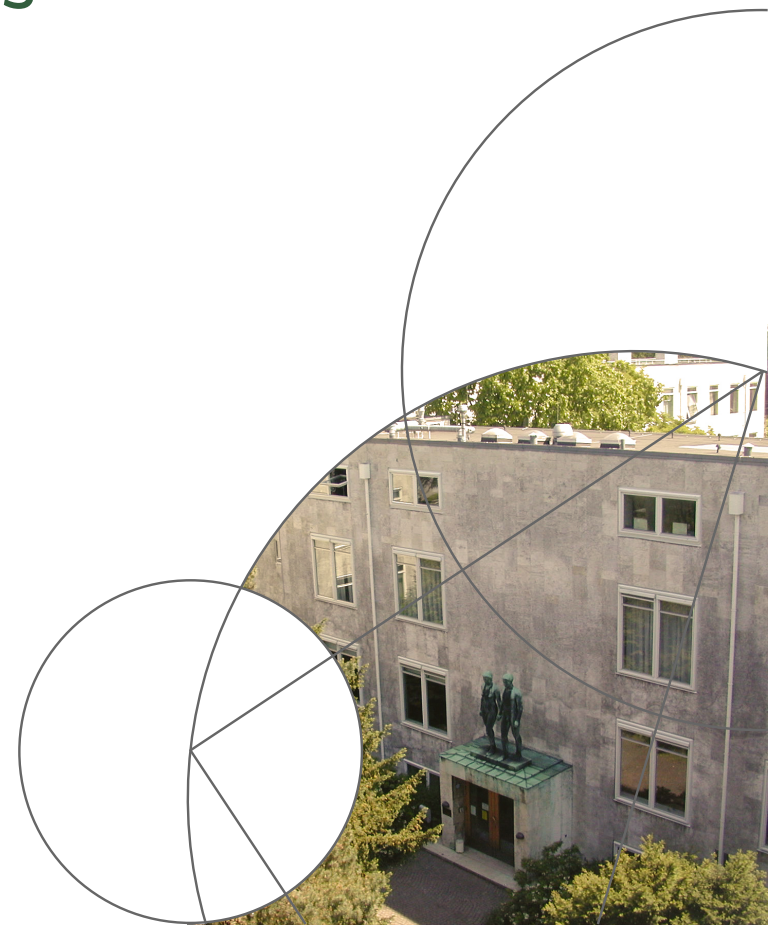
Principal Component Analysis

Machine Learning

Christian Igel
Department of Computer Science



Basic questions

- Principal component analysis (PCA, Karhunen-Loève transform) is arguably the most fundamental *unsupervised learning* algorithm.
- It is frequently used for (linear) reduction, (lossy) data compression, feature extraction, and data visualization.
- Let's consider $S = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ with $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N \in \mathcal{X} = \mathbb{R}^d$.
- We ask how
  - to reduce the length of the description to $k < d$ variables such that as much information as possible is preserved?
  - many dimensions are needed to capture a certain percentage of the variability of the data?
  - to visualize the data in two or three dimensions preserving as much of its variability as possible?

Outline

Outline

Basis vectors

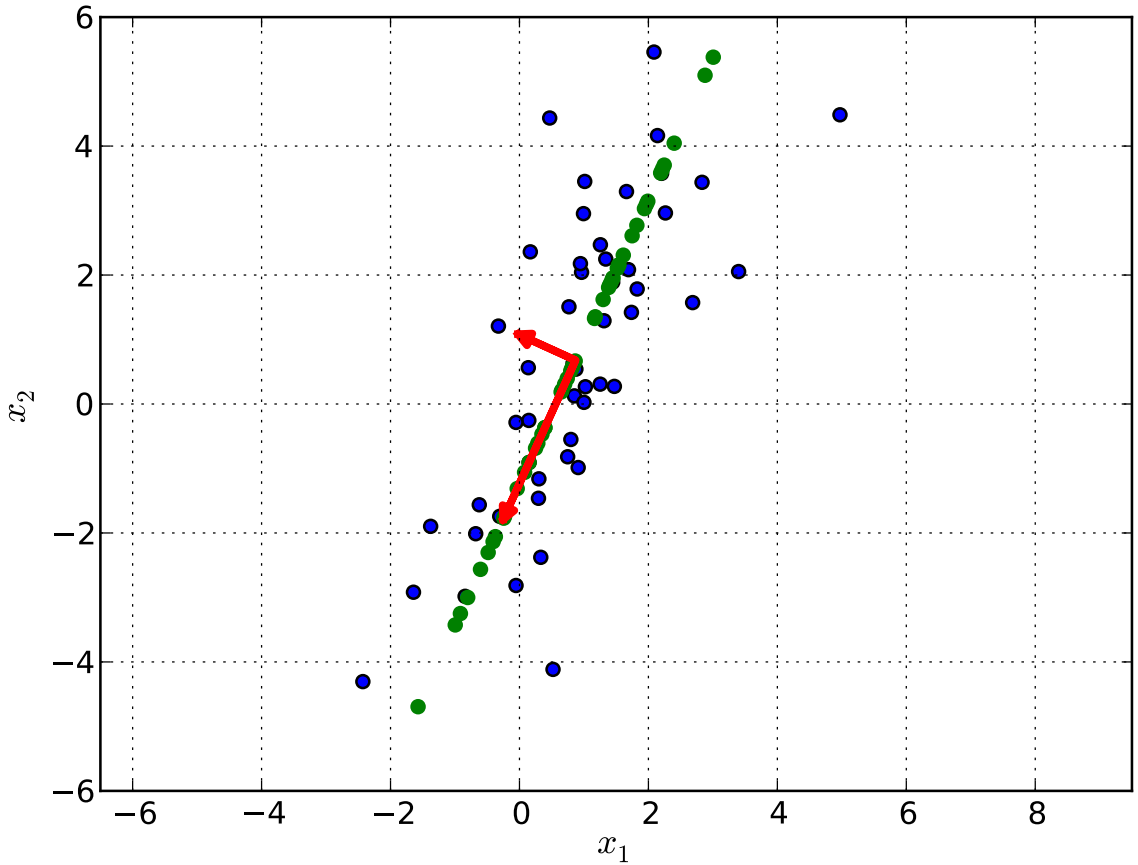
- *Basis*: Set of linearly independent *basis vectors* $\boldsymbol{u}_1, \ldots$ that, in a linear combination, can represent every vector in given vector space
- Ortho*normal* basis: Basis vectors are orthogonal (i.e., $\boldsymbol{u}_i^\mathsf{T}\boldsymbol{u}_j = 0$ for $i \neq j$) and have unit length, $\|\boldsymbol{u}_i\| = \sqrt{\boldsymbol{u}_i^\mathsf{T}\boldsymbol{u}_i} = 1$
- Example: $\boldsymbol{x} \in \mathbb{R}^2$, orthonormal basis

$$\begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix},$$

$\boldsymbol{x} = (1,0)^\mathsf{T} x_1 + (0,1)^\mathsf{T} x_2$

- Example can be read as "way from $\mathbf{0}$ to $\boldsymbol{x}$: First go x_1 units in the direction $(1,0)^\mathsf{T}$ and then x_2 units in direction $(1,0)^\mathsf{T}$"

2-dimensional example



Changing basis

- Changing basis means "expressing the way using two other, non-parallel directions $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$: First go z_1 units in the direction $\boldsymbol{u}_1$ and then z_2 units in direction $\boldsymbol{u}_2$":

$$\boldsymbol{x} = \sum_{i=1}^{d} z_i \boldsymbol{u}_i = \sum_{i=1}^{d} (\boldsymbol{x}^\mathsf{T} \boldsymbol{u}_i) \boldsymbol{u}_i$$

- A lower dimensional representation uses "fewer directions" and a (different) origin/starting point.

Orthogonal matrix

- Gather basis vectors in $d \times d$ matrix $\boldsymbol{U}$ such that the columns of $\boldsymbol{U}$ correspond to the basis vectors.
- ith column of $\boldsymbol{U}$ is given by $\boldsymbol{u}_i$.
- Square matrix composed of an orthonormal basis is an *orthogonal* matrix having the property $\boldsymbol{U}^\mathsf{T} = \boldsymbol{U}^{-1}$.
- Define $d \times k$ matrix $\boldsymbol{U}_k$ as the first k basis vectors.
- ith column of $\boldsymbol{U}_k$ is given by $\boldsymbol{u}_i$ and the ith row of $\boldsymbol{U}^\mathsf{T}$ by $\boldsymbol{u}_i^\mathsf{T}$.

Outline

Eigenvectors and eigenvalues

- For an eigenvector $\boldsymbol{u}_i \in \mathbb{R}^d$ of the $d \times d$ matrix $\boldsymbol{S}$ it holds by definition:

$$\boldsymbol{S}\boldsymbol{u}_i = \lambda_i \boldsymbol{u}_i$$

  λ_i is the corresponding eigenvalue.

- We consider only consider eigenvectors of unit length and assume that the eigenvectors are sorted according to $i < j \Rightarrow \lambda_i \geq \lambda_j$.

Eigendecomposition of real symmetric matrix

Any *real symmetric* $d \times d$ matrix $\boldsymbol{M}$ can be decomposed as:

$$\boldsymbol{M} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^\mathsf{T}$$

with

- $\boldsymbol{Q} \in \mathbb{R}^{d \times d}$ being orthogonal,
- $\boldsymbol{Q}^\mathsf{T} = \boldsymbol{Q}^{-1}$,
- $\boldsymbol{\Lambda} \in \mathbb{R}^{d \times d} = \mathrm{diag}(\lambda_1, \ldots, \lambda_d)$ being diagonal, with $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d$ being the eigenvalues of $\boldsymbol{M}$, and
- the columns of $\boldsymbol{Q}$ being the corresponding eigenvectors.

Singular value decomposition (SVD)

Any real $N \times d$ matrix $\boldsymbol{X}$ can be decomposed as:

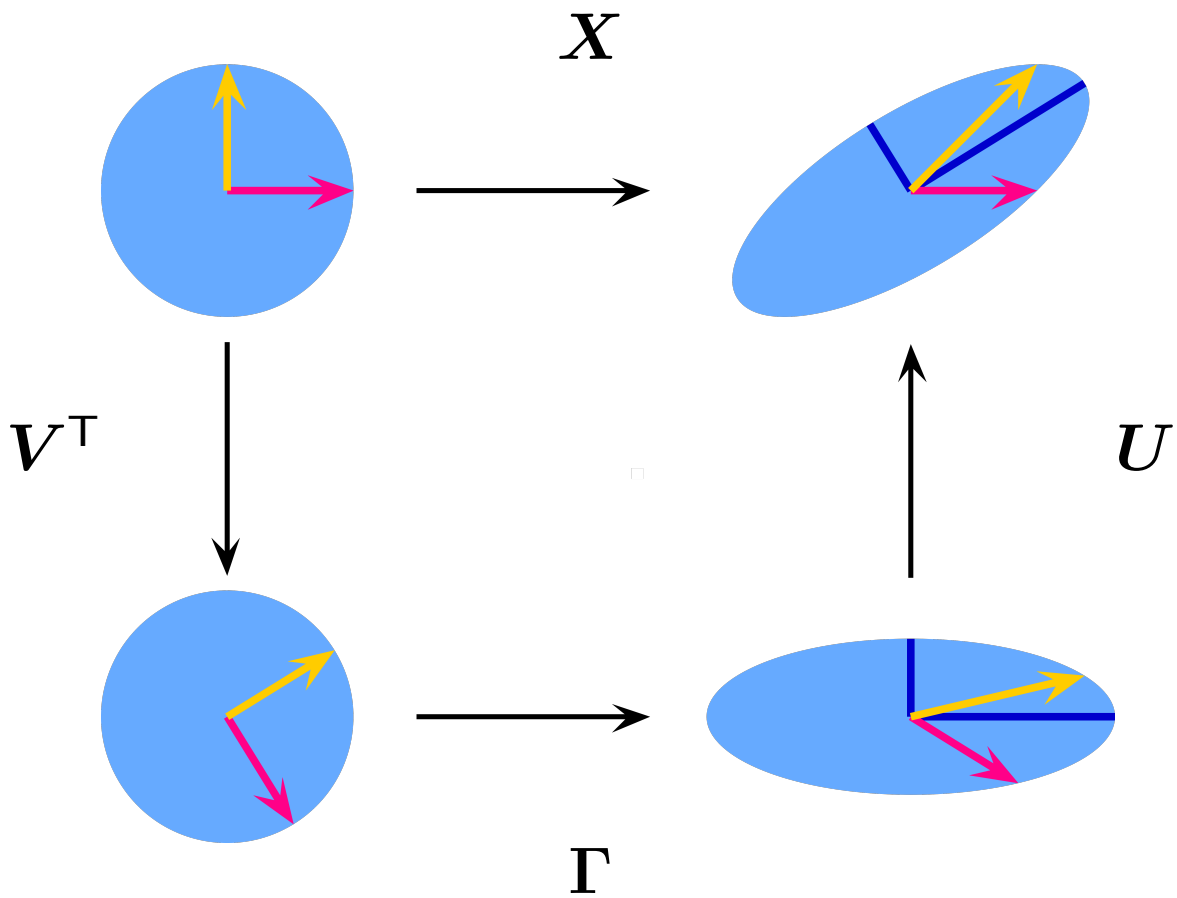
$$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Gamma}\boldsymbol{V}^\mathsf{T}$$

with

- $\boldsymbol{U} \in \mathbb{R}^{N \times N}$ orthogonal,
- $\boldsymbol{\Gamma} \in \mathbb{R}^{N \times d}$ with $\boldsymbol{\Gamma} = \begin{bmatrix} \mathrm{diag}(\gamma_1, \ldots, \gamma_N) & \mathbf{0} \end{bmatrix}$ if $d \geq N$ and $\boldsymbol{\Gamma} = \begin{bmatrix} \mathrm{diag}(\gamma_1, \ldots, \gamma_d) \\ \mathbf{0} \end{bmatrix}$ if $d \leq N$ and with $\gamma_1 \geq \gamma_2 \geq \cdots \geq 0$,
- $\boldsymbol{V} \in \mathbb{R}^{d \times d}$ orthogonal.

The columns of $\boldsymbol{V}$ are the *right singular vectors* of $\boldsymbol{X}$.

SVD example



drawing modified from Wikimedia Commons

SVD and eigendecomposition

- When $\boldsymbol{M}$ is positive semi-definite, its eigendecomposition is also a SVD.
- The right singular vectors of $\boldsymbol{X}$ are the eigenvectors of $\boldsymbol{X}^\mathsf{T}\boldsymbol{X}$.
- $\boldsymbol{X}^\mathsf{T}\boldsymbol{X}$ is positive (semi-) definite (i.e., all eigenvalues are non-negative), and the singular values of $\boldsymbol{X}$ are the square roots of the eigenvalues of $\boldsymbol{X}^\mathsf{T}\boldsymbol{X}$.

Outline

Basic questions

- Principal component analysis (PCA, Karhunen-Loève transform) is arguably the most fundamental *unsupervised learning* algorithm.
- It is frequently used for (linear) reduction, (lossy) data compression, feature extraction, and data visualization.
- Let's consider $S = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ with $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N \in \mathcal{X} = \mathbb{R}^d$.
- We ask how
  - to reduce the length of the description to $k < d$ variables such that as much information as possible is preserved?
  - many dimensions are needed to capture a certain percentage of the variability of the data?
  - to visualize the data in two or three dimensions preserving as much of its variability as possible?

Example: Cambridge face database








Basic idea

- Find k-dimensional affine linear model $f : \mathbb{R}^k \to \mathbb{R}^d$ of the d-dimensional data representing S as accurately as possible:

$$f(\boldsymbol{z}) = \boldsymbol{b} + \boldsymbol{U}_k \boldsymbol{z} \ ,$$

  where $\boldsymbol{z} \in \mathbb{R}^k$, $\boldsymbol{b} \in \mathbb{R}^d$, and $\boldsymbol{U}_k \in \mathbb{R}^{d \times k}$.

- Vectors $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_k \in \mathbb{R}^d$ are the columns of $\boldsymbol{U}_k$ and we require them to be pairwise orthogonal and of unit length.

- Model represents data in $\mathbb{R}^d$ by k-dimensional parameters $\boldsymbol{z}$.

Reconstruction error

- We measure model quality by sum-of-squares *reconstruction error*

$$J = \frac{1}{N} \sum_{i=1}^{N} \|\boldsymbol{x}_i - f(\boldsymbol{z}_i)\|^2 \quad .$$

- Formal goal of PCA

$$\min_{\boldsymbol{b}, \boldsymbol{U}_k, \{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_N\}} \frac{1}{N} \sum_{i=1}^{N} \|\boldsymbol{x}_i - f(\boldsymbol{z}_i)\|^2$$

  subject to the constraints on $\boldsymbol{U}_k$.

Solution

- The choice for $\boldsymbol{b}$ is the empirical mean:

$$\overline{\boldsymbol{x}} = \frac{1}{N}\sum_{i=1}^{N}\boldsymbol{x}_i$$

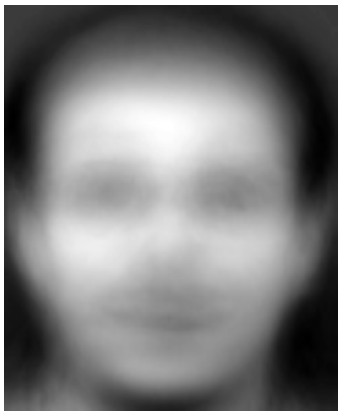
- Data points should be encode by

$$\boldsymbol{z}_i = \boldsymbol{U}_k^{\mathsf{T}}(\boldsymbol{x}_i - \overline{\boldsymbol{x}}) \quad .$$

- k columns of $\boldsymbol{U}_k$ should correspond to the first k eigenvectors of the *data covariance matrix* or *empirical covariance matrix*

$$\boldsymbol{S} = \frac{1}{N}\sum_{i=1}^{N}(\boldsymbol{x}_i - \overline{\boldsymbol{x}})(\boldsymbol{x}_i - \overline{\boldsymbol{x}})^{\mathsf{T}} \quad .$$

Mean face



$$\overline{\boldsymbol{x}} = \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{x}_i$$

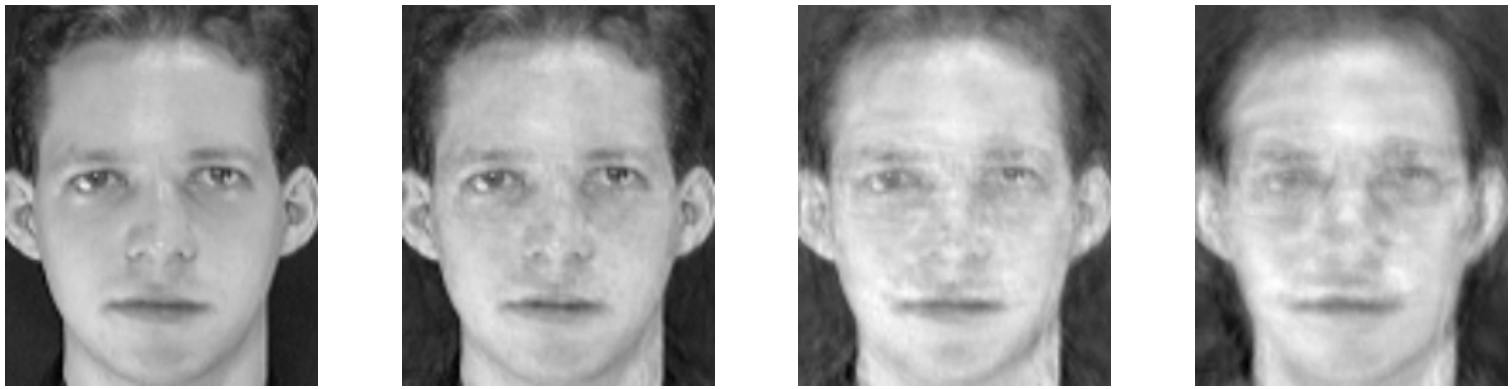
Eigenfaces








Reconstruction using eigenfaces



using all, 300, 200, 100 eigenfaces

PCA algorithm

Algorithm 1: dimensionality reduction using PCA

Input: data $S = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$, number of dimensions k

1 compute the empirical mean $\overline{\boldsymbol{x}} = \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{x}_i$

2 compute empirical covariance matrix
$\boldsymbol{S} = \frac{1}{N} \sum_{i=1}^{N} (\boldsymbol{x}_i - \overline{\boldsymbol{x}})(\boldsymbol{x}_i - \overline{\boldsymbol{x}})^\mathsf{T}$

3 compute the $d \times k$ matrix $\boldsymbol{U}_k$ composed of the first k eigenvectors of $\boldsymbol{S}$, where the eigenvectors are ordered by decreasing eigenvalue

4 compute $\boldsymbol{z}_i = \boldsymbol{U}_k^\mathsf{T}(\boldsymbol{x}_i - \overline{\boldsymbol{x}})$ for $i = 1, \ldots, N$

Output: mean $\overline{\boldsymbol{x}}$, principal components $\boldsymbol{U}_k$, projected data $\{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_N\}$, model $f(\boldsymbol{z}) = \overline{\boldsymbol{x}} + \boldsymbol{U}_k \boldsymbol{z}$

Eckart and Young theorem

The (squared) Frobenius norm of an $N \times d$ matrix $\boldsymbol{X}$ is

$$\|\boldsymbol{X}\|_F^2 = \sum_{i=1}^{N} \sum_{j=1}^{d} x_{ij}^2 \ .$$

The $N \times d$ matrix $\tilde{\boldsymbol{X}}$ with rank k approximating the $N \times d$ matrix $\boldsymbol{X}$ best in terms of the Frobenius norm $\|\boldsymbol{X} - \tilde{\boldsymbol{X}}\|_F^2$ is

$$\tilde{\boldsymbol{X}} = \boldsymbol{X}\boldsymbol{V}_k\boldsymbol{V}_k^\mathsf{T} \ ,$$

where $\boldsymbol{V}_k$ is the matrix of top-k right singular vectors of $\boldsymbol{X}$.

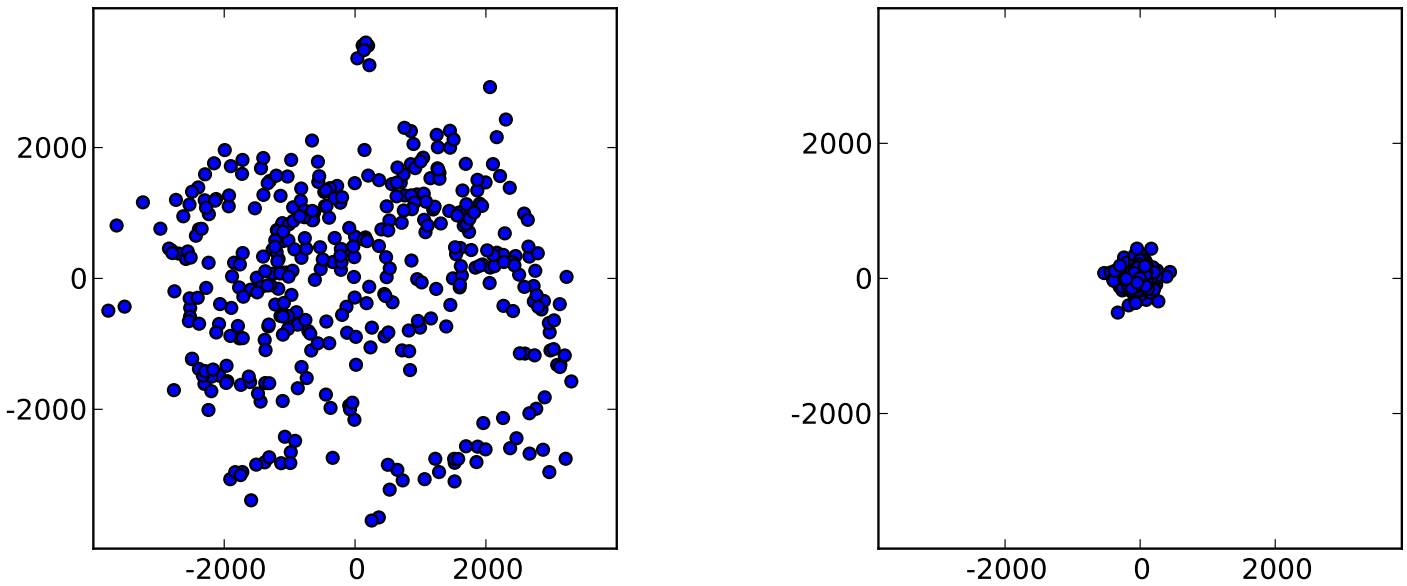
Maximizing variance

- We measure the variability of S by the trace $\mathrm{trace}\{\boldsymbol{S}\} = \sum_{i=1}^{d} s_{ii} = \sum_{i=1}^{d} \lambda_i = \sum_{j=1}^{d} \boldsymbol{u}_j^\mathsf{T} \boldsymbol{S} \boldsymbol{u}_j$. (Sum of the eigenvalues of a diagonalizable matrix is equal to its trace.)
- Accordingly the variability of $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_N$ is

$$\frac{1}{N} \sum_{i=1}^{N} \left[\boldsymbol{U}_k^\mathsf{T} \boldsymbol{x}_i - \boldsymbol{U}_k^\mathsf{T} \overline{\boldsymbol{x}}\right] \left[\boldsymbol{U}_k^\mathsf{T} \boldsymbol{x}_i - \boldsymbol{U}_k^\mathsf{T} \overline{\boldsymbol{x}}\right]^\mathsf{T} = \sum_{j=1}^{k} \boldsymbol{u}_j^\mathsf{T} \boldsymbol{S} \boldsymbol{u}_j \ .$$

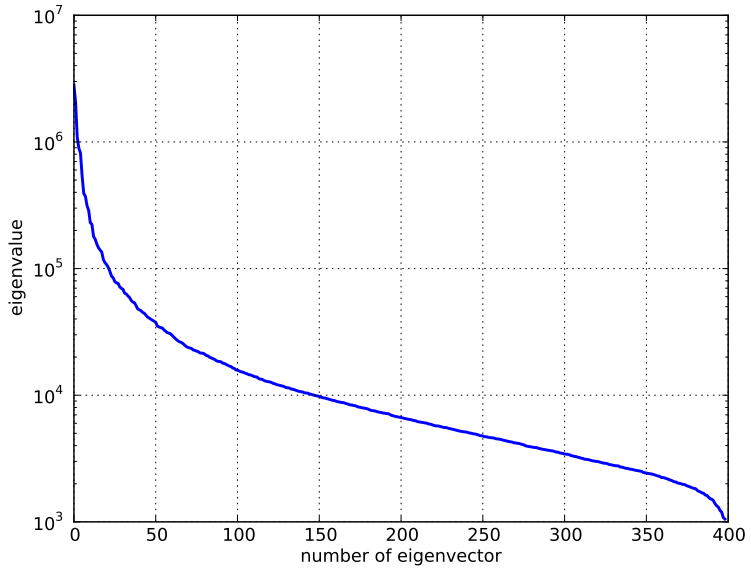
- Minimizing reconstruction error corresponds to maximizing variance.

Variance of faces



first two eigenvalues vs. eigenvalues 99 and 100

"Explained variance"



We have:

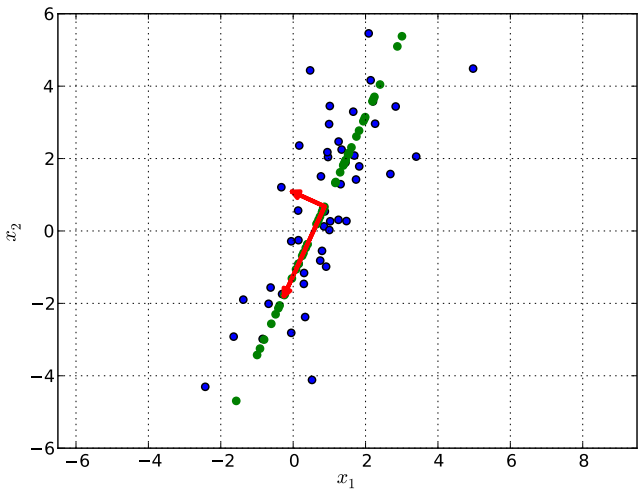
$$\text{trace}\{\boldsymbol{S}\} = \sum_{j=1}^{d} \boldsymbol{u}_j^\mathsf{T} \boldsymbol{S} \boldsymbol{u}_j = \sum_{i=1}^{d} \lambda_i$$

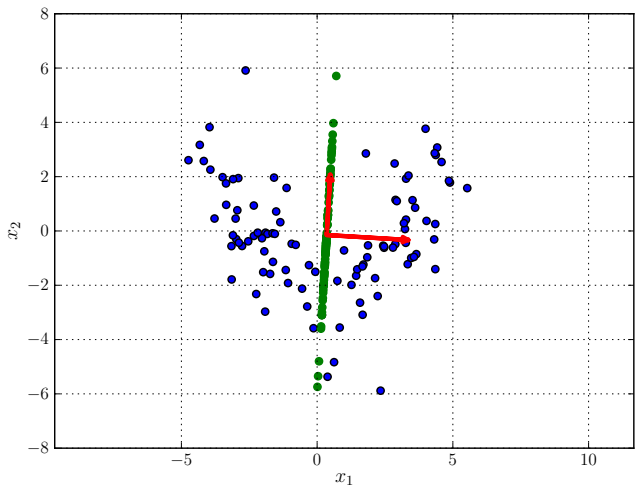
The quotient

$$\frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{d} \lambda_i}$$

measures the fraction of variance "explained" by the first k principal components.

Linear and non-linear example





Outline

Let's discuss I

[..] many standard design choices when learning from data intend each dimension to be treated equally (such as using the Euclidean distance metric in similarity methods or using the sum of squared weights as the regularization term in weight decay).

Unless there is some explicit reason not to do so, the data should be presented to the learning algorithm with each dimension on an equal footing. To do so, we need to transform the data to a standardized setting so that we can be immune to arbitrary choices made during data collection.

Quotes on this and the following slides from
Abu-Mostafa, Magdon-Ismail, and Lin, *Learning from Data*, AMLbook, 2012

Whitening

- Scaling each input component to zero mean and unit variance is a common preprocessing step.
- Whitening is a more advanced preprocessing transformation.
- Whitening transforms data such that after the transformation the covariance of the data is the identity matrix.
- Whitening can be achieved by transforming the data via PCA and then normalizing each component by the (square root of the) corresponding eigenvalue.

PCA, preprocessing, and feature selection

PCA is often used for preprocessing data:

- PCA can by used to implement whitening (see next slide)
- PCA can be used for feature selection

Important difference::

- Centering, scaling and whitening attempt to correct for arbitrary "choices that may have been made during data collection". They fix the coordinate system.
- Feature selection, e.g. using PCA, is different. It removes features. Feature selection using PCA removes dinensions that do not explain much variance in order to consider the "true" dimensionality of the input which removes noise and redundancies.

Let's discuss II

"What if the small fluctuations [...] were the actual important information on which f depends, and the large variability[...] are random fluctuations?"

- In this case, dimensionality reduction w/ PCA is a bad idea.
- One may argue, "Though possible, this rarely happens in practice, and if it does happen, then your input is corrupted by large random noise and you are in trouble anyway. So, let's focus on the case where we have a chance and discuss how to find this optimal rotation", but this is a strong assumption.
- If in doubt, keep all dimensions and let the algorithm work out which dimensions are important by explicit or implicit ($\rightarrow$ random forest) feature selection

Let's discuss III

"Whitening is a way to put your data into a spherically symmetric form so that all directions are 'equal'. This is recommended when you have no evidence to the contrary; you whiten the data because most learning algorithms treat every dimension equally (nearest neighbor, weight decay, etc.). PCA, on the other hand is highlighting specific directions which contain more variance.

There is no use doing PCA after doing whitening, since every direction will be on an equal footing after whitening. You use PCA precisely because the directions are not to be treated equally. PCA helps to identify and throw away the directions where the fluctuations are a result of small amounts of noise. After deciding which directions to throw away, you can now use whitening to put all the retained directions on an equal footing, if you wish."

Outline

Summary

Principal component analysis (PCA) is frequently used for

- dimensionality reduction,
- noise reduction, and
- visualization.

It

- is an affine linear model of the data,
- is a change of basis, new basis vectors are orthogonal,
- minimizes reconstruction error, and
- maximizes variance.