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

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

Agenda for this Unit

Section:

Introduction

- Why Dimensionality Reduction?
- Use Case I: Data Compression
- Use Case II: Data Visualization
- Further PCA Applications
- What is PCA?

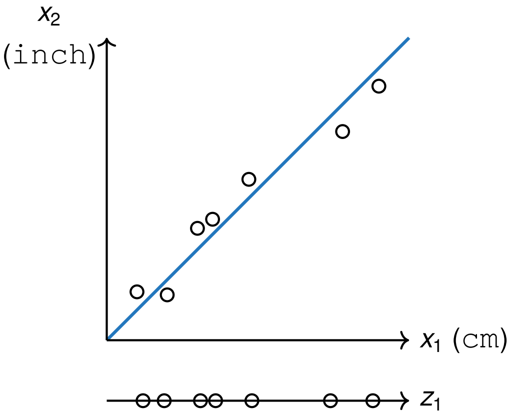
Why Dimensionality Reduction?

- 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

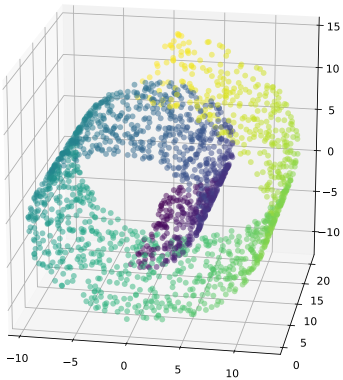
Dimensionality reduction can help **speed up** learning algorithms substantially. Too many (correlated) features usually **decrease the performance** of the learning algorithm **(curse of dimensionality)**.

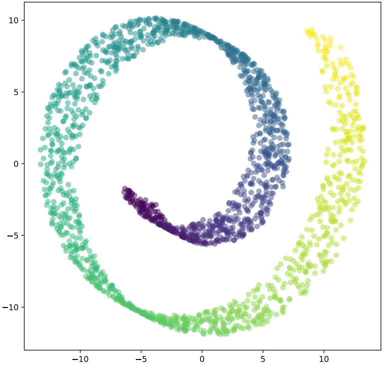
Use Case I: Data Compression / Feature Extraction

- 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 $(\mathbb{R}^2 \to \mathbb{R})$



Use Case II: Data Visualization





Application of PCA to Images: Eigenfaces

Figure: Original images

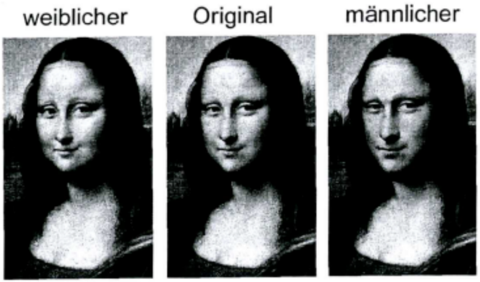
Figure: First 36 principal components

Application of PCA to Images: Eigenfaces (Ctd.)

Figure: Original images

Figure: Reconstructed images

Application of PCA to Images: Face Morphing



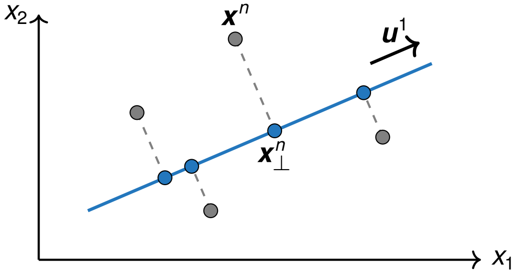
PCA: Principal Component Analysis

- 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 $\boldsymbol{X} := \left\{\boldsymbol{x}^1, \boldsymbol{x}^2, \dots, \boldsymbol{x}^N\right\}$
  - $\boldsymbol{x}^n \in \mathbb{R}^M$ $(1 \leqslant n \leqslant N)$ 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** $(\mathbb{R}^M \to \mathbb{R}^D)$

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

Important

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

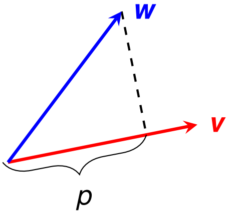


- $\boldsymbol{x}^n$ denotes the original data point
- $\boldsymbol{x}_\perp^n$ is the **orthogonal projection** of $\boldsymbol{x}^n$ onto the vector $\boldsymbol{u}^1$

The goal is to find $\boldsymbol{u}^1$ such 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 (orthogonal) projection of $\boldsymbol{w}$ onto $\boldsymbol{v}$ defined?



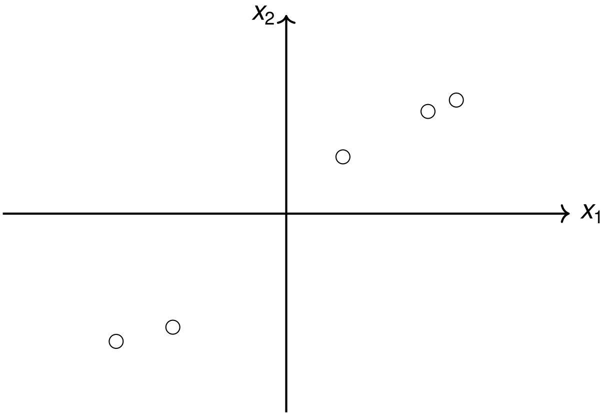
$$
\begin{aligned}
p &= \|\boldsymbol{w}\| \cos \measuredangle(\boldsymbol{v}, \boldsymbol{w}) \\
&= \|\boldsymbol{w}\| \frac{\boldsymbol{v}^\top \boldsymbol{w}}{\|\boldsymbol{v}\| \cdot \|\boldsymbol{w}\|} = \frac{\boldsymbol{v}^\top \boldsymbol{w}}{\|\boldsymbol{v}\|}
\end{aligned}
$$

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

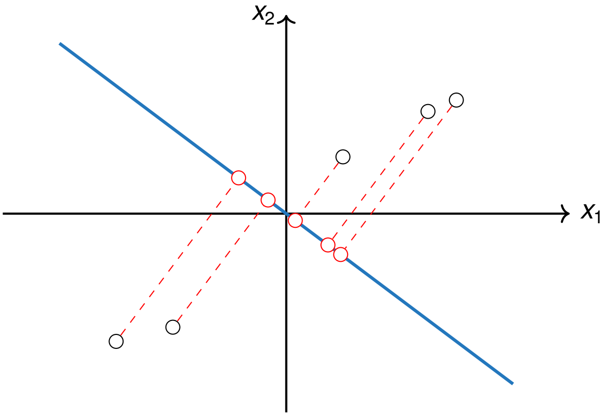
Section:

Derivation of the PCA Algorithm

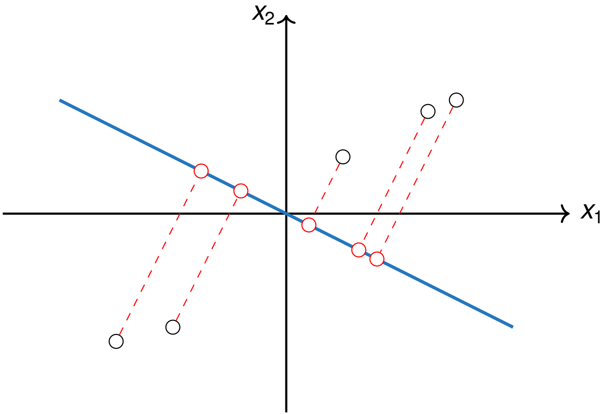
Maximum Variance Formulation



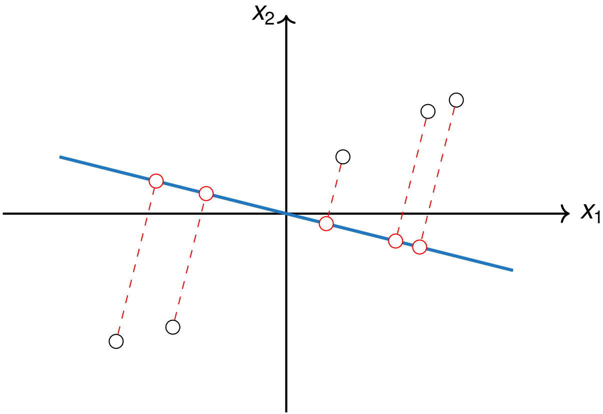
Maximum Variance Formulation



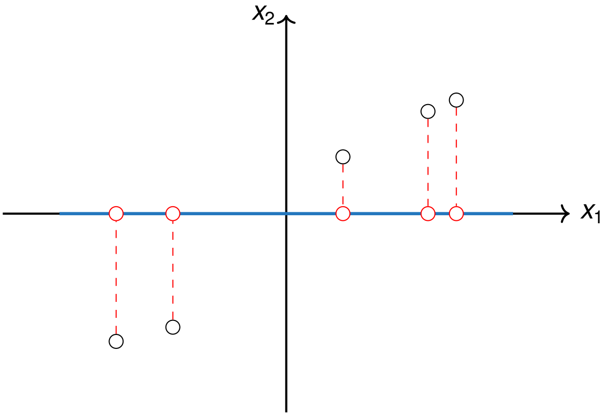
Maximum Variance Formulation



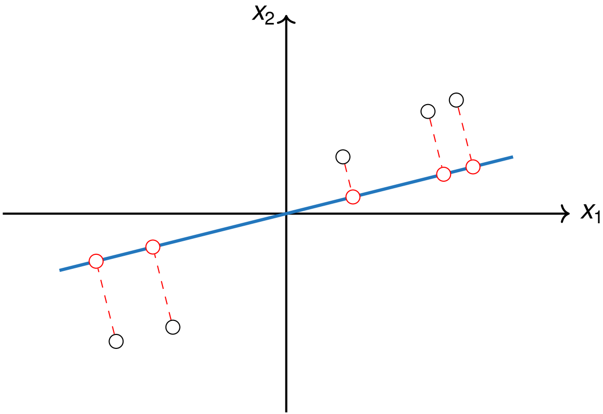
Maximum Variance Formulation



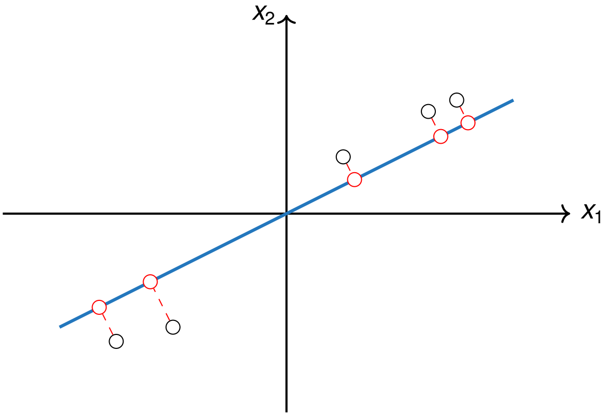
Maximum Variance Formulation



Maximum Variance Formulation



Maximum Variance Formulation



Important

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 $\boldsymbol{u}^1$*)*
- Each data point $\boldsymbol{x}^n \in \mathbb{R}^M$ is projected onto a scalar value $(\boldsymbol{u}^1)^\top \boldsymbol{x}^n \in \mathbb{R}$
- The **mean** of the projected data is $(\boldsymbol{u}^1)^\top \boldsymbol{\mu}$, where

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

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

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

Maximum Variance Formulation (Ctd.)

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

$$\boldsymbol{\Sigma} := \frac{1}{N} \sum_{n=1}^{N} \left(\boldsymbol{x}^n - \boldsymbol{\mu} \right) \left(\boldsymbol{x}^n - \boldsymbol{\mu} \right)^\top \tag{2}$$

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

$$\max_{\boldsymbol{u}^1} \left\{ (\boldsymbol{u}^1)^\top \boldsymbol{\Sigma} \boldsymbol{u}^1 + \lambda_1 \left(1 - (\boldsymbol{u}^1)^\top \boldsymbol{u}^1 \right) \right\} \tag{3}$$

Maximum Variance Formulation (Ctd.)

- We have to solve

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

- 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 $(\boldsymbol{u}^1)^\top$ and use $(\boldsymbol{u}^1)^\top \boldsymbol{u}^1 = 1$, we see: $(\boldsymbol{u}^1)^\top \boldsymbol{\Sigma} \boldsymbol{u}^1 = \lambda_1$

The variance is maximized by setting $\boldsymbol{u}^1$ equal to the eigenvector of $\boldsymbol{\Sigma}$ having the largest eigenvalue λ_1. This eigenvector is the first principal component and its eigenvalue λ_1 is the variance it retains.

Derivation of the Eigenvalue Problem

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

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

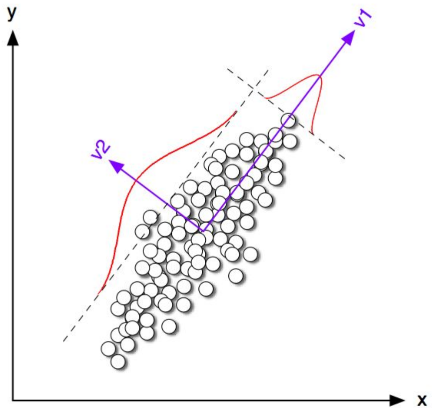
- Setting this derivative to zero and reordering the terms yields the eigenvalue problem $\boldsymbol{\Sigma}\boldsymbol{u}^1 = \lambda_1\boldsymbol{u}^1$

Important

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 $\boldsymbol{u}^1, \boldsymbol{u}^2, \dots, \boldsymbol{u}^D$ of the covariance matrix $\boldsymbol{\Sigma}$
  - These correspond to the D largest eigenvalues $\lambda_1^\star, \lambda_2^\star, \dots, \lambda_D^\star$

Principal Components

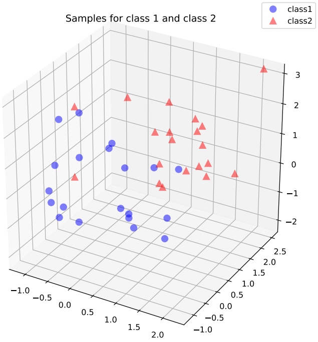


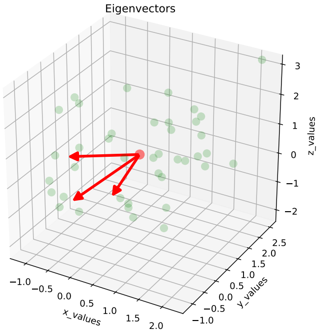
Here, $\boldsymbol{v}^1$ is the **first principal component**. It captures the most variance of the data. The **second principal component** is given by $\boldsymbol{v}^2$.

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

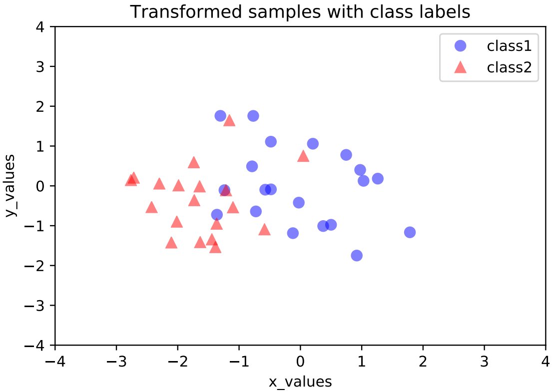
$$(\boldsymbol{v}^1)^\top \boldsymbol{v}^2 = 0.$$

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





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

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

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

Properties of the Covariance Matrix

The covariance matrix $\boldsymbol{\Sigma}$ is computed according to:

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

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

Properties of the Covariance Matrix (Ctd.)

Property ❷ The matrix $\boldsymbol{\Sigma}$ is **positive semi-definite**, i. e.

$$\boldsymbol{x}^{\top} \boldsymbol{\Sigma} \boldsymbol{x} \geqslant 0 \quad \forall \boldsymbol{x} \in \mathbb{R}^{M}$$

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

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

Properties of the Covariance Matrix (Ctd.)

Property ❹ The entries on the main diagonal of Σ 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 $\boldsymbol{X} = \left(\boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^n\right) \in \mathbb{R}^{N \times M}$, number of dimensions D

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

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

Example: Computation of the Covariance Matrix

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

$$\boldsymbol{X} := \{(1, 4), (4, 1), (1, 1)\}$$

- We begin by computing the sample set mean $\boldsymbol{\mu}$ of the dataset $\boldsymbol{X}$
- 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:

$$\boldsymbol{\Sigma}_1 := (\boldsymbol{x}^1 - \boldsymbol{\mu})(\boldsymbol{x}^1 - \boldsymbol{\mu})^\top = \begin{pmatrix} -1 \\ 2 \end{pmatrix} \begin{pmatrix} -1 & 2 \end{pmatrix} = \begin{pmatrix} 1 & -2 \\ -2 & 4 \end{pmatrix}$$

$$\boldsymbol{\Sigma}_2 := (\boldsymbol{x}^2 - \boldsymbol{\mu})(\boldsymbol{x}^2 - \boldsymbol{\mu})^\top = \begin{pmatrix} 2 \\ -1 \end{pmatrix} \begin{pmatrix} 2 & -1 \end{pmatrix} = \begin{pmatrix} 4 & -2 \\ -2 & 1 \end{pmatrix}$$

$$\boldsymbol{\Sigma}_3 := (\boldsymbol{x}^3 - \boldsymbol{\mu})(\boldsymbol{x}^3 - \boldsymbol{\mu})^\top = \begin{pmatrix} -1 \\ -1 \end{pmatrix} \begin{pmatrix} -1 & -1 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$$

Example: Computation of the Covariance Matrix (Ctd.)

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

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

$$= \begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix}$$

Eigenvalues and Eigenvectors

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

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

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

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

Example (continued): Computation of Eigenvalues

- The characteristic polynomial of $\boldsymbol{\Sigma}$ is given by

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

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

Finding the corresponding Eigenvectors

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

$$\begin{aligned} \boldsymbol{\Sigma}\boldsymbol{u} = \lambda_j \boldsymbol{u} \quad &\Longleftrightarrow \quad \boldsymbol{\Sigma}\boldsymbol{u} - \lambda_j \boldsymbol{u} = \boldsymbol{0} \\ &\Longleftrightarrow \quad (\boldsymbol{\Sigma} - \lambda_j \boldsymbol{I}_M)\boldsymbol{u} = \boldsymbol{0} \end{aligned}$$

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

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

Example (continued): Computation of Eigenvectors

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

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

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

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

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

The Eigendecomposition of $\boldsymbol{\Sigma}$

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

$$\boldsymbol{U} := \begin{pmatrix} | & | & & | \\ \boldsymbol{u}^1 & \boldsymbol{u}^2 & \dots & \boldsymbol{u}^M \\ | & | & & | \end{pmatrix} \in \mathbb{R}^{M \times M} \tag{7}$$

The Eigendecomposition of $\boldsymbol{\Sigma}$ (Ctd.)

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

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

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

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

Example (continued): The Eigendecomposition of $\boldsymbol{\Sigma}$

- For $\lambda_1 = 3$ we choose

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

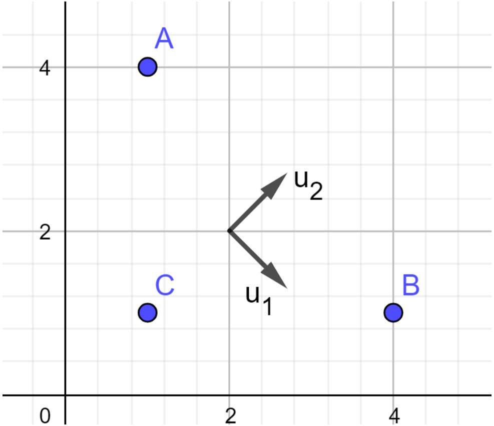
- For $\lambda_2 = 1$ we choose

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

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

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

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 $\boldsymbol{\Lambda}$

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

Choice of D: Strategy 1 (Ctd.)

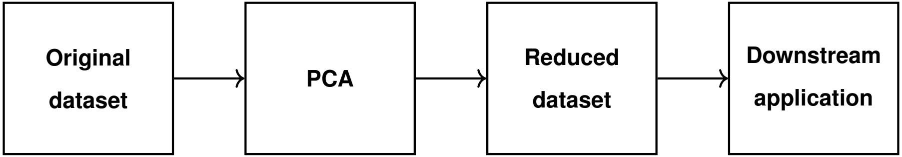
- Sort the eigenvalues in descending order
- Without loss of generality we assume that λ_1 is the largest, and λ_M the smallest eigenvalue *(otherwise we can rearrange the elements in the matrices accordingly)*
- Choose the smallest D which **satisfies the inequality**:

$$\frac{\sum_{j=1}^{D} \lambda_j}{\sum_{j=1}^{M} \lambda_j} \geqslant \gamma \qquad \gamma \in [0, 1] \tag{8}$$

- γ 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**



Important

Projection of the Data

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

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

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

$$\boldsymbol{Z} := \boldsymbol{X}\boldsymbol{V} \in \mathbb{R}^{N \times D} \tag{10}$$

Example (continued): Projection of the Data

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

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

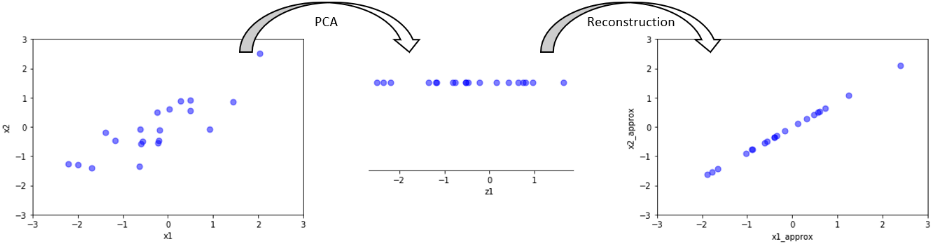
- The projected data $\boldsymbol{Z} \in \mathbb{R}^{N \times D}$ is then obtained by matrix multiplication:

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

Reconstruction from compressed Representation

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

$$\boldsymbol{X}_{\approx} := \boldsymbol{Z}\boldsymbol{V}^{\top} \tag{11}$$



Example (continued): Projection of the Data

The reconstructed data is given by

$$\boldsymbol{X}_{\approx} := \boldsymbol{Z}\boldsymbol{V}^\top = \begin{pmatrix} -2.121 \\ 2.121 \\ 0 \end{pmatrix} \begin{pmatrix} 1/\sqrt{2} & -1/\sqrt{2} \end{pmatrix}$$

$$= \begin{pmatrix} -1.5 & 1.5 \\ 1.5 & -1.5 \\ 0 & 0 \end{pmatrix}$$

Section:

FISHER's Linear Discriminant Analysis (FLDA)

Introduction
Derivation of the optimal 1D Projection

Dimensionality Reduction for Classification

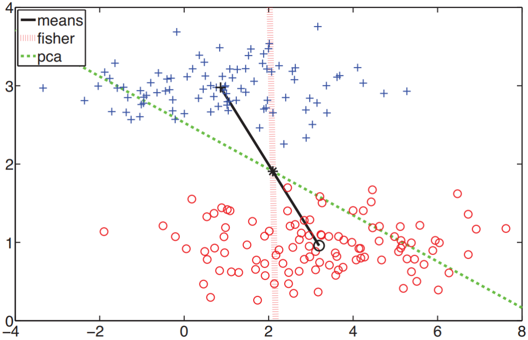
- We can use dimensionality reduction for classification
- However, using PCA often results in poor classification performance as it does not take the class labels into account
- Consider a labeled dataset comprising N training examples

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

- We consider two-class problems only, i. e. $y \in \{1, 2\}$

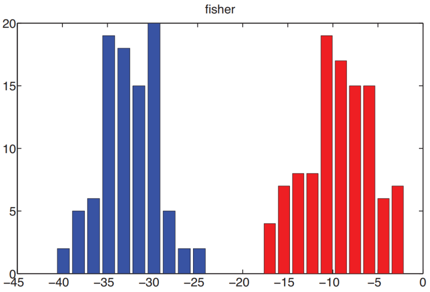
Goal: Find a 1D projection which maximizes the class separation

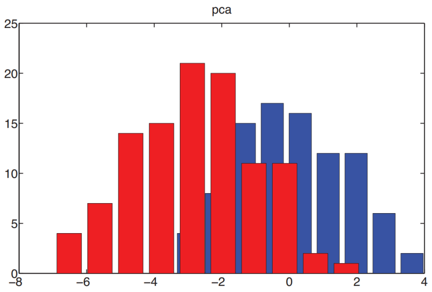
FLDA vs. PCA



cf. MURPHY.2012, page 272

FLDA vs. PCA (Ctd.)





cf. MURPHY.2012, page 272

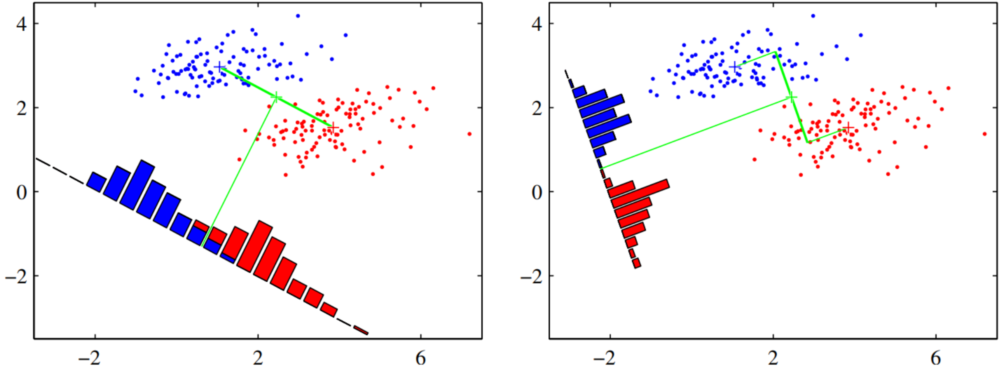
Projection of the Means

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

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

- Let $m_k := \boldsymbol{w}^\top \boldsymbol{\mu}^k$, $k = 1, 2$, be the projection of each mean onto the line $\boldsymbol{w}$
- 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 $z_n := \boldsymbol{w}^\top \boldsymbol{x}^n$ be the projection of the data points onto the line $\boldsymbol{w}$
- The variance of the projected points belonging to class k is

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

Goal: Find $\boldsymbol{w}$ 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:

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

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

$$\boldsymbol{S}_B := \left(\boldsymbol{\mu}^2 - \boldsymbol{\mu}^1\right)\left(\boldsymbol{\mu}^2 - \boldsymbol{\mu}^1\right)^\top \tag{15}$$

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

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

FISHER Criterion (Ctd.)

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

$$\begin{aligned}
\boldsymbol{w}^\top \boldsymbol{S}_B \boldsymbol{w} &= \boldsymbol{w}^\top (\boldsymbol{\mu}^2 - \boldsymbol{\mu}^1)(\boldsymbol{\mu}^2 - \boldsymbol{\mu}^1)^\top \boldsymbol{w} \\
&= (m_2 - m_1)(m_2 - m_1) = (m_2 - m_1)^2
\end{aligned}$$

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

Maximization of the Objective

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

$$\boldsymbol{S}_B\boldsymbol{w} = \lambda \boldsymbol{S}_W\boldsymbol{w} \qquad \text{where} \qquad \lambda := \frac{\boldsymbol{w}^\top \boldsymbol{S}_B \boldsymbol{w}}{\boldsymbol{w}^\top \boldsymbol{S}_W \boldsymbol{w}} \tag{17}$$

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

$$\boldsymbol{S}_W^{-1}\boldsymbol{S}_B\boldsymbol{w} = \lambda \boldsymbol{w} \tag{18}$$

Maximization of the Objective

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

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

$$\boldsymbol{w} \propto \boldsymbol{S}_W^{-1}\left(\boldsymbol{\mu}^2 - \boldsymbol{\mu}^1\right) \tag{20}$$

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

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, ...

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...?

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

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

Do you have any questions?