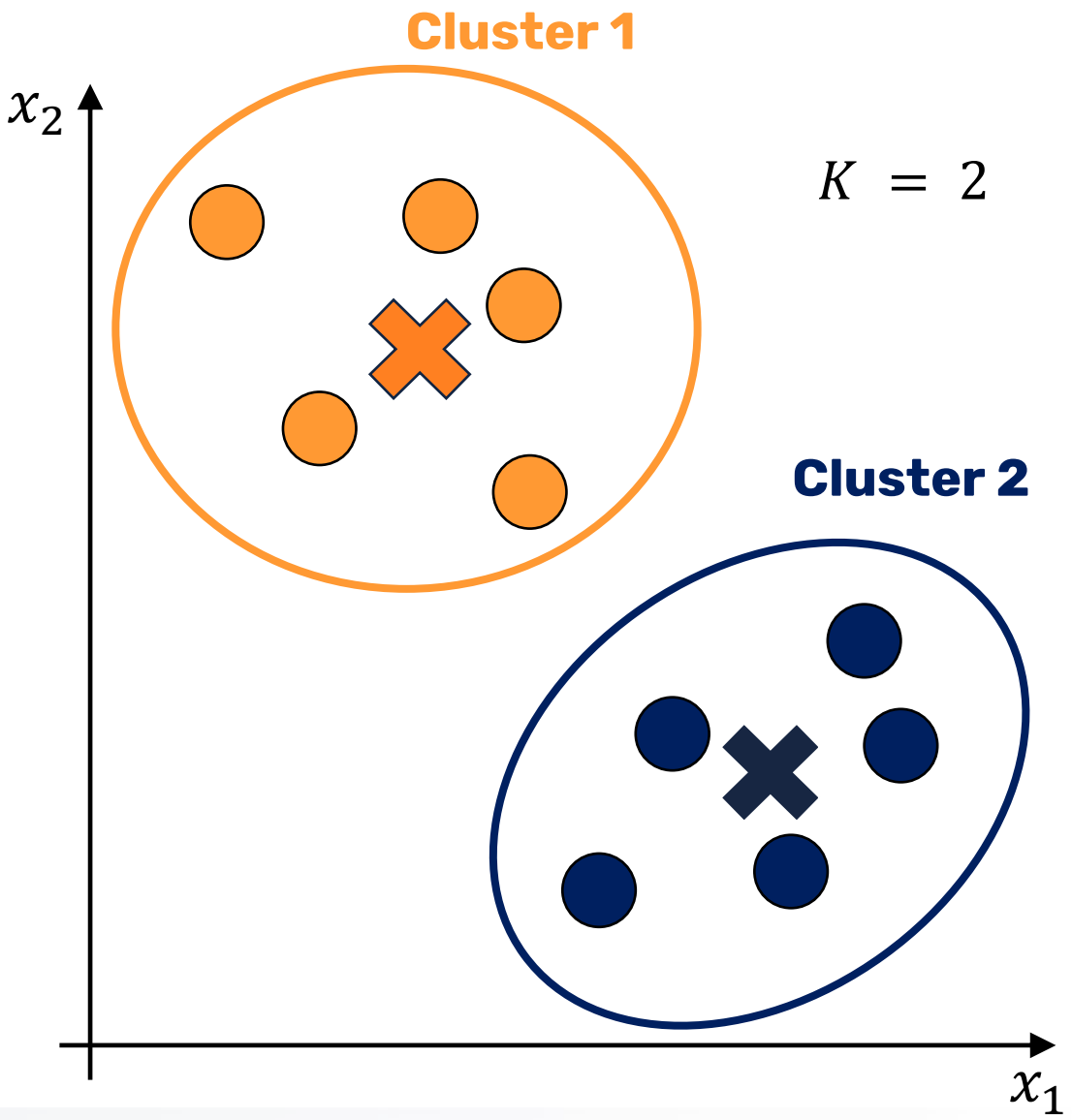
K-means in action

1. Choose the number of clusters K
2. Randomly initialize K cluster centroids
3. Assign each instance to the closest centroid
4. Re-estimate the centroid of each cluster
5. Iterate over 3. and 4. until there are no improvements



K-means in action

1. Choose the number of clusters K
2. Randomly initialize K cluster centroids
3. Assign each instance to the closest centroid
4. Re-estimate the centroid of each cluster
5. Iterate over 3. and 4. until there are no improvements



K-means algorithm

The K-means algorithm is an **iterative** method which provides a solution to the **clustering** problem

Inputs:

- The number of clusters K
- The dataset $\mathcal{D} = \{\boldsymbol{x}(1), \boldsymbol{x}(2), \dots, \boldsymbol{x}(N)\}, \ \boldsymbol{x} \in \mathbb{R}^{d \times 1}$

Notice how **the label y is not required** to perform the clustering procedure

The aim of K-means is to **minimize the total sum** of the squared **distances** of every **point** from its corresponding **cluster centroid**

K-means algorithm

The steps of the K-means algorithm are:

1. Randomly initialize K cluster centroids $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_K \in \mathbb{R}^{d \times 1}$
2. **Repeat**

   2.1 **For** $n = 1 : N$ `#loop through each data point`

   c_n = index of the cluster centroid **closest** to $\boldsymbol{x}(n)$

   2.2 **For** $k = 1 : K$ `#loop through each cluster`

   $\boldsymbol{\mu}_k$ = average of points assigned to cluster k

The steps **2.1** and **2.2** are repeated until the centroids do not change

K-means algorithm

The K-means can be viewed as a **minimization** problem. Define

- c_n: index of the cluster to which example $\boldsymbol{x}_n$ is currently assigned, $c_n \in \{1,2,\dots,K\}$
- $\boldsymbol{\mu}_k$: cluster centroid k, $\boldsymbol{\mu}_k \in \mathbb{R}^d$
- $\boldsymbol{\mu}_{cn}$: cluster centroid of the cluster to which example $\boldsymbol{x}_n$ has been assigned

Distortion cost function

$$J(c_1,\dots,c_2,\boldsymbol{\mu}_1,\dots,\boldsymbol{\mu}_K) = \frac{1}{N}\sum_{n=1}^{N}\left\|\boldsymbol{x}_n - \boldsymbol{\mu}_{c_n}\right\|^2$$

The minimization of this cost function is **computationally difficult** (NP-hard). For this reason, **greedy algorithms** are used that converge to a **local optimum**

K-means algorithm

2. **Repeat**

   2.1 **For** $n = 1 : N$ → Minimize $J(\dots)$ with respect to $c_1, c_2, \dots, c_N$ while holding $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \dots, \boldsymbol{\mu}_N$ constant

   c_n = index of the cluster centroid closest to $\boldsymbol{x}(n)$

   2.2 **For** $k = 1 : K$ → Minimize $J(\dots)$ with respect to $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \dots, \boldsymbol{\mu}_N$ while holding $c_1, c_2, \dots, c_N$ constant

   $\boldsymbol{\mu}_k$ = average of points assigned to cluster k

The distortion cost function $J(c_1, \dots, c_2, \boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_K)$ **decreases** at every step

If, at the end of the procedure, a cluster **has no points:**

- **Eliminate** the cluster (end with $K - 1$ clusters)
- **Re-initialize** the cluster centroids (if you need K clusters)

Random initialization

The **initialization** of the centroids is done randomly. A recommended way is:

- Randomly pick K training examples
- Set $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \dots, \boldsymbol{\mu}_K$ equal to these K examples

→ A different initialization may lead to a different clustering result

Try **multiple random initializations** (ex. Try 20-50 initializations and keep the best in terms of some clustering metrics, e.g. the K-means cost function)

- Pick the clustering that gave lowest $J(c_1, \dots, c_2, \boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_K)$
- With $K = 2 - 10$, trying a high number of initializations can improve performance
- With $K > 10 - 100$, multiple initializations do not change a lot the final result. The first clustering should be a fairly decent solution

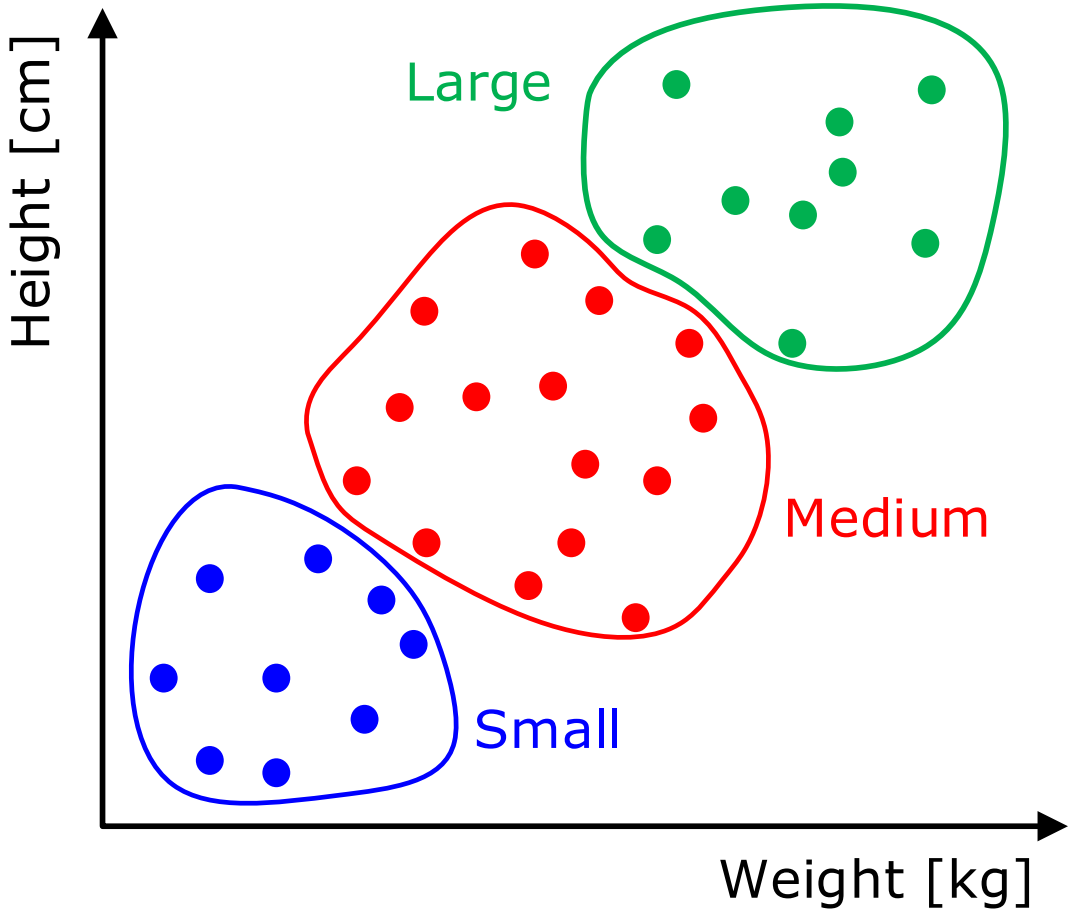
K-means for non-separated clusters

Clustering can be useful even when the clusters are not so separated

Consider a **t-shirt producer** that wants to know **which shirt size** to produce

By **clustering its customers** in three groups, he knows he has to satisfy the people in each cluster with that shirt size

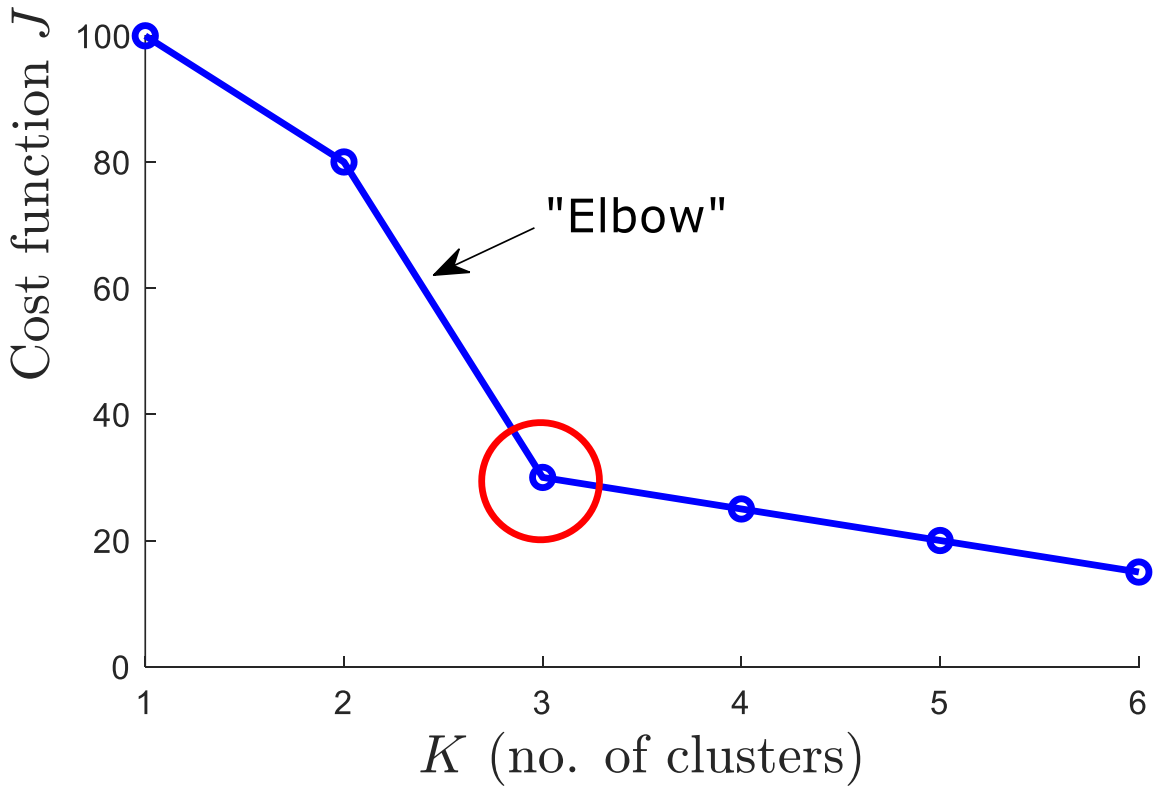
This is an example of **market segmentation**



Chosing the number of clusters

The number of clusters K should be dictated by **problem understanding:**

- In the T-shirts example, ask: if I have 5 clusters, **how well** they fit the customers?
- **How many** shirts can I sell? Will the customers be happier?
- However, there exist **heuristics** to choose K



Outline

1. Introduction
2. K-means clustering
3. **Hierarchial clustering**
4. Principal Component Analysis (PCA)

Hierarchical clustering

It is a clustering method which seeks to build a **hierarchy of clusters**. Two strategies:

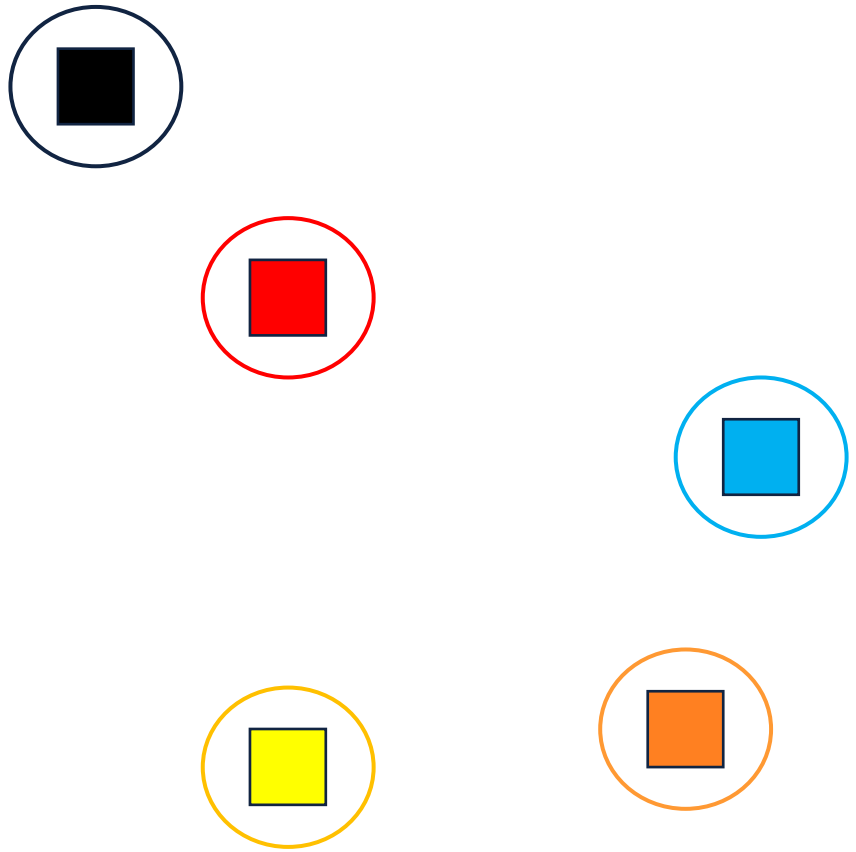
- **Agglomerative:** in the first step each sample represent a cluster. *Bottom-up approach*
- **Divisive:** only one big cluster which represents all the data at the beginning. *Top-down approach*

Differences with respect to K-means:

- Hierarchical clustering **does not need** a pre-specified **number of clusters**
- Results do not depend on any **initialization**
- K-means is faster but does not work well with **not spherical clusters**
- Hierchical clustering results can be more **intuive to visualize** (especially in high dimensions)

Hierarchical clustering - agglomerative

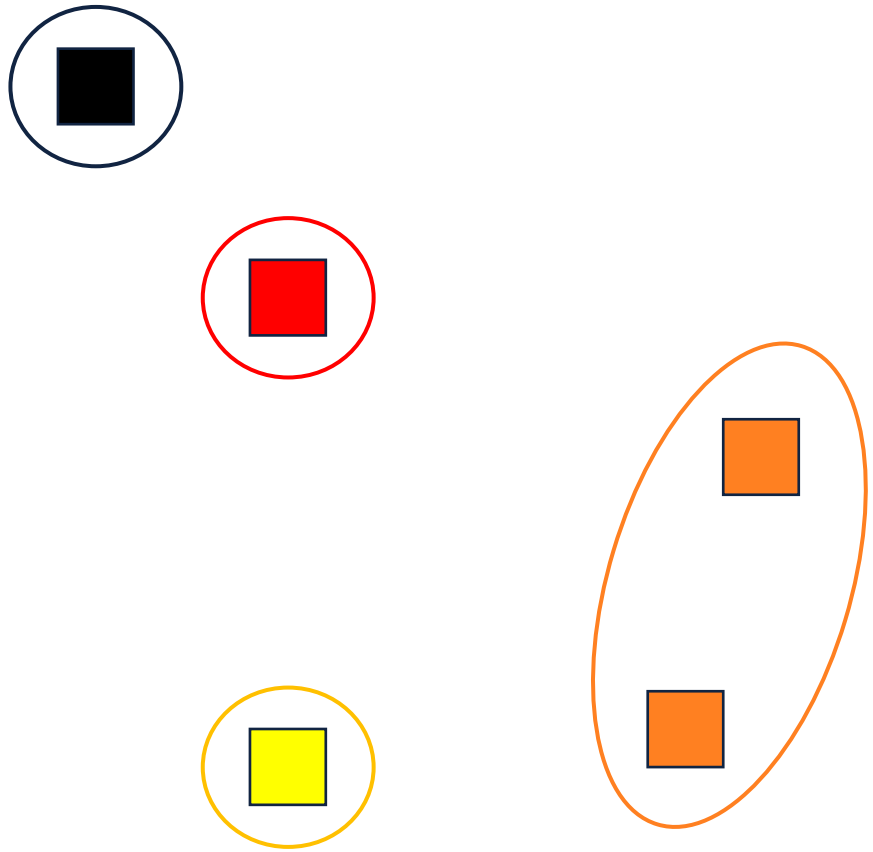
1. Assign a cluster to each sample



Hierarchical clustering - agglomerative

1. Assign a cluster to each sample

2. Combine the two closest clusters (choose your favorite distance method)

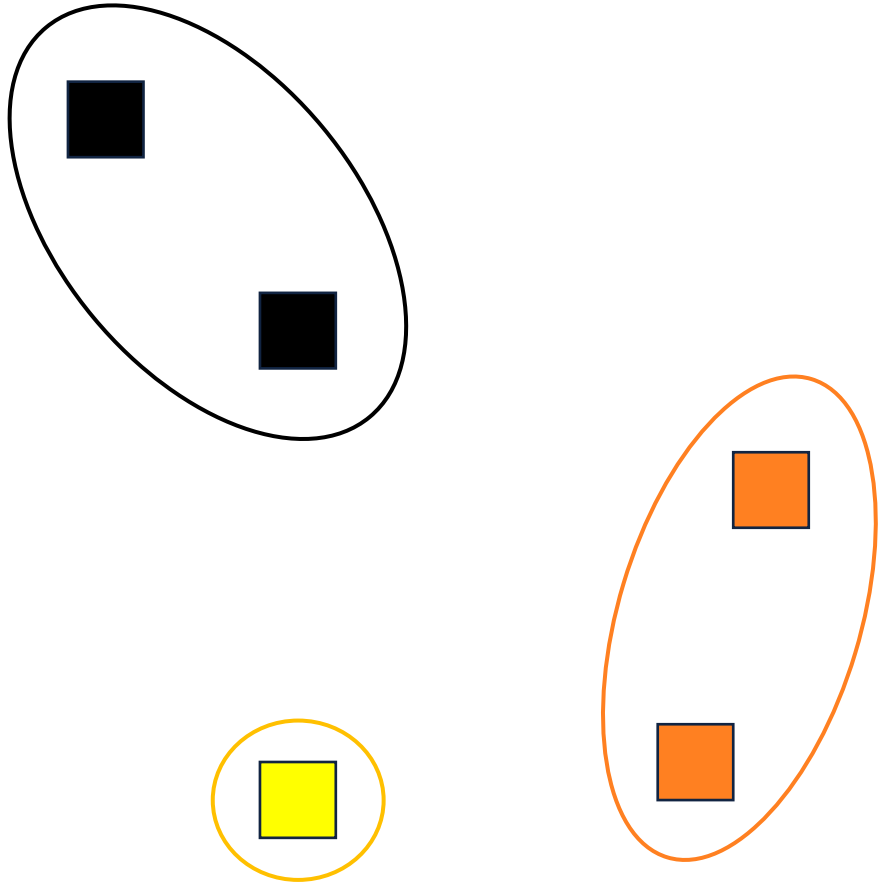


Hierarchical clustering - agglomerative

1. Assign a cluster to each sample

2. Combine the two closest clusters (choose your favorite distance method)

3. Repeat step 2.

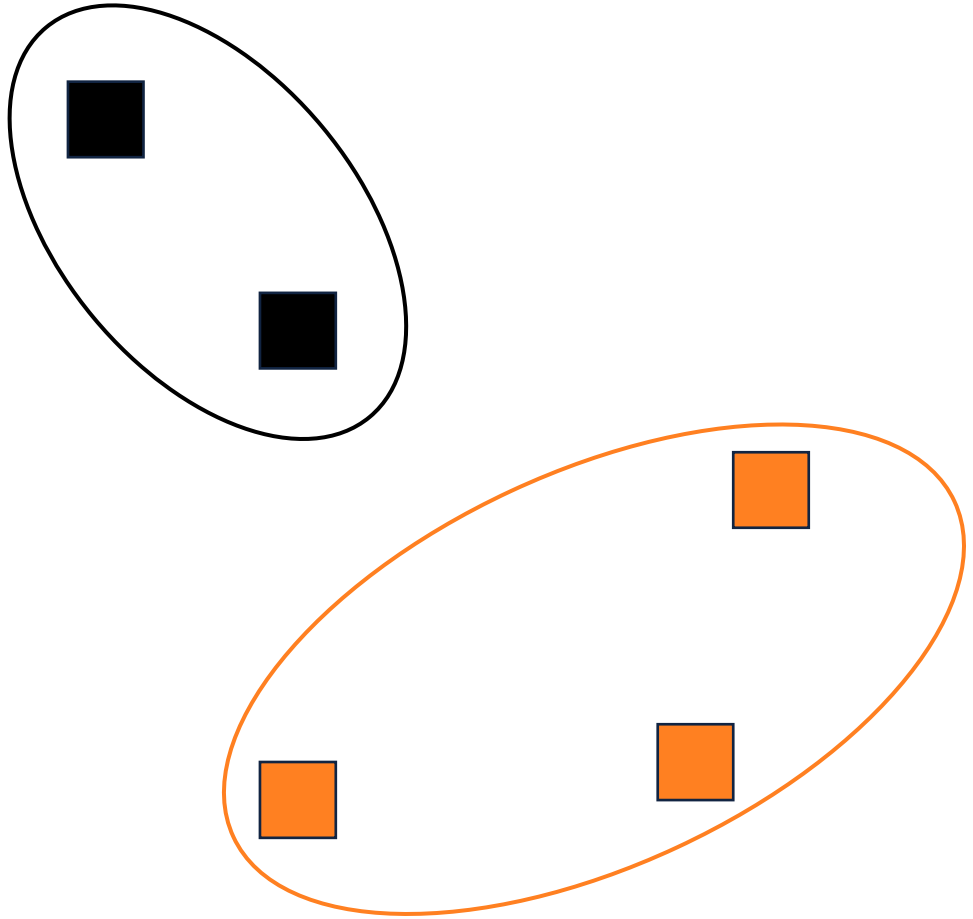


Hierarchical clustering - agglomerative

1. Assign a cluster to each sample

2. Combine the two closest clusters (choose your favorite distance method)

3. Repeat step 2.

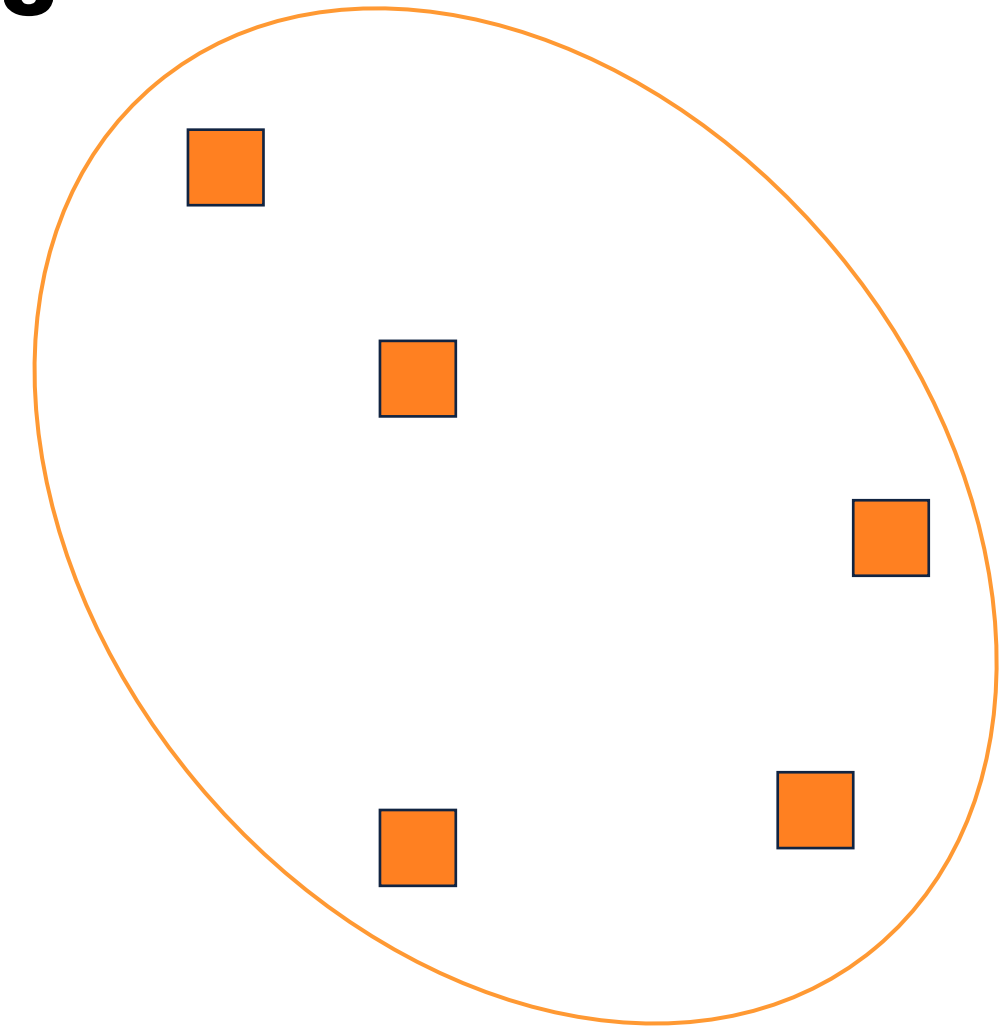


Hierarchical clustering - agglomerative

1. Assign a cluster to each sample

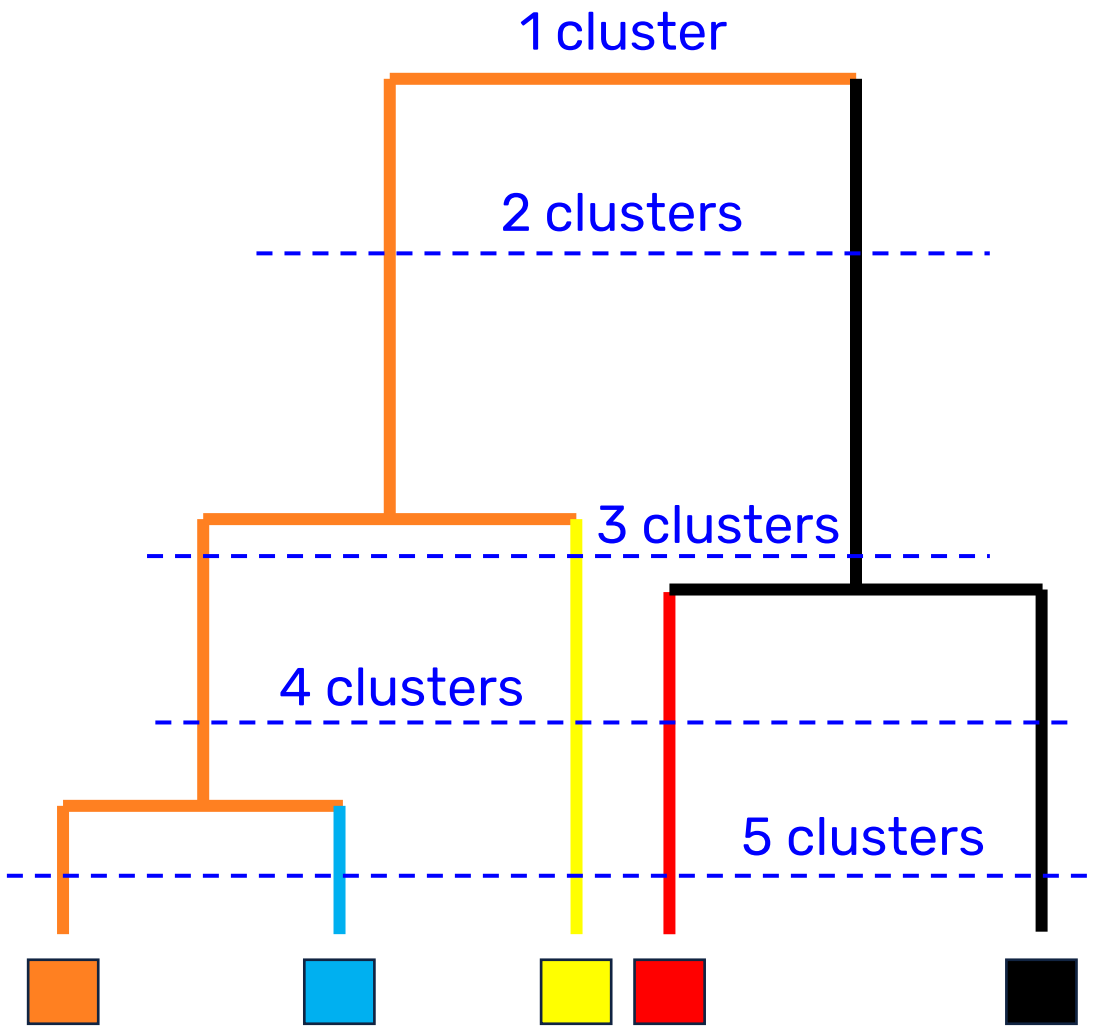
2. Combine the two closest clusters (choose your favorite distance method)

3. Repeat step 2.



Hierarchical clustering - agglomerative

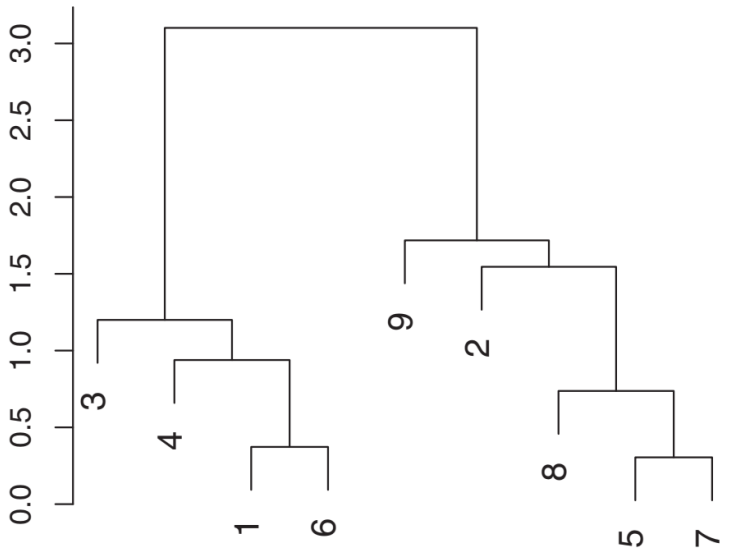
1. Assign a cluster to each sample
2. Combine the two closest clusters (choose your favorite distance method)
3. Repeat step 2.
4. Show the results using a dendogram

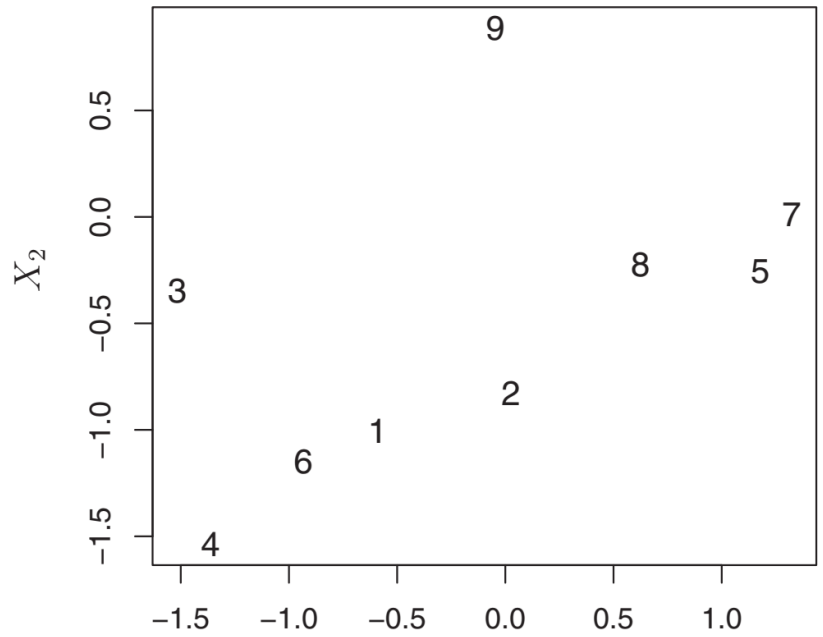


Hierarchical clustering - agglomerative

The distance metric used can lead to different dendograms

- Euclidean distance
- Correlation





Observations that fuse at the **very bottom** of the tree: **quite similar** to each other

Observations that fuse **close to the top** of the tree: **quite different** to each other

Example: marketing segmentation

Suppose you are in charge of a **marketing campaign** for a Telco company. You have to decide which offers to give to your customers, based on their **usage behaviour data**

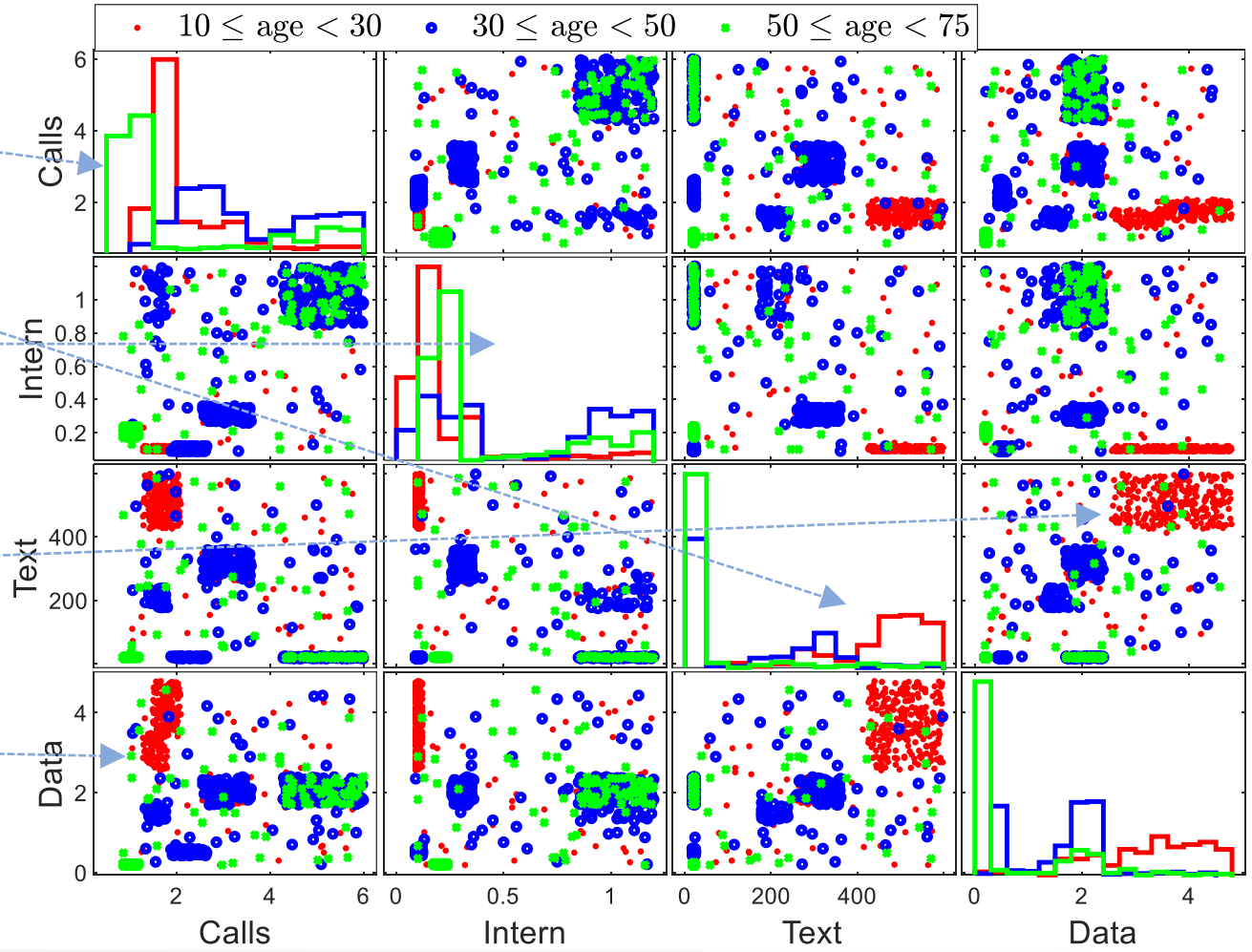
The dataset contains $N = 1000$ customers. The $d = 5$ variables for each client are:

- **call:** average number of call time hours per month
- **intern:** average number of international calls hours per month
- **text:** average number of text messages per month
- **data:** average number data used in GB per month
- **age:** age of the customer

Example: marketing segmentation

First try to have a look at the data. Plot each combination of 2 variables grouped by age

- Seasoned people call less
- Young people text more
- Young people don’t do international calls
- High data usage seems correlated to high text usage
- Youg people call less but use data more
- …



Example: marketing segmentation

Let's first apply hierarchical clustering, aiming at discovering **8 different groups**

Group	Mean calls	Mean intern	Mean text	Mean data	Mean age	Proportion
Young adults	1.706	0.10394	509.42	3.7312	18.953	25.4%
Silver	1.0323	0.20451	21.887	0.20897	60.826	19.5%
Pro	5.0681	1.0149	26.085	26.085	46.69	18.4%
30's	3.1366	0.30876	309.51	1.9767	30.18	16.1%
40's	2.2965	0.10391	21.402	0.52043	35.598	9.2%
40's	1.8552	0.97593	296.72	1.5111	37.185	5.4%
30's	4.9832	0.74324	343.83	2.1332	29.118	3.4%
50's	2.9569	0.56077	271.06	3.2804	53	2.6%

- **Group «Silver»:** low level of usage, but it is older
- **Group «Pro»:** lot of national anche international calls, few text messages
- **Group «Young adults»:** lot of text messages and data, young age

Example: marketing segmentation

Some clusters seem very similar. Redo the clustering using only **5 groups**

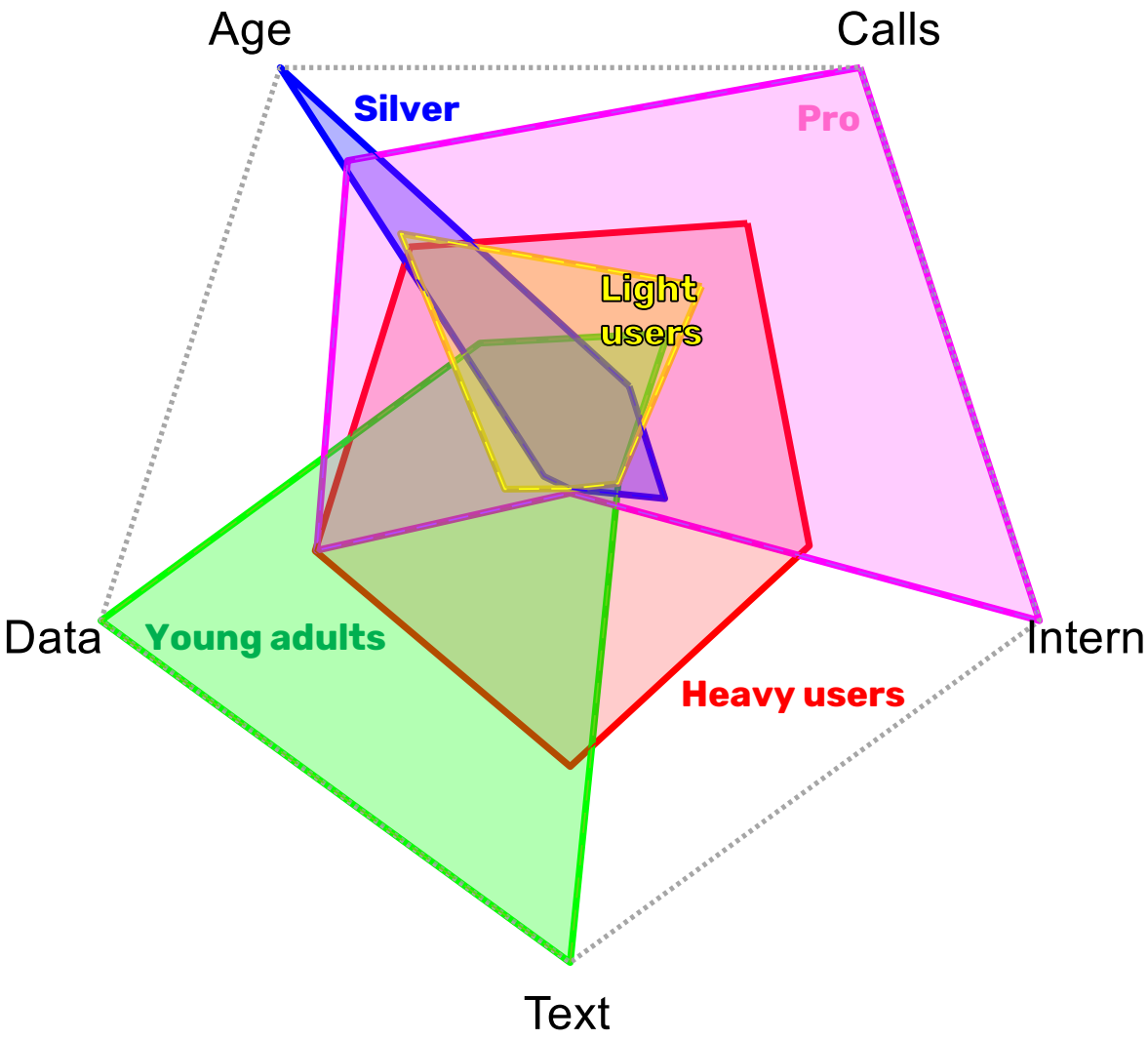
Group	Mean calls	Mean intern	Mean text	Mean data	Mean age	Proportion
Heavy users	3.0963	0.51731	307.61	2.0279	33.582	27.5%
Young adults	1.706	0.10394	509.42	3.7312	18.953	25.4%
Silver	1.0323	0.20451	21.887	0.20897	60.826	19.5%
Pro	5.0681	1.0149	26.085	2.0148	46.69	18.4%
Light users	2.2965	0.10391	21.402	0.52043	35.598	9.2%

- **Group «Heavy users»:** lot of calls, text and data
- **Group «Young adults»:** few calls, lot of text and date, low age
- **Group «Silver»:** low usage, seasoned
- **Group «Pro»:** lot of calls and data
- **Group «Light users»:** similar to the «silver» group, but younger

Example: marketing segmentation

We can compare the groups using a **radar plot**

- **Group «Heavy users»:** lot of calls, text and data
- **Group «Young adults»:** few calls, lot of text and date, low age
- **Group «Silver»:** low usage, seasoned
- **Group «Pro»:** lot of calls and data
- **Group «Light users»:** similar to the «silver» group, but younger



Outline

1. Introduction
2. K-means clustering
3. Hierarchial clustering
4. **Principal Component Analysis (PCA)**

Principal Component Analysis motivation

Principal Component Analysis (PCA) can be used for different purposes:

- **Data compression:** project the data to a lower dimensionality (from $\mathbb{R}^d$ to $\mathbb{R}^q$, $q < d$)
- **Data visualization:** visualize in a 2-D plot high-dimensional data

PCA produces a **low-dimensional representation** of a dataset:

- Finds **linear combinations** of the original features that have maximal variance
- The derived variables are **mutually uncorrelated**
- The derived variables can be used in supervised learning problems to **speed up** the computation

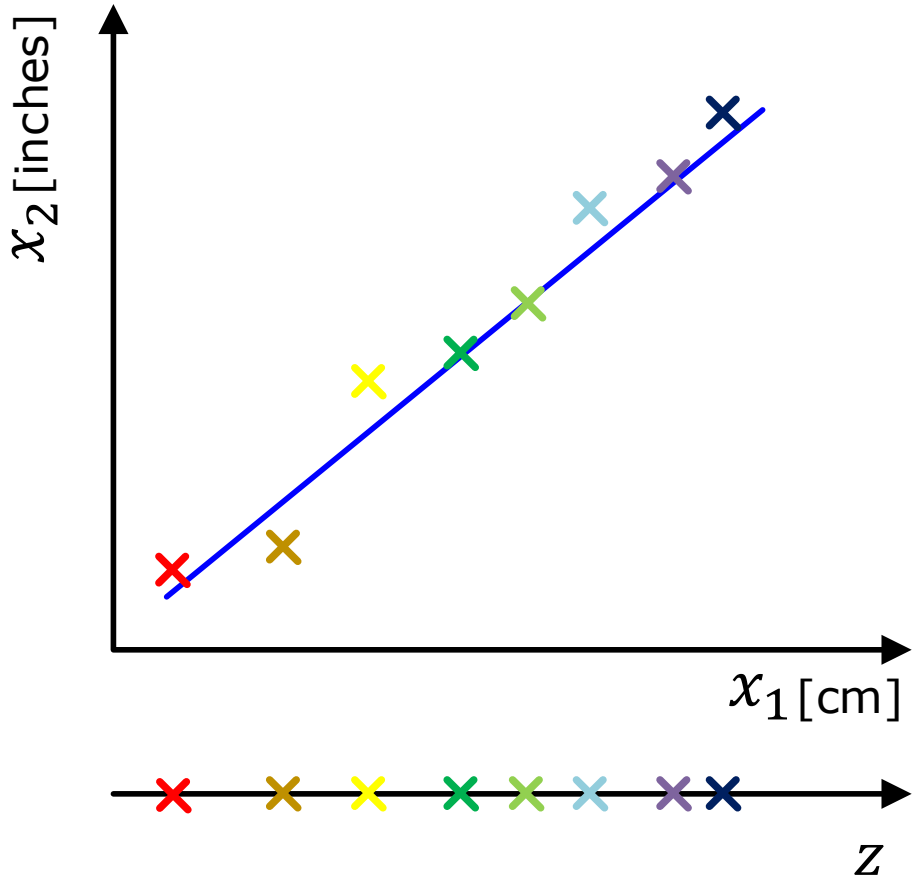
Data compression

Reduce the data dimensionality from $d = 2$ to $q = 1$

x_1 and x_2 are not perfectly correlated due to different measurement noise

- Highly **redundant** representation
- The compressed data retain the **main information**
- Approximation but **reduced space** requirement

$$\begin{aligned} \boldsymbol{x}(1) \in \mathbb{R}^2 &\rightarrow z(1) \in \mathbb{R} \\ \boldsymbol{x}(2) \in \mathbb{R}^2 &\rightarrow z(2) \in \mathbb{R} \\ &\vdots \\ \boldsymbol{x}(N) \in \mathbb{R}^2 &\rightarrow z(N) \in \mathbb{R} \end{aligned}$$



Data visualization

Suppose we want to explore the **`USarrest`** dataset[1], which contains crime statistics per 100.000 residents in 50 USA states. The variables are:

1. Murder arrests (per 100.000)
2. Assault arrests (per 100.000)
3. Percent urban population
4. Rape arrests (per 100.000)

State	Murder	Assault	Urban pop	Rape
Alabama	13.2	236	58	21.2
Alaska	10	260	48	44.5
Arizona	8.1	294	80	31
⋮	⋮	⋮	⋮	⋮

How to visualize a 4-dimensional dataset?

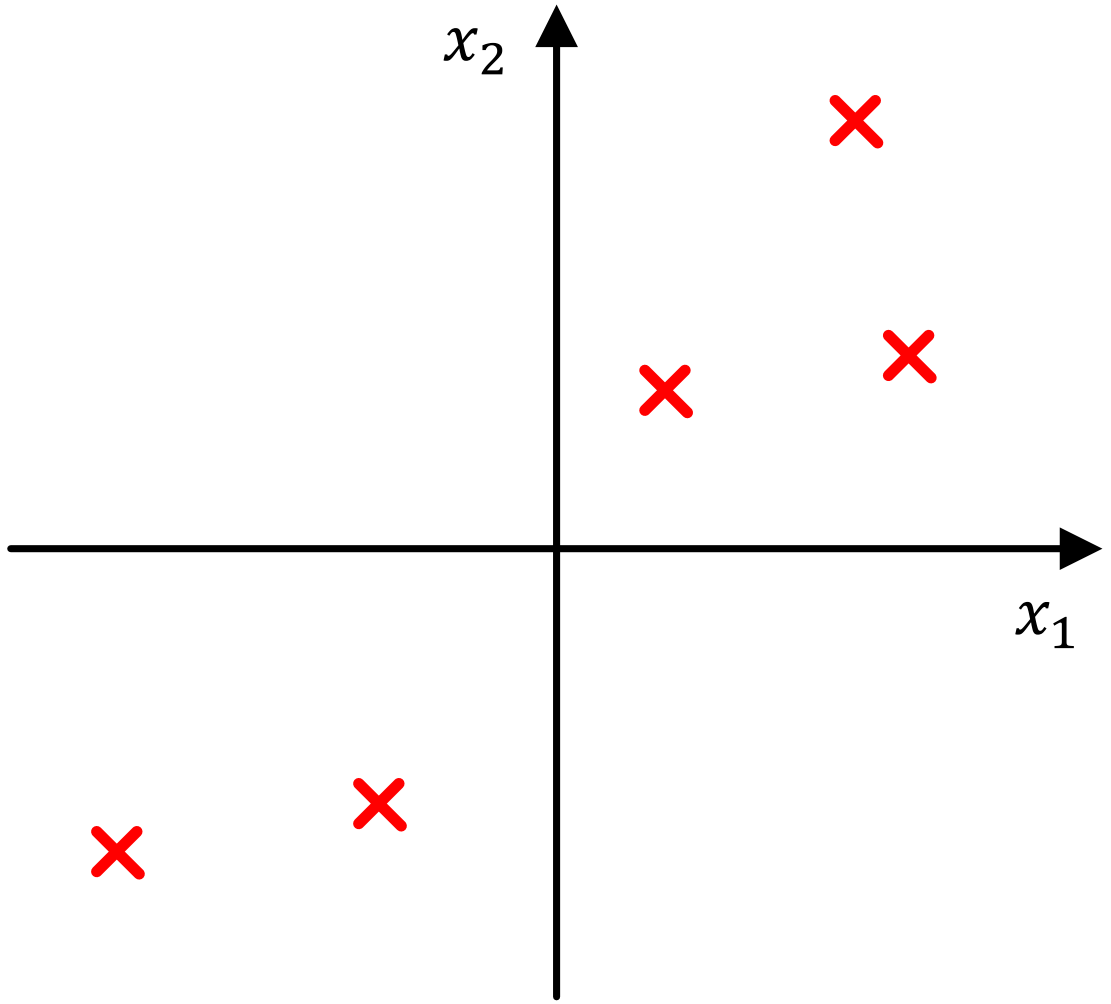
[1] https://forge.scilab.org/index.php/p/rdataset/source/tree/master/csv/datasets/USArrests.csv

Problem formulation

Onto which **direction** it is better to **project** the data?

Pick the direction which **minimizes the projection error**.

This direction is also that on which the data **vary the most**

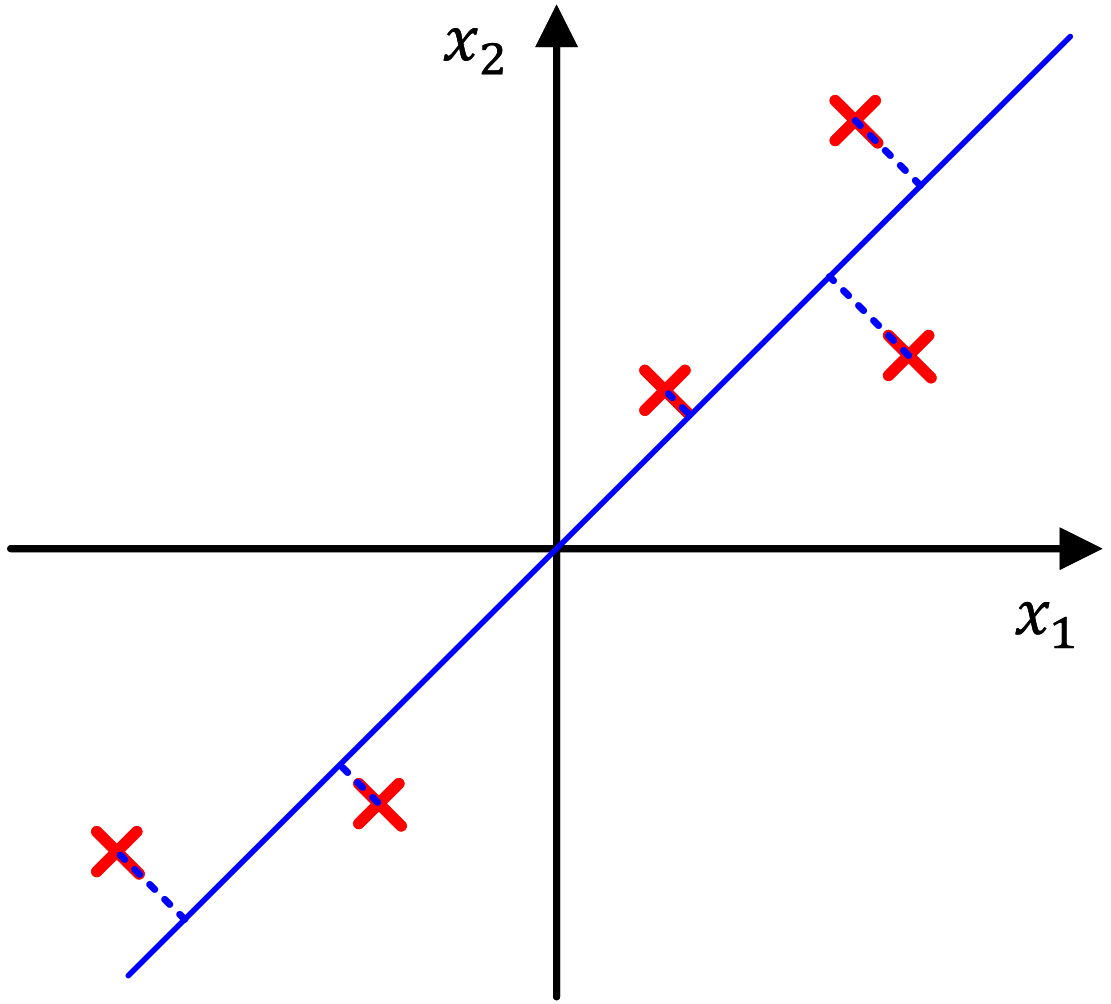


Problem formulation

Onto which **direction** it is better to **project** the data?

Pick the direction which **minimizes the projection error**.

This direction is also that on which the data **vary the most**

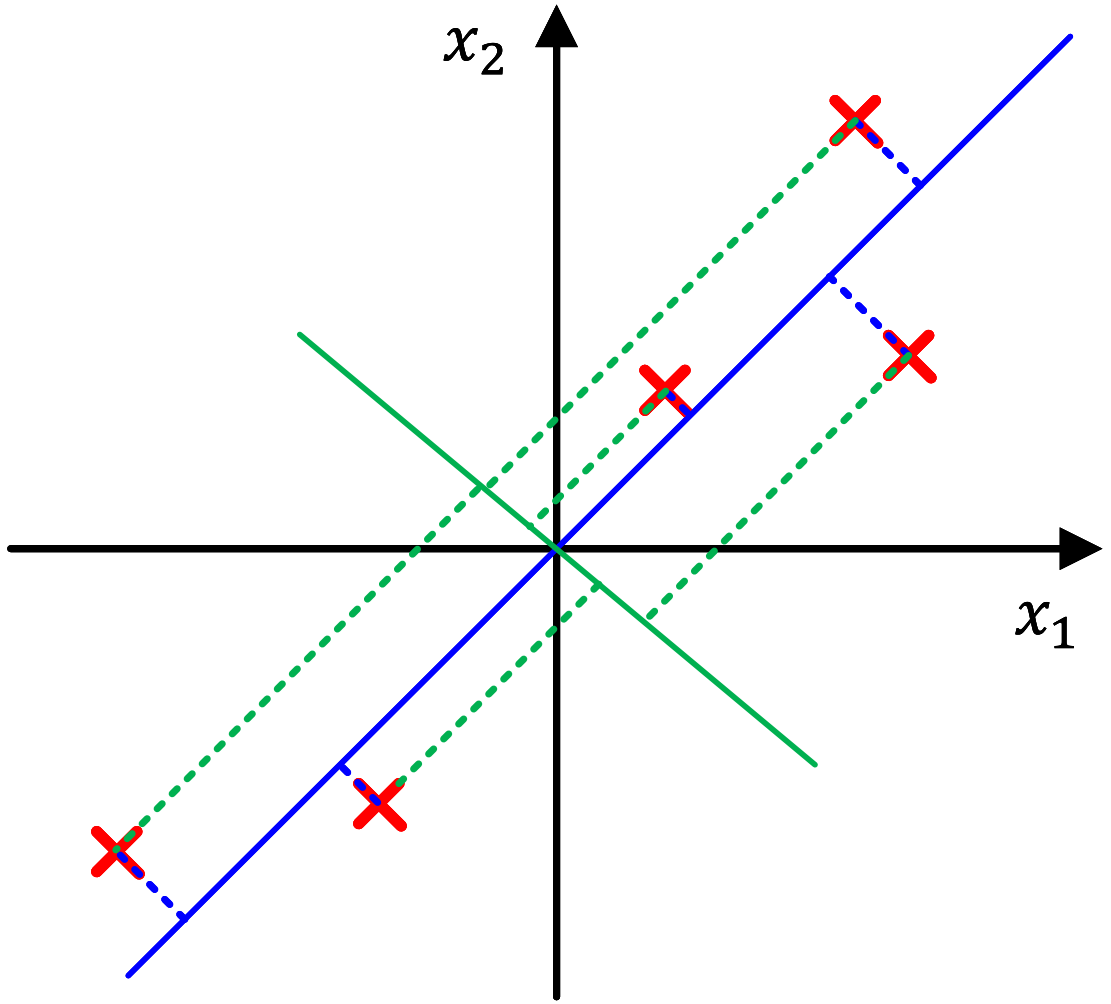


Problem formulation

Onto which **direction** it is better to **project** the data?

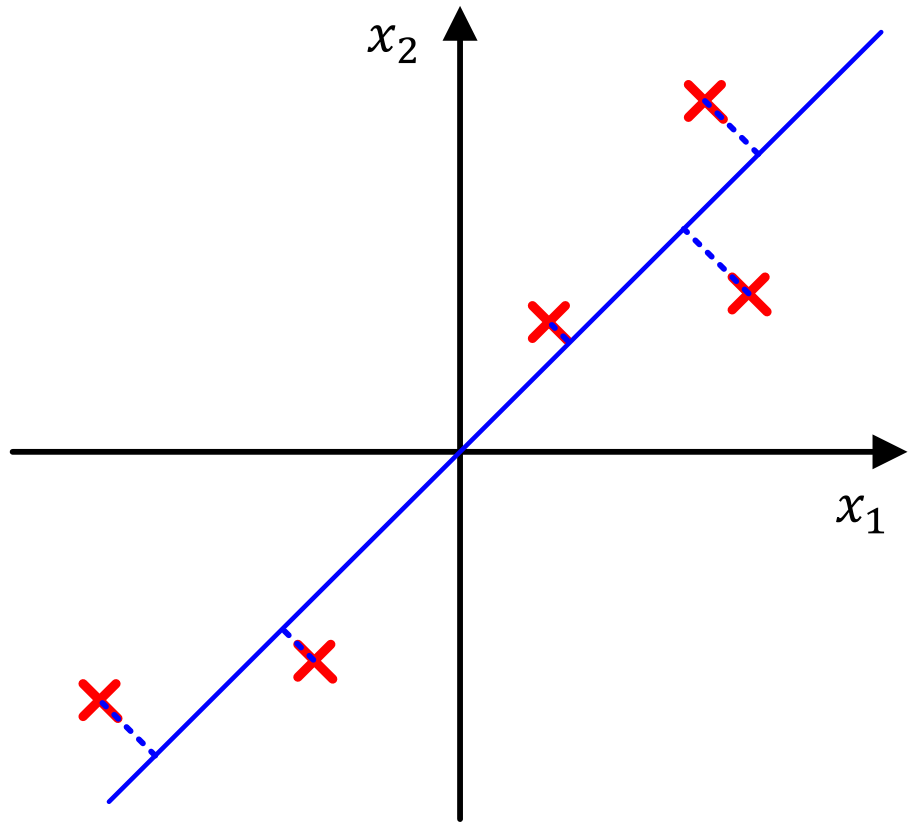
Pick the direction which **minimizes the projection error**.

This direction is also that on which the data **vary the most**

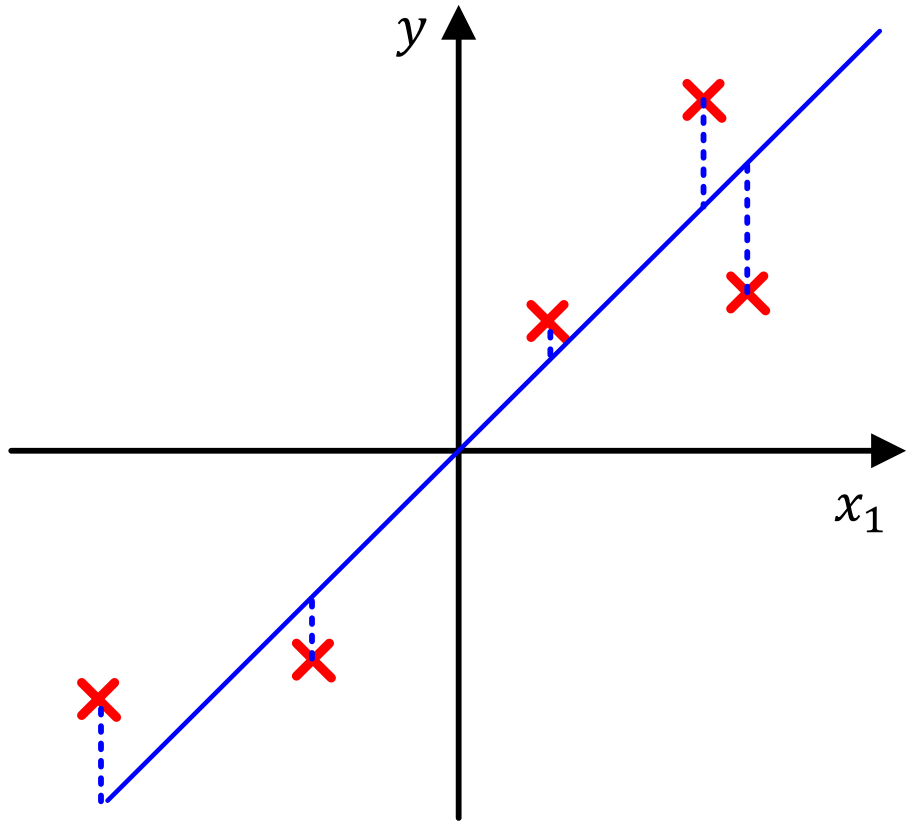


PCA is NOT Linear regression

PCA



Linear regression



PCA algorithm

Inputs

- q: the number of dimensions to retain (can be $q = d$)
- The training set $\mathcal{D} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_N\}$, $\boldsymbol{x} \in \mathbb{R}^d$ (do not consider dummy variables $\boldsymbol{x}_0 = 1$)

Data pre-processing

- Remove the feature (column) mean for each column of the data matrix $X \in \mathbb{R}^{N \times d}$
- Standardize each feature element for the respective feature standard deviation
- Each feature now has mean 0 and standard deviation 1
- Define the normalized data matrix as $\tilde{X} \in \mathbb{R}^{N \times d}$

PCA algorithm

Perform a **Singular Value Decomposition** (SVD) of the matrix $\tilde{X} \in \mathbb{R}^{N \times d}$

$$\tilde{X} = USV^T$$

$$\underset{N \times N}{U} = \begin{bmatrix} & \cdots & \\ \vdots & \ddots & \vdots \\ & \cdots & \end{bmatrix} \qquad \underset{N \times d}{S} = \begin{bmatrix} & \cdots & \\ \vdots & \ddots & \vdots \\ & \cdots & \end{bmatrix} \qquad \underset{d \times d}{V^T} = \begin{bmatrix} & \cdots & \\ \vdots & \ddots & \vdots \\ & \cdots & \end{bmatrix}$$

- The columns $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_N$ of U form an orthonormal basis of $\mathbb{R}^N$
- The columns $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_d$ of V form an orthonormal basis of $\mathbb{R}^d$
- The diagonal elements $s_1, s_2, \ldots, s_{\min\{N,d\}}$ in S are **nonnegative** and called **singular values** of $\tilde{X}$

PCA algorithm

From $\tilde{X} = USV^T$ we get $V = \begin{bmatrix} | & | & & | \\ \mathbf{v}_1 & \mathbf{v}_2 & \cdots & \mathbf{v}_d \\ | & | & & | \end{bmatrix} \in \mathbb{R}^{d \times d}$

- The columns $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_d$ of V are called **principal component loadings** and they are the **eigenvectors** of the covariance matrix $\Sigma = \frac{1}{N}\tilde{X}^T\tilde{X}$
- Select the first $q \leq d$ columns of V, obtaining the **reduced** matrix $V_q \in \mathbb{R}^{d \times q}$
- Compute the projected data $Z = \tilde{X}V_q \in \mathbb{R}^{N \times q}$ **(principal component scores)**
- It is possible to recover the original data, up to an **approximation**, by computing $\tilde{X}_r = Z \cdot V_q^T \in \mathbb{R}^{N \times d}$

Choice of the number of the components

The singular values $s_1, s_2, \dots, s_{\min\{N,d\}}$ in the matrix S can be used to compute the data **variance explained** by each principal component

The percentage of variance explained by the j-th component is:

$$e_j = \frac{s_j^2}{\sum_{i=1}^{\min\{N,d\}} s_i^2} \cdot 100$$

Choices:

- Retain a number of components that explain a **determined level** of variance in the data
- Retain a **fixed number** of components

Example with the USarrest data

The first 2 principal component loadings for the USarrest dataset are the first 2 columns of the matrix $V \in \mathbb{R}^{4\times4}$. Thus, $V_q \in \mathbb{R}^{4\times2}$

State	Murder	Assault	Urban pop	Rape
Alabama	13.2	236	58	21.2
Alaska	10	260	48	44.5
Arizona	8.1	294	80	31
⋮	⋮	⋮	⋮	⋮

PCA

Feature	PC 1	PC 2
Murder	−0.5359	0.4182
Assault	−0.5832	0.1880
UrbanPop	−0.2782	−0.8728
Rape	−0.5432	−0.1673

- It is possible to plot both the **projected data** and the **new dimensions**
- For example, the direction of the **Murder feature** is the direction of the line from the origin to the point $[-0.5359,\ \ 0.4182]$

Biplot

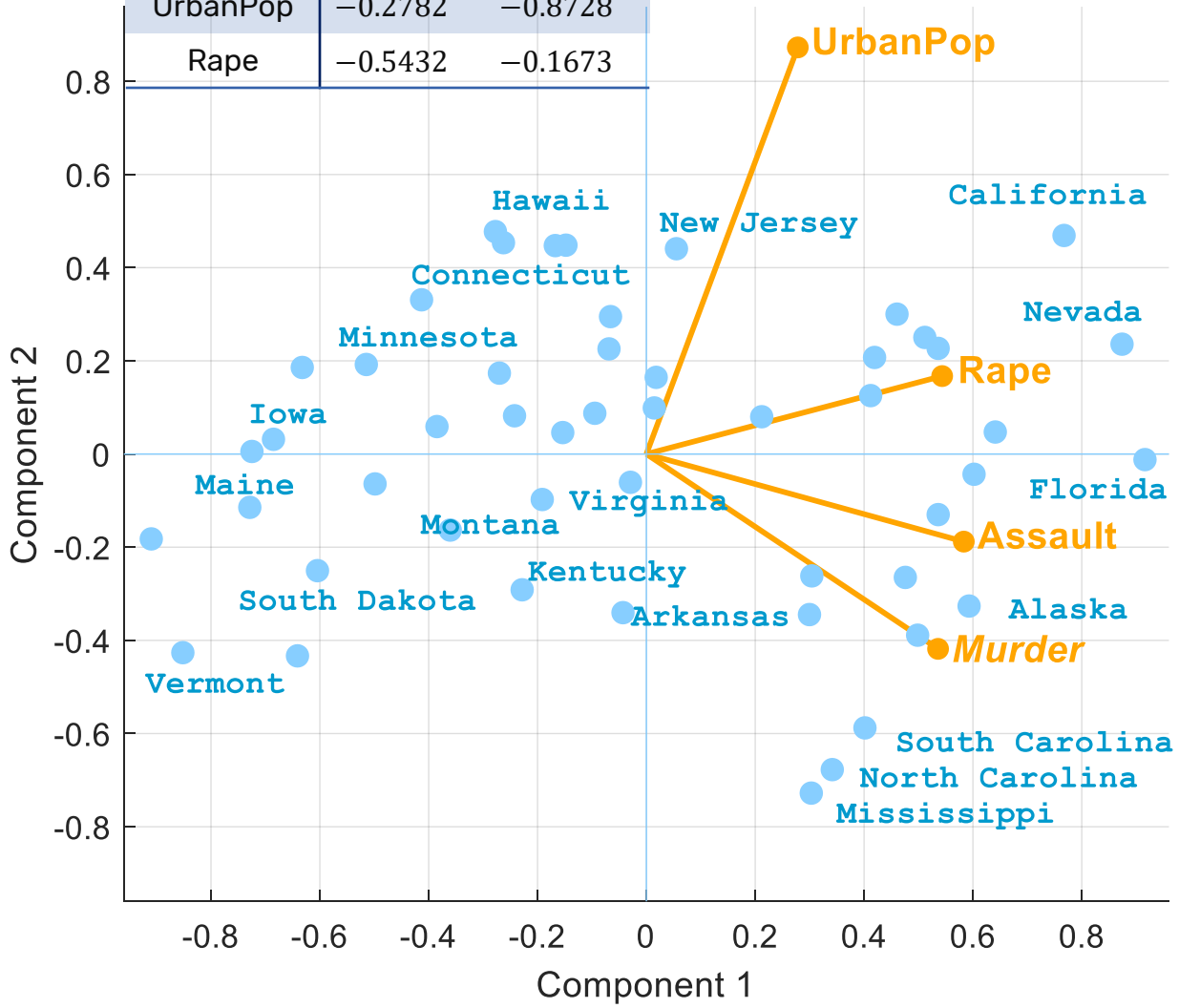
The 1st loading vector gives similar weight on **Assault**, **Murder**, and **Rape**, and less weight on **UrbanPop**

- The 1st component roughly corresponds to a **measure of overall rates of serious crimes**

The 2nd loading vector vector places most of its weight on **UrbanPop** and much less weight on the other three features

- The 2nd component roughly corresponds to **the level of urbanization of the state**

Feature	PC 1	PC 2
Murder	−0.5359	0.4182
Assault	−0.5832	0.1880
UrbanPop	−0.2782	−0.8728
Rape	−0.5432	−0.1673

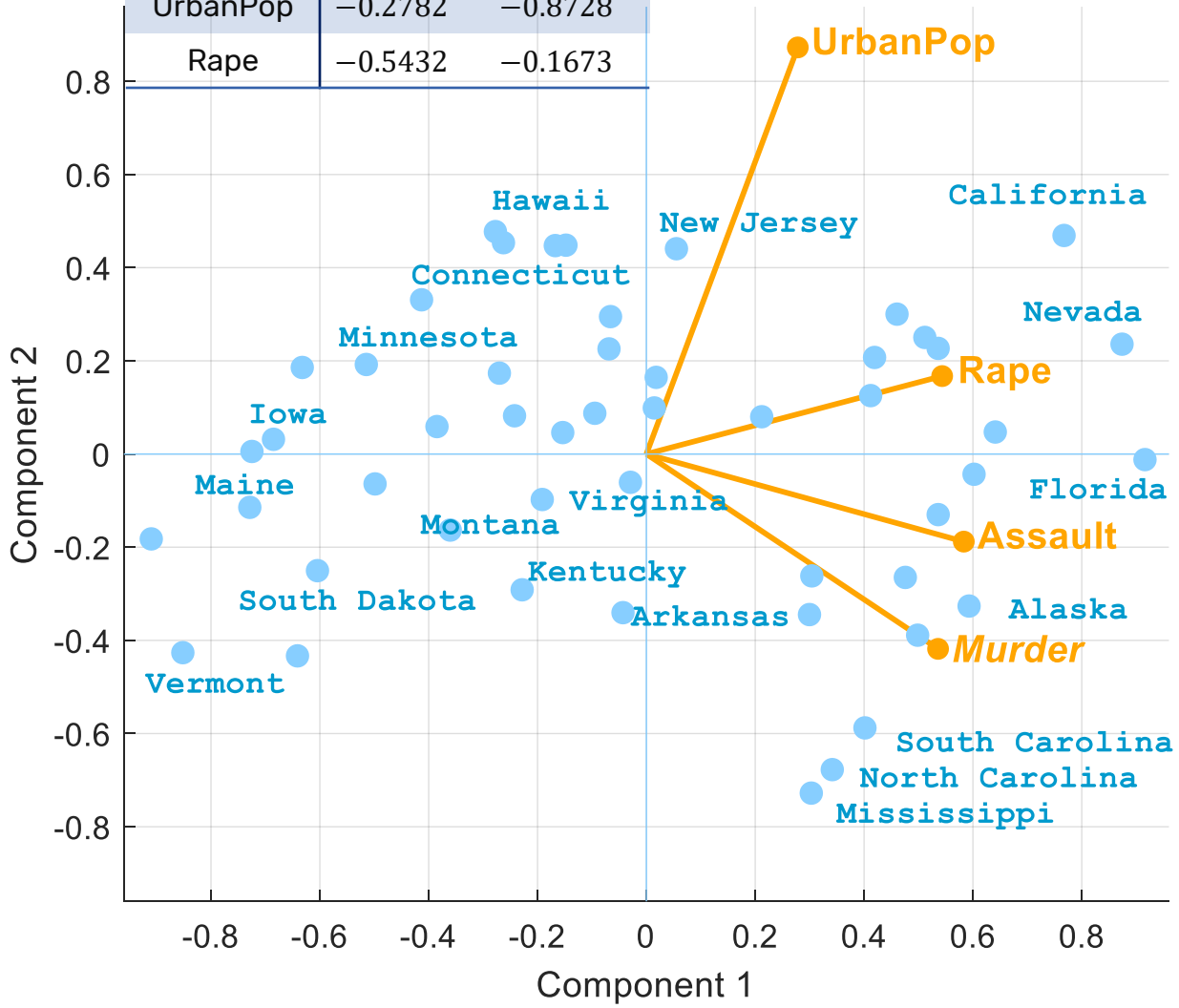


Biplot

The **crime-related variables** (**Murder**, **Assault**, and **Rape**) are located close to each other, and that the **UrbanPop** variable is far from the other three

- The crime variables are correlated with each other (states with high murder rates tend to have high assault and rape rates)
- The UrbanPop variable is less correlated with the other three.

Feature	PC 1	PC 2
Murder	−0.5359	0.4182
Assault	−0.5832	0.1880
UrbanPop	−0.2782	−0.8728
Rape	−0.5432	−0.1673



Biplot

We can observe that:

- states with **large positive scores on the 1st component**, such as California, Nevada and Florida, have **high crime rates**, while states like South Dakota, with negative scores on the first component, have low crime rates.
- California also has a **high score on the 2nd component**, indicating a **high level of urbanization**, while the opposite is true for states like Mississippi.
- States **close to zero** on both components, such as Virginia, have approximately **average levels** of both crime and urbanization.

Feature	PC 1	PC 2
Murder	−0.5359	0.4182
Assault	−0.5832	0.1880
UrbanPop	−0.2782	−0.8728
Rape	−0.5432	−0.1673