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

Daniel Wehner, M.Sc.

SAP SE / DHBW Mannheim

Winter term 2023/2024

Duale Hochschule Baden-Württemberg

Find all slides on GitHub (DaWe1992/Applied_ML_Fundamentals)

Lecture Overview

Unit I	Machine Learning Introduction
Unit II	Mathematical Foundations
Unit III	Bayesian Decision Theory
Unit IV	Regression
Unit V	Classification I
Unit VI	Evaluation
Unit VII	Classification II
Unit VIII	Clustering
Unit IX	**Dimensionality Reduction**

Agenda for this Unit

Section:

Introduction

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

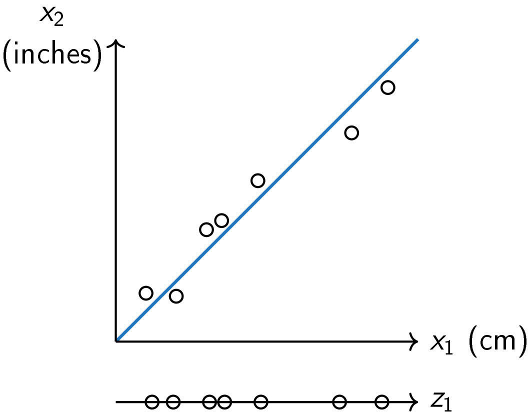
Why Dimensionality Reduction?

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

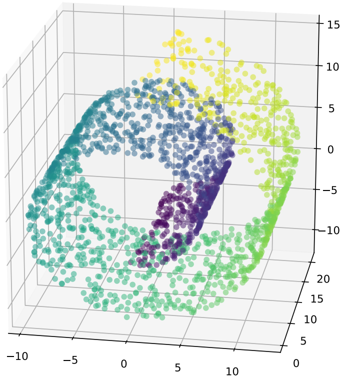
Dimensionality reduction can help to **speed up** learning algorithms substantially. Too many (correlated) features usually **decrease the performance** of the learning algorithm (**curse of dimensionality**).

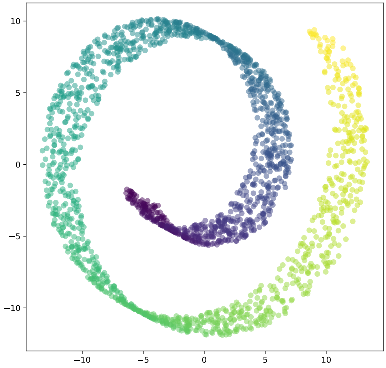
Use Case I: Data Compression / Feature Extraction

- 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





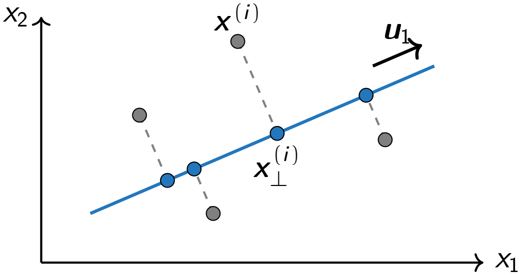
PCA: Principal Component Analysis

- 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: $\mathbb{R}^2 \rightarrow \mathbb{R}$)



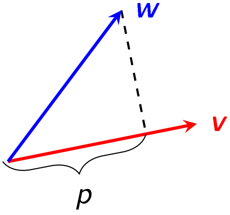
- $x^{(i)}$ denotes the original data point
- $x_\perp^{(i)}$ is the orthogonal projection of $x^{(i)}$ onto the vector u_1

The goal is to find a suitable vector u_1 so that the variance of the projection is maximized!

Important

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?



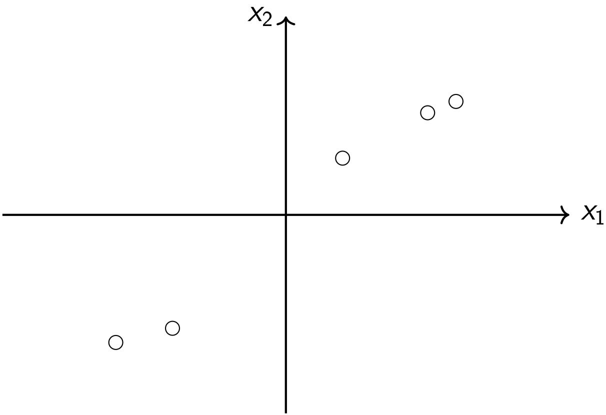
$$\begin{aligned} p &= \|\boldsymbol{w}\| \cos \measuredangle(\boldsymbol{v}, \boldsymbol{w}) \\ &= \|\boldsymbol{w}\| \frac{\boldsymbol{v}^\intercal \boldsymbol{w}}{\|\boldsymbol{v}\| \cdot \|\boldsymbol{w}\|} = \frac{\boldsymbol{v}^\intercal \boldsymbol{w}}{\|\boldsymbol{v}\|} \end{aligned}$$

- 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 $\boldsymbol{u}_1^\intercal \boldsymbol{x}^{(i)}$

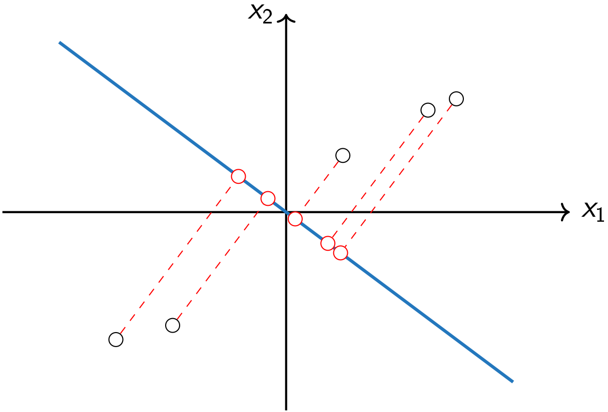
Section:

Derivation of the PCA Algorithm

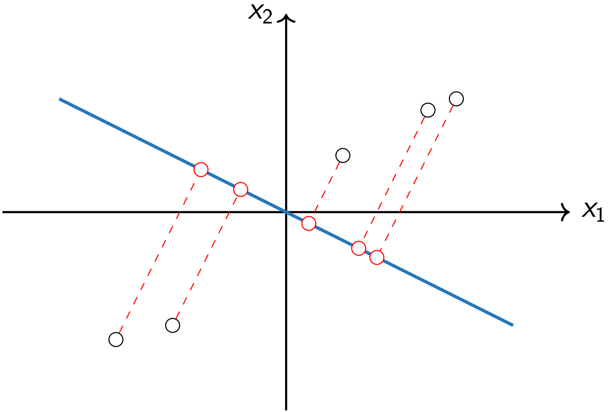
Maximum Variance Formulation



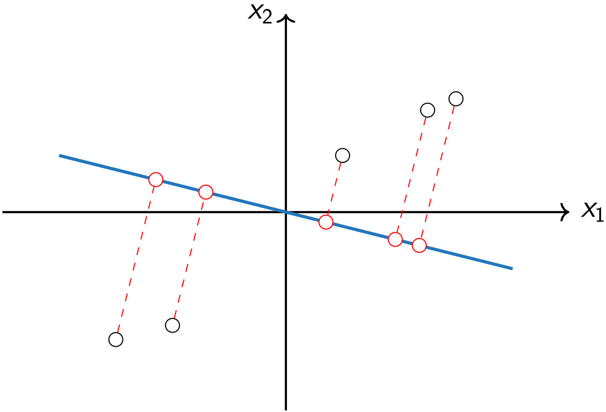
Maximum Variance Formulation



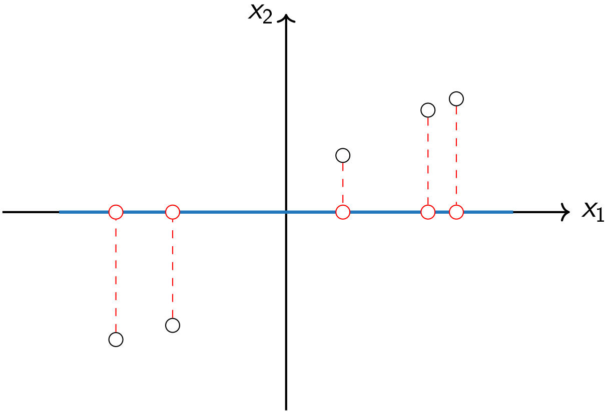
Maximum Variance Formulation



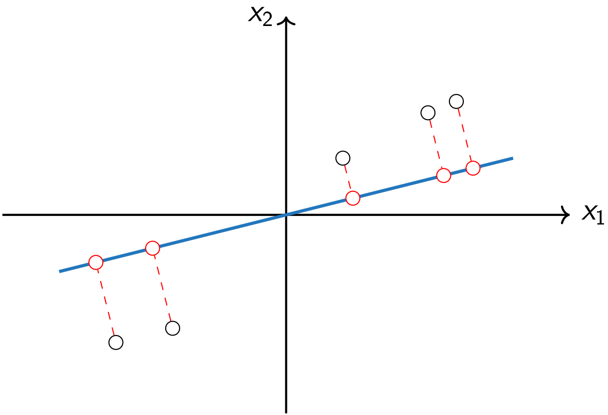
Maximum Variance Formulation



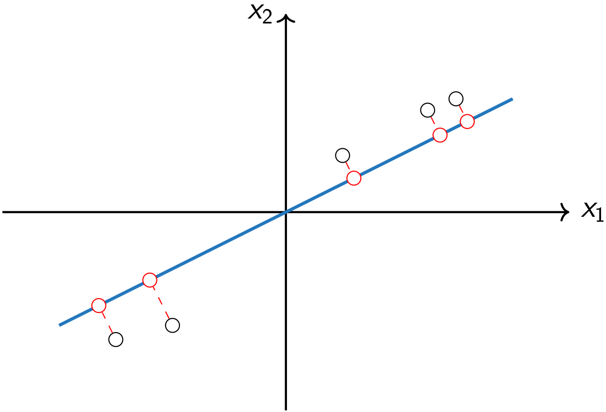
Maximum Variance Formulation



Maximum Variance Formulation



Maximum Variance Formulation



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:

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

Maximum Variance Formulation (Ctd.)

- We have to solve

$$\frac{\partial}{\partial \boldsymbol{u}_1} \left\{ \boldsymbol{u}_1^\intercal \boldsymbol{\Sigma} \boldsymbol{u}_1 + \lambda_1 (1 - \boldsymbol{u}_1^\intercal \boldsymbol{u}_1) \right\} \stackrel{!}{=} \mathbf{0}$$

- 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 $\boldsymbol{u}_1^\intercal$ 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.

Derivation of the Eigenvalue Problem

- Remember: $\frac{\partial}{\partial \boldsymbol{x}} \boldsymbol{x}^\intercal \boldsymbol{A} \boldsymbol{x} = 2\boldsymbol{A}\boldsymbol{x}$ if $\boldsymbol{A}$ is a symmetric matrix
- We get (because $\boldsymbol{\Sigma}$ is symmetric):

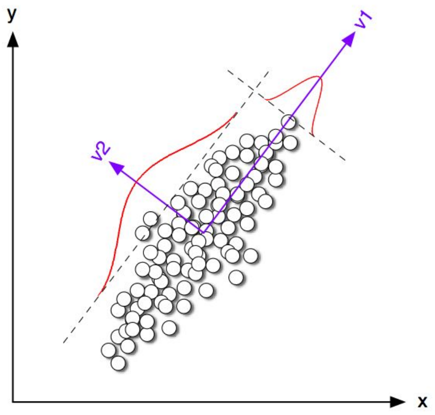
$$\begin{aligned} \frac{\partial}{\partial \boldsymbol{u}_1} \left\{ \boldsymbol{u}_1^\intercal \boldsymbol{\Sigma} \boldsymbol{u}_1 + \lambda_1 (1 - \boldsymbol{u}_1^\intercal \boldsymbol{u}_1) \right\} &= 2\boldsymbol{\Sigma}\boldsymbol{u}_1 - 2\lambda_1 \boldsymbol{u}_1 \\ &= 2(\boldsymbol{\Sigma}\boldsymbol{u}_1 - \lambda_1 \boldsymbol{u}_1) \stackrel{!}{=} \boldsymbol{0} \end{aligned}$$

- Setting this derivative to zero and reordering the terms yields the eigenvalue problem $\boldsymbol{\Sigma}\boldsymbol{u}_1 = \lambda_1 \boldsymbol{u}_1$

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 $\lambda_1, \lambda_2, \ldots, \lambda_k$

Principal Components

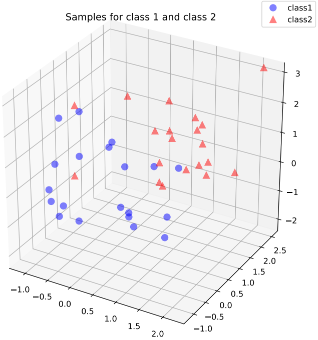


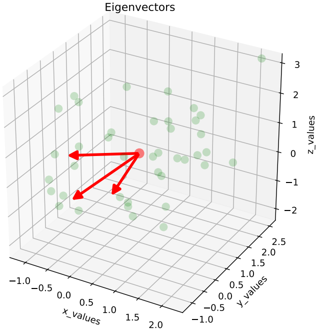
Here, $\boldsymbol{v}_1$ is the **first principal component**. It captures the most variance of the data. The **second principal component** is given by $\boldsymbol{v}_2$.

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

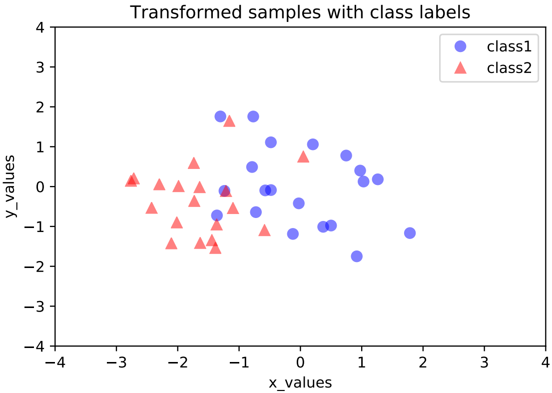
$$\boldsymbol{v}_1^{\mathsf{T}} \boldsymbol{v}_2 = 0.$$

PCA Result


Samples for class 1 and class 2
class1
class2


Eigenvectors
x_values
y_values
z_values

PCA Result (Ctd.)



Covariance Matrix

Let the m features $\mathcal{F}_1, \ldots, \mathcal{F}_m$ be given, then

$$\boldsymbol{\Sigma} := \begin{pmatrix} \text{cov}(\mathcal{F}_1, \mathcal{F}_1) & \text{cov}(\mathcal{F}_1, \mathcal{F}_2) & \ldots & \text{cov}(\mathcal{F}_1, \mathcal{F}_m) \\ \text{cov}(\mathcal{F}_2, \mathcal{F}_1) & \text{cov}(\mathcal{F}_2, \mathcal{F}_2) & \ldots & \text{cov}(\mathcal{F}_2, \mathcal{F}_m) \\ \vdots & \vdots & \ddots & \vdots \\ \text{cov}(\mathcal{F}_m, \mathcal{F}_1) & \text{cov}(\mathcal{F}_m, \mathcal{F}_2) & \ldots & \text{cov}(\mathcal{F}_m, \mathcal{F}_m) \end{pmatrix} \tag{4}$$

Remark: $\text{cov}(\mathcal{F}_i, \mathcal{F}_i) = \mathbb{V}(\mathcal{F}_i)$ for $i = 1, 2, \ldots, m$

Properties of the Covariance Matrix

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

$$\boldsymbol{\Sigma} := \frac{1}{n} \sum_{i=1}^{n} \overbrace{(\boldsymbol{x}^{(i)} - \overline{\boldsymbol{x}})(\boldsymbol{x}^{(i)} - \overline{\boldsymbol{x}})^\intercal}^{\textbf{Outer product} \to \textbf{matrix}} \tag{5}$$

- For m features, $\boldsymbol{\Sigma}$ is a **quadratic** $(m \times m)$ matrix
- The entries on the main diagonal are non-negative as they represent the variances of the individual features
- $\boldsymbol{\Sigma}$ is **positive-semidefinite**, i. e. $\boldsymbol{x}^\intercal \boldsymbol{\Sigma} \boldsymbol{x} \geqslant 0 \ \forall \boldsymbol{x} \in \mathbb{R}^m$

Properties of the Covariance Matrix (Ctd.)

- $\boldsymbol{\Sigma}$ is **symmetric**, i. e.

$$\boldsymbol{\Sigma}^{\intercal} = \boldsymbol{\Sigma}.$$

  The reason for this is that the covariance of two random variables is a symmetric function: $\operatorname{cov}(\mathcal{F}_i, \mathcal{F}_j) = \operatorname{cov}(\mathcal{F}_j, \mathcal{F}_i)\ \forall i, j$

- All eigenvalues of $\boldsymbol{\Sigma}$ are non-negative. The eigenvalues capture the **amount of variability** in an orthogonal basis given by the principal components

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 k
Step 4: Projection of the Data onto the Principal Subspace

Important

PCA Algorithm Overview

Algorithm 1: PCA Algorithm

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

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

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

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

Perform an eigendecomposition for $\boldsymbol{\Sigma}$: $\boldsymbol{\Sigma} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^{\mathsf{T}}$

Select the k eigenvectors with the largest eigenvalues and stack them column-wise into $\widetilde{\boldsymbol{U}}$

Project the data: $\boldsymbol{Z} \longleftarrow \boldsymbol{X}\widetilde{\boldsymbol{U}}$

Example: Computation of the Covariance Matrix

Example: Let the dataset

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

be given. We begin by computing the mean $\overline{x}$ of the dataset $\boldsymbol{X}$ (component-wise arithmetic mean). We obtain:

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

Example: Computation of the Covariance Matrix (Ctd.)

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

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

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

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

Example: Computation of the Covariance Matrix (Ctd.)

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

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

Eigenvalues and Eigenvectors

- 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**
- The eigenvalues λ are the roots (German: *Nullstellen*) of the **characteristic polynomial** of $\boldsymbol{\Sigma}$:

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

Example (continued): Computation of Eigenvalues

The characteristic polynomial is given by

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

Therefore, $\lambda_1 = 1$ and $\lambda_2 = 3$.

Finding the Eigenvectors

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

$$\boldsymbol{\Sigma u} = \lambda_j \boldsymbol{u} \iff \boldsymbol{\Sigma u} - \lambda_j \boldsymbol{u} = \mathbf{0}$$
$$\iff (\boldsymbol{\Sigma} - \lambda_j \boldsymbol{I}_m)\boldsymbol{u} = \mathbf{0}$$

- We have to find the solutions to the **homogeneous system of linear equations** (see ⇒ here)

$$\boldsymbol{Au} = \mathbf{0}$$

where $\boldsymbol{A} := \boldsymbol{\Sigma} - \lambda_j \boldsymbol{I}_m$

Example (continued): Computation of Eigenvectors

- We compute the eigenvectors for eigenvalue $\lambda_1 = 1$:

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

Therefore, the set of eigenvectors for eigenvalue $\lambda_1 = 1$ is given by

$$\left\{ t \cdot \begin{pmatrix} -1 & -1 \end{pmatrix}^{\intercal} : t \in \mathbb{R} \backslash \{0\} \right\}$$

- Similarly, we obtain $\left\{ t \cdot \begin{pmatrix} 1 & -1 \end{pmatrix}^{\intercal} : t \in \mathbb{R} \backslash \{0\} \right\}$ for eigenvalue $\lambda_2 = 3$

The Eigendecomposition of $\boldsymbol{\Sigma}$

- 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

$$\boldsymbol{\Sigma} = \boldsymbol{U} \boldsymbol{\Lambda} \boldsymbol{U}^\intercal$$

  where $\boldsymbol{U}, \boldsymbol{\Lambda} \in \mathbb{R}^{m \times m}$

- $\boldsymbol{U}$ is obtained by stacking the **normalized** eigenvectors column-wise:

$$\boldsymbol{U} := \begin{pmatrix} | & | & & | \\ \widetilde{\boldsymbol{u}}_1 & \widetilde{\boldsymbol{u}}_2 & \ldots & \widetilde{\boldsymbol{u}}_m \\ | & | & & | \end{pmatrix} \in \mathbb{R}^{m \times m} \tag{7}$$

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

- $\boldsymbol{\Lambda} := \text{diag}(\lambda_1, \dots, \lambda_m)$ is a **diagonal matrix** which contains the eigenvalues on its main diagonal:

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

- **Important:** The order of the eigenvalues and eigenvectors in the matrices is crucial! (If you put the eigenvector $\widetilde{\boldsymbol{u}}_j$ into the j-th column of $\boldsymbol{U}$, then you have to put the eigenvalue λ_j into the j-th column of $\boldsymbol{\Lambda}$)

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

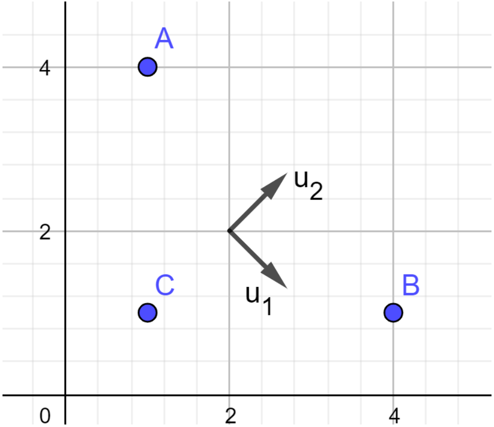
- For $\lambda_1 = 1$ we choose the eigenvector $\boldsymbol{u}_1 = \begin{pmatrix} 1 & 1 \end{pmatrix}^\intercal$ and normalize it. We get

$$\tilde{\boldsymbol{u}}_1 := \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \end{pmatrix}^\intercal$$

- Similarly, we obtain $\tilde{\boldsymbol{u}}_2 := \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & -1 \end{pmatrix}^\intercal$
- Finally we get the **eigendecomposition** of $\boldsymbol{\Sigma}$:

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

Example: Principal Components



Choice of k: 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 k: Strategy 1 (Ctd.)

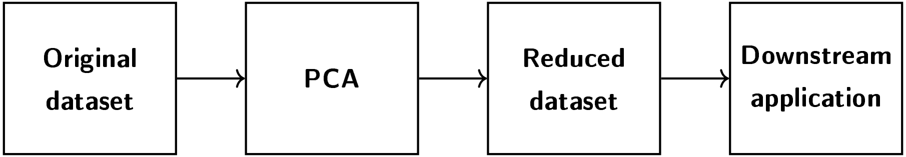
- Sort the eigenvalues in descending order: $(\lambda_1^\star, \lambda_2^\star, \dots, \lambda_m^\star)$
- $\lambda_1^\star$ is the largest eigenvalue, $\lambda_2^\star$ the second-largest, etc.
- Choose the smallest k which satisfies the inequality:

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

- γ specifies the fraction of variance to be retained overall *(this is a hyper-parameter of the algorithm)*

Choice of k: Strategy 2

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



Projection of the Data

- We have constructed the matrix $\widetilde{\boldsymbol{U}}$ (containing only the eigenvectors connected to the k largest eigenvalues) which is given by

$$\widetilde{\boldsymbol{U}} := \begin{pmatrix} | & | & & | \\ \widetilde{\boldsymbol{u}}_1 & \widetilde{\boldsymbol{u}}_2 & \dots & \widetilde{\boldsymbol{u}}_k \\ | & | & & | \end{pmatrix} \in \mathbb{R}^{m \times k} \tag{9}$$

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

$$\mathbb{R}^{n \times k} \ni \boldsymbol{Z} := \boldsymbol{X} \widetilde{\boldsymbol{U}} \tag{10}$$

Example (continued): Projection of the Data

We choose to reduce the dataset $\mathcal{D}$ to one dimension. We select the principal component $\widetilde{\boldsymbol{u}}_2 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & -1 \end{pmatrix}^\intercal$ connected to the eigenvalue $\lambda_2 = 3$. $\widetilde{\boldsymbol{U}}$ is therefore given by:

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

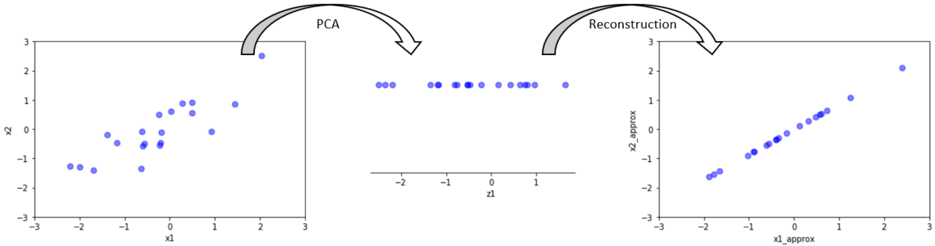
The projection $\boldsymbol{Z} \in \mathbb{R}^{n \times k}$ is given by

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

Reconstruction from compressed Representation

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

$$\boldsymbol{X}_{\approx} := \boldsymbol{Z}\widetilde{\boldsymbol{U}}^{\intercal} \tag{11}$$



Example (continued): Projection of the Data

The reconstructed data is given by

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

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

Section:

Further PCA Applications

Eigenfaces

Face Morphing

Application of PCA to Images: Eigenfaces

Figure: 100 images of faces

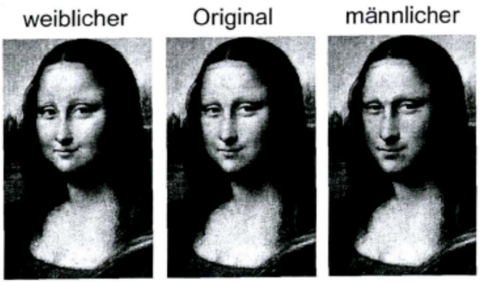
Figure: First 36 principal components

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?