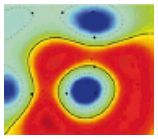
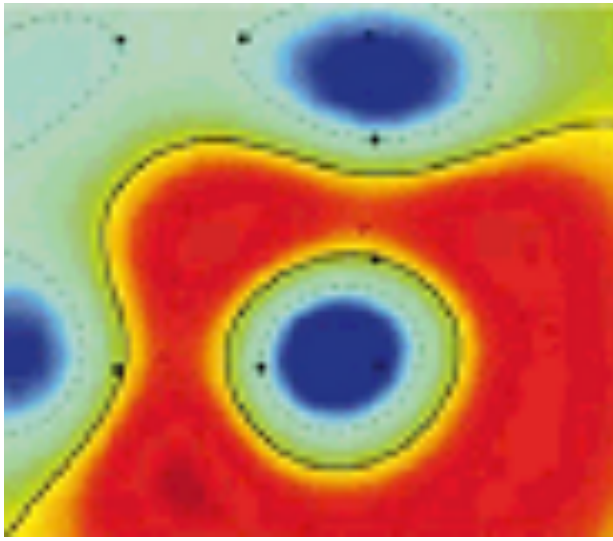


WiSe 2024/25

Machine Learning 1/1-X

Technische Universität Berlin

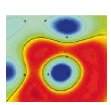


Lecture 4

Principal Component Analysis

Outline

- Motivation: Dimensionality reduction
- Lagrange Multipliers: Local Optima under Equality Constraints
- Principal Component Analysis (PCA)
  - What are principal components?
  - Maximum Variance Formulation
  - Minimum Error Formulation
  - How to find/calculate them
    - Lagrange multipliers: Closed-Form Solution
    - SVD vs. Eigenvalue Decomposition
    - Iterative approaches
- Further Applications



Curse of Dimensionality

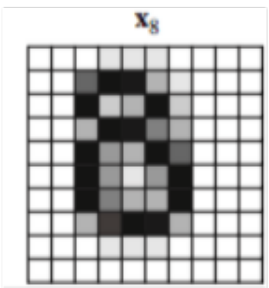
维度诅咒

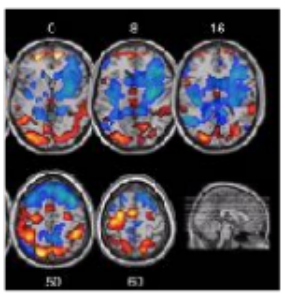
在许多机器学习问题中，数据具有高维度。

标准的回归/分类技术在以下情况中可能变得：

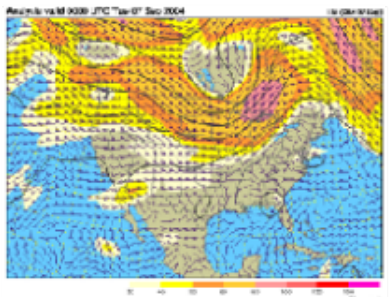
- 对于 $d > N$，定义不良。
- 即使对于 $d \leq N$，也可能因为数值不稳定而导致条件不良。

In many machine learning problems, the data are high-dimensional.





From Limb and Braun, 2008


University of Colorado


IEEE Spectrum

Standard regression/classification techniques can become

- ill-defined for $d > N$.
- ill-conditioned/numerically unstable even for $d \leq N$.

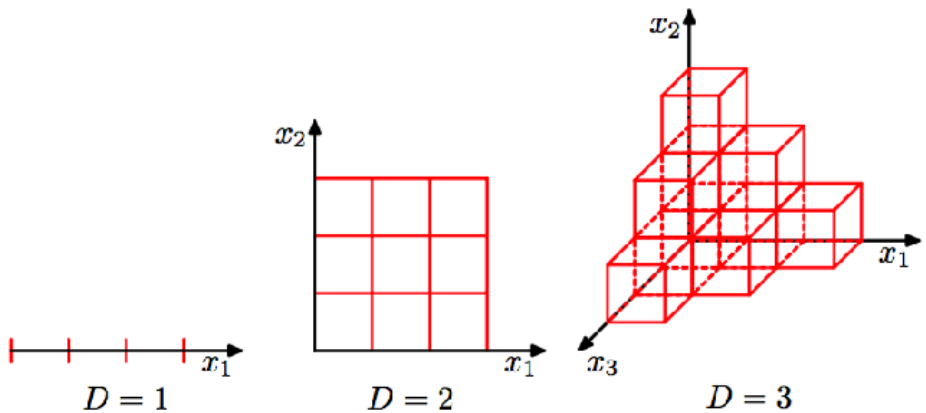
Curse of Dimensionality (2)

当维度增加时，空间的体积增加得非常快，以至于可用的数据变得稀疏。

为了得到可靠的结果（如避免过拟合）所需的数据量通常会随着维度指数增长。

这可以通过减少数据的维度（现在）或进行正则化（稍后）来解决。

- When the dimensionality increases, the volume of the space increases so fast that the available data becomes sparse.



- The amount of data needed for a reliable result (e.g. no overfitting) often grows exponentially with the dimensionality.
- Here again, this can be addressed by reducing the dimensionality of the data **(today)** or regularizing (later).

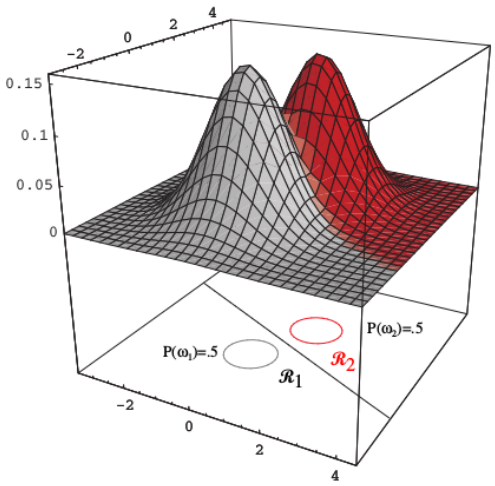
Example of ill-conditioning

Bayes Classifier with Estimated Covariance

Recap:

- Assume our data is generated for each class ω_j according to the multivariate Gaussian distribution $p(\boldsymbol{x}|\omega_j) = \mathcal{N}(\boldsymbol{\mu}_j, \Sigma)$ and with class priors $P(\omega_j)$. The Bayes optimal classifier is derived (using Bayes' Theorem) as

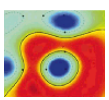
$$\begin{aligned} &\arg\max_j \{P(\omega_j|\boldsymbol{x})\} \\ &= \arg\max_j \{\log p(\boldsymbol{x}|\omega_j) + \log P(\omega_j)\} \\ &= \arg\max_j \left\{\boldsymbol{x}^\top \Sigma^{-1} \boldsymbol{\mu}_j - \frac{1}{2}\boldsymbol{\mu}_j \Sigma^{-1} \boldsymbol{\mu}_j + \log P(\omega_j)\right\} \end{aligned}$$



- In practice, the true Σ is unknown, so we use some estimator $\widehat{\Sigma}$ in place of Σ.

Question:

- Can we compute $\widehat{\Sigma}^{-1}$, and can we do so accurately?



Example of ill-conditioning

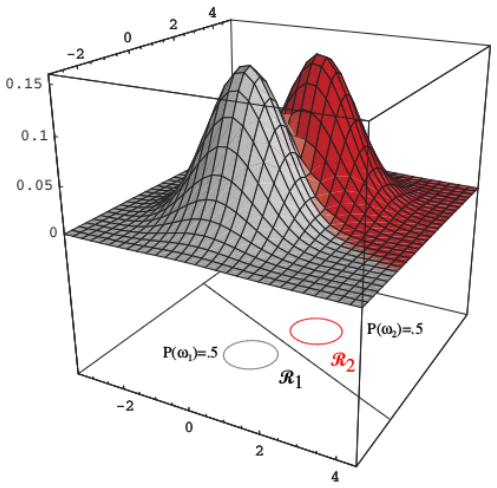
Bayes Classifier with Estimated Covariance

Example of estimator:

- Maximum likelihood estimator:

$$\widehat{\Sigma} = \frac{1}{N}\bar{X}\bar{X}^\top$$

where $\bar{X}$ is a matrix of size $d \times N$ containing the centered data.



Problem:

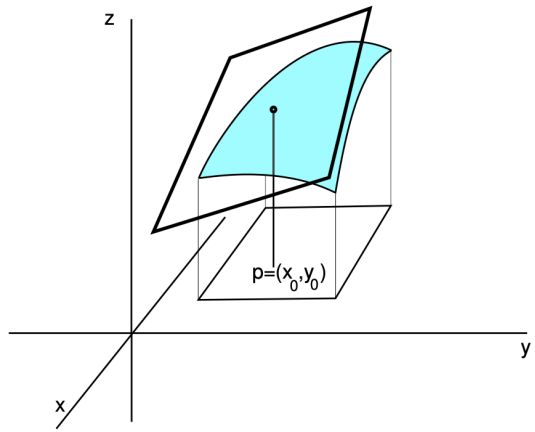
- The matrix $\widehat{\Sigma}^{-1}$ used in the classifier only exists when $d \leq N$ and the matrix $\bar{X}$ has full rank (due $\text{rank}(X) = \text{rank}(XX^\top)$).
- The matrix $\widehat{\Sigma}^{-1}$ used in the classifier is typically accurate only when $d \ll N$.

Solutions:

- Reduce the dimensionality of the data **(today)**
- Regularize the model (later)

Recap (Analysis II)

- For a differential mapping $f : G \subset \mathbb{R}^n \to \mathbb{R}^m$, the derivative $D_{\boldsymbol{p}} f : \mathbb{R}^n \to \mathbb{R}^m$ is a linear approximation of f at point $\boldsymbol{p} \in G$ defined as $f(\boldsymbol{x}) = f(\boldsymbol{p}) + D_{\boldsymbol{p}} f(\boldsymbol{x} - \boldsymbol{p}) + R(\boldsymbol{x})$ for $\lim_{\boldsymbol{x} \to \boldsymbol{p}} R(\boldsymbol{x}) / \|\boldsymbol{x} - \boldsymbol{p}\| = 0$.
- The geometric interpretation of the derivative is easiest in the case $f : \mathbb{R}^2 \to \mathbb{R}$. Then the graph of the mapping $\boldsymbol{x} \mapsto f(\boldsymbol{p}) + D_{\boldsymbol{p}} f(\boldsymbol{x} - \boldsymbol{p})$ is a hyperplane in $\mathbb{R}^3$, the tangent plane on the graph of f.



- For $\boldsymbol{v} \in G$, the directional derivative $\partial_{\boldsymbol{v}} f(\boldsymbol{p}) = D_{\boldsymbol{p}} f(\boldsymbol{v}) \in \mathbb{R}^m$ describes the rate at which each component f_i changes at point $\boldsymbol{p}$ in direction $\boldsymbol{v}$.
- Different notations: $\nabla_{\boldsymbol{x}} f(\boldsymbol{p}) = \frac{\partial f}{\partial \boldsymbol{x}}(\boldsymbol{p})$ or $\partial_i f(\boldsymbol{p}) = \frac{\partial f}{\partial x_i}(\boldsymbol{p})$.

Local Optima under Equality Constraints

Method of Lagrange Multipliers

- **Goal:** for differentiable mappings $f\colon G \to \mathbb{R}$ and $g\colon G \to \mathbb{R}^m$ on an open set $G \subset \mathbb{R}^n$, we look for solutions of a constraint optimization problem

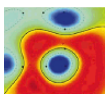
$$\underset{x \in G}{\text{minimize}} \;\; f(\boldsymbol{x}) \quad \text{subject to} \quad g_1(\boldsymbol{x}) = 0, \ldots, g_m(\boldsymbol{x}) = 0.$$

- **Recipe:** find all stationary points $\boldsymbol{p} \in G$ and $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_m)^\top \in \mathbb{R}^m$ by solving the following system of $n + m$ equations with $n + m$ variables :

$$g_1(\boldsymbol{p}) = 0, \ldots, g_m(\boldsymbol{p}) = 0$$

$$\partial_j f(\boldsymbol{p}) = \sum_{k=1}^{m} \lambda_k \partial_j g_k(\boldsymbol{p}), \quad j = 1, \ldots, n.$$

- More generally, the second set of equations corresponds to $\nabla_x \mathcal{L}(\boldsymbol{p}, \boldsymbol{\lambda}) = \nabla_x f(\boldsymbol{p}) + \boldsymbol{\lambda}^\top D_{\boldsymbol{p}} g = \boldsymbol{0}$ where $\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) + \boldsymbol{\lambda}^\top g(\boldsymbol{x})$ is the Lagrange function (next lecture).



Method of Lagrange Multipliers: Intuition

Consider maximizing an objective $f(\boldsymbol{x})$ subject to a single constraint $g(\boldsymbol{x}) = 0$.

- **Step 1:** Build the Lagrangian:

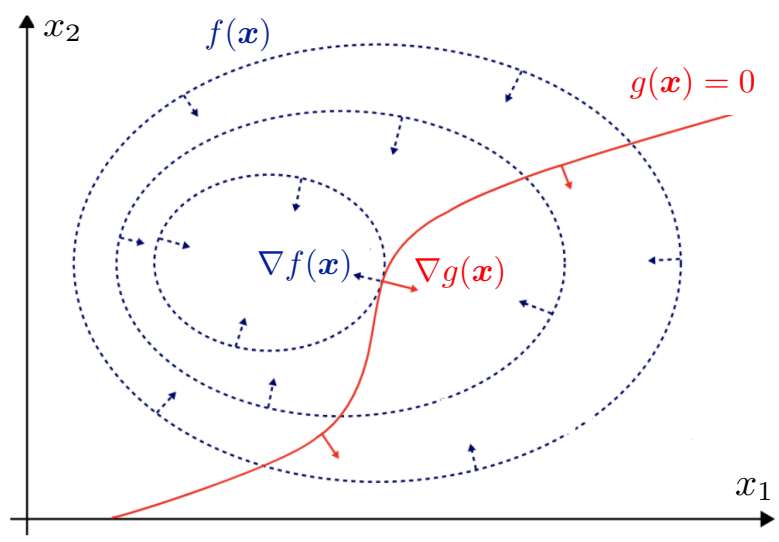
$$\mathcal{L}(\boldsymbol{x}, \lambda) = f(\boldsymbol{x}) + \lambda g(\boldsymbol{x})$$

  where λ is called the Lagrange multiplier.

- **Step 2:** Solve the equation

$$\nabla \mathcal{L}(\boldsymbol{x}, \lambda) = \mathbf{0}$$

  which is a necessary condition for the solution.



Intuition for step 2: the equation corresponds to $\nabla f(\boldsymbol{x}) = -\lambda \nabla g(\boldsymbol{x})$, i.e. the gradients of the objective and constraint function at a stationary point are collinear according to the parameter λ.

Lagrange Multipliers: Simple Example

Maximize $f(x_1, x_2) = x_1 + x_2$ subject to $x_1^2 + x_2^2 = 1$.

- **Step 1:** Write the problem in the canonical form

$$\underset{x \in \mathbb{R}^2}{\text{maximize}} \quad \underbrace{x_1 + x_2}_{f(x)} \quad \text{subject to} \quad \underbrace{x_1^2 + x_2^2 - 1}_{g(x)} = 0.$$

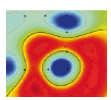
- **Step 1:** Build the Lagrangian

$$\mathcal{L}(x, \lambda) = f(x) + \lambda g(x) = (x_1 + x_2) + \lambda(x_1^2 + x_2^2 - 1)$$

- **Step 2:** Solve the system $\nabla_x \mathcal{L} = \mathbf{0}$, $g(x) = 0$

$$\frac{\partial \mathcal{L}}{\partial x_1}(x, \lambda) = 0 \quad \Rightarrow \quad -\frac{1}{2x_1} = \lambda$$
$$\frac{\partial \mathcal{L}}{\partial x_2}(x, \lambda) = 0 \quad \Rightarrow \quad -\frac{1}{2x_2} = \lambda$$
$$g(x) = x_1^2 + x_2^2 - 1 = 0 \quad \Rightarrow \quad x_1 = x_2 = \pm\frac{\sqrt{2}}{2}$$

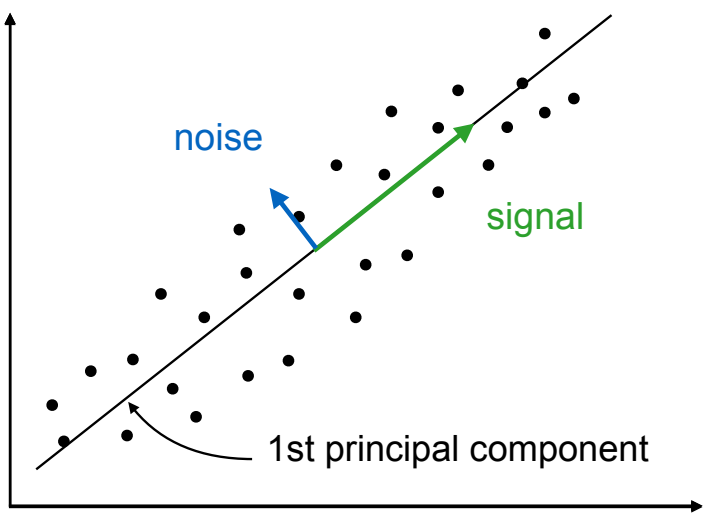
which gives us the solution $(x_1, x_2) = (\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2})$.



Principal Component Analysis (PCA)

- Principal Component Analysis (PCA) is a statistical technique used to simplify the complexity in high-dimensional data while retaining most of the variation present in the dataset.
- PCA is commonly used for dimensionality reduction, data visualization, noise reduction, and feature extraction.
- The basic idea of PCA is to transform a set of correlated variables into a new set of uncorrelated variables by projecting them onto the principal components, which are ordered by the amount of variance they capture from the data.

Which line fits data best?
The line $\boldsymbol{w}$ that minimizes the noise and maximizes the signal [Pearson 1901].



PCA: Maximum Variance Formulation

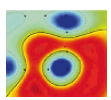
- Given a multivariate random variable $X : \Omega \to \mathbb{R}^d$ (representing our data), we look for a direction $\boldsymbol{w}$ along which X varies the most by maximizing the variance of the projected data:

$$\underset{\boldsymbol{w} \in \mathbb{R}^d}{\text{maximize}} \ \operatorname{Var}[\boldsymbol{w}^\top X] \quad \text{subject to} \quad \|\boldsymbol{w}\|^2 = 1$$

- Let us rewrite the objective:

$$\begin{aligned}
\operatorname{Var}[\boldsymbol{w}^\top X] &= \mathbb{E}\left[(\boldsymbol{w}^\top X - \mathbb{E}[\boldsymbol{w}^\top X])^2\right] = \mathbb{E}[(\boldsymbol{w}^\top X)^2] - \mathbb{E}[\boldsymbol{w}^\top X]^2 \\
&= \mathbb{E}[\boldsymbol{w}^\top X \cdot X^\top \boldsymbol{w}] - \mathbb{E}[\boldsymbol{w}^\top X] \cdot \mathbb{E}[X^\top \boldsymbol{w}] \\
&= \boldsymbol{w}^\top \mathbb{E}[XX^\top]\boldsymbol{w} - \boldsymbol{w}^\top \mathbb{E}[X]\mathbb{E}[X^\top]\boldsymbol{w} \\
&= \boldsymbol{w}^\top (\mathbb{E}[XX^\top] - \mathbb{E}[X]\mathbb{E}[X^\top])\boldsymbol{w} \\
&= \boldsymbol{w}^\top (\mathbb{E}[(X - \mathbb{E}[X])(X - \mathbb{E}[X])^\top]\boldsymbol{w} = \boldsymbol{w}^\top \operatorname{Cov}[X]\boldsymbol{w}
\end{aligned}$$

- Given the first principal components $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_k$, we can find the next component $\boldsymbol{u}_{k+1}$ by maximizing the objective $f_0(\boldsymbol{w}) = \boldsymbol{w}^\top \operatorname{Cov}[X]\boldsymbol{w}$ subject to the constraints $\|\boldsymbol{w}\|^2 = 1$ and $\boldsymbol{w}^\top \boldsymbol{u}_j = 0$ for all $j = 1, \ldots, k$.

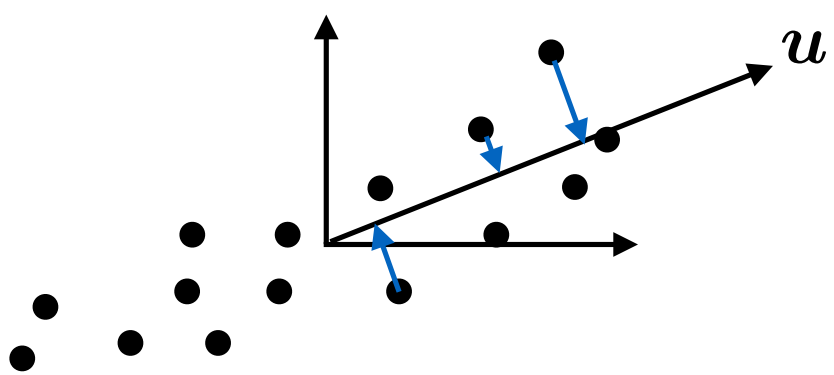


PCA: Minimum Error Formulation

- Given a multivariate random variable $X : \Omega \to \mathbb{R}^d$ and a number $k \in \mathbb{N}$, we look for vectors $U := [\boldsymbol{u}_1, ..., \boldsymbol{u}_k] \in \mathbb{R}^{d \times k}$ minimizing the expected distance between the original point and its projection:

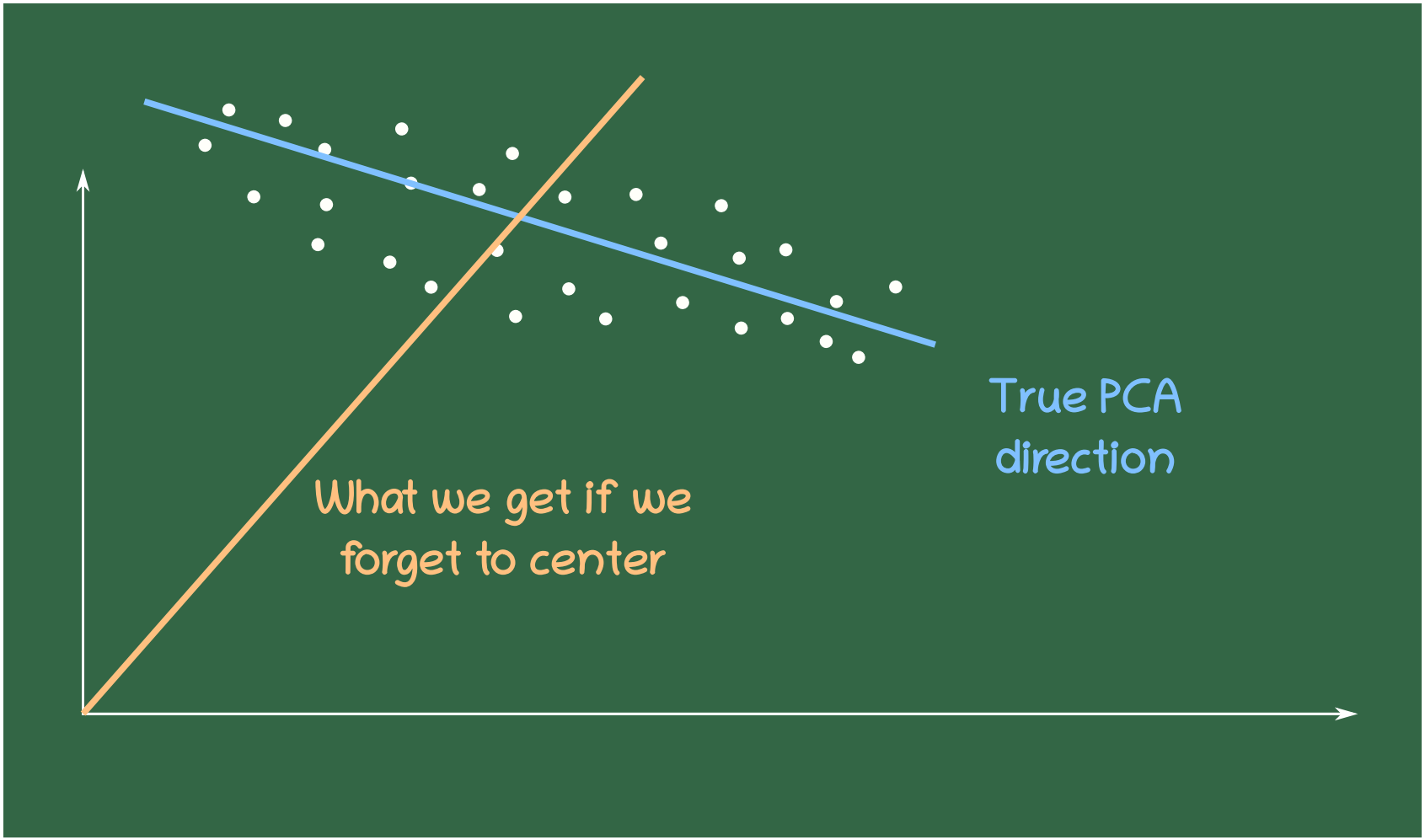
$$\underset{\boldsymbol{u}_1,...,\boldsymbol{u}_k}{\text{minimize}} \ \mathbb{E}[\|X - UU^\top X\|^2] \quad \text{subject to} \quad U^\top U = I.$$

- Note: we assume here centered data, i.e., $\mathbb{E}[X] = \mathbf{0}$.



What if the Data is Uncentered?

未居中数据的主成分方向会偏离真实方向，因为原始数据的均值并不位于原点，而是在一个偏移的位置。如果数据没有居中，那么得到的方向不仅反映了数据的主要变化方向，还包含了偏移信息，从而导致结果不准确。



Equivalence of both Formulations

- Consider the following derivations for $U := [\boldsymbol{u}_1, ..., \boldsymbol{u}_k] \in \mathbb{R}^{d \times k}$, $k \leqslant d$:

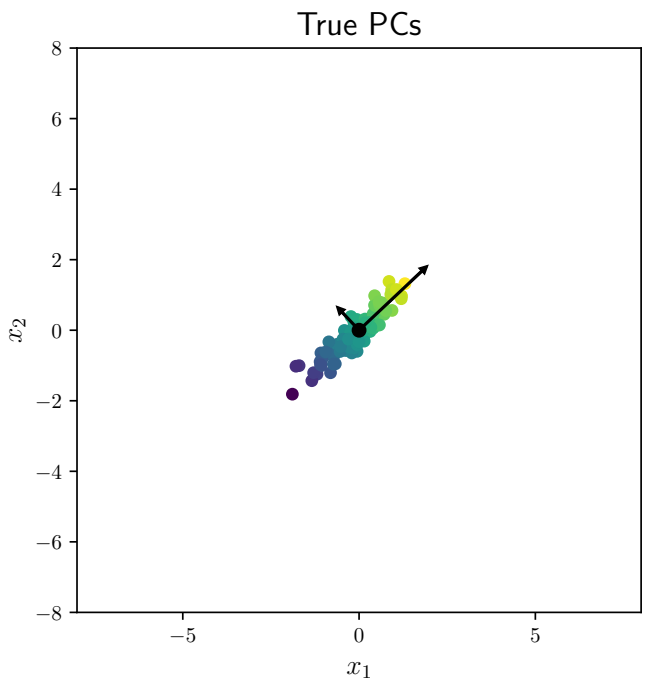
$$\begin{aligned}
\mathbb{E}[\|X - UU^\top X\|^2] &= \mathbb{E}[(X - UU^\top X)^\top (X - UU^\top X)] \\
&= \mathbb{E}[\|X\|^2 - 2\underbrace{\langle X, UU^\top X\rangle}_{=\langle U^\top X, U^\top X\rangle} + \underbrace{\|UU^\top X\|^2}_{=\|U^\top X\|^2}] \\
&= \mathbb{E}\left[\|X\|^2 - \|U^\top X\|^2\right] = \underbrace{\mathbb{E}\left[\|X\|^2\right]}_{\text{const. w.r.t. } U} - \mathbb{E}\left[\|U^\top X\|^2\right],
\end{aligned}$$

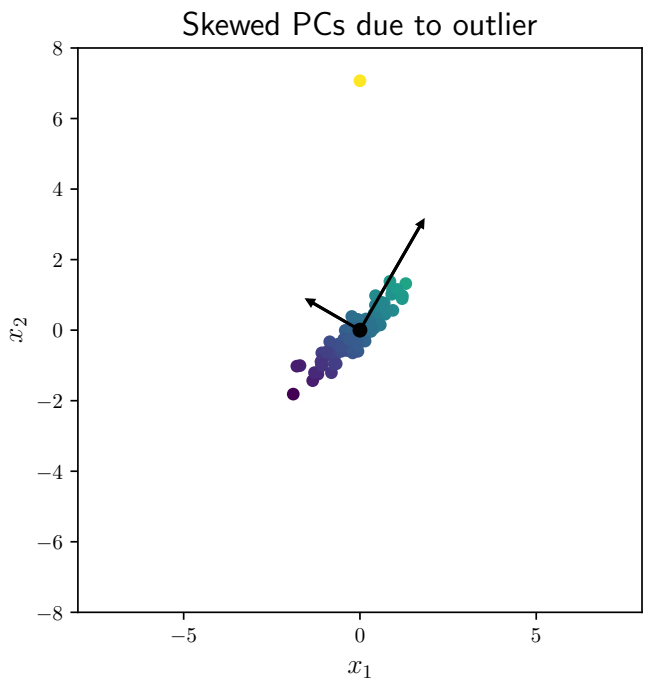
  where $\|UU^\top X\|^2 = \|U^\top X\|^2$ holds because the column of U are orthonormal to each other.

- That is, minimizing the projection error is equivalent to maximizing $\mathbb{E}\left[\|U^\top X\|^2\right]$ subject to the constraint $U^\top U = I$. For $U = \boldsymbol{u} \in \mathbb{R}^d$ and $\mathbb{E}[X] = \mathbf{0}$, this reduces to maximizing $\mathbb{E}\left[(\boldsymbol{u}^\top X)^2\right] = \mathrm{Var}[\boldsymbol{u}^\top X]$ subject to $\|\boldsymbol{u}\|^2 = 1$.

Is PCA Robust to Outliers?

One outlier vs. 100 "good" points.





Closed-Form Solution for PCA

- Finding the first principal component (maximum-variance formulation):

$$\underset{\boldsymbol{w}\in\mathbb{R}^d}{\text{minimize}} \;\; \boldsymbol{w}^\top \boldsymbol{\Sigma} \boldsymbol{w} \quad \text{subject to} \quad \|\boldsymbol{w}\|^2 = 1,$$

We now apply the method Lagrange multipliers

- **Step 1:** Build the Lagrangian

$$\mathcal{L}(\boldsymbol{w}, \lambda) = \boldsymbol{w}^\top \boldsymbol{\Sigma} \boldsymbol{w} + \lambda \cdot (\|\boldsymbol{w}\|^2 - 1)$$

- **Step 2:** Set gradient of Lagrangian to zero:

$$\nabla_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w}, \lambda) = \mathbf{0} \quad \Rightarrow \quad \boldsymbol{\Sigma} \boldsymbol{w} = \lambda \boldsymbol{w}$$

! PCA solution is an eigenvector of the covariance matrix Σ

(*Homework:* Show that it is the eigenvector with highest eigenvalue.)

Solving PCA (more components)

- Given an algorithm for finding the first principal component, we can easily adapt it to extract the remaining components by repeating the following two steps (i) compute the first principal component and (ii) remove it from the data.

Finding multiple PCA components for $X \in \mathbb{R}^{d \times N}$

for $i = 1$ **to** h **do**

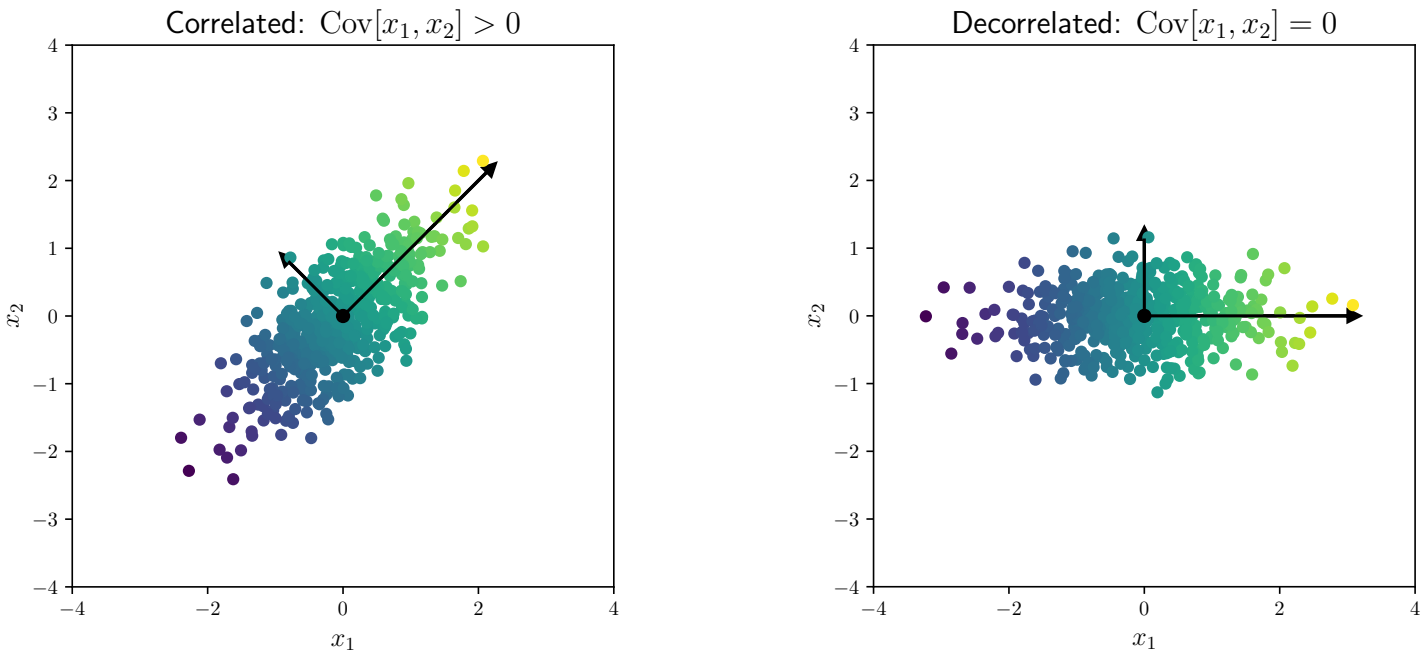
$\boldsymbol{w}_i \leftarrow \mathrm{PC}_1(X)$ (i)

$X \leftarrow X - \boldsymbol{w}_i \boldsymbol{w}_i^\top X$ (ii)

end for

- It can be shown that the resulting sequence of principal components $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_h$ corresponds to the leading eigenvectors of the estimated matrix $\widehat{\Sigma}$.
- Therefore, in practice, is is sufficient to have a procedure which only can extract the leading eigenvector of $\widehat{\Sigma}$.

Interpreting PCA: Decorrelation



- PCA rotates data into new coordinate system with the directions of largest variance being the new coordinate axes. As a result the corresponding variables become decorrelated, i.e., $\mathrm{Cov}[x_1, x_2] = 0$.

Stable PCA Computation via SVD

右奇异向量矩阵V给出了投影方向，即主成分。

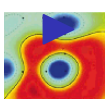
Singular Value Decomposition (SVD)

SVD factorizes a matrix $X \in \mathbb{R}^{d \times n}$ as a product $X = UDV^\top$ where

- $U \in \mathbb{R}^{d \times d}$ is orthogonal and contains the eigenvectors of $XX^\top$
- $V \in \mathbb{R}^{n \times n}$ is orthogonal and contains the eigenvectors of $X^\top X$
- The diagonal entries in $D = \text{diag}(\sigma_1, \ldots, \sigma_r) \in \mathbb{R}^{d \times n}$, where $r = \min(d, n)$, $\sigma_1 \geqslant \cdots \geqslant \sigma_r \geqslant 0$ are the singular values of X.
- The singular values $\sigma_i = \sqrt{\lambda_i}$ are the square roots of the eigenvalues λ_i of $XX^\top$ (or equivalently $X^\top X$).

Observation:

- SVD for $\tilde{X} := \frac{1}{\sqrt{N}} \bar{X}$ (here $\bar{X} \in \mathbb{R}^{d \times n}$ is centered!) provides
  - a matrix $U = (\boldsymbol{u}_1 | \ldots | \boldsymbol{u}_d)$ that contains the eigenvectors (or principal components) of the estimated covariance matrix $\widehat{C} = \frac{1}{N} \bar{X} \bar{X}^\top$ and
  - a matrix $D = \text{diag}(\sqrt{\lambda_1}, \ldots, \sqrt{\lambda_r})$ (note: square roots do not change the ordering of eigenvalues).



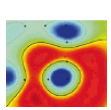
SVD is computationally faster and more stable than classical PCA

Comparing Classical PCA and SVD

In [2]:
```
# Prepare and center the data
N,d = 5,2
X = numpy.random.normal(0,1,[d,N])
Xc = X - X.mean(axis=1,keepdims=True)

# Classical PCA
C = numpy.dot(Xc,Xc.T)/N
L,U = numpy.linalg.eigh(C)
print('eigvals',L[::-1])
print('PCA1',U[:,-1],)
print('PCA2',U[:,-2],'\n')

# PCA via SVD
U,L,V = numpy.linalg.svd(Xc/N**.5,full_matrices=False)
print('eigvals',L**2)
print('PCA1',U[:,0])
print('PCA2',U[:,1])
```

```
eigvals [2.43755297 0.35240585]
PCA1 [-0.57228568  0.82005433]
PCA2 [-0.82005433 -0.57228568]

eigvals [2.43755297 0.35240585]
PCA1 [-0.57228568  0.82005433]
PCA2 [0.82005433 0.57228568]
```

- Eigenvalues are the **same**.
- PCA projection vector is the **same** (up to a sign flip).



PCA Computation via SVD, limitations

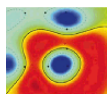
- The SVD has computational complexity of $\mathcal{O}(\min(N^2 d, d^2 N))$.
- When $d \approx N$, the complexity is roughly as bad as the that of computing the eigenvectors of $\widehat{\Sigma}$, that is, $\mathcal{O}(d^3)$.
- This makes the computation of principal components using SVD prohibitive for large datasets.

Note: 高得负担不起的

- In practice, we often need **only the first few principal components**, whereas SVD computes all of them.

Question:

- Can we design a faster procedure that only extracts the first few principal components?



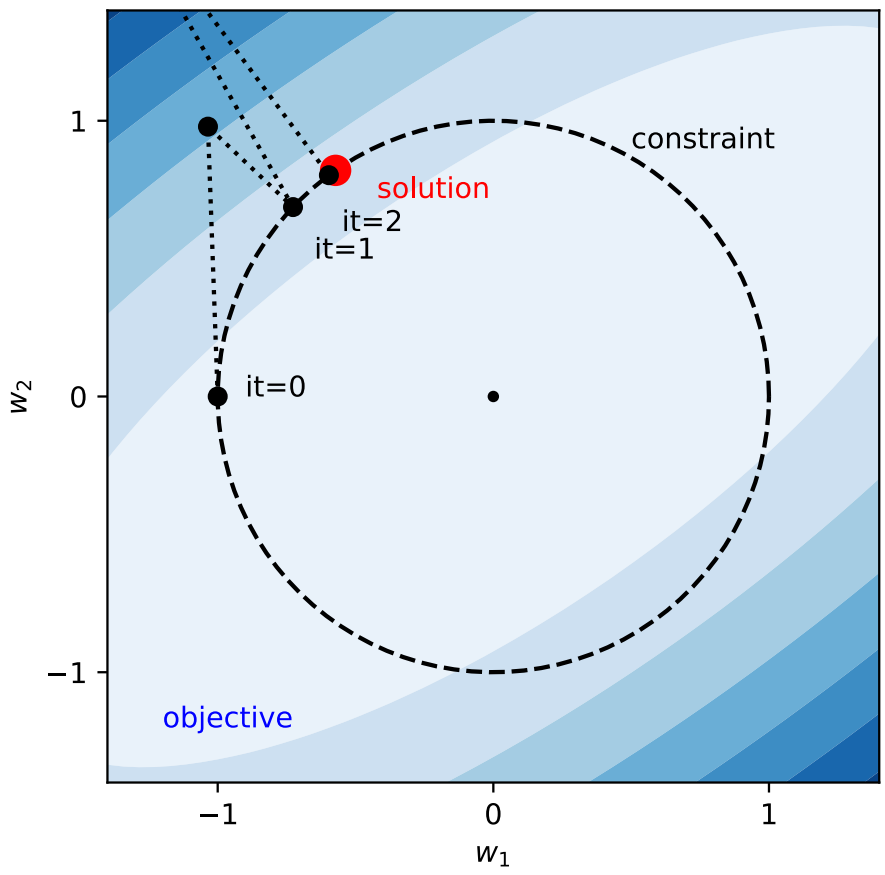
Power Iteration Algorithm

Find the first component by applying the *iteration*

$$\boldsymbol{v}^{(t)} \leftarrow \widehat{\boldsymbol{\Sigma}}\boldsymbol{w}^{(t-1)}$$

$$\boldsymbol{w}^{(t)} \leftarrow \frac{\boldsymbol{v}^{(t)}}{\|\boldsymbol{v}^{(t)}\|}$$

T times, starting from a random (or any) unit-norm vector $\boldsymbol{w}^{(0)}$.



Questions:

- Does it always converge? **yes**
- How fast it converges? **exponentially fast**

(Homework: prove this.)

Power Iteration Algorithm: In

幂迭代算法（Power Iteration Algorithm）：直觉

► 考虑一个对称矩阵 A 及其一组正交归一基 $\mathbf{v}_1, \ldots, \mathbf{v}_n$，这些基向量是其特征向量。

主成分 $\langle \mathbf{w}, \mathbf{v}_i \rangle \lambda_i$ 的大小决定了该方向上的伸缩程度，其中对应于**最大特征值（按绝对值计算）**的分量会比其他分量增长得更快。

经过归一化步骤，剩余分量相对减少，从而使得算法最终收敛于最大特征值对应的特征向量。

- Consider a symmetric matrix A and an orthonormal basis $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n$ consisting of eigenvectors. The magnitude $\langle \boldsymbol{w}, \boldsymbol{v}_i \rangle \lambda_i$ of the principal component corresponding to the highest eigenvalue (in magnitude) is stretched much stronger than the remaining components. The subsequent normalization step leads to a proportional reduction of the remaining components.

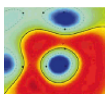
$$A\boldsymbol{w} = A\Big(\sum_{i=1}^{d} \langle \boldsymbol{w}, \boldsymbol{v}_i \rangle \boldsymbol{v}_i\Big) = \sum_{i=1}^{d} \langle \boldsymbol{w}, \boldsymbol{v}_i \rangle A\boldsymbol{v}_i = \sum_{i=1}^{d} \langle \boldsymbol{w}, \boldsymbol{v}_i \rangle \lambda_i \boldsymbol{v}_i.$$

► 结果：非主导方向的分量在每次迭代中逐渐变小。

► 幂迭代法并不总是收敛。然而，如果矩阵 A 是对称的，并且具有唯一的最大特征值（按绝对值计算），则收敛是可以保证的。

► 如果最大特征值的重数（即特征值的重复次数）大于 1，则算法仍可能收敛，但收敛的特征向量可能不是唯一的。

- As a result, the components of the non-dominant directions become smaller in each iteration.
- Power Iteration does not always converge. However, if the matrix A is symmetric and has a unique largest eigenvalue in magnitude, the convergence is guaranteed.
- If the largest eigenvalue has multiplicity greater than one, the algorithm still may converge – but not to a unique eigenvector.



Power Iteration (more components)

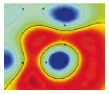
The power iteration (POWIT) method only gives us the leading eigenvector. If we want further PCA components, we need to compute them iteratively.

Naive approach (cf. slide 17): Recompute covariance at each step

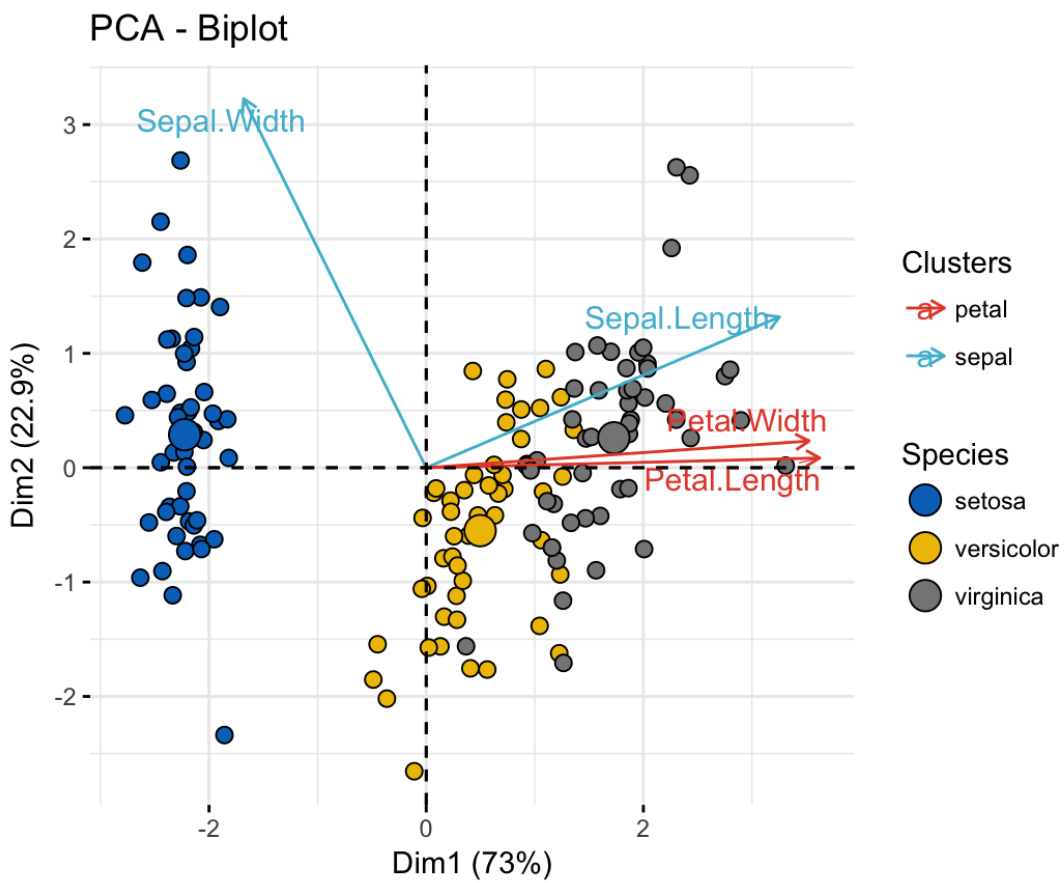
for $i = 1$ **to** h **do**
$\widehat{\Sigma} = \frac{1}{N} \bar{X} \bar{X}^\top$
$\boldsymbol{w}_i \leftarrow \mathrm{POWIT}(\widehat{\Sigma})$
$\bar{X} \leftarrow \bar{X} - \boldsymbol{w}_i \boldsymbol{w}_i^\top \bar{X}$
end for

Better approach: Apply changes directly to the covariance matrix

$\widehat{\Sigma} = \frac{1}{N} \bar{X} \bar{X}^\top$
for $i = 1$ **to** h **do**
$\boldsymbol{w}_i \leftarrow \mathrm{POWIT}(\widehat{\Sigma})$
$\widehat{\Sigma} \leftarrow \widehat{\Sigma} - \boldsymbol{w}_i \boldsymbol{w}_i^\top \widehat{\Sigma}$
end for



Applications of PCA: Data Visualization



- PCA can be used to visualize how examples of different classes (e.g. different species) are related, and whether they form clusters.
- Canonical coordinates representing individual features can also be projected in the PCA space. This gives a 'Biplot'.

在这个应用中，PCA通过将原始四个特征（花瓣和花萼的宽度和长度）转换为两个主成分，减少了数据的维度，使得原本难以可视化的高维数据可以直观地在二维空间中显示。

Applications of PCA: Data Compression

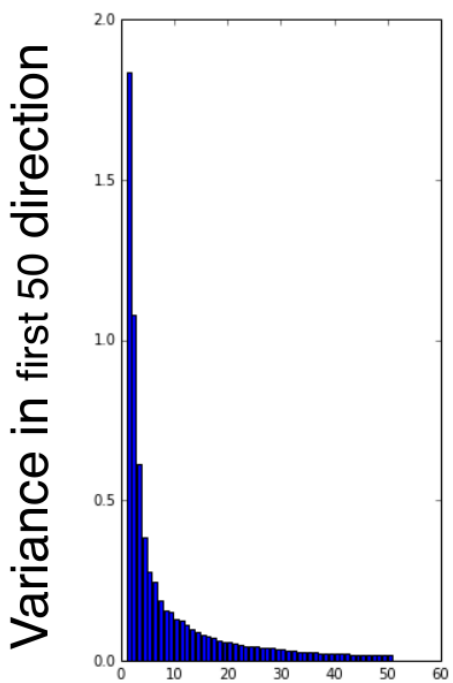
Idea: Faces look very similar in comparison to other random images. How many principle components would we need to accurately describe all faces?

An 64x64 pixel image of a face can be represented as a 4096 dimensional vector where each entry has the pixel's grayscale value.

Reference: Turk, Matthew A and Pentland, Alex P. Face recognition using eigenfaces. Computer Vision and Pattern Recognition, 1991.

Applications of PCA: Data Compression

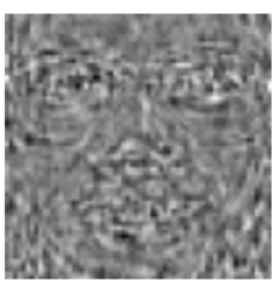
The principle components are directions in our faces space.
Thus, each principle component is a face representation, too.



Principle component 1

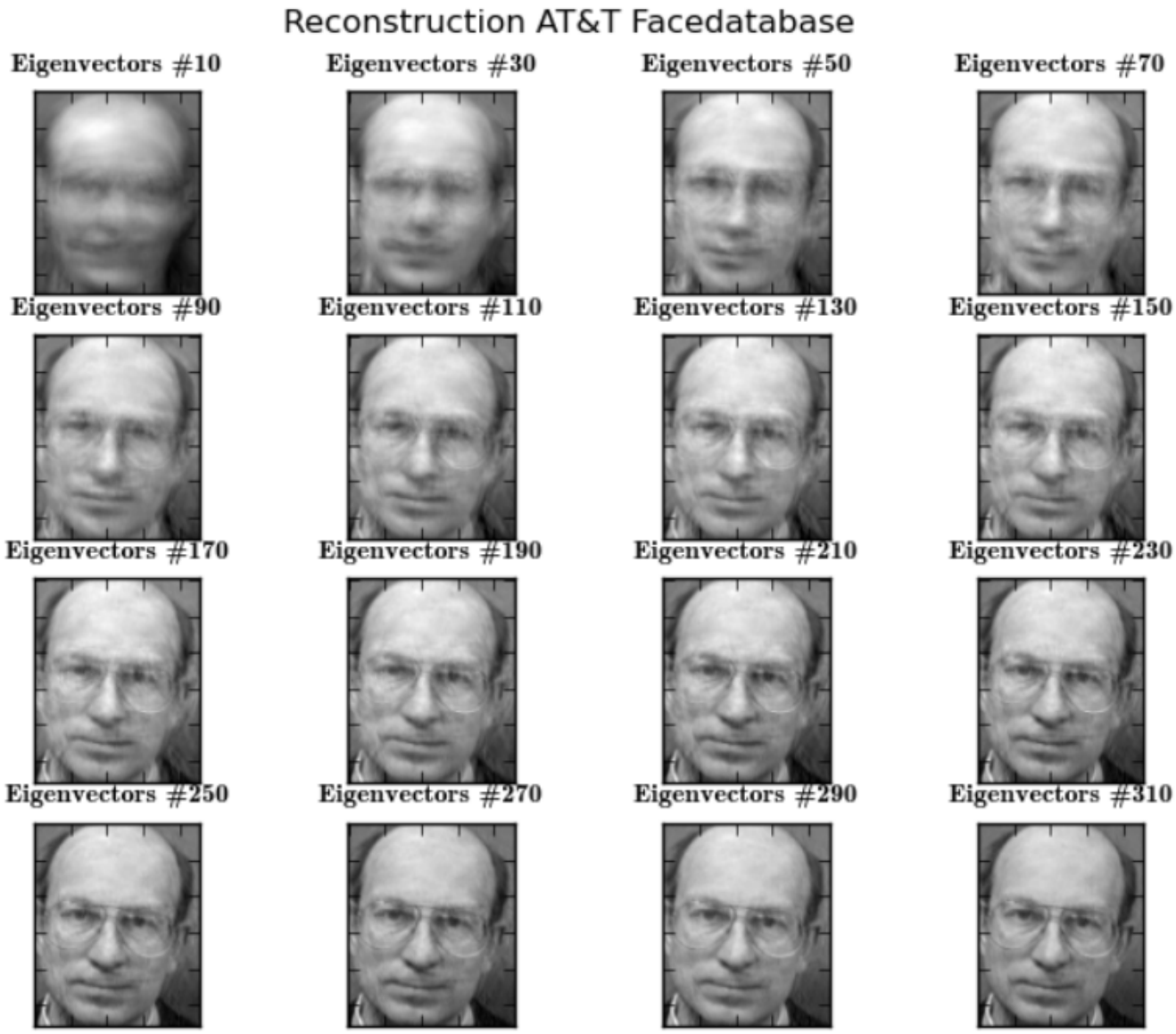
Principle component 2

Principle component 3



Principle component 350

Applications of PCA: Data Compression

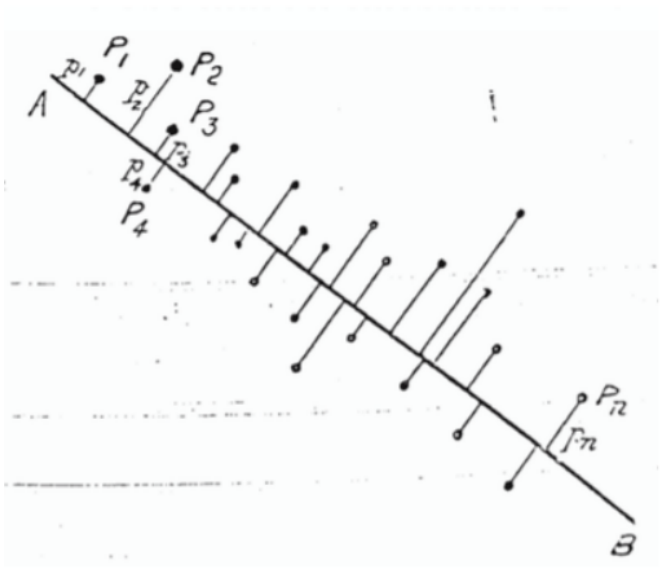


Applications of PCA: Denoising

We can use PCA to denoise data:

Step 1: Reduce dimensionality to filter out "noise" directions: $(w_1^{\mathrm{T}} x, \ldots, w_k^{\mathrm{T}} x)^{\mathrm{T}}$

Step 2: Project back into original space:



$$\sum_{i=1}^{k} (w_i^{\mathrm{T}} x) w_i$$

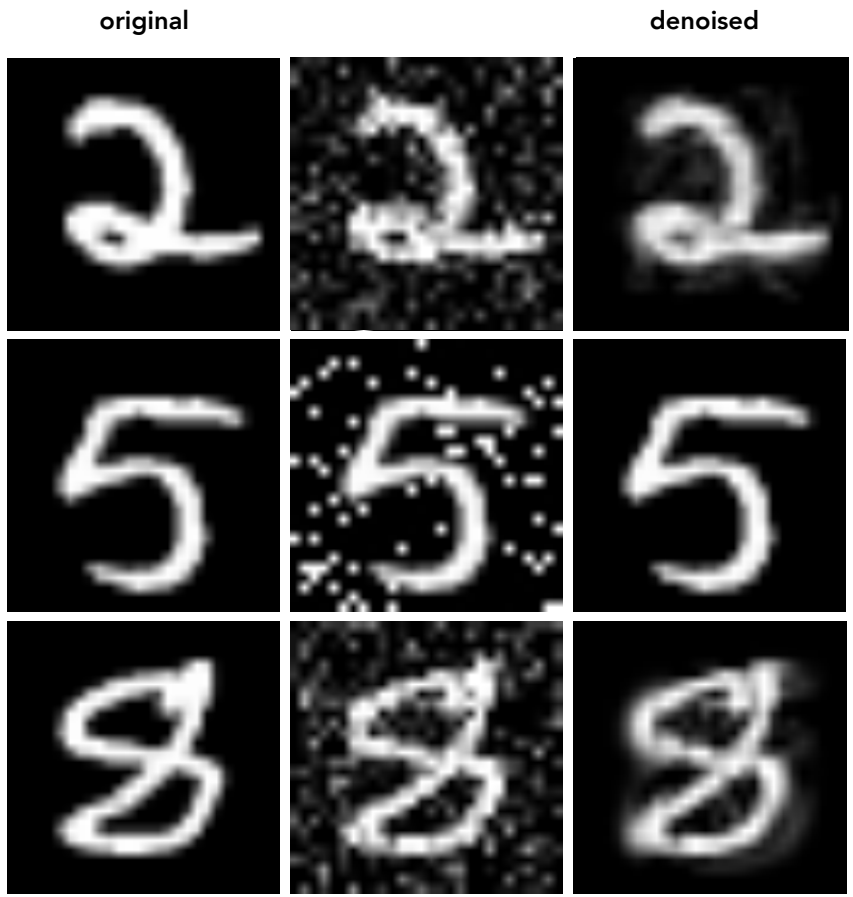
Step 3: Undo centering:

$$\sum_{i=1}^{k} (w_i^{\mathrm{T}} x) w_i + \sum_{i=1}^{n} x_i$$

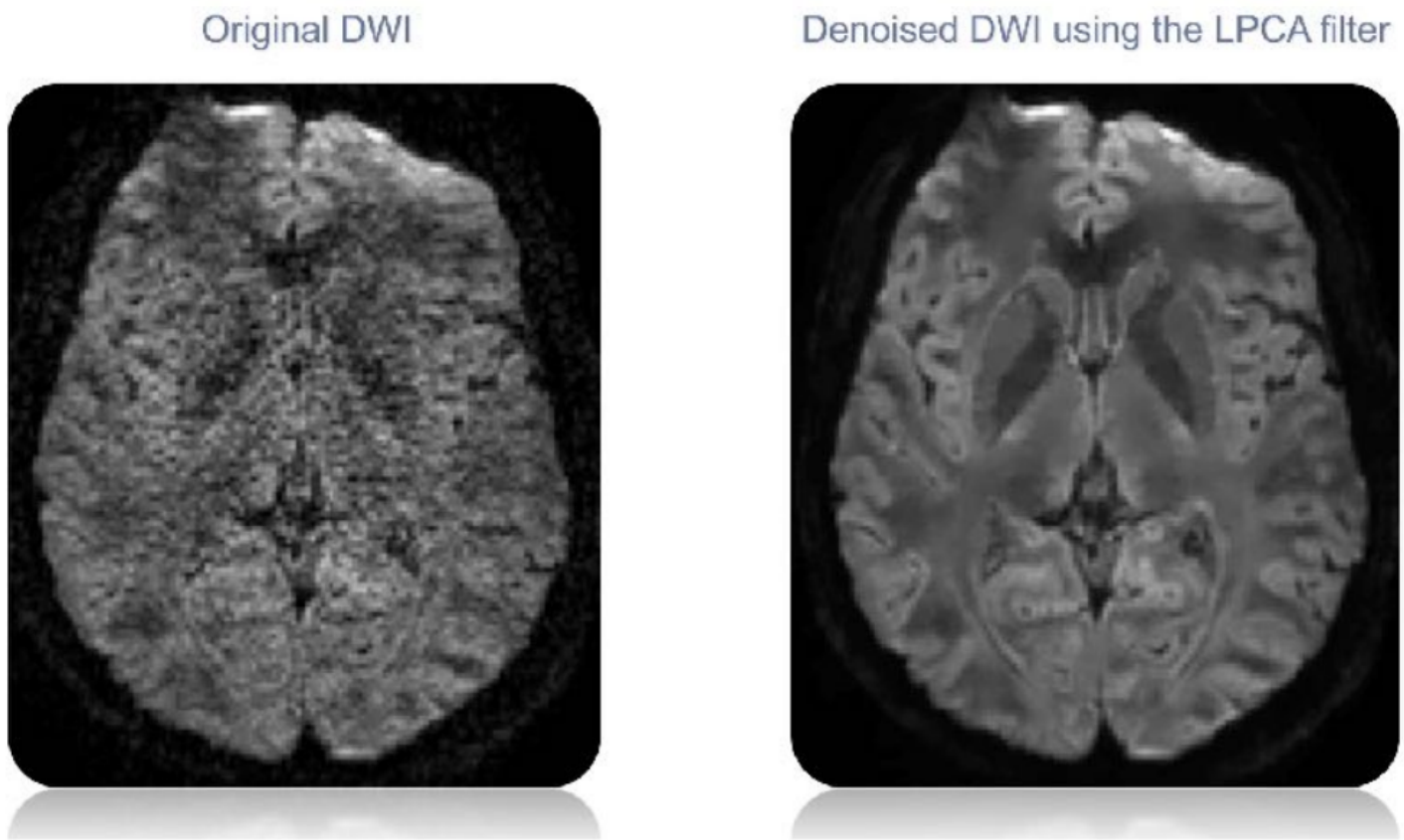
47

Applications of PCA: Denoising

- We project noisy images 28×28 images ($X \in [0,1]^{784}$) to a subspace of dimension $k = 100$:



Applications of PCA: Denoising

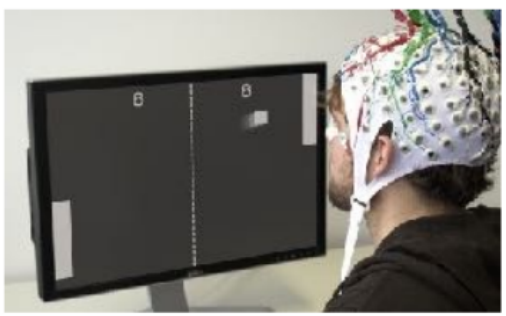


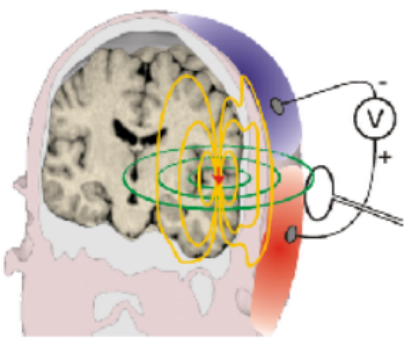
Mann et al., 2013

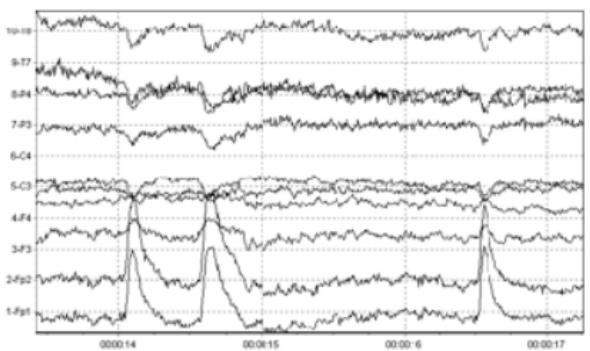
Applying PCA locally on the individual patches allows for som non-linearity in the PCA algorithm.

Applications of PCA: Artifact Removal

In electroencephalographic (EEG) recordings, eye blink artifacts can be stronger than the neuronal activity.

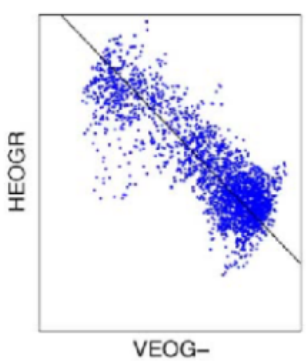






BCI2000

→ reasonable to remove first principal components



Blink

Applications of PCA: Portfolio Management

- Consider historical data of returns for the individual stocks over some time period $X \in \mathbb{R}^d$.
- PCA decomposes a set of stock returns into principal components that represent the main sources of variance (or risk) in the data. In the context of a portfolio, each component often aligns with an economic or market factor (e.g., broad market movement, sector trends, interest rate changes). The first PC describes the investment shares in the stocks with a highest volatility (or market risk).
- PCA creates uncorrelated principal components, allowing the portfolio to diversify across independent sources of risk and return. PCA identifies how correlated the stocks are by analyzing which stocks load heavily onto the same principal components.

PCA应用：投资组合管理

- 考虑个别股票在某段时间内的历史收益数据，记作 $X \in \mathbb{R}^d$。
- PCA将股票收益数据分解为主成分，这些主成分代表数据中方差（或风险）的主要来源。在投资组合的上下文中，每个成分通常与某个经济或市场因素对齐（例如，广泛的市场波动、行业趋势、利率变化）。第一个主成分（PC）描述了波动性（或市场风险）最高的股票的投资份额。
- PCA创建了不相关的主成分，使得投资组合可以在风险和收益的独立来源之间实现分散化。PCA通过分析哪些股票在主要成分上负载较重，识别出这些股票之间的相关性。

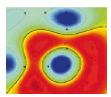
Beyond PCA

- **核PCA**：在特征空间中执行PCA（即对输入特征进行非线性变换）。这种方法可以实现更好的压缩/去噪效果。
- **CCA、ICA等**：不是最大化方差，而是最大化某些感兴趣的量（例如相关性、峰度等）。
- **自编码器网络**：PCA相当于一个线性自编码器网络。通过在自编码器中引入非线性，可以使分析变为非线性。
- **稀疏编码**：在约束条件下（例如稀疏性）最大化投影空间的重构。

Many methods that extend PCA or relate to it have been developed:

- **Kernel PCA**: Perform PCA in feature space (where input features are transformed nonlinearly). Gives better compression/denoising.
- **CCA, ICA, etc.** Does not maximize variance but some other quantity of interest (e.g. correlation, kurtosis, etc.)
- **Autoencoder Networks**. PCA is equivalent to a linear autoencoder network. The analysis can be made nonlinear by incorporating nonlinearities in the autoencoder.
- **Sparse Coding.** Maximize reconstruction subject to constraints in the projected space (e.g. sparsity).

Kernel PCA will be presented in ML1. Other approaches will be presented in ML2.



Summary

许多应用（例如提高计算的稳定性、数据压缩或去噪）都与减少维度的任务密切相关。
主成分分析（PCA）是一种实现Pearson（1901年）关于最小化噪声和最大化信号的原则的降维技术。（它同时做到了这两点！）
拉格朗日乘数框架表明，PCA的解是协方差矩阵的特征向量。
存在几种方法来计算主成分（例如SVD、幂迭代），选择哪种方法取决于考虑的场景，例如数据量大小和我们需要的PCA成分数量。
PCA有许多应用（例如可视化、压缩、去噪、伪影减少）。

- Many applications such as improving stability of computation, data compression, or denoising are closely related to the task of **reducing dimensionality**.
- **Principal Component Analysis** is a dimensionality reduction technique that implements [Pearson 1901]'s principle of minimizing **noise** and maximizing **signal**. (It does both simultaneously**!**).
- The framework of **Lagrange Multipliers** shows us that the solution of PCA are the **eigenvectors** of the covariance matrix.
- Several methods exist to compute principal components (e.g. **SVD**, **Power Iteration**) which one to choose depends on the considered scenario, e.g. how large is our data, how many PCA components we need.
- PCA has many **applications** (e.g. visualization, compression, denoising, artifact reduction).