Chapter 2 - Unsuperised Learning

Dimensionality Reduction and Clustering

Maschinelles Lernen 1 -
Grundverfahren WS19/20

Prof. Gerhard Neumann
KIT, Institut für Anthrophomatik und Robotik

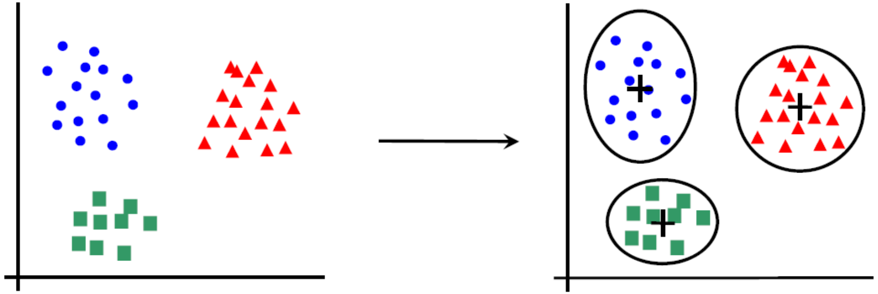
Wrap-Up for Chapter 1: “Simple” Supervised Learning

Algorithm	Reg / Class	Representation	Optimization	Loss

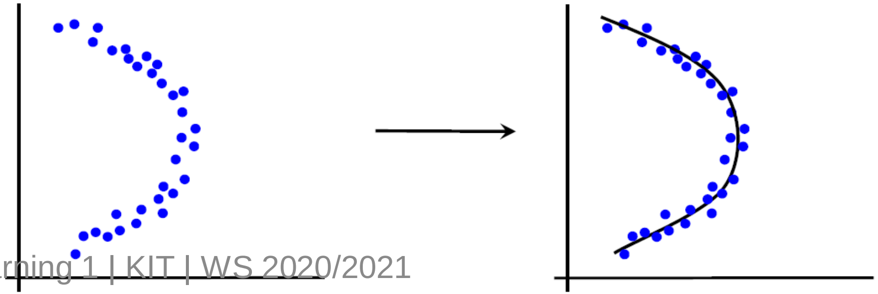
Unsupervised Learning

Trainings data does not include target values

- **Density estimation:** Model the data
- **Clustering:**



- **Dimensionality reduction:**



Learning Outcomes

- Understand what dimensionality reduction means and why do use it
- Understand what we mean with a “projection” of a vector
- What makes a dimensionality reduction a “good” reduction
- What are the principal components in the data and what is the relation to the covariance matrix
- Learn about constraint convex optimization

Today‘s Agenda!

Dimensionality Reduction:

- Linear Dimensionality Reduction
- Linear Orthogonal Projections
- Reproduction Error
- Principal Component Analysis

Basics: Convex Constraint Optimization

- Lagrangian Multipliers and Constraint Optimization
- Dual Optimization Problem

Slides are largely based on
Slides from Jan Peters

Dimensionality Reduction

Supervised Learning:

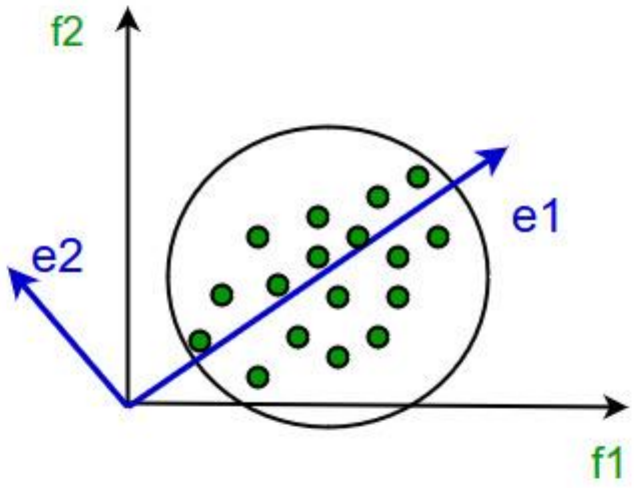
- Learn a mapping from input x to output y

Sometimes, it is quite helpful to analyze the data points themselves

- Unsupervised learning
- Particularly:
  - Reduce the dimensionality of the data

Possible application:

- Visualization of the data
- Preprocessing for any learning algorithm



Motivation from Linear Least-squares Regression

- In least-squares linear regression the parameters are computed as

$$\boldsymbol{w} = (\boldsymbol{X}^T \boldsymbol{X})^{-1} \boldsymbol{X}^T \boldsymbol{y}$$

where $\boldsymbol{X} \in \mathbb{R}^{N \times d}$ and $\boldsymbol{y} \in \mathbb{R}^{n \times 1}$

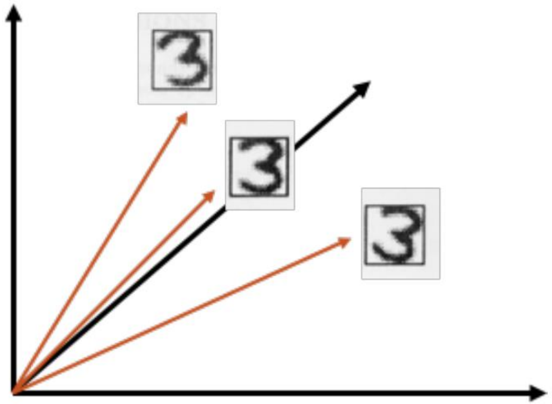
- We need to invert a d × d matrix, which naively costs $O(d^3)$

- Hence, it would be helpful to find a new d_{new} << d to gain computational advantage while not loosing prediction performance

Dimensionality Reduction

- How can we find more efficient representations for our data?
- How can we capture the “essence” of the data?

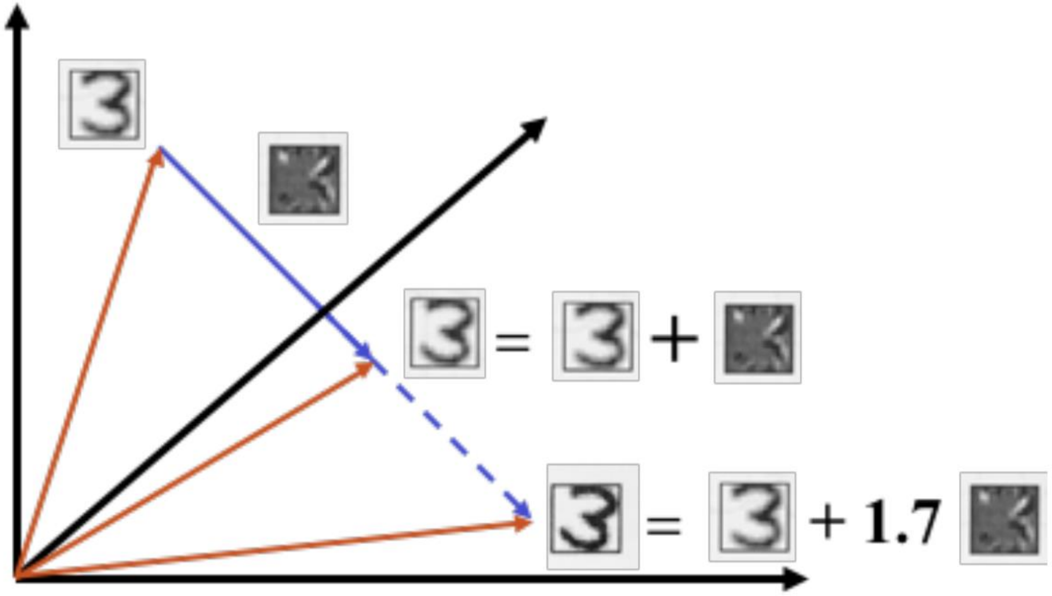
Example: images of the digit 3



- The images can be represented as points in a high-dimensional space (e.g., with one dimension per pixel, in a 4k image there are around 9 million dimensions!)

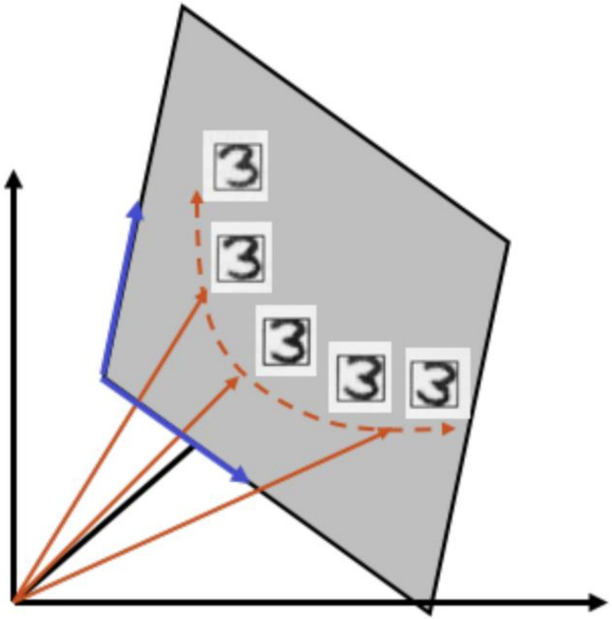
Linear Dimensionality Reduction

To make things easier, we will once again assume linear models. A data point (here: one image) can be written as a linear combination of bases (here: basis images)



Linear Dimensionality Reduction

- What linear transformations of the data can be used to define a lower-dimensional subspace that captures most of the structure?



Linear Dimensionality Reduction

Problem definition:

- Original data point i: $\boldsymbol{x}_i \in \mathbb{R}^D$
- Low-dimensional representation of data point i: $\boldsymbol{z}_i \in \mathbb{R}^M$ with D << M
- **Goal:** find a mapping

$$\boldsymbol{x}_i \rightarrow \boldsymbol{z}_i$$

- Restrict this mapping to be a linear function

$$\boldsymbol{z}_i = \boldsymbol{W}\boldsymbol{x}_i, \text{ with } \boldsymbol{W} \in \mathbb{R}^{M \times D}$$

Orthonormal Basis Vectors

We can always write a vector in terms of an orthonormal basis coordinate system

$$\boldsymbol{x} = \sum_{i=1}^{D} z_i \boldsymbol{u}_i, \text{ where } \boldsymbol{u}_i^T \boldsymbol{u}_j = \delta_{ij} \text{ and } \delta_{ij} = 1 \text{ if } i = j, \ 0 \text{ otherwise}$$

- **Orthonormality condition:** The product of 2 different basis vectors is 0. The norm of each basis vector is 1.

Example:

$$\begin{bmatrix} 3 \\ 7 \end{bmatrix} = 3 \begin{bmatrix} 1 \\ 0 \end{bmatrix} + 7 \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Projections

The coefficients z_i can be obtained by projecting **x** on the basis vector u_i

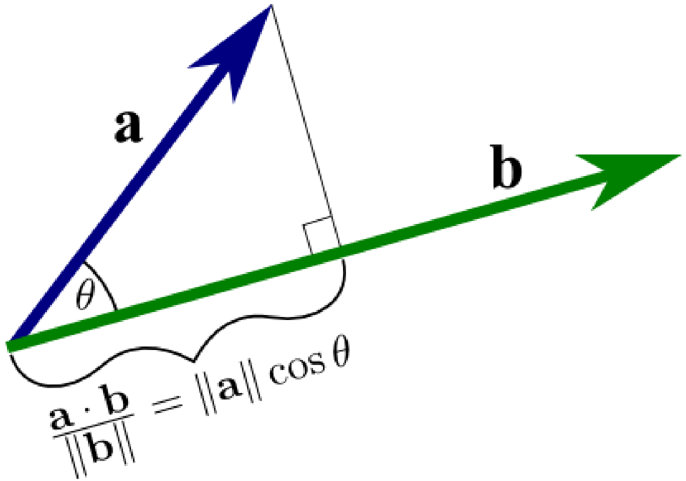
$$\underbrace{z_i}_{\text{scalar coefficient}} = \underbrace{\boldsymbol{u}_i^T \boldsymbol{x}}_{\text{projection}}$$

Example:

$$\boldsymbol{x} = z_1 \boldsymbol{u}_1 + z_2 \boldsymbol{u}_2$$

$$\boldsymbol{u}_1^T \boldsymbol{x} = z_1 \underbrace{\boldsymbol{u}_1^T \boldsymbol{u}_1}_{=1} + z_2 \underbrace{\boldsymbol{u}_2^T \boldsymbol{u}_2}_{=0} = z_1$$

Projection of 2 vectors



Decomposition

Use M << D basis vectors:

$$\boldsymbol{x} = \underbrace{\sum_{i=1}^{M} z_i \boldsymbol{u}_i}_{\tilde{\boldsymbol{x}} \approx \boldsymbol{x}} + \underbrace{\sum_{j=M+1}^{D} z_j \boldsymbol{u}_j}_{\text{skip}}$$

Find the M basis vectors u_i that minimize the mean squared reproduction error:

$$\arg\min_{\boldsymbol{u}_1,\ldots,\boldsymbol{u}_M} E(\boldsymbol{u}_1, \ldots, \boldsymbol{u}_M) = \arg\min_{\boldsymbol{u}_1,\ldots,\boldsymbol{u}_M} \sum_{i=1}^{N} ||\boldsymbol{x}_i - \tilde{\boldsymbol{x}}_i||^2$$

Minimizing the error

Assuming a single basis vector, the error can be written as

$$E(\boldsymbol{u}_1) = \sum_{i=1}^{N} ||\boldsymbol{x}_i - \tilde{\boldsymbol{x}}_i||^2 = \sum_{i=1}^{N} ||\boldsymbol{x}_i - \underbrace{(\boldsymbol{u}_1^T \boldsymbol{x}_i)}_{z_{i1}} \boldsymbol{u}_1||^2$$

$$= \sum_{i=1}^{N} \boldsymbol{x}_i^T \boldsymbol{x}_i - 2(\boldsymbol{u}_1^T \boldsymbol{x}_i)^2 + (\boldsymbol{u}_1^T \boldsymbol{x}_i)^2 \boldsymbol{u}_1^T \boldsymbol{u}_1 = \sum_{i=1}^{N} \boldsymbol{x}_i^T \boldsymbol{x}_i - (\boldsymbol{u}_1^T \boldsymbol{x}_i)^2$$

$$= \sum_{i=1}^{N} \boldsymbol{x}_i^T \boldsymbol{x}_i - z_{i1}^2$$

Minimizing the error

The error can be written as

$$E(\boldsymbol{u}_1) = \sum_{i=1}^{N} \boldsymbol{x}_i^T \boldsymbol{x}_i - z_{i1}^2$$

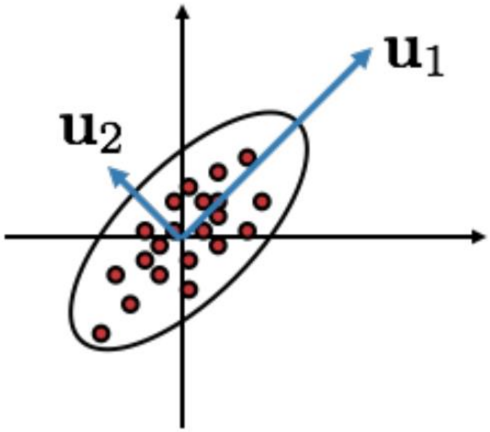
$$\Rightarrow \arg\min_{\boldsymbol{u}_1} E(\boldsymbol{u}_1) = \arg\max_{\boldsymbol{u}_1} \sum_{i=1}^{N} z_{i1}^2 = \arg\max_{\boldsymbol{u}_1} \sum_{i=1}^{N} (\boldsymbol{u}_1^T \boldsymbol{x}_i)^2$$

- Minimizing the error is equivalent to maximizing the variance of the projection. (Assuming a zero mean on the data)
- We can ensure a zero mean projection by subtracting the mean from the data

$$\bar{\boldsymbol{x}}_i = \boldsymbol{x}_i - \boldsymbol{\mu}$$

Illustration

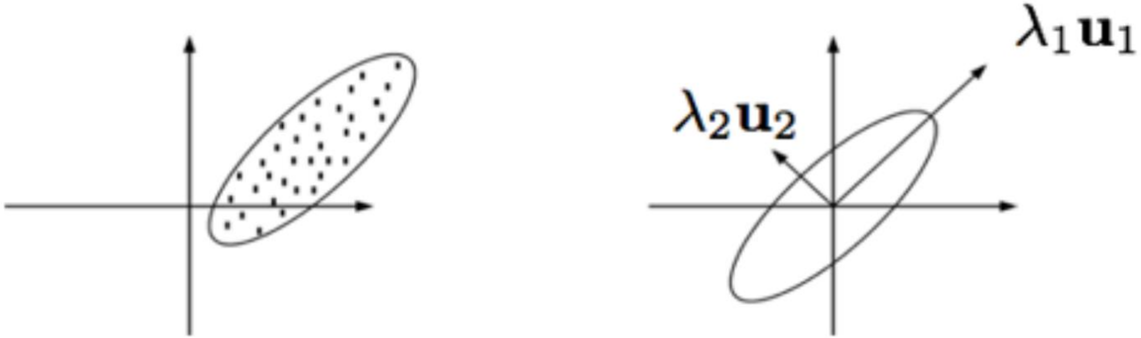
$$\tilde{\boldsymbol{x}} = \sum_{i=1}^{M} z_i \boldsymbol{u}_i + \boldsymbol{\mu}$$



- Projecting onto $\mathbf{u_1}$ captures the majority of the variance and hence projecting onto it minimizes the error
- Note that these axes are orthogonal and decorrelate the data
  - i.e. in the coordinate frame of these axes, the data is uncorrelated (side note: this only works for Gaussians)

Principle component analysis (PCA)

Goal: find the so-called principal directions, and the variance of the data along each principal direction



- λ_i is the marginal variance along the principal direction $\boldsymbol{u}_i$

Principle component analysis

- The first principal direction $\mathbf{u_1}$ is the direction along which the variance of the projected data is maximal

$$\boldsymbol{u}_1 = \arg\max_{\boldsymbol{u}} \frac{1}{N} \sum_{i=1}^{N} \Big(\boldsymbol{u}^T \underbrace{(\boldsymbol{x}_i - \boldsymbol{\mu})}_{\bar{\boldsymbol{x}}_i}\Big)^2 \quad \text{s.t. } \boldsymbol{u}^T \boldsymbol{u} = 1$$

  - The directions all have unit norm

- The second principal direction maximizes the variance of the data in the orthogonal complement of the first principal direction

Derivation…

- **Objective in matrix form…**

$$E(\boldsymbol{u}) = \frac{1}{N}\sum_{i=1}^{N}\left(\boldsymbol{u}^T(\boldsymbol{x}_i - \boldsymbol{\mu})\right)^2$$

$$= \frac{1}{N}\sum_{i=1}^{N}\left(\boldsymbol{u}^T(\boldsymbol{x}_i - \boldsymbol{\mu})(\boldsymbol{x}_i - \boldsymbol{\mu})^T\boldsymbol{u}\right)$$

$$= \boldsymbol{u}^T \underbrace{\left(\frac{1}{N}\sum_{i=1}^{N}(\boldsymbol{x}_i - \boldsymbol{\mu})(\boldsymbol{x}_i - \boldsymbol{\mu})^T\right)}_{\text{covariance } \boldsymbol{\Sigma}} \boldsymbol{u} = \boldsymbol{u}^T\boldsymbol{\Sigma}\boldsymbol{u}$$

  - The objective can be written in terms of the sample covariance!

Derivation…

We obtain the following constrained optimization problem

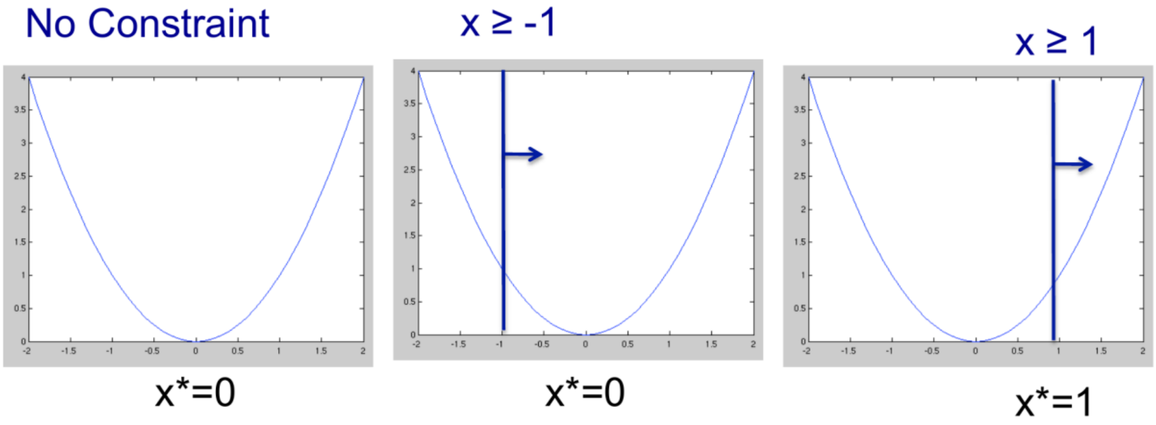
$$\boldsymbol{u}_1 = \arg\max_{\boldsymbol{u}} \ \boldsymbol{u}^T \boldsymbol{\Sigma} \boldsymbol{u} \quad \text{s.t. } \boldsymbol{u}^T \boldsymbol{u} = 1$$

We need to look at constraint optimization first!

Constraint Optimization

Basics: Constrained Optimization

Simple constrained optimization problem: $\arg\min_x x^2 \quad \text{s.t.} \quad x \geq b$



How do we solve the constrained optimization problem? **Lagrangian Multipliers!**

Basics: Lagrangian Multipliers

$$\min_x x^2 \quad \text{s.t.} \;\; x \geq b$$

The Lagrangian:

- L = objective - multiplier * constraint

$$L(x, \alpha) = \underbrace{x^2}_{\text{objective}} - \underbrace{\alpha}_{\text{multiplier}} \cdot \underbrace{(x - b)}_{\text{constraint}}$$

Lagrangian optimization:

$$\min_x \max_\alpha L(x, \alpha), \quad \text{s.t.} \;\; \alpha \geq 0$$

Why is this equivalent?

***Min* fights *max*!**

- $x < b:$
  - $(x - b) < 0 \rightarrow \max_\alpha -\alpha(x - b) = \infty$
  - *min* won't let that happen
- $x > b:$
  - $(x - b) > 0, \alpha \geq 0 \rightarrow \alpha^* = 0$
  - L is the same as original objective
- $x = b:$
  - α can be anything
  - L is the same as original objective

***Min* forces *max* to behave such that constraints are satisfied**

General formulation

General Formulation:
$$\min_{\boldsymbol{x}} f(\boldsymbol{x}),$$
$$\text{s.t.} \quad h_i(\boldsymbol{x}) \geq b_i, \;\; \text{for } i = 1 \dots K$$

- Several inequality constraints (equality constraints also possible)

Lagrangian optimization:
$$\min_{\boldsymbol{x}} \max_{\boldsymbol{\lambda}} L(\boldsymbol{x}, \boldsymbol{\lambda}), \qquad L(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) - \sum_{i=1}^{K} \lambda_i \big(h_i(\boldsymbol{x}) - b_i\big)$$
$$\text{s.t. } \lambda_i \geq 0, \;\; \text{for } i = 1 \dots K$$

Dual formulation

Primal optimization problem:

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}),$$
$$\text{s.t.} \quad h_i(\boldsymbol{x}) \geq b_i, \text{ for } i = 1 \dots K$$

Dual optimization problem:

$$\boldsymbol{\lambda}^* = \arg\max_{\boldsymbol{\lambda}} g(\boldsymbol{\lambda}), \quad g(\boldsymbol{\lambda}) = \min_{\boldsymbol{x}} L(\boldsymbol{x}, \boldsymbol{\lambda})$$
$$\text{s.t.} \quad \lambda_i \geq 0, \text{ for } i = 1 \dots K$$

- *g* is also called the dual function of the optimization problem
- We essentially swapped min and max in the definition of L

Slaters condition: For a convex objective and convex constraints, solving the dual is equivalent to solving the primal!

- Optimal primal parameters can be obtained from optimal dual parameters, i.e.

$$\boldsymbol{x}^* = \arg\min_{\boldsymbol{x}} L(\boldsymbol{x}, \boldsymbol{\lambda}^*)$$

Example:

$$\min_x x^2 \quad \text{s.t.} \quad x \geq 1$$

Back to the PCA Derivation…

We obtain the following constrained optimization problem

$$\boldsymbol{u}_1 = \arg\max_{\boldsymbol{u}} \ \boldsymbol{u}^T \boldsymbol{\Sigma} \boldsymbol{u} \quad \text{s.t. } \boldsymbol{u}^T \boldsymbol{u} = 1$$

- We now know what to do… Lagrangian optimization

The Lagrangian is given by:

$$L(\boldsymbol{u}, \lambda) = \boldsymbol{u}^T \boldsymbol{\Sigma} \boldsymbol{u} + \lambda(\boldsymbol{u}^T \boldsymbol{u} - 1)$$

- Optimal solution for u:

$$\frac{\partial L(\boldsymbol{u}, \lambda)}{\partial \boldsymbol{u}} = 2\boldsymbol{\Sigma}\boldsymbol{u} + 2\lambda\boldsymbol{u} \stackrel{!}{=} \boldsymbol{0} \quad \Rightarrow \boldsymbol{\Sigma}\boldsymbol{u} = \lambda\boldsymbol{u}$$

This is an Eigen-value problem!

Basics: Eigenvalues and Eigenvectors

- Let the Eigenvectors and Eigenvalues of **C** be $\mathbf{u}_k$ and λ_k for $k \leq D$ i.e.,

$$\boldsymbol{C}\boldsymbol{u}_k = \lambda_k \boldsymbol{u}_k \quad \text{with } \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_D$$

Ordered list of Eigenvalues

- In matrix form:

$$\boldsymbol{C}\boldsymbol{U} = \boldsymbol{U}\boldsymbol{\Lambda} \quad \text{with } \boldsymbol{\Lambda} = \mathrm{diag}(\lambda_1, \ldots, \lambda_D) \text{ and } \boldsymbol{U} = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_D]$$

- Because **U** is orthonormal (eigenvectors have unit norm), we know that $\boldsymbol{U}\boldsymbol{U}^T = \boldsymbol{I}$

- This mean that we can decompose **C** as

$$(\boldsymbol{C}\boldsymbol{U})\boldsymbol{U}^T = (\boldsymbol{U}\boldsymbol{\Lambda})\boldsymbol{U}^T \Rightarrow \boldsymbol{C} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T$$

Basics: Eigenvalues and Eigenvectors

Every positive definite symmetric matrix can be decomposed in its Eigendecomposition

$$\boldsymbol{C} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T = \underbrace{\begin{bmatrix} \boldsymbol{u}_1 & \dots & \boldsymbol{u}_D \end{bmatrix}}_{\text{Eigenvectors}} \underbrace{\begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_D \end{bmatrix}}_{\text{Eigenvalues}} \begin{bmatrix} \boldsymbol{u}_1^T \\ \vdots \\ \boldsymbol{u}_D^T \end{bmatrix}$$

Back to PCA

Eigenvalues-Eigenvectors of the covariance matrix

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

- The largest Eigenvalue gives us the maximal variance
- The corresponding Eigenvector gives us the direction with maximal variance

Principal Component Analysis

- **Observation:** If $\lambda_k \approx 0$ for k > M for some M << D, then we can use the subset of the first D eigenvectors to define a basis for approximating the data vectors with loosing accuracy

$$\boldsymbol{x}_i - \boldsymbol{\mu} = \underbrace{\sum_{j=1}^{M} z_{ij}\boldsymbol{u}_j}_{\tilde{\boldsymbol{x}}} + \underbrace{\sum_{j=M+1}^{D} z_{ij}\boldsymbol{u}_j}_{\text{close to } 0} \Rightarrow \boldsymbol{x}_i \approx \boldsymbol{\mu} + \sum_{j=1}^{M} z_{ij}\boldsymbol{u}_j$$

- This representation has the minimal mean squared error (MSE) of all linear representations of dimension D

$$\arg\min_{\boldsymbol{u}_1,\ldots,\boldsymbol{u}_M} E(\boldsymbol{u}_1,\ldots,\boldsymbol{u}_M) = \arg\min_{\boldsymbol{u}_1,\ldots,\boldsymbol{u}_M} \sum_{i=1}^{N} ||\boldsymbol{x}_i - \tilde{\boldsymbol{x}}_i||^2$$

Principal Component Analysis

Now we know how we can represent our data in a lower dimensional space in a principled way

- Center the data around the mean (compute the mean of the data and subtract it)
- Compute the covariance matrix, decompose it, and choose the first D largest Eigenvalues and corresponding Eigenvectors
- This gives us an (Eigen)basis for representing the data

  – **Projection to low-D:** $\boldsymbol{z}_i = \boldsymbol{B}^T(\boldsymbol{x}_i - \boldsymbol{\mu})$

  – **Reprojection to high-D:** $\tilde{\boldsymbol{x}}_i = \boldsymbol{\mu} + \boldsymbol{B}\boldsymbol{z}_i$

  with $\boldsymbol{B} = \begin{bmatrix} \boldsymbol{u}_1 & \dots & \boldsymbol{u}_M \end{bmatrix}$

- It is also common to normalize the variance of each dimension (i.e. unit variance)

How to choose M

- A larger M leads to a better approximation. In the limit, when M = D we stay in the initial data dimensions
- There are at least 2 good possibilities for choosing D
  - Choose D based on application performance, i.e. choose the smallest D that makes the application work well enough
  - Choose D so that the Eigenbasis captures some fraction of the variance (for example η = 0.9).
    The eigenvalue λ_i describes the marginal variance captured by $\mathbf{u}_i$

$$\text{Choose } D \text{ s.t. } \sum_{i=1}^{M} \lambda_i = \eta \underbrace{\sum_{i=1}^{D} \lambda_i}_{}$$

Total variance of the data

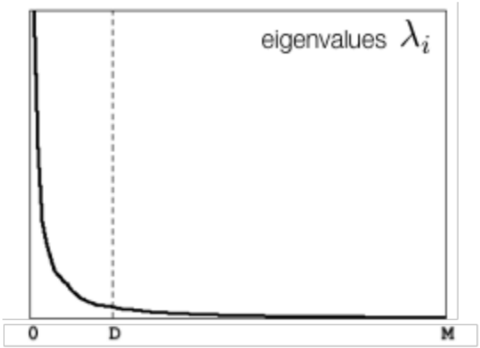


Image representation with PCA

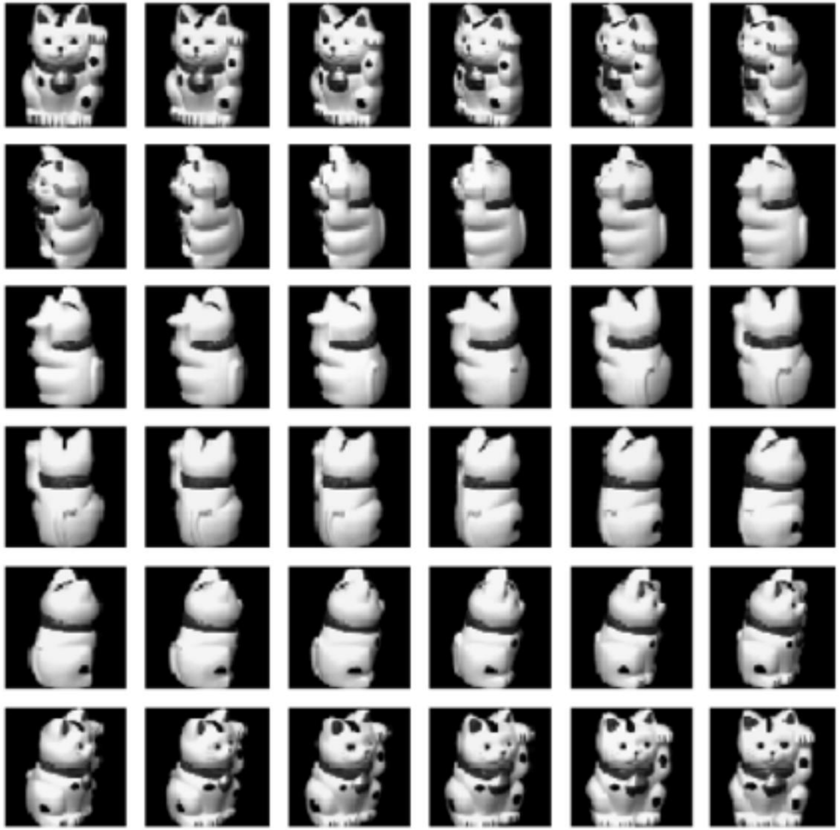
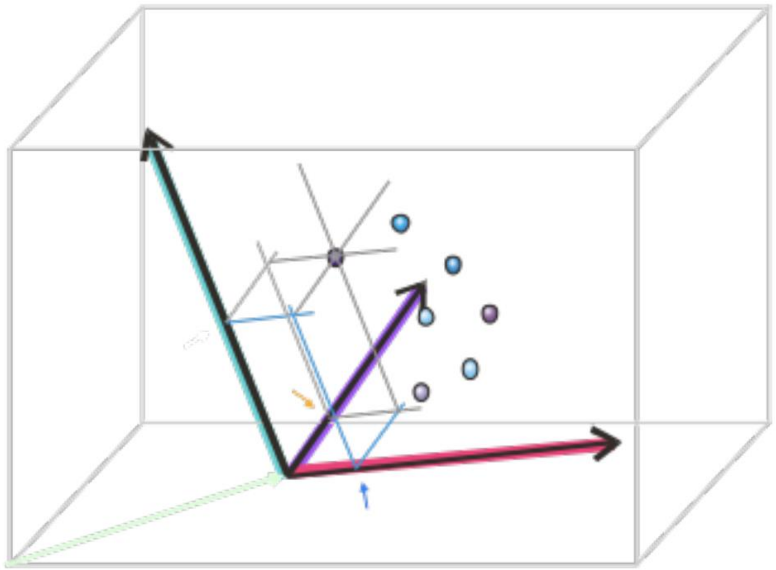
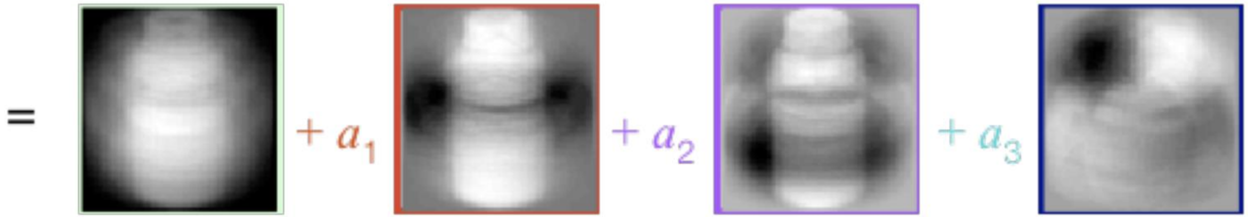


Image representation with PCA





Eigenfaces

- The first popular use of PCA for object recognition was for the detection and recognition of faces [Turk and Pentland, 1991]
- Collect a face ensemble
- Normalize for contrast, scale, & orientation
- Remove backgrounds
- Apply PCA & choose the first D eigen-images that account for most of the variance of the data

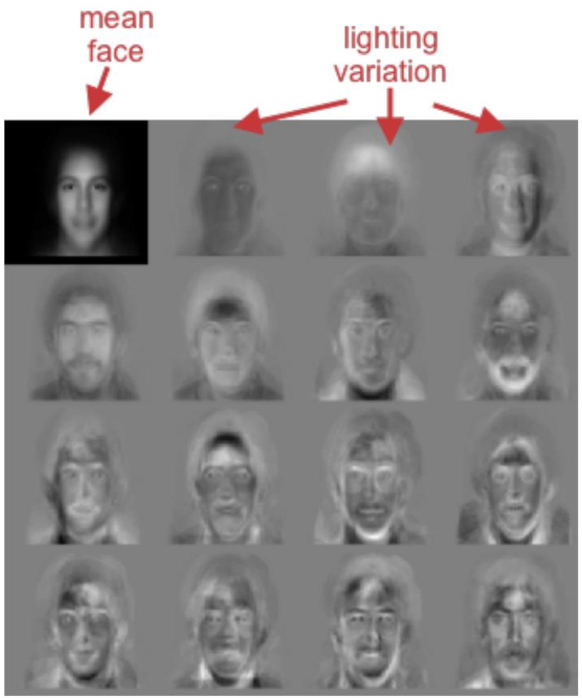
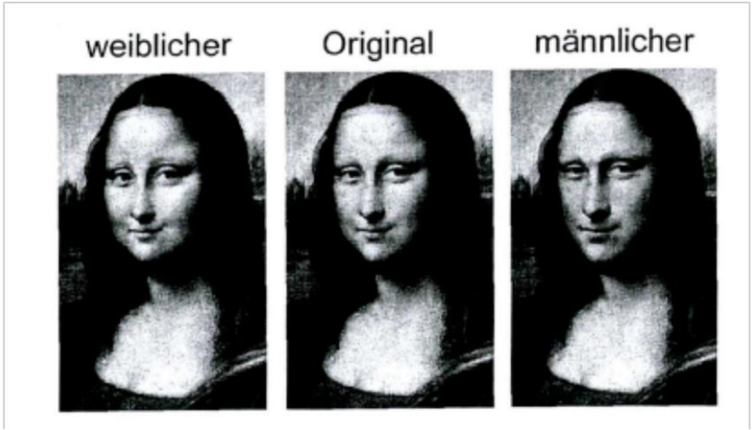


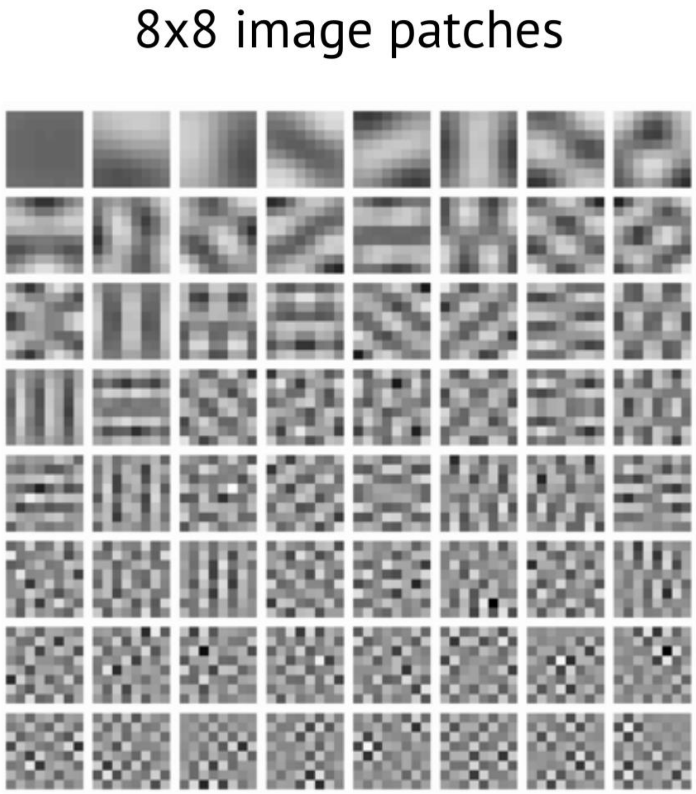
Image Morphing with PCA



Generic Image Ensembles

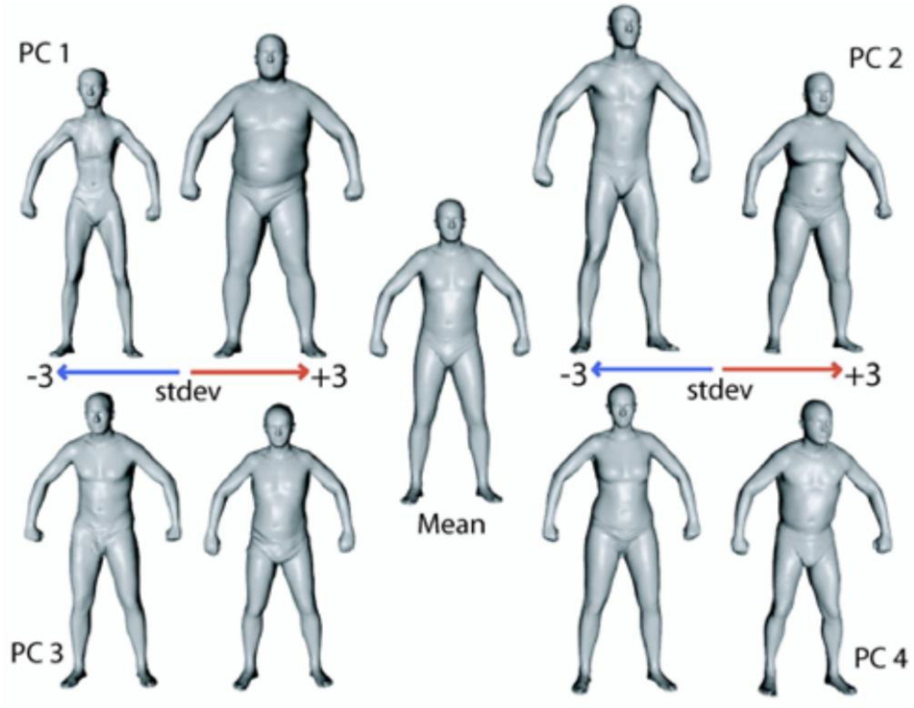
Is there a low-dimensional model describing natural images?

PCA of natural image patches



PCA Model of body shapes

- PCA on a detailed triangle model of human bodies [Anguelov et al. 05]



Wrap-up

Summary:

- PCA projects the data into a linear subspace
- PCA maximizes the variance of the projection
- PCA minimizes the error of the reconstruction

Applications:

- PCA allows us to transform a high-dimensional input space to a low-dimensional feature space, while capturing the essence of the data
- PCA finds a more natural coordinate system for the data
- PCA is a very common preprocessing step for high-dimensional input data

Takeaway messages

What have we learned today?

- What does dimensionality reduction mean?
- What is PCA? What are the three things that it does?
- What are the roles of the Eigenvectors and Eigenvalues in PCA?
- Can you describe applications of PCA?