

*** Applied Machine Learning Fundamentals ***

Principal Component Analysis

Daniel Wehner, M.Sc.

SAP SE / DHBW Mannheim

Winter term 2021/2022



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	Probability Density Estimation
Unit V	Regression
Unit VI	Classification I
Unit VII	Evaluation
Unit VIII	Classification II
Unit IX	Clustering
Unit X	Dimensionality Reduction

Agenda for this Unit

1 Introduction

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

2 Maximum Variance Formulation

- Example
- Formalization of the Problem

3 PCA Algorithm

- The Algorithm
- Example

- Data Reconstruction
- Choice of k

4 PCA Applications

- Eigenfaces
- Face Morphing

5 Wrap-Up

- Summary
- Self-Test Questions
- Lecture Outlook
- Recommended Literature and further Reading
- Meme of the Day

Section:
Introduction



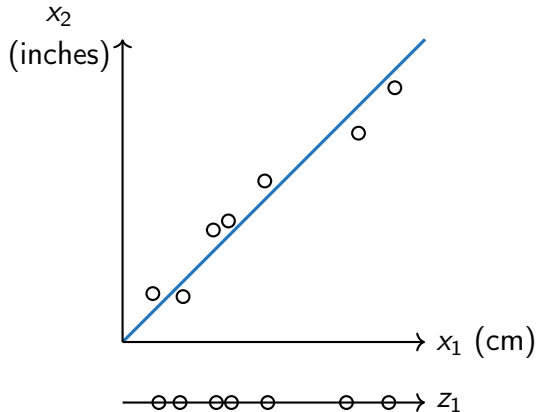
Why Dimensionality Reduction?

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

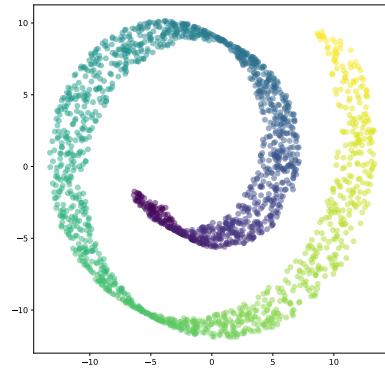
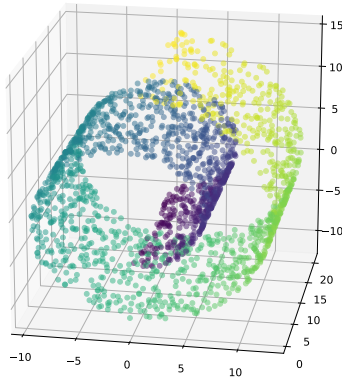
Dimensionality reduction can help to **speed up** learning algorithms substantially. Too many (correlated) features usually **decrease the performance** of the learning algorithm (cf. **curse of dimensionality**).

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 ($\mathbb{R}^2 \rightarrow \mathbb{R}$)



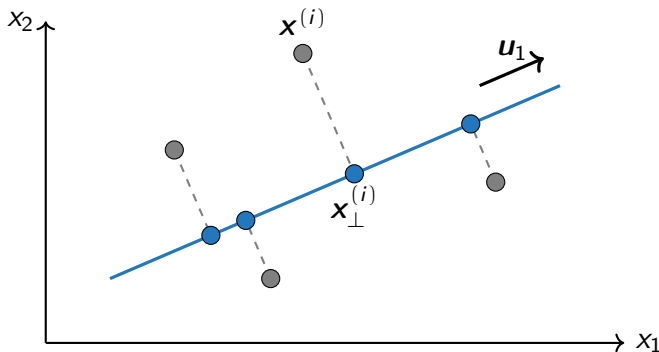
Use Case II: Data Visualization



PCA: Principal Component Analysis

- PCA is an **unsupervised** algorithm
- It is known as the *Karhunen-Loève* transform
- PCA can be defined as the **orthogonal projection** of the data onto a lower dimensional **linear space** (*principal subspace*)
- Consider a data set of n observations $\mathbf{X} = \{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(n)}\}$
 - $\mathbf{x}^{(i)}$ is a real-valued vector in \mathbb{R}^m (m -dimensional)
 - We want to project the data onto a space having dimensionality $k \ll m$, while **maximizing the variance of the projected data** ($\mathbb{R}^m \rightarrow \mathbb{R}^k$)
- **Remove dimensions which are the least informative of the data**

Orthogonal Projections

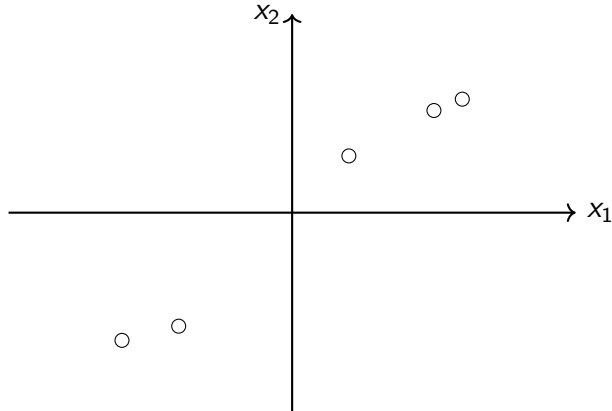


- $\mathbf{x}^{(i)}$ denote the original data points
- $\mathbf{x}_{\perp}^{(i)}$ is the orthogonal projection of $\mathbf{x}^{(i)}$ onto vector \mathbf{u}_1

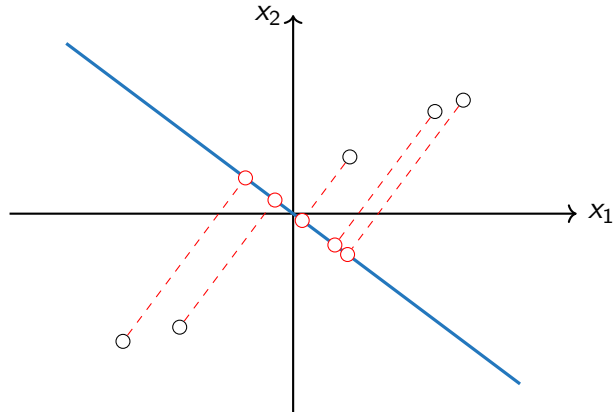
Section:
Maximum Variance Formulation



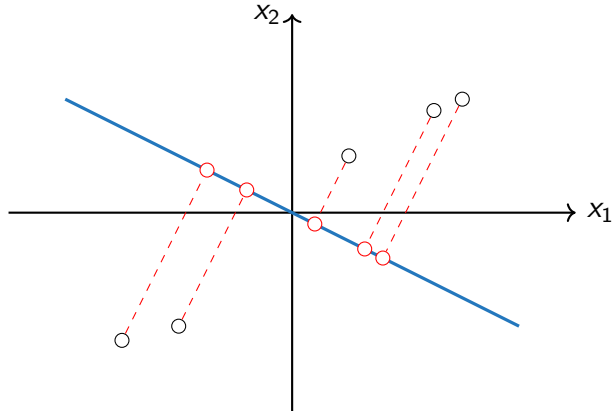
Maximum Variance Formulation



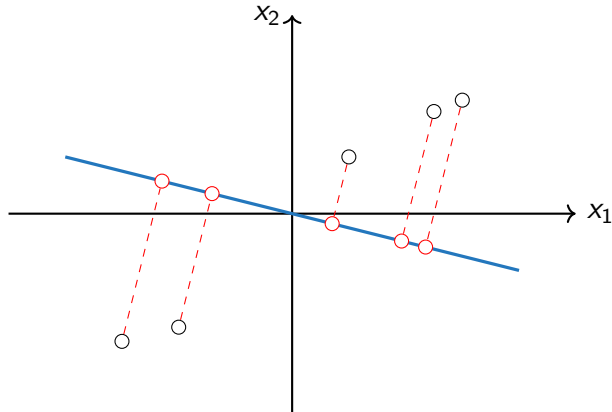
Maximum Variance Formulation



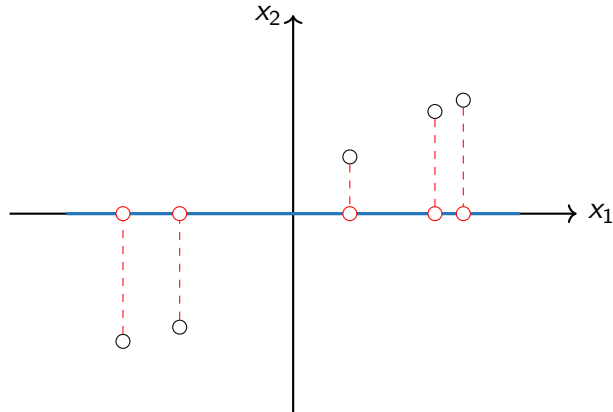
Maximum Variance Formulation



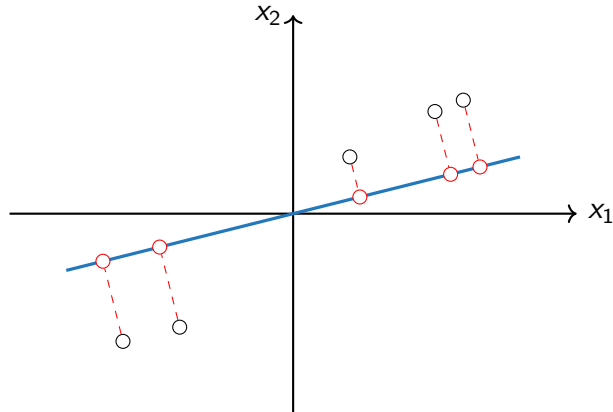
Maximum Variance Formulation



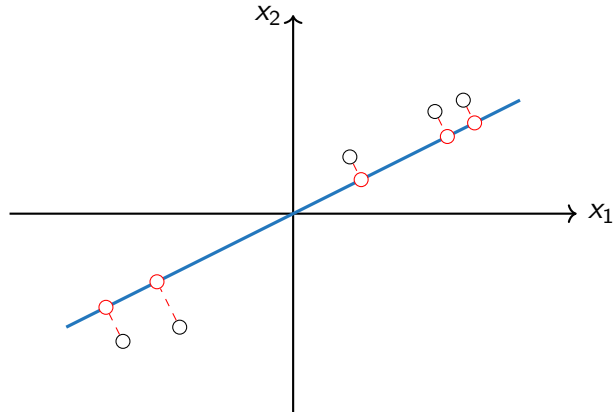
Maximum Variance Formulation



Maximum Variance Formulation



Maximum Variance Formulation



Maximum Variance Formulation (Ctd.)

- In the following we assume $k = 1$ (projection onto a line defined by a unit vector \mathbf{u}_1)
- Each data point $\mathbf{x}^{(i)}$ is projected onto a scalar value $\mathbf{u}_1^\top \mathbf{x}^{(i)}$
- The mean of the projected data is $\mathbf{u}_1^\top \bar{\mathbf{x}}$, where $\bar{\mathbf{x}}$ is the sample set mean:

$$\bar{\mathbf{x}} = \frac{1}{n} \sum_{i=1}^n \mathbf{x}^{(i)} \quad (1)$$

- The variance of the projected data is given by:

$$\frac{1}{n} \sum_{i=1}^n (\mathbf{u}_1^\top \mathbf{x}^{(i)} - \mathbf{u}_1^\top \bar{\mathbf{x}})^2 = \mathbf{u}_1^\top \Sigma \mathbf{u}_1 \quad (2)$$

Maximum Variance Formulation (Ctd.)

- Σ is the covariance matrix defined by:

$$\Sigma = \frac{1}{n} \sum_{i=1}^n \overbrace{(\mathbf{x}^{(i)} - \bar{\mathbf{x}})(\mathbf{x}^{(i)} - \bar{\mathbf{x}})^{\top}}^{\text{Outer product} \rightarrow \text{matrix}} \quad (3)$$

- The projected variance $\mathbf{u}_1^{\top} \Sigma \mathbf{u}_1$ is maximized with respect to \mathbf{u}_1
- Constraint: $\|\mathbf{u}_1\| = 1$, otherwise \mathbf{u}_1 grows unboundedly
- We have to solve the following optimization problem:

$$\max_{\mathbf{u}_1} \{ \mathbf{u}_1^{\top} \Sigma \mathbf{u}_1 + \lambda_1 (1 - \mathbf{u}_1^{\top} \mathbf{u}_1) \} \quad (4)$$



Maximum Variance Formulation (Ctd.)

- $\nabla_{\mathbf{u}_1} \{ \mathbf{u}_1^\top \Sigma \mathbf{u}_1 + \lambda_1 (1 - \mathbf{u}_1^\top \mathbf{u}_1) \} \stackrel{!}{=} 0 \quad \implies \Sigma \mathbf{u}_1 = \lambda_1 \mathbf{u}_1$
- This is an **eigenvalue problem**
- The equation tells us that \mathbf{u}_1 must be an eigenvector of Σ
- If we left-multiply by \mathbf{u}_1^\top and use $\mathbf{u}_1^\top \mathbf{u}_1 = 1$, we see: $\mathbf{u}_1^\top \Sigma \mathbf{u}_1 = \lambda_1$

The variance reaches a maximum, if we set \mathbf{u}_1 equal to the eigenvector having the largest eigenvalue λ_1 . This eigenvector is the first principal component.

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 $\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_k$ of the covariance matrix Σ
 - These correspond to the k largest eigenvalues $\lambda_1, \lambda_2, \dots, \lambda_k$

Section:
PCA Algorithm



Algorithm 1: PCA Algorithm

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

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

- 1 $\bar{\mathbf{x}} \leftarrow \frac{1}{n} \sum_{i=1}^n \mathbf{x}^{(i)}$ // sample set mean
- 2 $\Sigma \leftarrow \frac{1}{n} \sum_{i=1}^n (\mathbf{x}^{(i)} - \bar{\mathbf{x}})(\mathbf{x}^{(i)} - \bar{\mathbf{x}})^\top$ // covariance matrix
- 3 Perform singular value decomposition to find the eigenvectors of matrix Σ :

$$[\mathbf{U}, \mathbf{S}, \mathbf{V}] = \text{SVD}(\Sigma)$$

- 4 Select first k eigenvectors: $\mathbf{U}_k \leftarrow \mathbf{U}_{(:, :k)}$ // eig.vecs with largest eig.vals.
 - 5 $\mathbf{Z} \leftarrow \mathbf{U}_k^\top \mathbf{X}$
-

Projection of the Data

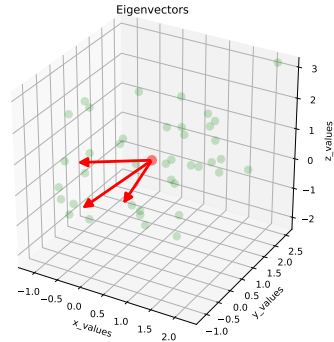
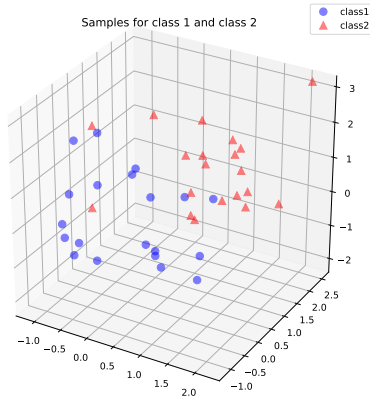
- Matrix \mathbf{U} is obtained by applying **singular value decomposition** to Σ

$$\mathbf{U} = \begin{bmatrix} | & | & \dots & | \\ \mathbf{u}_1 & \mathbf{u}_2 & \dots & \mathbf{u}_m \\ | & | & & | \end{bmatrix} \in \mathbb{R}^{m \times m} \quad (5)$$

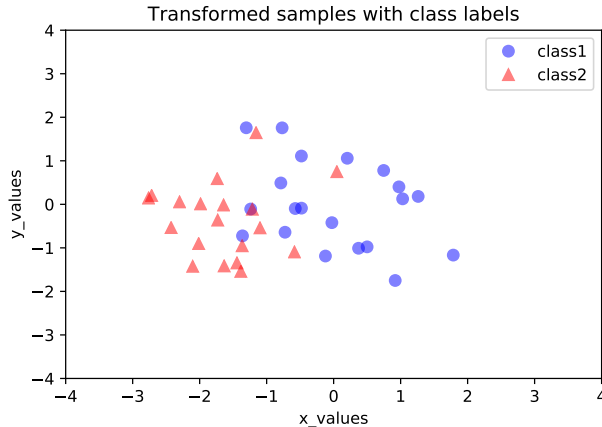
- The projection $\mathbb{R}^m \rightarrow \mathbb{R}^k (k \ll m)$ is performed as follows:

$$\begin{bmatrix} z_1^{(i)} \\ \vdots \\ z_k^{(i)} \end{bmatrix} = \begin{bmatrix} | & | & \dots & | \\ \mathbf{u}_1 & \mathbf{u}_2 & \dots & \mathbf{u}_k \\ | & | & & | \end{bmatrix}^T \begin{bmatrix} x_1^{(i)} \\ \vdots \\ x_m^{(i)} \end{bmatrix} \quad (6)$$

PCA Result



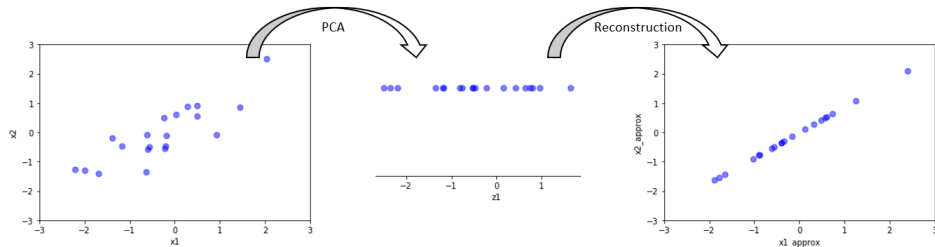
PCA Result (Ctd.)



Reconstruction from compressed Representation

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

$$\mathbf{x}_{\approx}^{(i)} = \mathbf{U}_k \mathbf{z}^{(i)} \quad (7)$$



Choosing the Number of Components

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

$$\frac{1}{n} \sum_{i=1}^n \| \mathbf{x}^{(i)} - \mathbf{x}_{\approx}^{(i)} \|^2 \quad (8)$$

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

$$\frac{1}{n} \sum_{i=1}^n \| \mathbf{x}^{(i)} \|^2 \quad (9)$$

Choosing the Number of Components (Ctd.)

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

$$\frac{\overbrace{\frac{1}{n} \sum_{i=1}^n \|\mathbf{x}^{(i)} - \mathbf{x}_{\approx}^{(i)}\|^2}^{\text{average projection error}}}{\underbrace{\frac{1}{n} \sum_{i=1}^n \|\mathbf{x}^{(i)}\|^2}_{\text{total variation}}} \leq \gamma \quad (10)$$

- 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:

$$[\mathbf{U}, \mathbf{S}, \mathbf{V}] = \text{SVD}(\mathbf{\Sigma}) \quad (11)$$

- We can use the $(m \times m)$ -matrix \mathbf{S} (eigenvalues on the main diagonal):

$$\mathbf{S} = \begin{bmatrix} S_{11} & 0 & \dots & 0 \\ 0 & S_{22} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & S_{mm} \end{bmatrix} \quad (12)$$



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

- For a given k , the fraction of variance retained can be computed as follows:

$$1 - \frac{\sum_{j=1}^k S_{jj}}{\sum_{j=1}^m S_{jj}} \leq 1 - \gamma \quad (13)$$

- The matrix has to be computed only once and can be reused for all k

Simplification:

$$\frac{\sum_{j=1}^k S_{jj}}{\sum_{j=1}^m S_{jj}} \geq \gamma$$

Section:
PCA Applications



Application of PCA to Images: Eigenfaces



Figure: 100 images of faces



Figure: First 36 principal components

Application of PCA to Images: Eigenfaces (Ctd.)

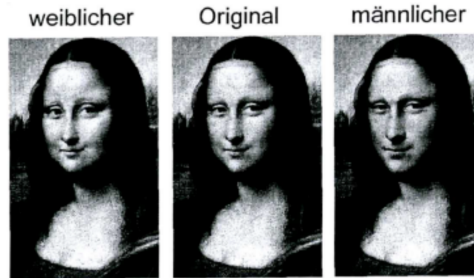


Figure: Original images



Figure: Reconstructed images

Application of PCA to Images: Face Morphing



Section:
Wrap-Up



Summary

- 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 where PCA is useful!

What's next...?



The Exam



Just kidding... (maybe)

Recommended Literature and further Reading I



[1] Pattern Recognition and Machine Learning

Christopher Bishop. Springer. 2006.

→ [Link](#), cf. chapter 12.1



[2] Machine Learning: A Probabilistic Perspective

Kevin Murphy. MIT Press. 2012.

→ [Link](#), cf. chapter 12.2

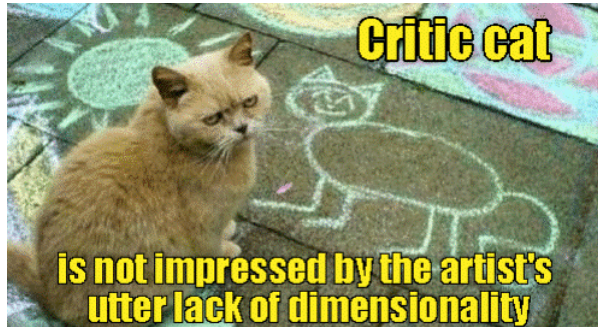


[3] Mathematics for Machine Learning

Deisenroth et al. Cambridge University Press. 2019.

→ [Link](#), cf. chapter 10

Meme of the Day



Thank you very much for the attention!

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

Term: Winter term 2021/2022

Contact:

Daniel Wehner, M.Sc.

SAP SE / DHBW Mannheim

daniel.wehner@sap.com

Do you have any questions?