

# Principal Component Analysis

Daniel Wehner, M.Sc.

SAP SE / DHBW Mannheim

Summer term 2025



Find all slides on [GitHub](#) (DaWe1992/Applied\_ML\_Fundamentals)

# Lecture Overview

- |             |                                   |               |                              |
|-------------|-----------------------------------|---------------|------------------------------|
| <b>I</b>    | Machine Learning Introduction     | <b>IX</b>     | Evaluation                   |
| <b>II</b>   | Optimization Techniques           | <b>X</b>      | Decision Trees               |
| <b>III</b>  | Bayesian Decision Theory          | <b>XI</b>     | Support Vector Machines      |
| <b>IV</b>   | Non-parametric Density Estimation | <b>XII</b>    | Clustering                   |
| <b>V</b>    | Probabilistic Graphical Models    | • <b>XIII</b> | Principal Component Analysis |
| <b>VI</b>   | Linear Regression                 | <b>XIV</b>    | Reinforcement Learning       |
| <b>VII</b>  | Logistic Regression               | <b>XV</b>     | Advanced Regression          |
| <b>VIII</b> | Deep Learning                     |               |                              |

# Agenda for this Unit

- 1 Introduction
- 2 Derivation of the PCA Algorithm
- 3 Implementation of the PCA Algorithm
- 4 FISHER's Linear Discriminant Analysis (FLDA)
- 5 Wrap-Up

## Section: Introduction

Why Dimensionality Reduction?

Use Case I: Data Compression

Use Case II: Data Visualization

Further PCA Applications

What is PCA?

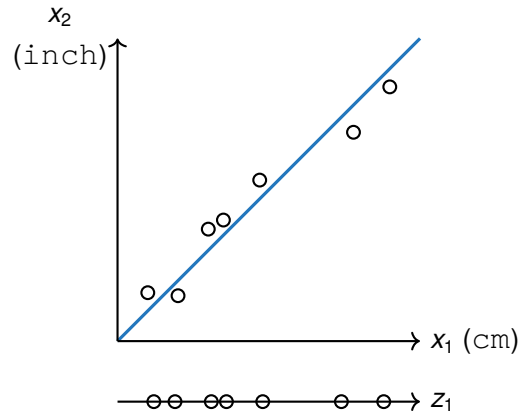
# Why Dimensionality Reduction?

- Most datasets are high-dimensional (*i. e. they have a large amount of features*)
- Dimensionality reduction can be used for:
  - **Lossy (!)** data compression,
  - Feature extraction, and
  - Data visualization

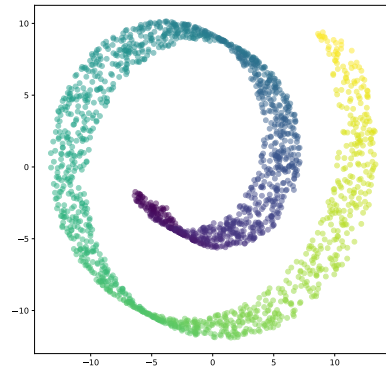
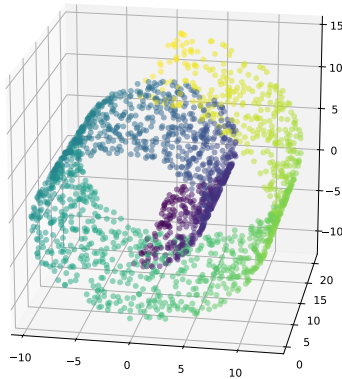
Dimensionality reduction can help **speed up** learning algorithms substantially. Too many (correlated) features usually **decrease the performance** of the learning algorithm (**curse of dimensionality**).

# Use Case I: Data Compression / Feature Extraction

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



# Use Case II: Data Visualization



# Application of PCA to Images: Eigenfaces



Figure: Original images



Figure: First 36 principal components



## Application of PCA to Images: Eigenfaces (Ctd.)

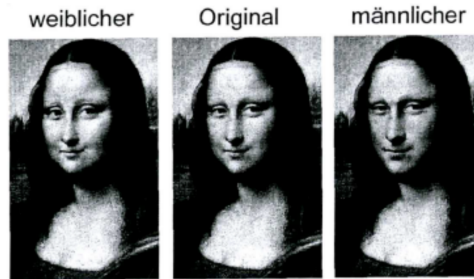


Figure: Original images



Figure: Reconstructed images

# Application of PCA to Images: Face Morphing

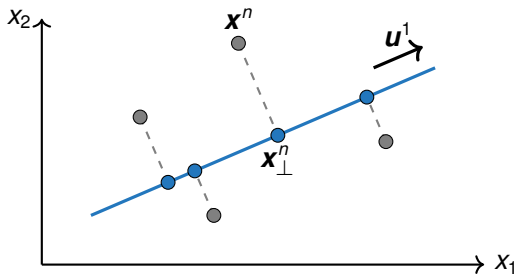


# PCA: Principal Component Analysis

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

**Goal: Remove dimensions which are the least informative of the data!**

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



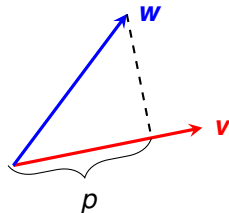
- $\mathbf{x}^n$  denotes the original data point
- $\mathbf{x}_{\perp}^n$  is the **orthogonal projection** of  $\mathbf{x}^n$  onto the vector  $\mathbf{u}^1$

**The goal is to find  $\mathbf{u}^1$  such that the variance of the projection is maximized!**

## Recall: Projection of Vectors

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

$$\begin{aligned} p &= \|\mathbf{w}\| \cos \angle(\mathbf{v}, \mathbf{w}) \\ &= \|\mathbf{w}\| \frac{\mathbf{v}^\top \mathbf{w}}{\|\mathbf{v}\| \cdot \|\mathbf{w}\|} = \frac{\mathbf{v}^\top \mathbf{w}}{\|\mathbf{v}\|} \end{aligned}$$



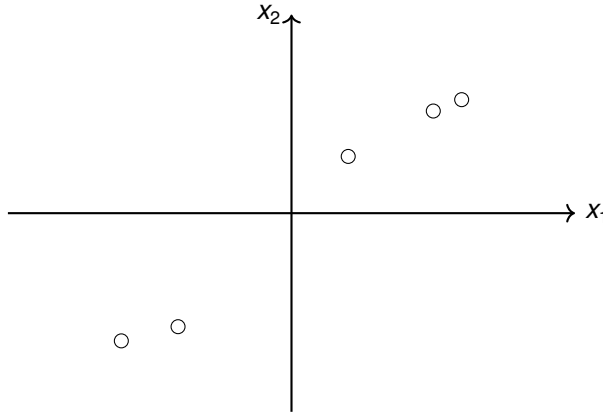
- We will assume  $\mathbf{u}^1$  to be a unit vector, i. e.  $\|\mathbf{u}^1\| = 1$
- $\frac{(\mathbf{u}^1)^\top \mathbf{x}^n}{\|\mathbf{u}^1\|}$  then reduces to the scalar product  $(\mathbf{u}^1)^\top \mathbf{x}^n$

## Section:

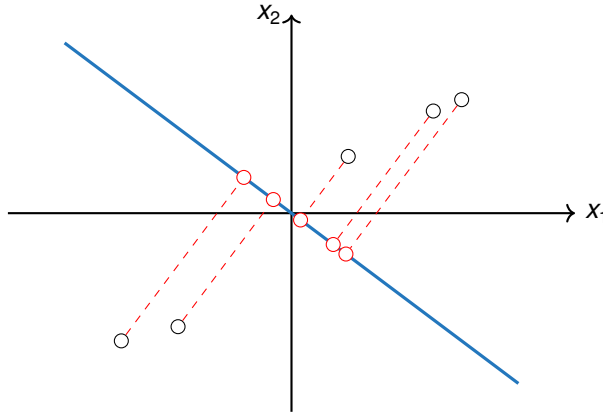
# Derivation of the PCA Algorithm

Introduction / Maximum Variance Formulation  
Formalization of the Problem  
An Example  
Properties of Covariance Matrices

# Maximum Variance Formulation

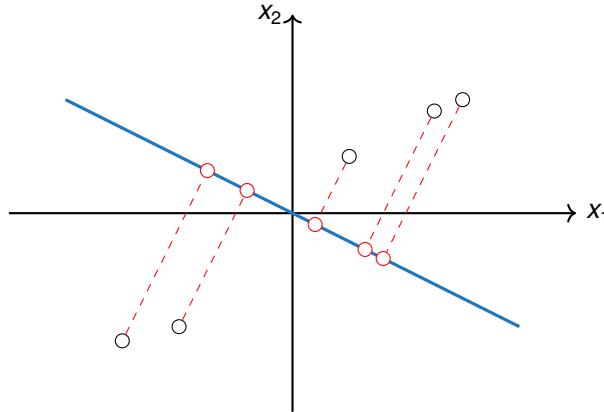


# Maximum Variance Formulation

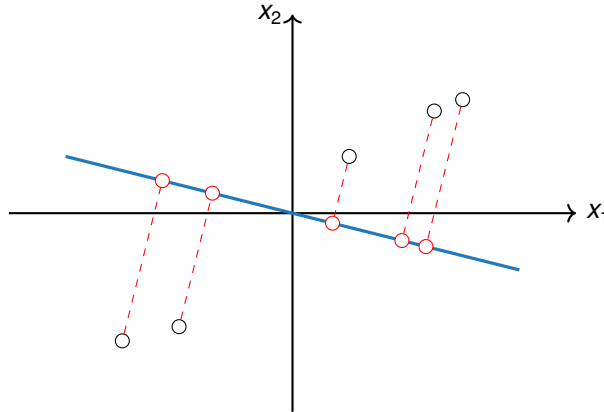




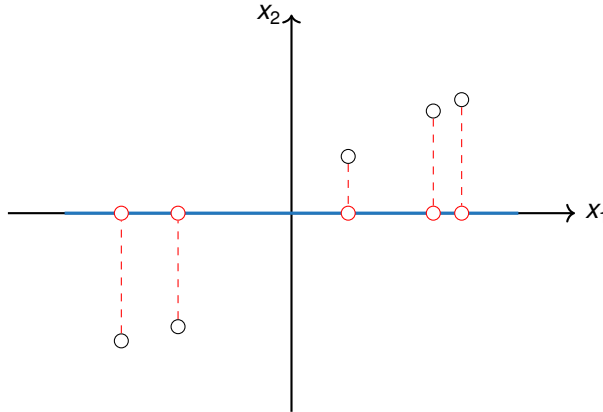
# Maximum Variance Formulation



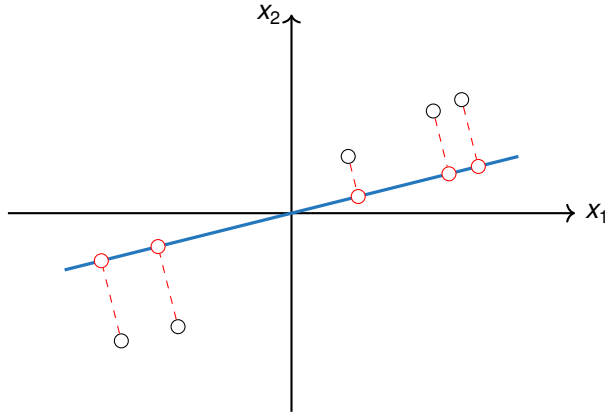
# Maximum Variance Formulation



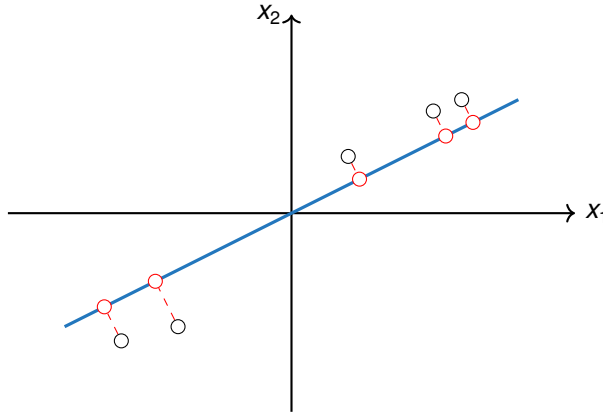
# Maximum Variance Formulation



# Maximum Variance Formulation



# Maximum Variance Formulation



## Maximum Variance Formulation (Ctd.)

- In the following we shall assume  $D = 1$   
(i. e. we project the data onto a line defined by a unit vector  $\mathbf{u}^1$ )
- Each data point  $\mathbf{x}^n \in \mathbb{R}^M$  is projected onto a scalar value  $(\mathbf{u}^1)^\top \mathbf{x}^n \in \mathbb{R}$
- The **mean** of the projected data is  $(\mathbf{u}^1)^\top \boldsymbol{\mu}$ , where

$$\boldsymbol{\mu} := \frac{1}{N} \sum_{n=1}^N \mathbf{x}^n$$

- The **variance** of the projected data is given by *(expand the square and simplify!)*:

$$\frac{1}{N} \sum_{n=1}^N ((\mathbf{u}^1)^\top \mathbf{x}^n - (\mathbf{u}^1)^\top \boldsymbol{\mu})^2 = (\mathbf{u}^1)^\top \boldsymbol{\Sigma} \mathbf{u}^1 \quad (1)$$

## Maximum Variance Formulation (Ctd.)

- $\Sigma \in \mathbb{R}^{M \times M}$  is the **covariance matrix** defined by:

$$\Sigma := \frac{1}{N} \sum_{n=1}^N (\mathbf{x}^n - \mu)(\mathbf{x}^n - \mu)^\top \quad (2)$$

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

$$\max_{\mathbf{u}^1} \{ (\mathbf{u}^1)^\top \Sigma \mathbf{u}^1 + \lambda_1 (1 - (\mathbf{u}^1)^\top \mathbf{u}^1) \} \quad (3)$$



## Maximum Variance Formulation (Ctd.)

- We have to solve

$$\frac{\partial}{\partial \mathbf{u}^1} \left[ (\mathbf{u}^1)^\top \Sigma \mathbf{u}^1 + \lambda_1 (1 - (\mathbf{u}^1)^\top \mathbf{u}^1) \right] \stackrel{!}{=} \mathbf{0}$$

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

**Important:** The variance is maximized by setting  $\mathbf{u}^1$  equal to the eigenvector of  $\Sigma$  having the largest eigenvalue  $\lambda_1$ . This eigenvector is the first principal component and its eigenvalue  $\lambda_1$  is the variance it retains.





# Derivation of the Eigenvalue Problem

- Remember:  $\frac{\partial}{\partial \mathbf{x}} \mathbf{x}^\top \mathbf{A} \mathbf{x} = 2\mathbf{A} \mathbf{x}$ , if  $\mathbf{A}$  is a symmetric matrix
- Remember:  $\mathbf{x}^\top \mathbf{x} = \|\mathbf{x}\|^2$ , and  $\frac{\partial}{\partial \mathbf{x}} \|\mathbf{x}\|^2 = 2\mathbf{x}$  (see exercise sheet #1)
- We get (because  $\Sigma$  is symmetric):

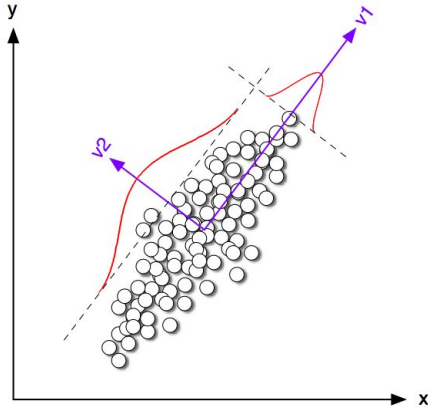
$$\begin{aligned} \frac{\partial}{\partial \mathbf{u}^1} \left[ (\mathbf{u}^1)^\top \Sigma \mathbf{u}^1 + \lambda_1 (1 - (\mathbf{u}^1)^\top \mathbf{u}^1) \right] &= 2\Sigma \mathbf{u}^1 - 2\lambda_1 \mathbf{u}^1 \\ &= 2(\Sigma \mathbf{u}^1 - \lambda_1 \mathbf{u}^1) \stackrel{!}{=} \mathbf{0} \end{aligned}$$

- Setting this derivative to zero and reordering the terms yields the eigenvalue problem  $\Sigma \mathbf{u}^1 = \lambda_1 \mathbf{u}^1$

## 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  $D$  dimensions:
  - The lower-dimensional subspace is defined by the  $D$  eigenvectors  $\mathbf{u}^1, \mathbf{u}^2, \dots, \mathbf{u}^D$  of the covariance matrix  $\Sigma$
  - These correspond to the  $D$  largest eigenvalues  $\lambda_1^*, \lambda_2^*, \dots, \lambda_D^*$

# Principal Components

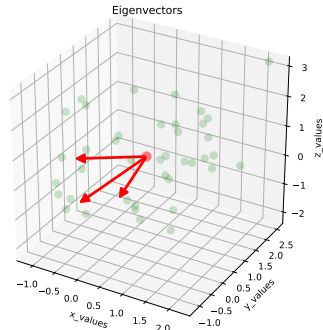
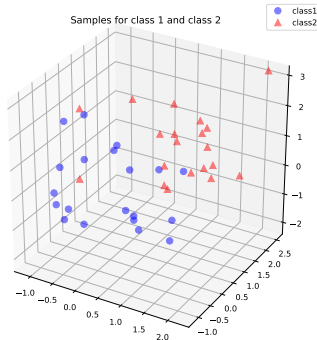


Here,  $\mathbf{v}^1$  is the **first principal component**. It captures the most variance of the data. The **second principal component** is given by  $\mathbf{v}^2$ .

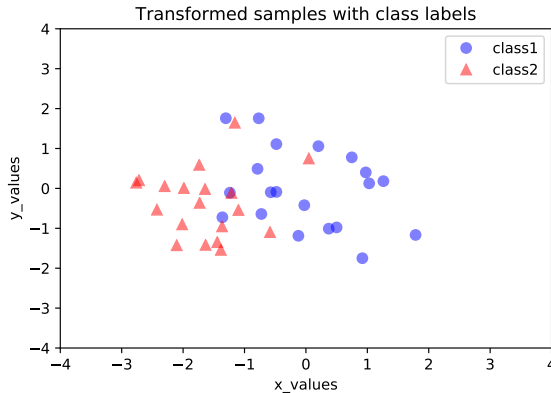
We see that both principal components are orthogonal, i. e.

$$(\mathbf{v}^1)^\top \mathbf{v}^2 = 0.$$

# PCA Example: Projection $\mathbb{R}^3 \rightarrow \mathbb{R}^2$



## PCA Example: Projection $\mathbb{R}^3 \rightarrow \mathbb{R}^2$ (Ctd.)



# Covariance Matrix

Let the  $M$  features  $F_1, \dots, F_M$  be given, then

$$\Sigma := \begin{pmatrix} \text{cov}(F_1, F_1) & \text{cov}(F_1, F_2) & \dots & \text{cov}(F_1, F_M) \\ \text{cov}(F_2, F_1) & \text{cov}(F_2, F_2) & \dots & \text{cov}(F_2, F_M) \\ \vdots & \vdots & \ddots & \vdots \\ \text{cov}(F_M, F_1) & \text{cov}(F_M, F_2) & \dots & \text{cov}(F_M, F_M) \end{pmatrix} \in \mathbb{R}^{M \times M} \quad (4)$$

**Remark:**  $\text{cov}(F_m, F_m) = \mathbb{V}(F_m)$  for  $m = 1, 2, \dots, M$

# Properties of the Covariance Matrix

The covariance matrix  $\Sigma$  is computed according to:

$$\Sigma := \frac{1}{N} \sum_{n=1}^N (\mathbf{x}^n - \boldsymbol{\mu})(\mathbf{x}^n - \boldsymbol{\mu})^\top \quad (5)$$

**Property ①** The matrix  $\Sigma$  is a **square** ( $M \times M$ )-matrix, where  $M$  is the number of features in the dataset

## Properties of the Covariance Matrix (Ctd.)

**Property ②** The matrix  $\Sigma$  is **positive semi-definite**, i. e.

$$\mathbf{x}^\top \Sigma \mathbf{x} \geq 0 \quad \forall \mathbf{x} \in \mathbb{R}^M$$

It follows that all eigenvalues of  $\Sigma$  are **non-negative** and capture the **amount of variability** in an orthogonal basis given by the principal components

**Property ③** The matrix  $\Sigma$  is always a **symmetric** matrix, i. e. we have  $\Sigma^\top = \Sigma$ , because  $\text{cov}(F_i, F_j) = \text{cov}(F_j, F_i)$  for all features  $F_i$  and  $F_j$



# Properties of the Covariance Matrix (Ctd.)

**Property ④** The entries on the main diagonal of  $\Sigma$  are **non-negative** as they represent the variances of the individual features

## Section:

# Implementation of the PCA Algorithm

### Algorithm Overview

- Step 1: Computation of the Covariance Matrix
- Step 2: Computation of Eigenvalues and Eigenvectors
- Step 3: Choice of the Number of Dimensions  $D$
- Step 4: Projection of the Data onto the Principal Subspace

# PCA Algorithm

**Input:** Input data  $\mathbf{X} = (\mathbf{x}^1, \mathbf{x}^2, \dots, \mathbf{x}^N)^\top \in \mathbb{R}^{N \times M}$ , number of dimensions  $D$

**Output:** Projected data  $\mathbf{Z} \in \mathbb{R}^{N \times D}$

- 1 Compute the sample set mean  $\boldsymbol{\mu} \leftarrow \frac{1}{N} \sum_{n=1}^N \mathbf{x}^n$
- 2 Compute the covariance matrix  $\boldsymbol{\Sigma} \leftarrow \frac{1}{N} \sum_{n=1}^N (\mathbf{x}^n - \boldsymbol{\mu})(\mathbf{x}^n - \boldsymbol{\mu})^\top$
- 3 Eigendecomposition: Find matrices  $\mathbf{U}, \boldsymbol{\Lambda} \in \mathbb{R}^{M \times M}$  such that:  $\boldsymbol{\Sigma} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^\top$
- 4 Select the  $D$  eigenvectors with the largest eigenvalues to form the columns of  $\mathbf{V}$
- 5 Project the data:  $\mathbf{Z} \leftarrow \mathbf{X}\mathbf{V}$

## Example: Computation of the Covariance Matrix

- **Example:** Let the following dataset be given:

$$\mathbf{X} := \left( (1, 4)^\top, (4, 1)^\top, (1, 1)^\top \right)^\top$$

- We begin by computing the sample set mean  $\boldsymbol{\mu}$  of the dataset  $\mathbf{X}$
- We obtain (*by calculating the component-wise arithmetic mean*):

$$\boldsymbol{\mu} = \frac{1}{3} \left[ \begin{pmatrix} 1 \\ 4 \end{pmatrix} + \begin{pmatrix} 4 \\ 1 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \end{pmatrix} \right] = \begin{pmatrix} 2 \\ 2 \end{pmatrix}$$

## Example: Computation of the Covariance Matrix (Ctd.)

- We compute the outer products which we need to compute the covariance matrix:
- We get:

$$\Sigma_1 := (\mathbf{x}^1 - \boldsymbol{\mu})(\mathbf{x}^1 - \boldsymbol{\mu})^\top = \begin{pmatrix} -1 \\ 2 \end{pmatrix} \begin{pmatrix} -1 & 2 \end{pmatrix} = \begin{pmatrix} 1 & -2 \\ -2 & 4 \end{pmatrix}$$

$$\Sigma_2 := (\mathbf{x}^2 - \boldsymbol{\mu})(\mathbf{x}^2 - \boldsymbol{\mu})^\top = \begin{pmatrix} 2 \\ -1 \end{pmatrix} \begin{pmatrix} 2 & -1 \end{pmatrix} = \begin{pmatrix} 4 & -2 \\ -2 & 1 \end{pmatrix}$$

$$\Sigma_3 := (\mathbf{x}^3 - \boldsymbol{\mu})(\mathbf{x}^3 - \boldsymbol{\mu})^\top = \begin{pmatrix} -1 \\ -1 \end{pmatrix} \begin{pmatrix} -1 & -1 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$$

## Example: Computation of the Covariance Matrix (Ctd.)

- The covariance matrix is then computed by adding the matrices  $\Sigma_n$  ( $n = 1, 2, 3$ ) followed by component-wise division by the number of data points (here:  $N = 3$ )
- The covariance matrix of  $\mathbf{X}$  is:

$$\begin{aligned}\Sigma &= \frac{1}{3} \left[ \begin{pmatrix} 1 & -2 \\ -2 & 4 \end{pmatrix} + \begin{pmatrix} 4 & -2 \\ -2 & 1 \end{pmatrix} + \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} \right] \\ &= \begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix}\end{aligned}$$

# Eigenvalues and Eigenvectors

- As a next step we have to find vectors  $\mathbf{u}$  and scalars  $\lambda$  which satisfy the equation

$$\Sigma \mathbf{u} = \lambda \mathbf{u}$$

- The vectors  $\mathbf{u}$  are called **eigenvectors** and the scalars  $\lambda$  are referred to as **eigenvalues** of the covariance matrix  $\Sigma$
- The eigenvalues  $\lambda$  are the roots (German: *Nullstellen*) of the **characteristic polynomial**  $\chi_{\Sigma}$  of  $\Sigma$  defined by:

$$\chi_{\Sigma}(\lambda) := \det(\lambda \mathbf{I}_M - \Sigma) \tag{6}$$

## Example (continued): Computation of Eigenvalues

- The characteristic polynomial of  $\Sigma$  is given by

$$\begin{aligned}\chi_{\Sigma}(\lambda) &= \det \left[ \begin{pmatrix} \lambda & 0 \\ 0 & \lambda \end{pmatrix} - \begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix} \right] = \det \begin{pmatrix} \lambda - 2 & 1 \\ 1 & \lambda - 2 \end{pmatrix} \\ &= (\lambda - 2)^2 - 1 = \lambda^2 - 4\lambda + 3 \\ &= (\lambda - 3)(\lambda - 1)\end{aligned}$$

- Therefore, the eigenvalues are given by  $\lambda_1 = 3$  and  $\lambda_2 = 1$



## Finding the corresponding Eigenvectors

- Let  $\lambda_j$  be an eigenvalue of  $\Sigma$
- We want to find the corresponding eigenvectors  $\mathbf{u}$  such that

$$\begin{aligned}\Sigma \mathbf{u} = \lambda_j \mathbf{u} &\iff \Sigma \mathbf{u} - \lambda_j \mathbf{u} = \mathbf{0} \\ &\iff (\Sigma - \lambda_j \mathbf{I}_M) \mathbf{u} = \mathbf{0}\end{aligned}$$

- Therefore, we have to find the solutions to the following **homogeneous system of linear equations** (see  $\Rightarrow$  [here](#) how this is done), where we set  $\mathbf{A}_j := \Sigma - \lambda_j \mathbf{I}_M$

$$\mathbf{A}_j \mathbf{u} = \mathbf{0}$$

## Example (continued): Computation of Eigenvectors

- We compute the eigenvectors for eigenvalue  $\lambda_1 = 3$ :

$$(\Sigma - 3 \cdot I_M) = \begin{pmatrix} -1 & -1 \\ -1 & -1 \end{pmatrix} \xrightarrow{-I+II} \begin{pmatrix} -1 & -1 \\ 0 & 0 \end{pmatrix} \xrightarrow{(-1) \cdot I} \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix}$$

Therefore, the eigenspace connected to eigenvalue  $\lambda_1 = 3$  is given by

$$\mathcal{E}(3) = \{t \cdot (1, -1)^\top : t \in \mathbb{R}, t \neq 0\}$$

- Similarly, we obtain  $\mathcal{E}(1) = \{t \cdot (1, 1)^\top : t \in \mathbb{R}, t \neq 0\}$  for  $\lambda_2 = 1$

# The Eigendecomposition of $\Sigma$

- Without loss of generality we can assume that the eigenvectors are normalized, i. e.  $\|\mathbf{u}\| = 1$  (since  $\mathbf{u}/\|\mathbf{u}\|$  is an eigenvector connected to the same eigenvalue)
- The eigenvalues and eigenvectors of  $\Sigma$  can be used to decompose  $\Sigma \in \mathbb{R}^{M \times M}$  into a product of three matrices  $\Sigma = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$ , where  $\mathbf{U} \in \mathbb{R}^{M \times M}$  and  $\mathbf{\Lambda} \in \mathbb{R}^{M \times M}$
- $\mathbf{U}$  is obtained by stacking the **normalized** eigenvectors column-wise:

$$\mathbf{U} := \begin{pmatrix} | & | & \dots & | \\ \mathbf{u}^1 & \mathbf{u}^2 & \dots & \mathbf{u}^M \\ | & | & & | \end{pmatrix} \in \mathbb{R}^{M \times M} \quad (7)$$



## The Eigendecomposition of $\Sigma$ (Ctd.)

- $\Lambda := \text{diag}(\lambda_1, \dots, \lambda_M)$  is a **diagonal matrix** with the eigenvalues on the diagonal:

$$\Lambda := \begin{pmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_M \end{pmatrix}$$

- If you put an eigenvector into column  $m$  of  $\mathbf{U}$ , you have to make sure to put the corresponding eigenvalue in column  $m$  of  $\Lambda$

**Important: The order of eigenvectors and eigenvalues has to be consistent**

## Example (continued): The Eigendecomposition of $\Sigma$

- For  $\lambda_1 = 3$  we choose

$$\mathbf{u}^1 := 1/\sqrt{2} \cdot (1, -1)^\top$$

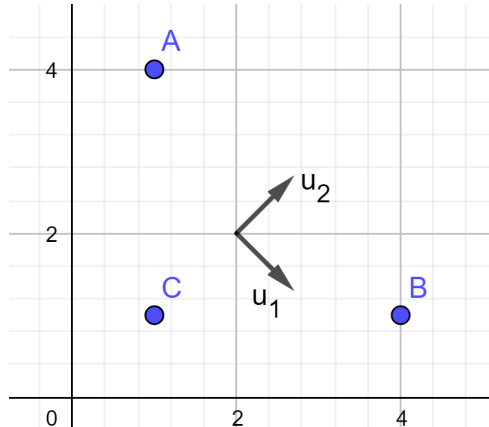
- For  $\lambda_2 = 1$  we choose

$$\mathbf{u}^2 := 1/\sqrt{2} \cdot (1, 1)^\top$$

- Finally, we are able to write down the **eigendecomposition** of  $\Sigma$ :

$$\Sigma = \begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix} = \begin{pmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ -1/\sqrt{2} & 1/\sqrt{2} \end{pmatrix} \begin{pmatrix} 3 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1/\sqrt{2} & -1/\sqrt{2} \\ 1/\sqrt{2} & 1/\sqrt{2} \end{pmatrix} = \mathbf{U} \mathbf{\Lambda} \mathbf{U}^\top$$

## Example (continued): Visualization Principal Components



## Choice of $D$ : Strategy 1

- The goal is to preserve **as much variance as possible**
- In the derivation we have seen that the **eigenvalues represent the amount of variance** captured by the respective principal components
- Again, we have a look at the  $(M \times M)$ -matrix  $\mathbf{\Lambda}$

$$\mathbf{\Lambda} = \begin{pmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_M \end{pmatrix}$$

## Choice of $D$ : Strategy 1 (Ctd.)

- Sort the eigenvalues in descending order
- Without loss of generality we assume that  $\lambda_1$  is the largest, and  $\lambda_M$  the smallest eigenvalue (*otherwise we can rearrange the elements in the matrices accordingly*)
- Choose the smallest  $D$  which **satisfies the inequality**:

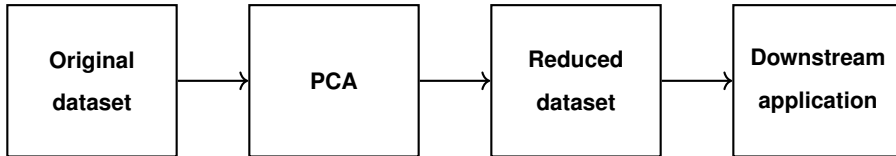
$$\frac{\sum_{j=1}^D \lambda_j}{\sum_{j=1}^M \lambda_j} \geq \gamma \quad \gamma \in [0, 1] \quad (8)$$

- $\gamma$  specifies the fraction of variance to be retained overall (*this is a hyperparameter of the algorithm*)



## Choice of $D$ : Strategy 2

- PCA is rarely used on its own, but in combination with a downstream application or classification task
- Another possible strategy therefore is to choose  $D$  so as to **maximize the performance in this downstream application**





# Projection of the Data

- We construct the matrix  $\mathbf{V}$  (containing only the **normalized** eigenvectors connected to the  $D$  largest eigenvalues) which is given by

$$\mathbf{V} := \begin{pmatrix} | & | & \dots & | \\ \mathbf{u}^1 & \mathbf{u}^2 & \dots & \mathbf{u}^D \\ | & | & & | \end{pmatrix} \in \mathbb{R}^{M \times D} \quad (9)$$

- The projection of the data from  $M$  to  $D$  dimensions ( $D \ll M$ ) is then performed by matrix multiplication:

$$\mathbf{Z} := \mathbf{XV} \in \mathbb{R}^{N \times D} \quad (10)$$

## Example (continued): Projection of the Data

- We choose to reduce  $\mathbf{X}$  to one dimension and select the principal component  $\mathbf{u}^1 = \frac{1}{\sqrt{2}} \cdot (1, -1)^\top$  connected to the larger eigenvalue  $\lambda_1 = 3$
- $\mathbf{V}$  is therefore given by

$$\mathbf{V} := \begin{pmatrix} 1/\sqrt{2} \\ -1/\sqrt{2} \end{pmatrix}$$

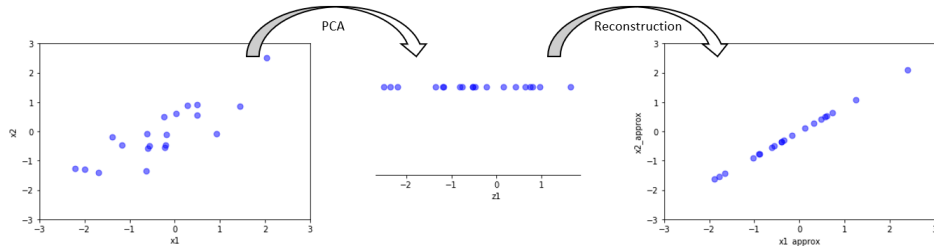
- The projected data  $\mathbf{Z} \in \mathbb{R}^{N \times D}$  is then obtained by matrix multiplication:

$$\mathbf{Z} := \mathbf{XV} = \begin{pmatrix} 1 & 4 \\ 4 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 1/\sqrt{2} \\ -1/\sqrt{2} \end{pmatrix} \approx \begin{pmatrix} -2.121 \\ 2.121 \\ 0 \end{pmatrix}$$

# Reconstruction from compressed Representation

It is possible to compute an **approximate reconstruction** of the data after having applied PCA:

$$\mathbf{X}_{\approx} := \mathbf{ZV}^T \quad (11)$$



## Example (continued): Projection of the Data

The reconstructed data is given by

$$\begin{aligned}\mathbf{x}_{\approx} &:= \mathbf{ZV}^T = \begin{pmatrix} -2.121 \\ 2.121 \\ 0 \end{pmatrix} \begin{pmatrix} 1/\sqrt{2} & -1/\sqrt{2} \end{pmatrix} \\ &= \begin{pmatrix} -1.5 & 1.5 \\ 1.5 & -1.5 \\ 0 & 0 \end{pmatrix}\end{aligned}$$

## Section:

# FISHER's Linear Discriminant Analysis (FLDA)

Introduction

Derivation of the optimal 1D Projection

# Dimensionality Reduction for Classification

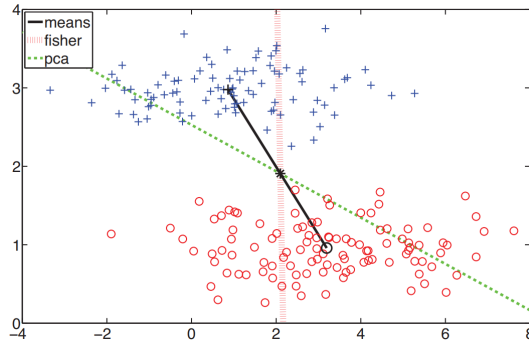
- We can use dimensionality reduction for classification
- However, using PCA often results in poor classification performance as it does not take the class labels into account
- Consider a labeled dataset comprising  $N$  training examples

$$\mathcal{D} := \{(\mathbf{x}^1, y_1), (\mathbf{x}^2, y_2), \dots, (\mathbf{x}^N, y_N)\}$$

- We consider two-class problems only, i. e.  $y_n \in \{1, 2\}$  for  $n = 1, 2, \dots, N$

**Goal:** Find a 1D projection which maximizes the class separation

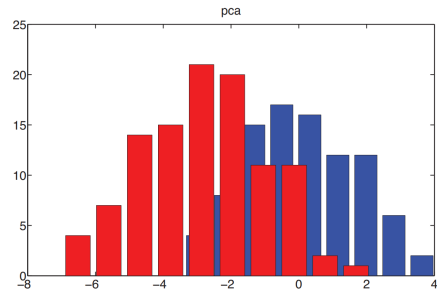
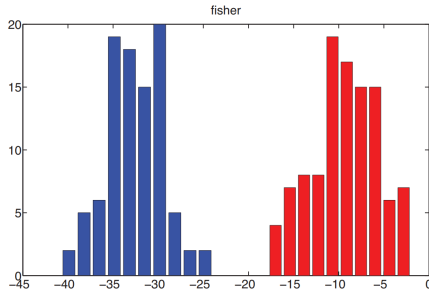
# FLDA vs. PCA



cf. MURPHY.2012, page 272



## FLDA vs. PCA (Ctd.)



cf. MURPHY.2012, page 272

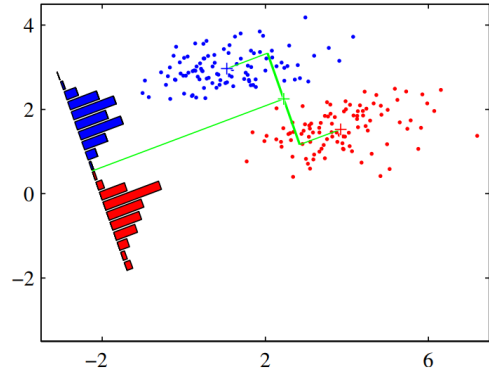
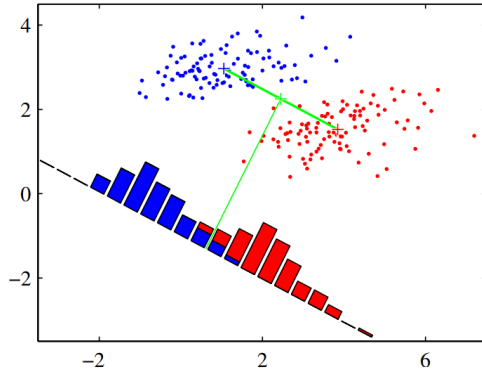
## Projection of the Means

- We derive the optimal direction  $\mathbf{w}$  for the two-class case
- The class-conditional means are defined as ( $N_1$  examples from  $\mathcal{C}_1$ ,  $N_2$  from  $\mathcal{C}_2$ )

$$\boldsymbol{\mu}^1 := \frac{1}{N_1} \sum_{n: y_n=1} \mathbf{x}^n \quad \text{and} \quad \boldsymbol{\mu}^2 := \frac{1}{N_2} \sum_{n: y_n=2} \mathbf{x}^n \quad (12)$$

- Let  $m_k := \mathbf{w}^\top \boldsymbol{\mu}^k$ ,  $k = 1, 2$ , be the projection of each mean onto the line  $\mathbf{w}$
- One approach could be to maximize the distance between these means, i. e.  
 $\max \boldsymbol{\mu}^2 - \boldsymbol{\mu}^1$
- However, this does usually not result in a good model

# Maximizing the Distance between the Means



cf. BISHOP.2006, page 188

# Projected Variance

- Let  $z_n := \mathbf{w}^\top \mathbf{x}^n$  be the projection of the data points onto the line  $\mathbf{w}$
- The variance of the projected data points belonging to class  $k$  is

$$s_k^2 := \sum_{n: y_n = k} (z_n - m_k)^2 \quad (13)$$

**Goal:** Find  $\mathbf{w}$  so as to maximize the distance between the projected means, i. e.  $m_2 - m_1$ , while also ensuring the projected clusters are *tight*, i. e. have low variance

# FISHER Criterion

## FISHER criterion:

$$\mathfrak{J}_F(\mathbf{w}) := \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2} = \frac{\mathbf{w}^\top \mathbf{S}_B \mathbf{w}}{\mathbf{w}^\top \mathbf{S}_W \mathbf{w}} \quad (14)$$

- We define the **between-class scatter matrix**  $\mathbf{S}_B$ :

$$\mathbf{S}_B := (\boldsymbol{\mu}^2 - \boldsymbol{\mu}^1)(\boldsymbol{\mu}^2 - \boldsymbol{\mu}^1)^\top \quad (15)$$

- We define the **within-class scatter matrix**  $\mathbf{S}_W$

$$\mathbf{S}_W := \sum_{n:y_n=1} (\mathbf{x}^n - \boldsymbol{\mu}^1)(\mathbf{x}^n - \boldsymbol{\mu}^1)^\top + \sum_{n:y_n=2} (\mathbf{x}^n - \boldsymbol{\mu}^2)(\mathbf{x}^n - \boldsymbol{\mu}^2)^\top \quad (16)$$

## FISHER Criterion (Ctd.)

**Proof:** We proof that we can rewrite the FISHER criterion as in equation (14)

$$\begin{aligned}\mathbf{w}^\top \mathbf{S}_B \mathbf{w} &= \mathbf{w}^\top (\boldsymbol{\mu}^2 - \boldsymbol{\mu}^1) (\boldsymbol{\mu}^2 - \boldsymbol{\mu}^1)^\top \mathbf{w} \\ &= (m_2 - m_1)(m_2 - m_1) = (m_2 - m_1)^2 \\ \mathbf{w}^\top \mathbf{S}_W \mathbf{w} &= \sum_{n:y_n=1} \mathbf{w}^\top (\mathbf{x}^n - \boldsymbol{\mu}^1) (\mathbf{x}^n - \boldsymbol{\mu}^1)^\top \mathbf{w} + \sum_{n:y_n=2} \mathbf{w}^\top (\mathbf{x}^n - \boldsymbol{\mu}^2) (\mathbf{x}^n - \boldsymbol{\mu}^2)^\top \mathbf{w} \\ &= \sum_{n:y_n=1} (z_n - m_1)^2 + \sum_{n:y_n=2} (z_n - m_2)^2 = s_1^2 + s_2^2\end{aligned}$$

## Maximization of the Objective

- We have to maximize equation (14) to find the optimal  $\mathbf{w}$
- For this we take the derivative of (14) with respect to  $\mathbf{w}$  and set it to zero
- One can show that  $\mathfrak{J}_F$  is maximized when

$$\mathbf{S}_B \mathbf{w} = \lambda \mathbf{S}_W \mathbf{w} \quad \text{where} \quad \lambda := \frac{\mathbf{w}^\top \mathbf{S}_B \mathbf{w}}{\mathbf{w}^\top \mathbf{S}_W \mathbf{w}} \quad (17)$$

- Equation (17) is called **generalized eigenvalue problem**
- If  $\mathbf{S}_W$  is invertible, we can convert it to the regular eigenvalue problem

$$\mathbf{S}_W^{-1} \mathbf{S}_B \mathbf{w} = \lambda \mathbf{w} \quad (18)$$

## Maximization of the Objective

- We know  $\mathbf{S}_B \mathbf{w} = (\boldsymbol{\mu}^2 - \boldsymbol{\mu}^1)(\boldsymbol{\mu}^2 - \boldsymbol{\mu}^1)^\top \mathbf{w} = (\boldsymbol{\mu}^2 - \boldsymbol{\mu}^1)(m_2 - m_1)$
- From equation (18) we have

$$\lambda \mathbf{w} = \mathbf{S}_W^{-1} (\boldsymbol{\mu}^2 - \boldsymbol{\mu}^1)(m_2 - m_1) \quad (19)$$

$$\mathbf{w} \propto \mathbf{S}_W^{-1} (\boldsymbol{\mu}^2 - \boldsymbol{\mu}^1) \quad (20)$$


Since we only care about the directionality, and not the scale factor, we simply set  $\mathbf{w} = \mathbf{S}_W^{-1} (\boldsymbol{\mu}^2 - \boldsymbol{\mu}^1)$



## Section: Wrap-Up

Summary  
Recommended Literature  
Self-Test Questions  
Lecture Outlook

# Summary

- Dimensionality reduction is important when we want 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) subspace called the **principal subspace**
- We want to **keep the dimensions with the most variance**
- These dimensions are called **principal components**
- Many applications: Data visualization, eigenfaces, morphing, ...
- FLDA can be used to reduce the dimensionality in classification problems

# Recommended Literature

## 1 PCA

- [BISHOP.2006], chapter 12
- [MURPHY.2012], chapter 12.2

## 2 FLDA

- [BISHOP.2006], chapter 4.1.4
- [MURPHY.2012], chapter 8.6.3

(For free PDF versions, see list in GitHub readme!)



# 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!
- 7 Describe what FLDA is! How do you find the optimal direction?

# What's next...?

- |             |                                   |              |                              |
|-------------|-----------------------------------|--------------|------------------------------|
| <b>I</b>    | Machine Learning Introduction     | <b>IX</b>    | Evaluation                   |
| <b>II</b>   | Optimization Techniques           | <b>X</b>     | Decision Trees               |
| <b>III</b>  | Bayesian Decision Theory          | <b>XI</b>    | Support Vector Machines      |
| <b>IV</b>   | Non-parametric Density Estimation | <b>XII</b>   | Clustering                   |
| <b>V</b>    | Probabilistic Graphical Models    | <b>XIII</b>  | Principal Component Analysis |
| <b>VI</b>   | Linear Regression                 | • <b>XIV</b> | Reinforcement Learning       |
| <b>VII</b>  | Logistic Regression               | <b>XV</b>    | Advanced Regression          |
| <b>VIII</b> | Deep Learning                     |              |                              |

**Thank you very much for the attention!**

**\*\*\* Artificial Intelligence and Machine Learning \*\*\***

**Topic:** Principal Component Analysis

**Term:** Summer term 2025

**Contact:**

Daniel Wehner, M.Sc.

SAP SE / DHBW Mannheim

[daniel.wehner@sap.com](mailto:daniel.wehner@sap.com)

**Do you have any questions?**