*** Applied Machine Learning Fundamentals ***
Principal Component Analysis

Daniel Wehner, M.Sc.

SAP SE / DHBW Mannheim

Winter term 2023/2024

Find all slides on GitHub (DaWe1992/Applied_ML_Fundamentals)

Lecture Overview

Agenda for this Unit

Section:
Introduction

Why Dimensionality Reduction?
Data Compression
Data Visualization
What is PCA?

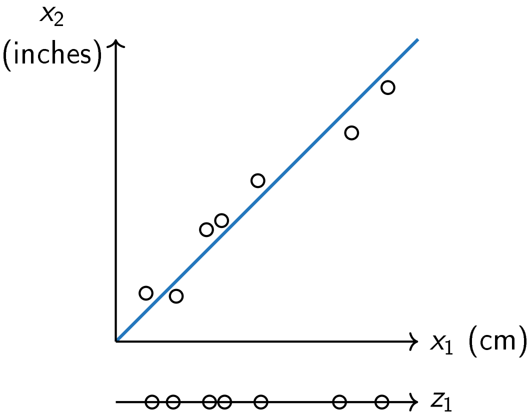
Why Dimensionality Reduction?

- Most data is high-dimensional
- Dimensionality reduction can be used for:
  - **Lossy (!)** data compression
  - Feature extraction
  - Data visualization

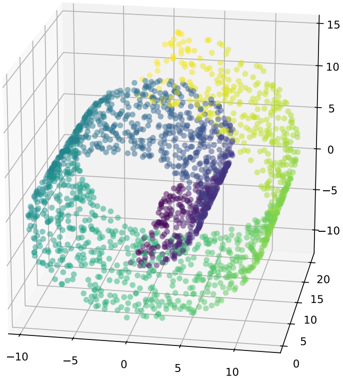
Dimensionality reduction can help to **speed up** learning algorithms substantially. Too many (correlated) features usually **decrease the performance** of the learning algorithm (**curse of dimensionality**).

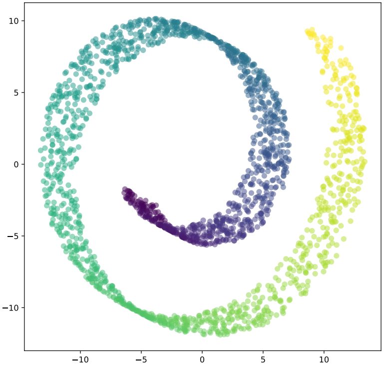
Use Case I: Data Compression / Feature Extraction

- The features *inches* and *cm* are closely related
- **Problems**:
  - Redundancy
  - More memory needed
  - Algorithms become slow
- **Solution**: Convert x_1 and x_2 into a new feature z_1 ($\mathbb{R}^2 \to \mathbb{R}$)



Use Case II: Data Visualization





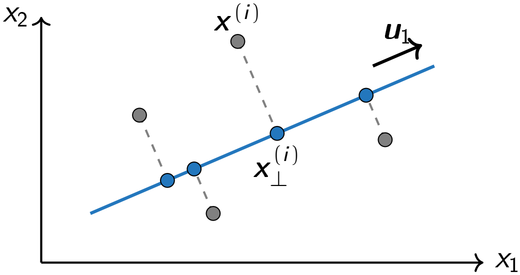
PCA: Principal Component Analysis

- PCA is an **unsupervised** algorithm
- PCA can be defined as the **orthogonal projection** of the data onto a lower dimensional **linear space** (*principal subspace*)
- Consider a dataset of n observations $\boldsymbol{X} = \{\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \dots, \boldsymbol{x}^{(n)}\}$
  - $\boldsymbol{x}^{(i)} \in \mathbb{R}^m$ $(1 \leqslant i \leqslant n)$ is an m-dimensional feature vector
  - We want to project the data onto a space having dimensionality $k \ll m$, while **maximizing the variance of the projected data** $(\mathbb{R}^m \to \mathbb{R}^k)$

Remove dimensions which are the least informative of the data!

Important

Orthogonal Projections (Case: $\mathbb{R}^2 \to \mathbb{R}$)

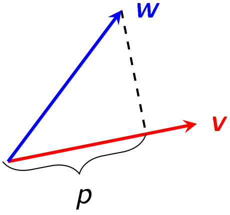


- $\boldsymbol{x}^{(i)}$ denotes the original data point
- $\boldsymbol{x}_{\perp}^{(i)}$ is the orthogonal projection of $\boldsymbol{x}^{(i)}$ onto the vector $\boldsymbol{u}_1$

The goal is to find a suitable vector $\boldsymbol{u}_1$ so that the variance of the projection is maximized!

Recall: Projection of Vectors

- Let $\boldsymbol{w}, \boldsymbol{v} \in \mathbb{R}^2$ be two vectors
- How is the projection of $\boldsymbol{w}$ onto $\boldsymbol{v}$ defined?



$$p = \|\boldsymbol{w}\| \cos \measuredangle(\boldsymbol{v}, \boldsymbol{w})$$

$$= \|\boldsymbol{w}\| \frac{\boldsymbol{v}^\intercal \boldsymbol{w}}{\|\boldsymbol{v}\| \cdot \|\boldsymbol{w}\|} = \frac{\boldsymbol{v}^\intercal \boldsymbol{w}}{\|\boldsymbol{v}\|}$$

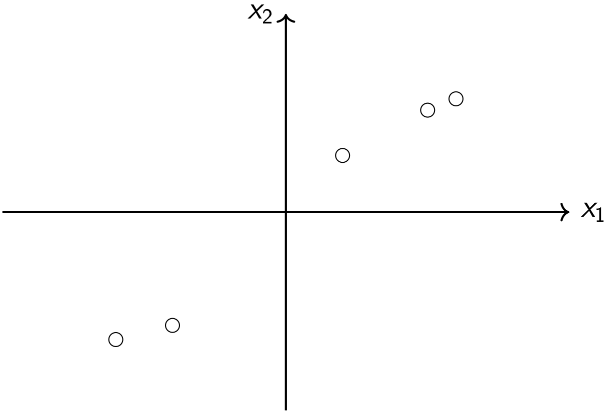
- We will assume $\boldsymbol{u}_1$ to be a unit vector, i. e. $\|\boldsymbol{u}_1\| = 1$
- $\frac{\boldsymbol{u}_1^\intercal \boldsymbol{x}^{(i)}}{\|\boldsymbol{u}_1\|}$ then reduces to the scalar product $\boldsymbol{u}_1^\intercal \boldsymbol{x}^{(i)}$

Section:

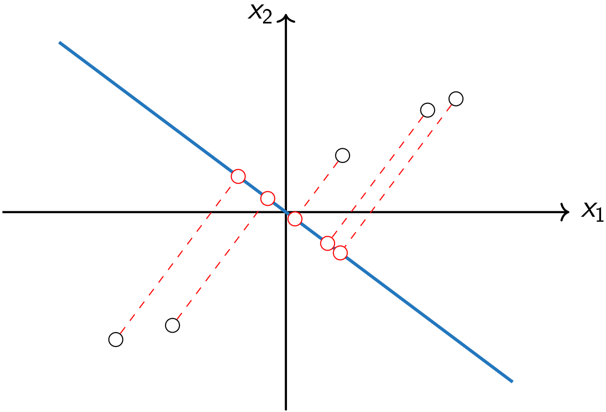
Maximum Variance Formulation

An Example
Formalization of the Problem

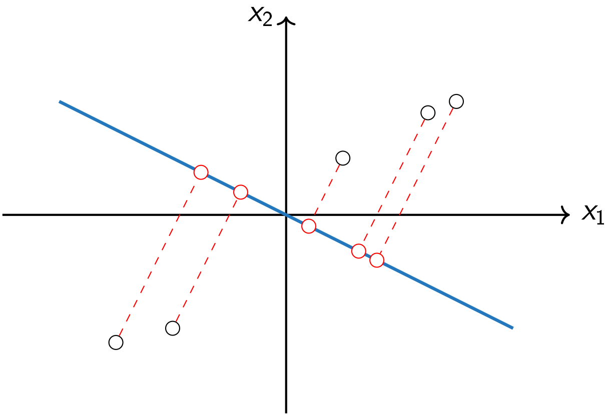
Maximum Variance Formulation



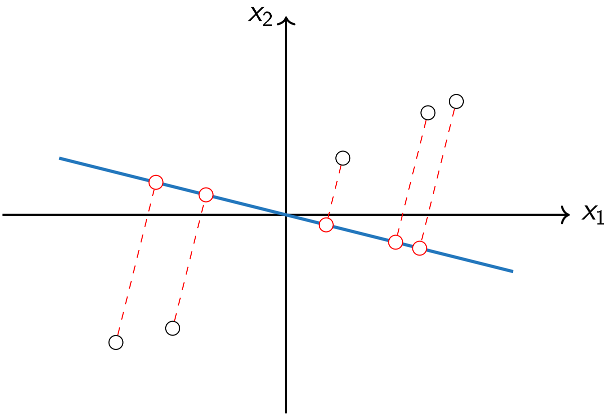
Maximum Variance Formulation



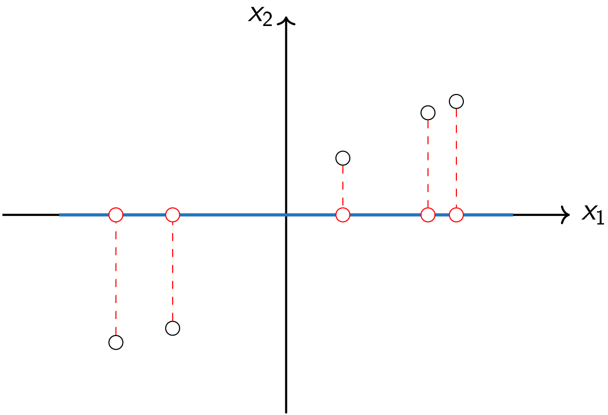
Maximum Variance Formulation



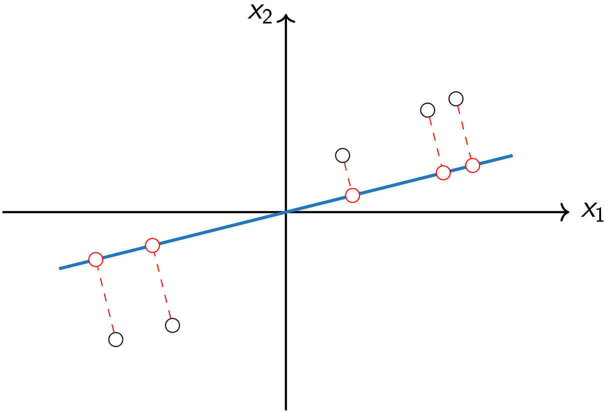
Maximum Variance Formulation



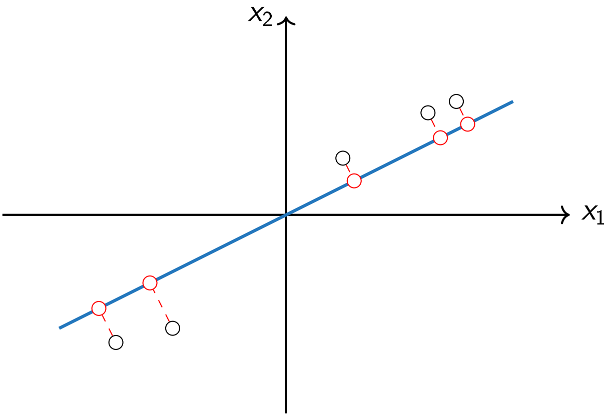
Maximum Variance Formulation



Maximum Variance Formulation



Maximum Variance Formulation



Maximum Variance Formulation (Ctd.)

- In the following we assume $k = 1$
  (i. e. we project the data onto a line defined by a unit vector $\boldsymbol{u}_1$)
- Each data point $\boldsymbol{x}^{(i)} \in \mathbb{R}^m$ is projected onto a scalar value $\boldsymbol{u}_1^\intercal \boldsymbol{x}^{(i)} \in \mathbb{R}$
- The mean of the projected data is $\boldsymbol{u}_1^\intercal \overline{\boldsymbol{x}}$, where $\overline{\boldsymbol{x}} := \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{x}^{(i)}$
- The variance of the projected data is given by:

$$\frac{1}{n} \sum_{i=1}^{n} \left(\boldsymbol{u}_1^\intercal \boldsymbol{x}^{(i)} - \boldsymbol{u}_1^\intercal \overline{\boldsymbol{x}} \right)^2 = \boldsymbol{u}_1^\intercal \boldsymbol{\Sigma} \boldsymbol{u}_1 \tag{1}$$

Maximum Variance Formulation (Ctd.)

- $\boldsymbol{\Sigma}$ is the covariance matrix defined by:

$$\boldsymbol{\Sigma} := \frac{1}{n} \sum_{i=1}^{n} \overbrace{(\boldsymbol{x}^{(i)} - \overline{\boldsymbol{x}})(\boldsymbol{x}^{(i)} - \overline{\boldsymbol{x}})^{\intercal}}^{\textbf{Outer product} \rightarrow \textbf{matrix}} \tag{2}$$

- We have to maximize the projected variance $\boldsymbol{u}_1^{\intercal} \boldsymbol{\Sigma} \boldsymbol{u}_1$ with respect to $\boldsymbol{u}_1$
- **Constraint:** $\|\boldsymbol{u}_1\| = 1$, otherwise $\boldsymbol{u}_1$ grows unboundedly
- We have to solve the following (Lagrangian) optimization problem:

$$\max_{\boldsymbol{u}_1} \{ \boldsymbol{u}_1^{\intercal} \boldsymbol{\Sigma} \boldsymbol{u}_1 + \lambda_1 (1 - \boldsymbol{u}_1^{\intercal} \boldsymbol{u}_1) \} \tag{3}$$

Maximum Variance Formulation (Ctd.)

- We have to solve

$$\nabla_{\boldsymbol{u}_1}\left\{\boldsymbol{u}_1^\intercal \boldsymbol{\Sigma} \boldsymbol{u}_1 + \lambda_1(1 - \boldsymbol{u}_1^\intercal \boldsymbol{u}_1)\right\} \overset{!}{=} 0$$

- This leads to the **eigenvalue problem** $\boldsymbol{\Sigma}\boldsymbol{u}_1 = \lambda_1 \boldsymbol{u}_1$
- The equation tells us that $\boldsymbol{u}_1$ must be an eigenvector of $\boldsymbol{\Sigma}$
- If we left-multiply by $\boldsymbol{u}_1^\intercal$ and use $\boldsymbol{u}_1^\intercal \boldsymbol{u}_1 = 1$, we see: $\boldsymbol{u}_1^\intercal \boldsymbol{\Sigma} \boldsymbol{u}_1 = \lambda_1$

The variance reaches a maximum, if we set u_1 equal to the eigenvector having the largest eigenvalue λ_1. This eigenvector is the first principal component and its eigenvalue λ_1 is the variance retained by it.

Maximum Variance Formulation (Ctd.)

- Additional principal components can be defined in an **incremental fashion**
- Choose each new component such that it **maximizes the remaining projected variance**
- All principal components are **orthogonal to each other**
- Projection onto k dimensions:
  - The lower-dimensional space is defined by the k eigenvectors $\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_k$ of the covariance matrix $\boldsymbol{\Sigma}$
  - These correspond to the k largest eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_k$

Section:

PCA Algorithm

PCA Algorithm

Algorithm 1: PCA Algorithm

Input: Input data $\boldsymbol{X} = \left(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(n)}\right) \in \mathbb{R}^{n \times m}$, number of dimensions k

Output: Projected data $\boldsymbol{Z} \in \mathbb{R}^{n \times k}$

Compute $\bar{\boldsymbol{x}} \longleftarrow \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{x}^{(i)}$ `// sample set mean`

Compute $\boldsymbol{\Sigma} \longleftarrow \frac{1}{n} \sum_{i=1}^{n} (\boldsymbol{x}^{(i)} - \bar{\boldsymbol{x}})(\boldsymbol{x}^{(i)} - \bar{\boldsymbol{x}})^{\intercal}$ `// covariance matrix`

Perform **s**ingular **v**alue **d**ecomposition (SVD) for $\boldsymbol{\Sigma}$: $(\boldsymbol{U}, \boldsymbol{S}, \boldsymbol{V}) = SVD(\boldsymbol{\Sigma})$

Select the k eigenvectors with the largest eigenvalues: $\boldsymbol{U}_k \longleftarrow \boldsymbol{U}_{(:,:k)}$

Project the data: $\boldsymbol{Z} \longleftarrow \boldsymbol{U}_k^{\intercal} \boldsymbol{X}$

Projection of the Data

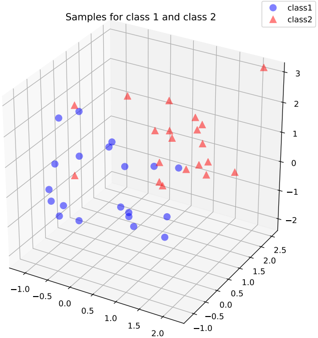
- Matrix $\boldsymbol{U}$ is obtained by applying **singular value decomposition** to $\boldsymbol{\Sigma}$

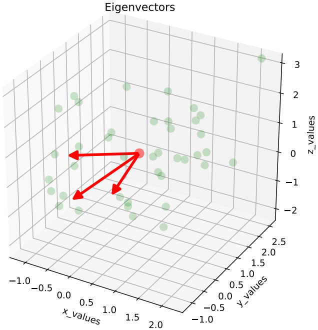
$$\boldsymbol{U} = \begin{pmatrix} | & | & & | \\ \boldsymbol{u}_1 & \boldsymbol{u}_2 & \dots & \boldsymbol{u}_m \\ | & | & & | \end{pmatrix} \in \mathbb{R}^{m \times m} \tag{4}$$

- The projection $\mathbb{R}^m \rightarrow \mathbb{R}^k (k \ll m)$ is performed as follows:

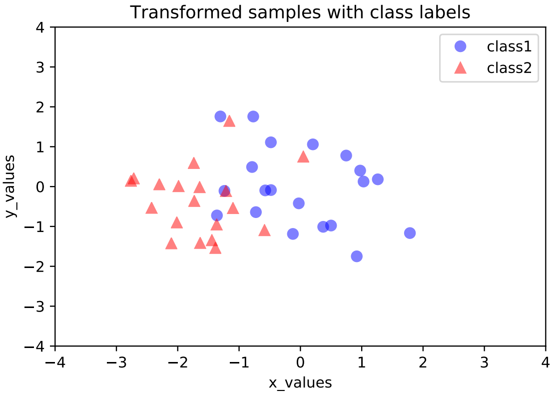
$$\begin{pmatrix} z_1^{(i)} \\ \vdots \\ z_k^{(i)} \end{pmatrix} = \begin{pmatrix} | & | & & | \\ \boldsymbol{u}_1 & \boldsymbol{u}_2 & \dots & \boldsymbol{u}_k \\ | & | & & | \end{pmatrix}^{\intercal} \begin{pmatrix} x_1^{(i)} \\ \vdots \\ x_m^{(i)} \end{pmatrix} \tag{5}$$

PCA Result


Samples for class 1 and class 2
class1
class2


Eigenvectors
x_values
y_values
z_values

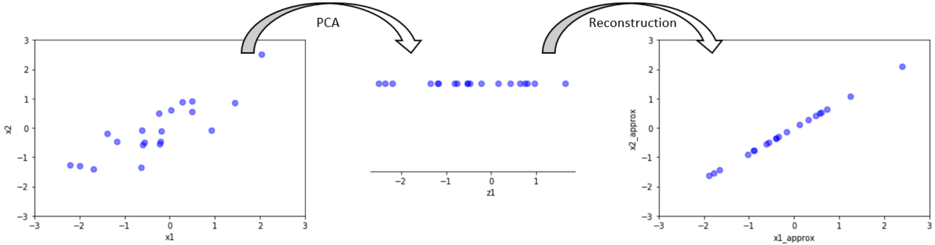
PCA Result (Ctd.)



Reconstruction from compressed Representation

It is possible to compute an **approximate reconstruction** of the data after having applied PCA ($\mathbb{R}^k \rightarrow \mathbb{R}^m$):

$$\boldsymbol{x}_{\approx}^{(i)} = \boldsymbol{U}_k \boldsymbol{z}^{(i)} \tag{6}$$



Choosing the Number of Components

- The goal is to preserve as much variance as possible
- Minimize the **average projection error** given by:

$$\frac{1}{n}\sum_{i=1}^{n}\|\boldsymbol{x}^{(i)} - \boldsymbol{x}_{\approx}^{(i)}\|^2 \tag{7}$$

- **Total variation** in the data is computed as follows:

$$\frac{1}{n}\sum_{i=1}^{n}\|\boldsymbol{x}^{(i)}\|^2 \tag{8}$$

Choosing the Number of Components (Ctd.)

- Typically, k is chosen to be the smallest value such that:

$$\frac{\overbrace{1/n \sum_{i=1}^{n} \|\boldsymbol{x}^{(i)} - \boldsymbol{x}_{\approx}^{(i)}\|^2}^{\text{average projection error}}}{\underbrace{1/n \sum_{i=1}^{n} \|\boldsymbol{x}^{(i)}\|^2}_{\text{total variation}}} \leqslant \gamma \qquad \gamma \in [0, 1] \tag{9}$$

- This means that $(1 - \gamma) \cdot 100\,\%$ of the variance is retained

You can be more efficient...

- The above algorithm is computationally very expensive
- The same result can be computed much more efficient
- Remember: $(\boldsymbol{U}, \boldsymbol{S}, \boldsymbol{V}) = SVD(\boldsymbol{\Sigma})$
- We can use the $(m \times m)$-matrix $\boldsymbol{S}$
- $\boldsymbol{S}$ is a diagonal matrix containing the eigenvalues on its main diagonal:

$$\boldsymbol{S} = \begin{pmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_m \end{pmatrix}$$

You can be more efficient... (Ctd.)

- Sort the eigenvalues in descending order: $(\lambda_1^\star, \lambda_2^\star, \dots, \lambda_m^\star)$
- $\lambda_1^\star$ is the largest eigenvalue, $\lambda_2^\star$ the second-largest, etc.
- Choose the smallest k which satisfies the inequality:

$$\frac{\sum_{j=1}^{k} \lambda_j^\star}{\sum_{j=1}^{m} \lambda_j^\star} \geqslant \gamma \qquad \gamma \in [0, 1] \tag{10}$$

- γ specifies the fraction of variance to be retained overall
- **Advantage:** The matrix $\boldsymbol{S}$ has to be computed only once and can be reused for all k

Section:
PCA Applications

Eigenfaces
Face Morphing

Application of PCA to Images: Eigenfaces

Figure: 100 images of faces

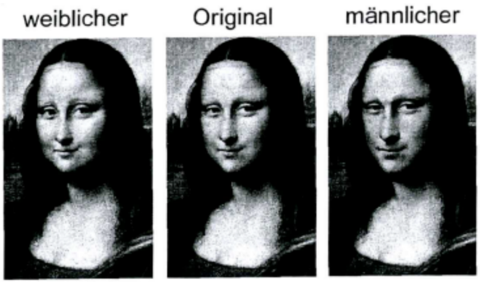
Figure: First 36 principal components

Application of PCA to Images: Eigenfaces (Ctd.)

Figure: Original images

Figure: Reconstructed images

Application of PCA to Images: Face Morphing



Section:

Wrap-Up

Summary
Self-Test Questions
Lecture Outlook

Summary

- Dimensionality reduction is important to avoid the **curse of dimensionality** ☠...
- ...or simply to **visualize high-dimensional data**
- It is defined as the **orthogonal projection** of the data onto a lower-dimensional (linear) space
- We want to **keep the dimensions with the most variance**
- These dimensions are called **principal components**
- Lots of applications: Eigenfaces, Morphing, ...

Self-Test Questions

1. How can PCA be defined?
2. What is the geometric relationship between the principal components?
3. Outline the PCA algorithm!
4. How can you recover the original data? Will you get the exact same data?
5. Explain how the number of components / dimensions can be chosen!
6. Name some use cases of PCA!

What's next...?

☠ **The Exam** ☠

Just kidding... (maybe)

Thank you very much for the attention!

Topic: *** Applied Machine Learning Fundamentals *** Principal Component Analysis
Term: Winter term 2023/2024

Contact:
Daniel Wehner, M.Sc.
SAP SE / DHBW Mannheim
daniel.wehner@sap.com

Do you have any questions?