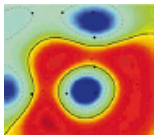
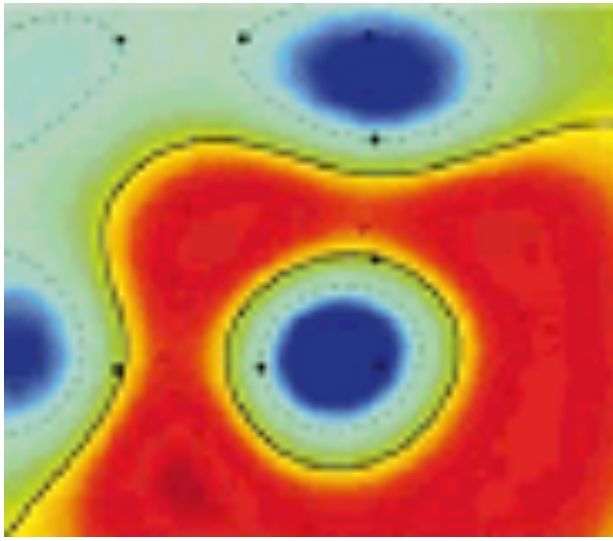


WiSe 2023/24

Machine Learning 1/1-X

Technische Universität Berlin

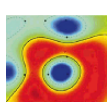


Lecture 4

Principal Component Analysis

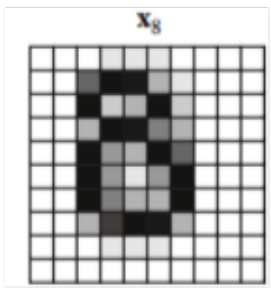
Outline

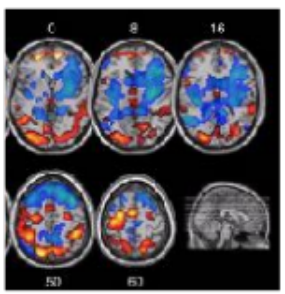
- Dimensionality reduction
- Principal Component Analysis (PCA)
  - What are principal components?
  - How to find/calculate them
    - Lagrange multipliers
    - Eigenvalues formulation
    - Iterative approaches
- What can we do with them? / Applications



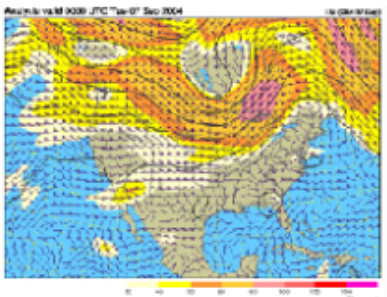
Curse of Dimensionality

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 $d \leq N$.

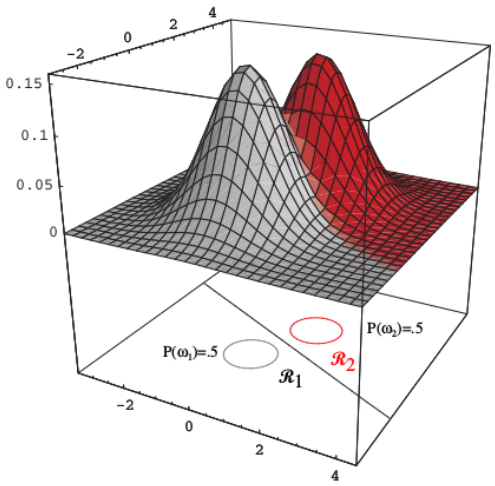
Example of ill-conditioning

Bayes Classifier with Estimated Covariance

Recap:

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

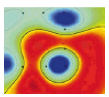
$$\begin{aligned} &\arg\max_j \{P(\omega_j|\mathbf{x})\} \\ &= \arg\max_j \{\log p(\mathbf{x}|\omega_j) + \log P(\omega_j)\} \\ &= \arg\max_j \left\{\mathbf{x}^\top \Sigma^{-1}\boldsymbol{\mu}_j - \frac{1}{2}\boldsymbol{\mu}_j \Sigma^{-1}\boldsymbol{\mu}_j + \log P(\omega_j)\right\} \end{aligned}$$



- 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

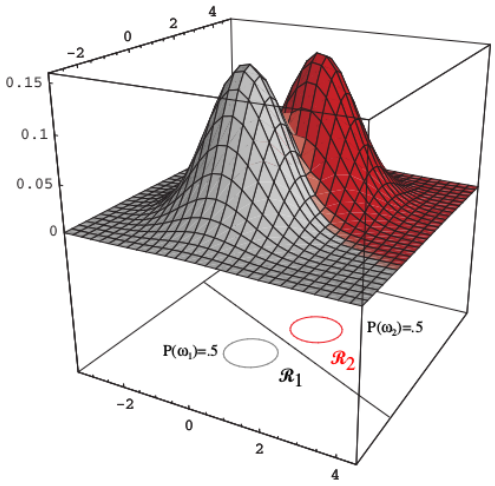
Bayes Classifier with Estimated Covariance

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:

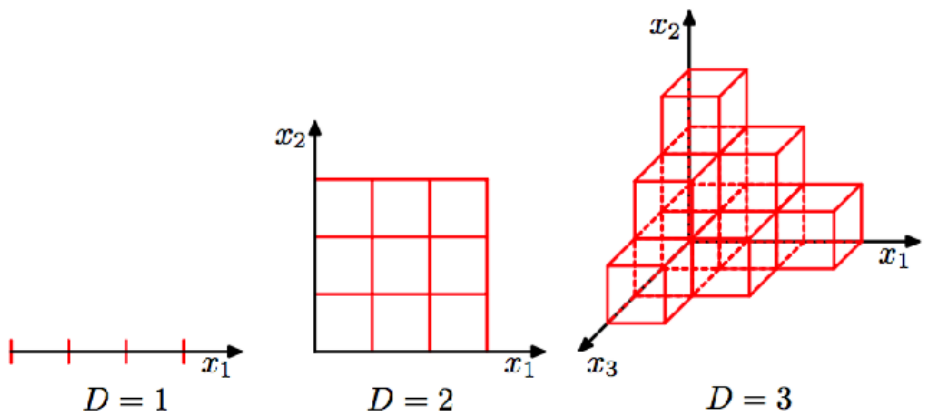
- The matrix $\widehat{\Sigma}^{-1}$ used in the classifier only exists when $d \leq N$ and the matrix $\bar{X}$ has full rank.
- The matrix $\widehat{\Sigma}^{-1}$ used in the classifier is typically accurate only when $d \ll N$.

Solutions:

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

Curse of Dimensionality (2)

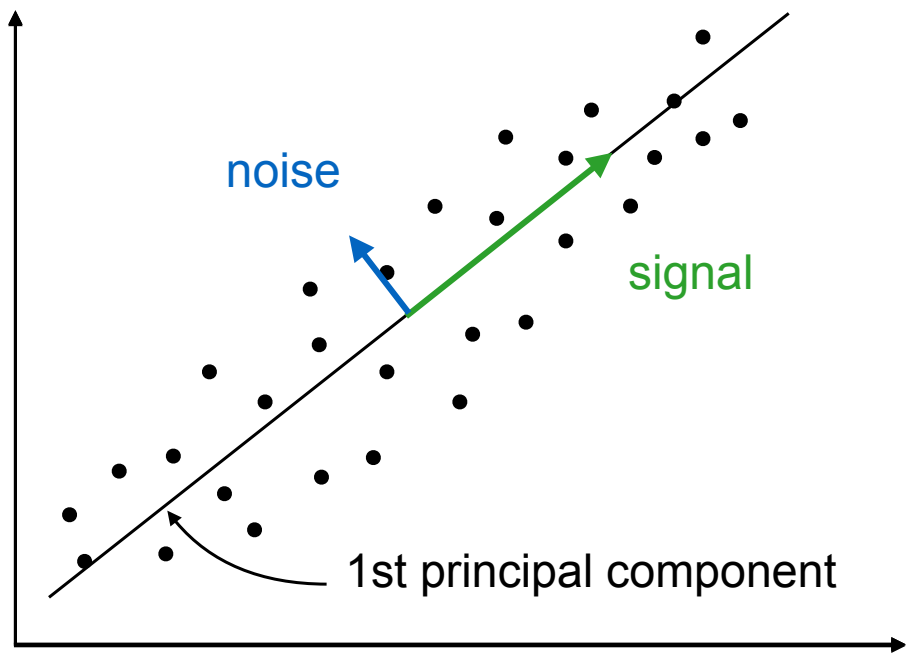
- When the dimensionality increases, the volume of the space increases so fast that the available data becomes sparse.



- 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).

Principal Component Analysis (PCA)

PCA is an analysis that linearly maps the data from a high-dimensional space to a low-dimensional space. In the example below, from a multi-dimensional space (2d) to a line (1d).



Which line fits data best?
The line **w** that minimizes the noise and maximizes the signal [Pearson 1901].

Reference: Bishop, Pattern Recognition and Machine Learning, Chapter 12.1

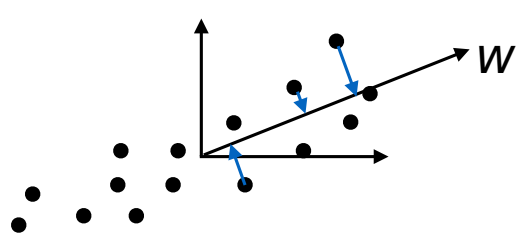
PCA as Noise Minimization

- Given a dataset $\mathbf{x}_1, \ldots, \mathbf{x}_N \in \mathbb{R}^d$, find a one-dimensional subspace, so that the data projected on that subspace has **minimum distance** to the original data.
- The reconstruction model is given by:

$$\widehat{\mathbf{x}} = \mathbf{w}\mathbf{w}^\top \mathbf{x}$$

  with $\|\mathbf{w}\| = 1$.
- Minimizing the noise (i.e. reconstruction error) can be written as:

$$\arg\min_{\mathbf{w}} \left[\frac{1}{N} \sum_{k=1}^{N} \| \mathbf{x} - \underbrace{\mathbf{w}\mathbf{w}^\top \mathbf{x}}_{\widehat{\mathbf{x}}} \|^2 \right]$$



PCA as Signal Maximization

- Given a dataset $\mathbf{x}_1, \ldots, \mathbf{x}_N \in \mathbb{R}^d$. Find a one-dimensional subspace, so that the data projected has **maximum variance**.
- The projection model is given by:

$$h = \mathbf{w}^\top \mathbf{x}$$

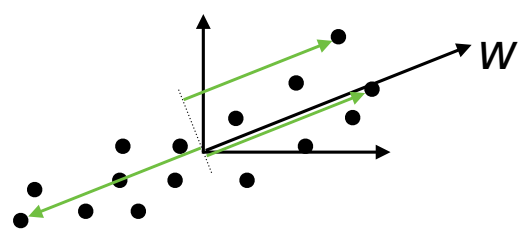
with $\|\mathbf{w}\| = 1$.

- Maximizing the signal (i.e. variance of projected data) can be written as:

$$\arg\max_{\mathbf{w}} \left[\frac{1}{N} \sum_{k=1}^{N} (\mathbf{w}^\top \mathbf{x}_k - \mathbb{E}[\mathbf{w}^\top \mathbf{x}])^2 \right]$$

and assuming centered data, i.e. $\mathbb{E}[\mathbf{x}] = \mathbf{0}$, it simplifies to:

$$arg\max_{\mathbf{w}} \left[\frac{1}{N} \sum_{k=1}^{N} (\mathbf{w}^\top \mathbf{x}_k)^2 \right]$$



From Min Noise to Max Signal

$$\arg\min_{\mathbf{w}} \left[\frac{1}{N}\sum_{k=1}^{N} \|\mathbf{x}_k - \mathbf{w}\mathbf{w}^\top \mathbf{x}_k\|^2\right]$$

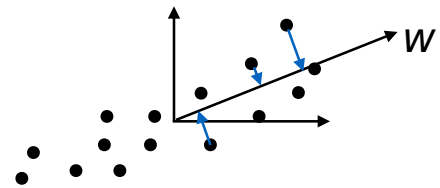
$$= \arg\min_{\mathbf{w}} \left[\frac{1}{N}\sum_{k=1}^{N} (\mathbf{x}_k - \mathbf{w}\mathbf{w}^\top \mathbf{x}_k)^\top (\mathbf{x}_k - \mathbf{w}\mathbf{w}^\top \mathbf{x}_k)\right]$$

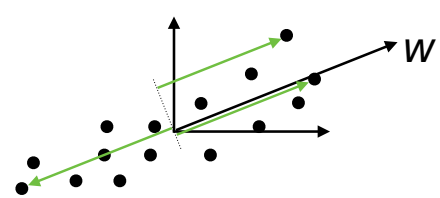
$$= \arg\min_{\mathbf{w}} \left[\frac{1}{N}\sum_{k=1}^{N} \underbrace{\mathbf{x}_k^\top \mathbf{x}_k}_{\text{cst.}} - 2\mathbf{x}_k^\top \mathbf{w}\mathbf{w}^\top \mathbf{x}_k + (\mathbf{w}\mathbf{w}^\top \mathbf{x}_k)^\top (\mathbf{w}\mathbf{w}^\top \mathbf{x}_k)\right]$$

$$= \arg\min_{\mathbf{w}} \left[\frac{1}{N}\sum_{k=1}^{N} -2(\mathbf{x}_k^\top \mathbf{w})^2 + \mathbf{x}_k^\top \mathbf{w}\underbrace{\mathbf{w}^\top \mathbf{w}}_{=1}\mathbf{w}^\top \mathbf{x}_k\right]$$

$$= \arg\min_{\mathbf{w}} \left[\frac{1}{N}\sum_{k=1}^{N} -2(\mathbf{x}_k^\top \mathbf{w})^2 + (\mathbf{x}_k^\top \mathbf{w})^2\right]$$

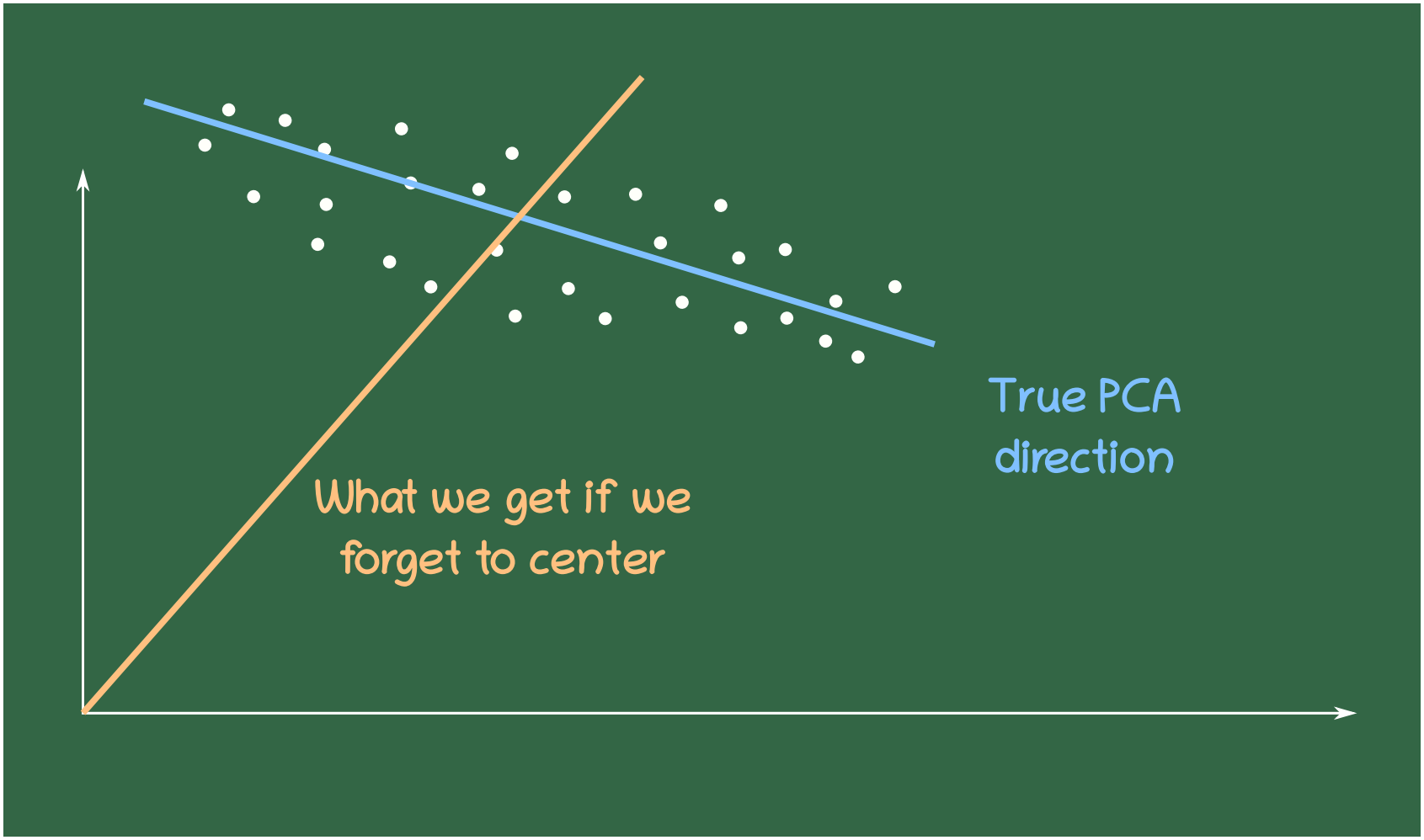
$$= \arg\max_{\mathbf{w}} \left[\frac{1}{N}\sum_{k=1}^{N} (\mathbf{x}_k^\top \mathbf{w})^2\right]$$



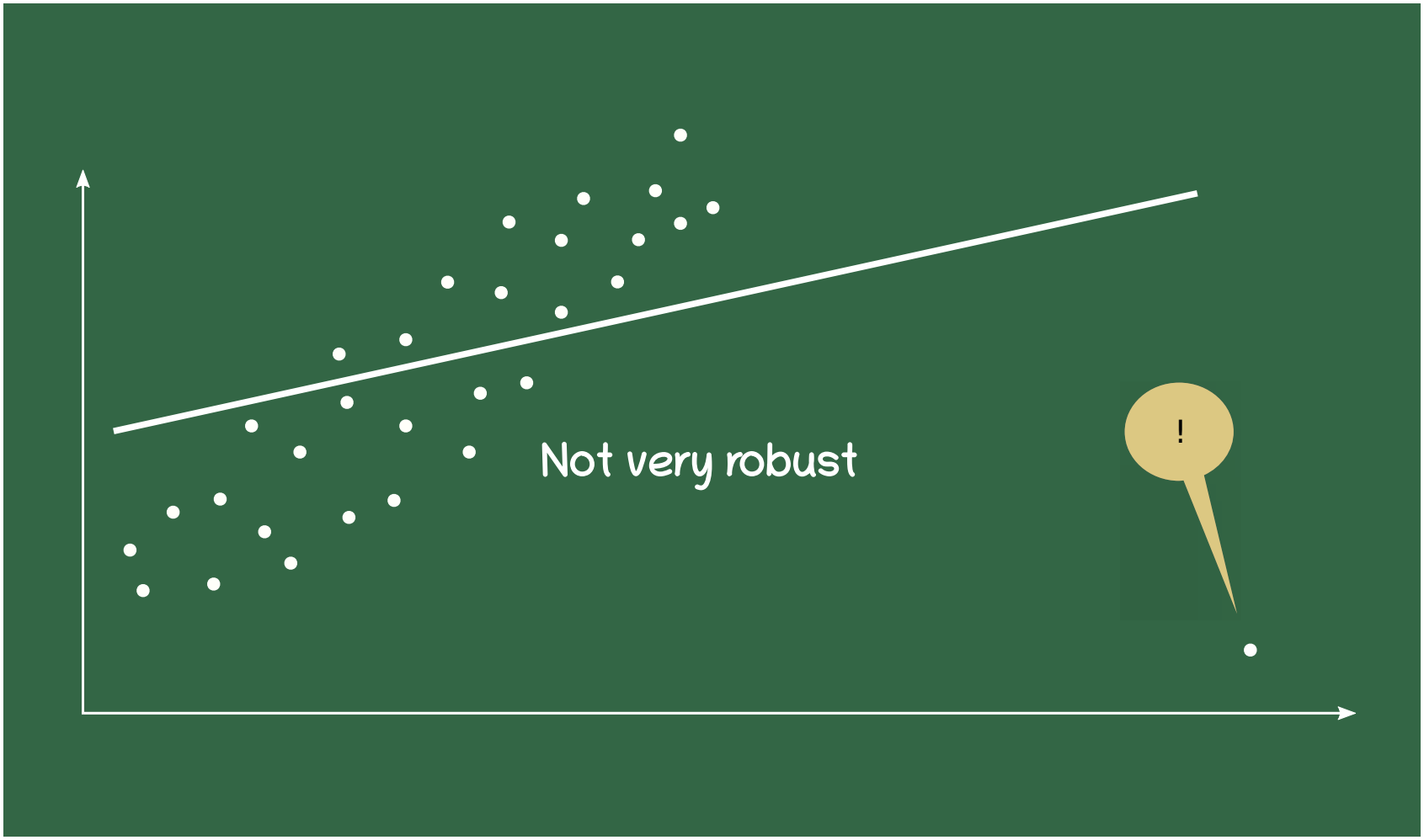


⚠ **Data needs to be centered**

What if the Data is Uncentered?

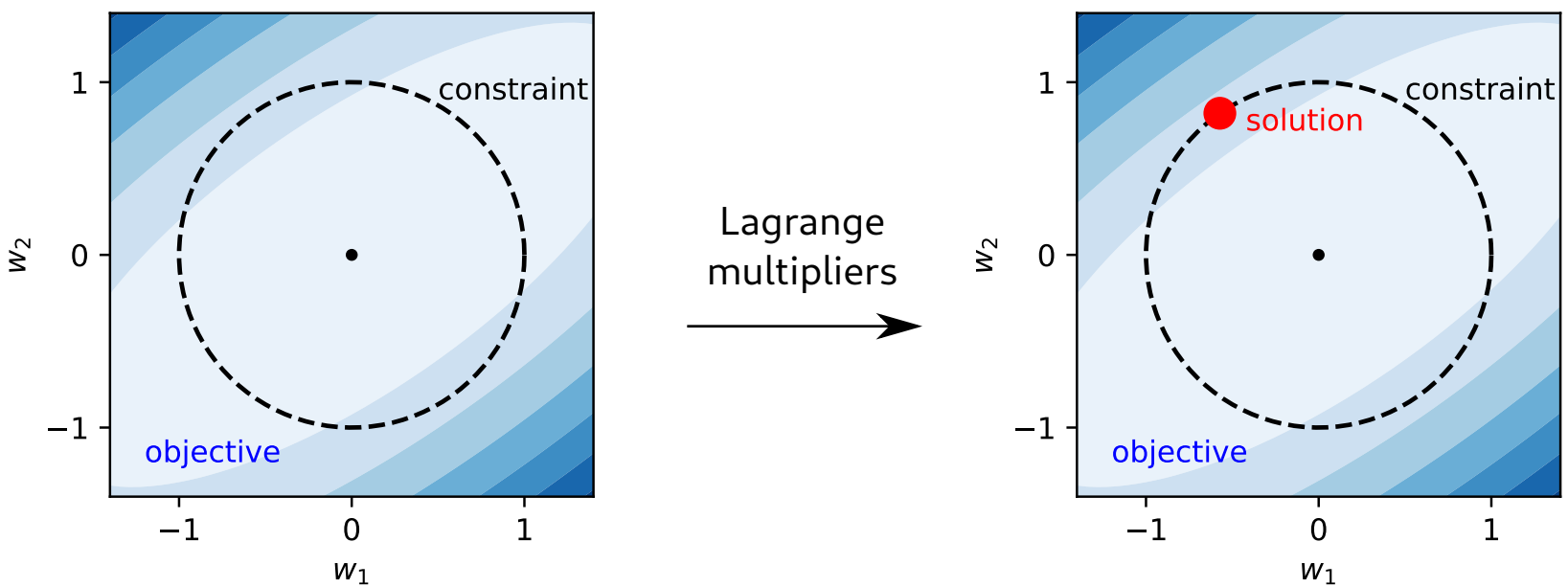


Is PCA Robust to Outliers?



Computing Principal Components

- So far, we have formulated PCA as maximizing some objective $f(\mathbf{w})$ subject to a constraint ($\|\mathbf{w}\| = 1$), but we do not have yet a procedure to quickly find the solution.
- Now, we will use the method of **Lagrange Multipliers** to effectively find a solution $\mathbf{w}$.



Method of Lagrange Multipliers

General framework for finding solutions of constrained optimization problems of the type $\arg\max_\theta f(\theta)$ subject to $g(\theta) = 0$.

- **Step 1:** Construct the 'Lagrangian':

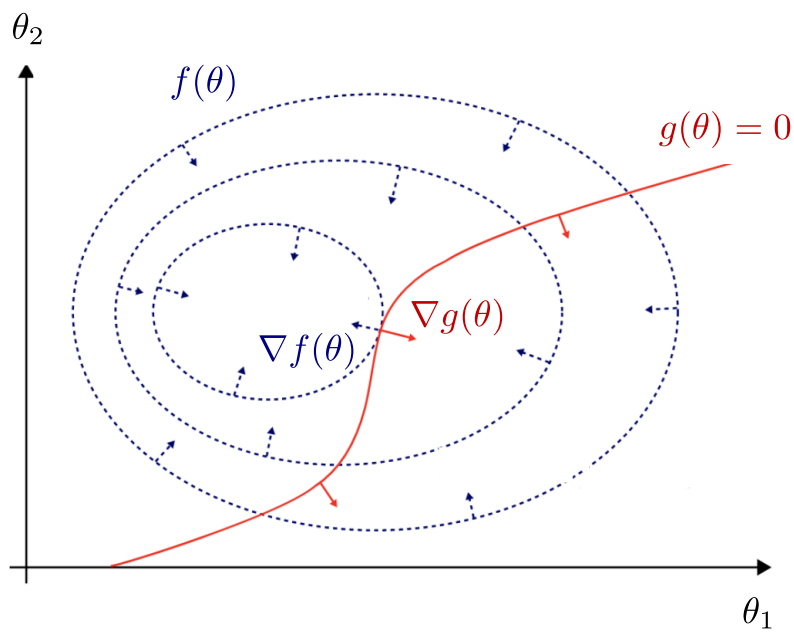
$$\mathcal{L}(\theta, \lambda) = f(\theta) + \lambda g(\theta)$$

  where λ is called the Lagrange multiplier.

- **Step 2:** Solve the equation

$$\nabla \mathcal{L}(\theta, \lambda) = 0$$

  which is a necessary condition for the solution.



Intuition for step 2: the equation includes the equation $\nabla f(\theta) = -\lambda \nabla g(\theta)$, i.e. the gradient of objective and constraint are aligned, but point in opposite directions (cf. 2d plot).

Lagrange Multipliers: 2D Example

Consider the constrained optimization problem

$$\boxed{\arg\max_{\theta} \left[1 - (\theta_1^2 + \theta_2^2)\right] \quad \text{s.t.} \quad \theta_1 + \theta_2 = 1}$$

- **Step 1:** Build the Lagrangian

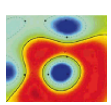
$$\mathcal{L}(\theta, \lambda) = 1 - (\theta_1^2 + \theta_2^2) + \lambda \cdot (\theta_1 + \theta_2 - 1)$$

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

$$\begin{aligned} \nabla_{\theta_1} \mathcal{L}(\theta, \lambda) = 0 \quad &\Rightarrow \quad 2\theta_1 = \lambda \\ \nabla_{\theta_2} \mathcal{L}(\theta, \lambda) = 0 \quad &\Rightarrow \quad 2\theta_2 = \lambda \\ \nabla_{\lambda} \mathcal{L}(\theta, \lambda) = 0 \quad &\Rightarrow \quad \theta_1 + \theta_2 = 1 \end{aligned}$$

which gives us the solution:

$$\theta = (0.5, 0.5)$$



The Solution of PCA

- Before applying Lagrange Multipliers, we first observe that the PCA optimization problem given in slide 8:

$$\arg\max_{\mathbf{w}} \left[\frac{1}{N} \sum_{k=1}^{N} (\mathbf{x}_k^\top \mathbf{w})^2 \right] \qquad \text{s.t.} \quad \|\mathbf{w}\| = 1$$

can be rewritten as:

$$= \arg\max_{\mathbf{w}} \left[\frac{1}{N} \sum_{k=1}^{N} (\mathbf{w}^\top \mathbf{x}_k)(\mathbf{x}_k^\top \mathbf{w}) \right] \qquad \text{s.t.} \quad \|\mathbf{w}\|^2 = 1$$

$$= \arg\max_{\mathbf{w}} \left[\mathbf{w}^\top \underbrace{\left(\frac{1}{N} \sum_{k=1}^{N} \mathbf{x}_k \mathbf{x}_k^\top \right)}_{\widehat{\Sigma}} \mathbf{w} \right] \qquad \text{s.t.} \quad \|\mathbf{w}\|^2 = 1$$

where $\widehat{\Sigma}$ is the covariance matrix (recall: the data is assumed to be centered). The covariance matrix does not depend on $\mathbf{w}$ and can be precomputed.

The Solution of PCA

Starting from our rewritten PCA optimization problem

$$\arg\max_{\mathbf{w}} \left[\mathbf{w}^\top \widehat{\Sigma}\mathbf{w}\right] \quad \text{s.t.} \quad \|\mathbf{w}\|^2 = 1$$

we look for a solution by applying the method of Lagrange multipliers.

- **Step 1:** Build the Lagrangian

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

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

$$\begin{aligned} \nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}, \lambda) = \mathbf{0} \quad &\Rightarrow \quad \widehat{\Sigma}\mathbf{w} = \lambda\mathbf{w} \\ \nabla_{\lambda} \mathcal{L}(\mathbf{w}, \lambda) = 0 \quad &\Rightarrow \quad \|\mathbf{w}\|^2 = 1 \end{aligned}$$

! PCA solution is an eigenvector of $\widehat{\Sigma}$

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

Solving PCA (more components)

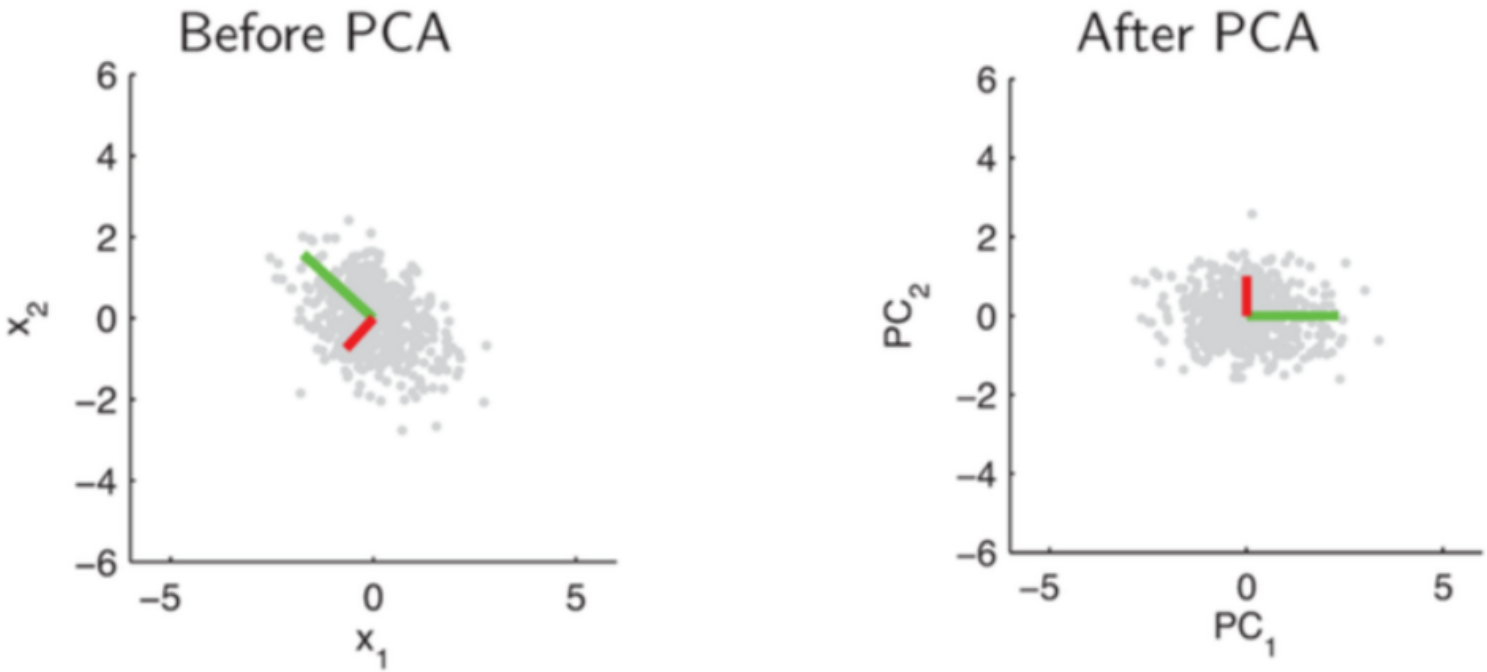
- PCA can be generalized to extracting h components $\mathbf{w}_1, \ldots, \mathbf{w}_h$.
- Conceptually it can be described by the following algorithm, which repeats multiple times the steps (i) compute the first principal component and (ii) remove it from the data.

Finding multiple PCA components

for $i = 1$ **to** h **do**
  $\mathbf{w}_i \leftarrow \mathrm{PC}_1(\bar{X})$ (i)
  $\bar{X} \leftarrow \bar{X} - \mathbf{w}_i \mathbf{w}_i^\top \bar{X}$ (ii)
end for

- It can be shown that the resulting sequence of principal components $\mathbf{w}_1, \ldots, \mathbf{w}_h$ corresponds to the leading eigenvectors of the matrix $\widehat{\Sigma}$.
- Therefore, in practice, we just need to compute the leading eigenvectors of $\widehat{\Sigma}$, without an iterative procedure.

Interpreting PCA



- PCA rotates data into new coordinate system with the directions of largest variance being the new coordinate axes.

Stable PCA Computation via SVD

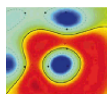
Singular Value Decomposition (SVD)

A singular value decomposition factorizes a matrix M as $U\Lambda V = M$ where

- U contains the eigenvectors of $MM^\top$
- V contains the eigenvectors of $M^\top M$
- The square roots of the eigenvalues of $MM^\top$ are on the diagonal of Λ.

Observation:

- Setting $M = \frac{1}{\sqrt{N}}\bar{X}$ and computing SVD results in
  - a matrix $U = (\mathbf{w}_1 | \ldots | \mathbf{w}_d)$ that contains the eigenvectors of $\widehat{\Sigma}$, i.e. the principal components, and
  - a matrix $\Lambda = \mathrm{diag}(\sqrt{\lambda_1}, \ldots, \sqrt{\lambda_d})$ (but applying the square root does not change the ordering of eigenvalues).
- SVD is computationally faster and more stable than covariance computation + eigenvalue decomposition.

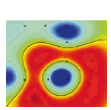


Comparing Classical PCA and SVD

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:

- 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*

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

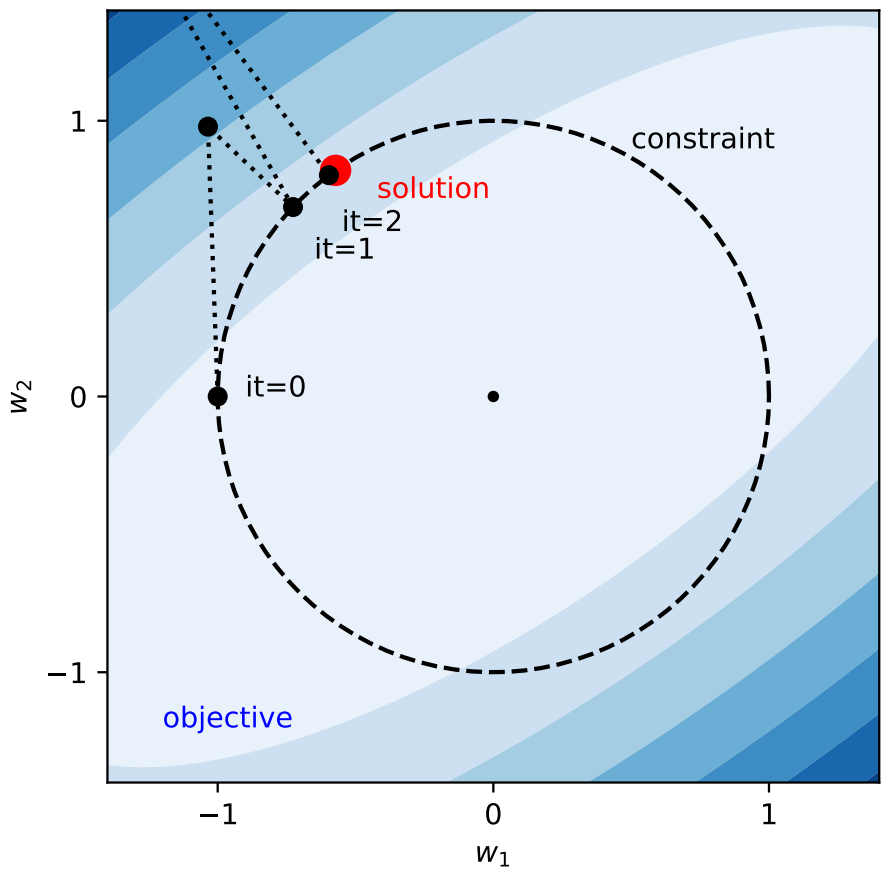
$$\mathbf{w}^{(t)} \leftarrow \frac{\mathbf{v}^{(t)}}{\|\mathbf{v}^{(t)}\|}$$

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

Questions:

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

(Homework: prove this.)



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.

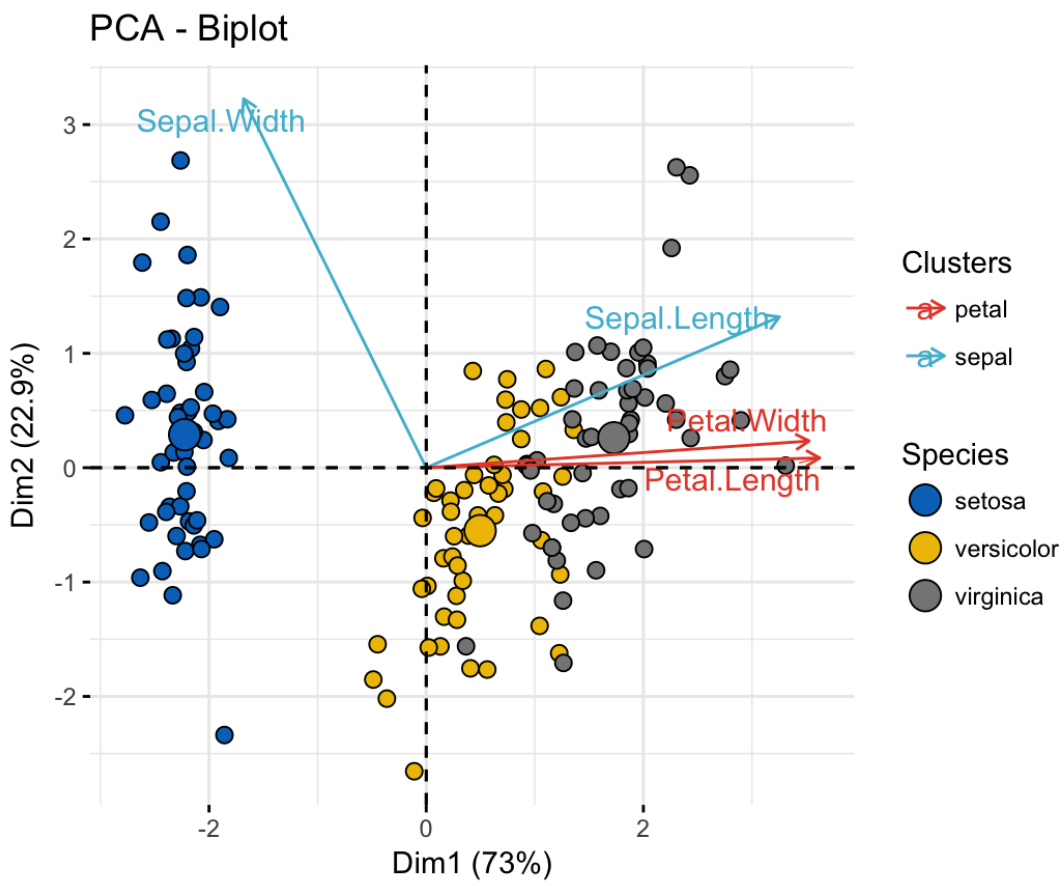
Naive approach (cf. slide 17): Recompute covariance at each step

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

Better approach: Work directly in covariance space

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

Applications of PCA: Data Compression

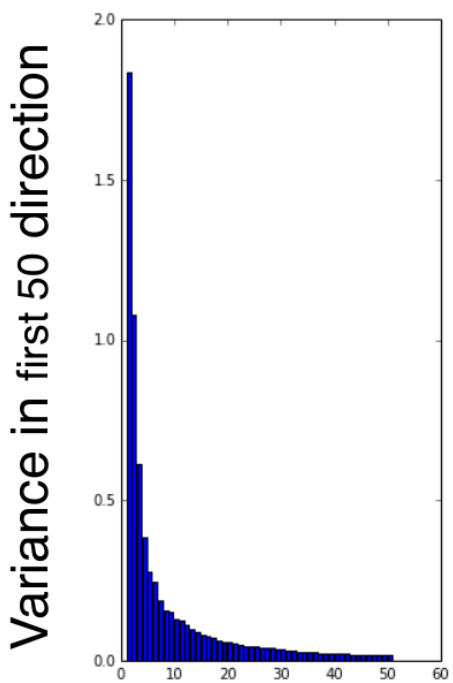
Idea: Faces look very similar in comparison to other random images. How many principle components would we need to accurately describe all faces?

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

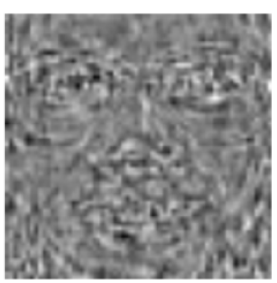
The principle components are directions in our faces space.
Thus, each principle component is a face representation, too.



Principle component 1

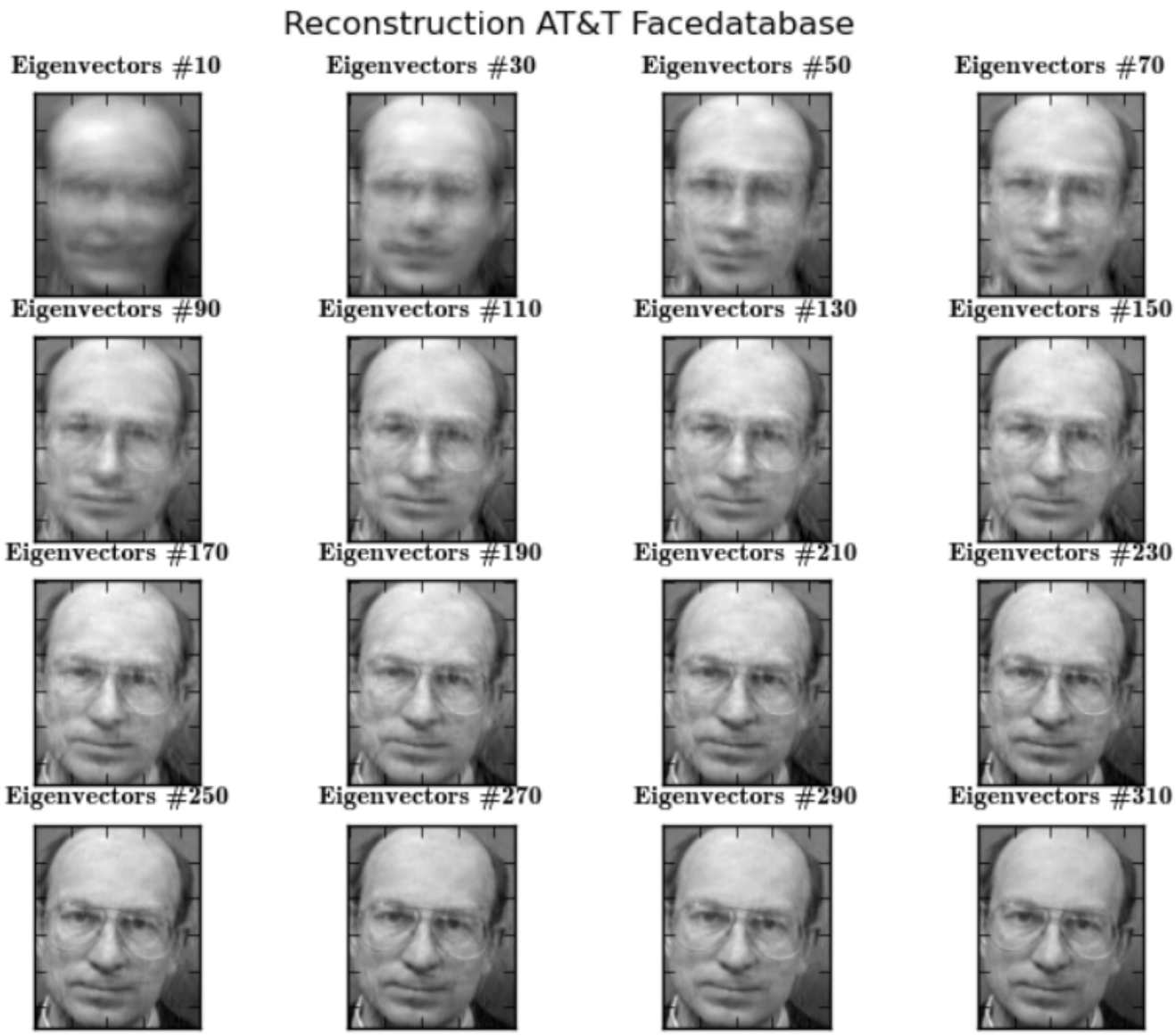
Principle component 2

Principle component 3



Principle component 350

Applications of PCA: Data Compression



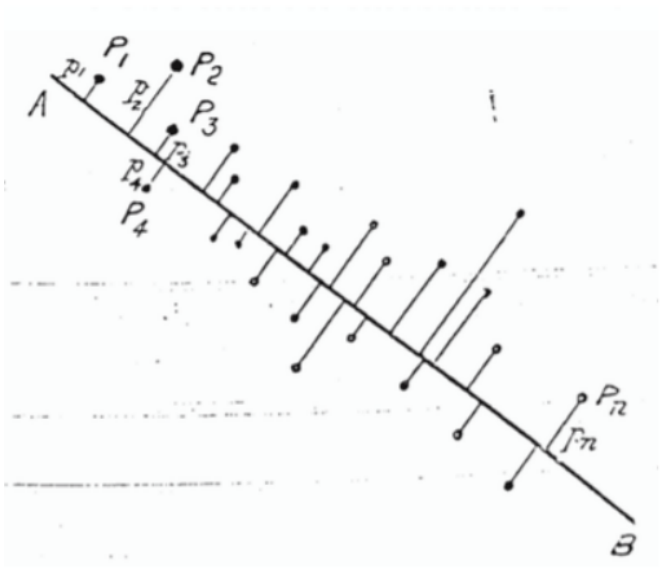
Applications of PCA: Denoising

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:

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



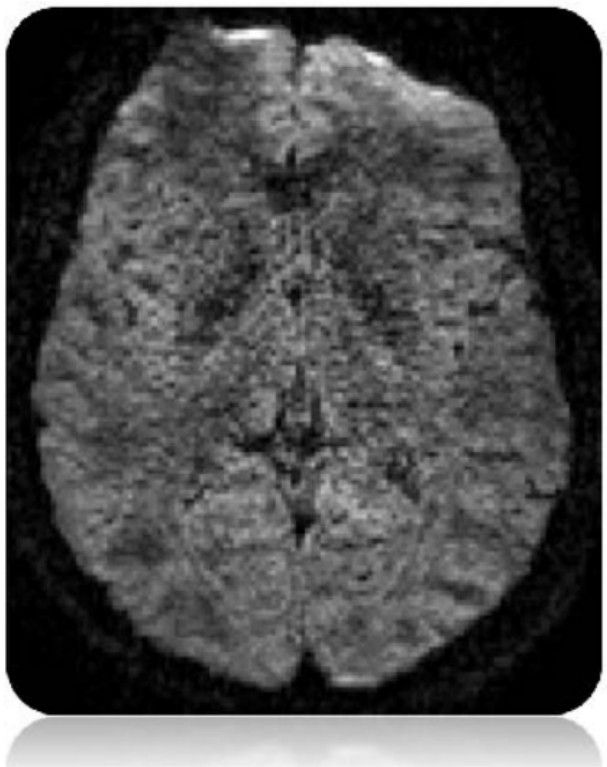
47

Step 3: Undo centering:

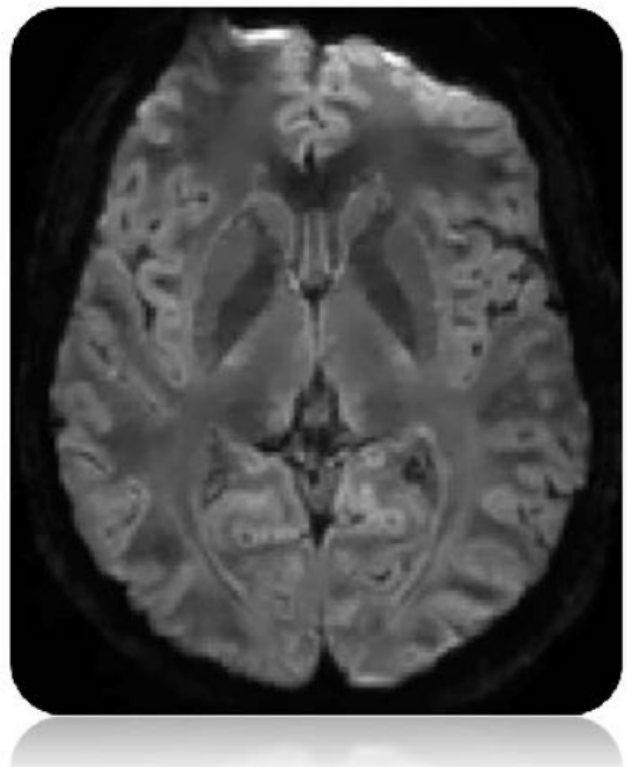
$$\sum_{i=1}^{k} (w_i^{\mathrm{T}} x) w_i + \sum_{i=1}^{n} x_i$$

Applications of PCA: Denoising

Original DWI



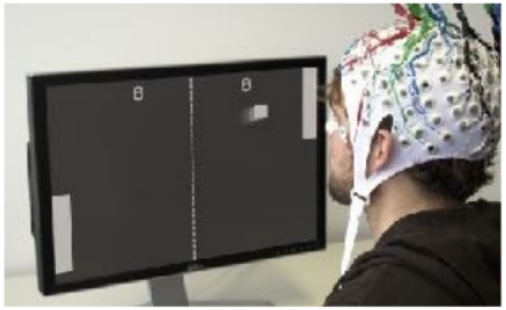
Denoised DWI using the LPCA filter

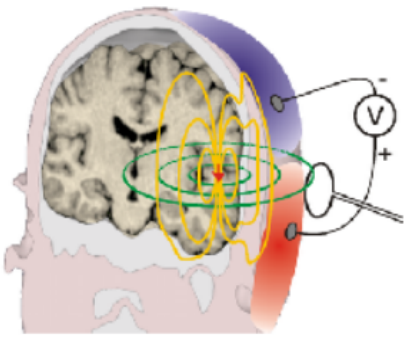


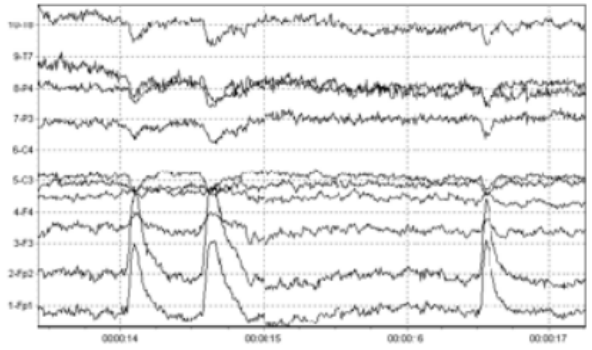
Mann et al., 2013

Applications of PCA: Artifact Removal

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

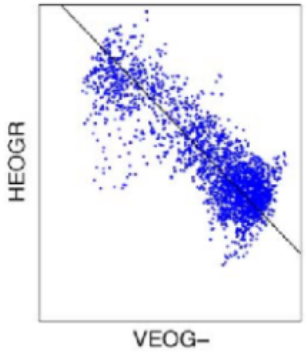






BCI2000

→ reasonable to remove first principal components



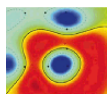
Blink

Beyond PCA

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

- Learning algorithms can be instable in high dimensional spaces. An approach to alleviate this problem is to **reduce the dimensionality** of the data.
- **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 is an **eigenvector** of the data covariance.
- 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).