# Homework 1: Principal Component Analysis (PCA)

Leslie O'Bray
leslie.obray@bsse.ethz.ch

Christian Bock
christian.bock@bsse.ethz.ch

Dr. Michael Moor
michael.moor@bsse.ethz.ch

Prof. Dr. Karsten Borgwardt
karsten.borgwardt@bsse.ethz.ch

*Submission deadline: 17.03.2021 at 14:00*

## Objectives

The goals of this homework are:

- to understand the crucial derivation steps of PCA.
- to implement PCA in `Python`.
- to apply PCA on simulated data.
- to visualise the results.

## Dataset

As for applying PCA, you will work on a simulated dataset. It is based on random draws from several multivariate Gaussian distributions with a total of 320 samples (divided into 4 classes) and a total of 40 features per sample.

## Exercises

First, you are asked to show a key step in the derivation of PCA. The second part is then about actually implementing different functions in order to perform a principal component analysis (PCA). For this, we prepared three `Python` files for this exercise:

1. `run_pca.py`: Main file which will be executed to run all exercises. Here you call the different functions which you have implemented throughout this exercise.
2. `pca.py`: File with a variety of template functions which you have to implement throughout this exercise and which will be called in the main file.

3. `utils.py`: File with many helper functions, such as data loading routines, functions to simulate data etc ...

You only have to edit the first two files for this task. Have a look at these files to familiarise yourself with them.

**Note:**
**1. In order to run the code the following `Python` dependencies have to be fulfilled: `numpy`, `scipy`, `matplotlib` and `scikitlearn`**
If you have not installed these packages, please make sure you install them before you begin with the exercise. You will need these libraries throughout the whole course.
**2. Your code has to be executable with Python 3!**
For instance, make sure you replace the old print formatting `print 'Hello, world'` with `print('Hello, world')`

## Exercise 1

In the first exercise we approach PCA on a more theoretical level and briefly revisit the derivation of PCA. In the lecture slides 18–20 the objective of PCA has been rephrased as a trace maximization problem. Specifically, the following objective (1)

$$\arg\min_{U \in \mathbb{R}^{d \times r}: U^\top U = I} \sum_{i=1}^{n} ||\mathbf{x}_i - UU^\top \mathbf{x}_i||_2^2 \tag{1}$$

has been converted to a trace maximization problem:

$$\arg\max_{U \in \mathbb{R}^{d \times r}: U^\top U = I} \text{trace}\left( U^\top \sum_{i=1}^{n} \mathbf{x}_i \mathbf{x}_i^\top U \right) \tag{2}$$

But where does this trace come from? In this exercise you are asked to derive and explain how the objective (1) can be rephrased into the problem (2). Specifically, elucidate the intermediate steps to arrive at the trace. That is, in the lecture slide 19, how to arrive at Equation 8 from Equation 7. (There will be no awarded points for merely copying the known equations from the slides). Hints: i) Given two vectors $\mathbf{a}, \mathbf{b} \in \mathbb{R}^d$, their inner product can be rewritten as the trace of their outer product: $\mathbf{a}^\top \mathbf{b} = \text{trace}(\mathbf{a}\ \mathbf{b}^\top)$. ii) The trace is a linear mapping. This implies that for two matrices $A, B \in \mathbb{R}^{n \times n}$ we have $\text{trace}(A + B) = \text{trace}(A) + \text{trace}(B)$.

## Exercise 2

In the second exercise you will implement PCA and apply it on simulated data with a total of 40 features and 320 samples. The dataset is already loaded in your main script file (`run_pca.py`). The data container contains a `data` matrix (#samples $\times$ #features $= 320 \times 40$) and a `target` vector ($320 \times 1$) with four different class labels.
You have to implement a variety of functions in the file `pca.py` and to update the main file (`run_pca.py`) in order to perform the following experiment:

(a) First, implement the following two functions: `computeCov` and `computePCA` (Note: do **not** use the `NumPy/SciPy cov` function or an already implemented PCA solution. Implement the covariance function yourself. You are **allowed** to use standard linear algebra functions, such as dot products or solvers to solve the eigen-value problem!)

(b) Next you should transform your input data onto a 2-dimensional subspace by using the first two principal components. For this purpose, you should implement the function: `transformData`.

(c) Plot the transformed data and highlight the samples according to their class labels in different colours (implement the function `plotTransformedData`). Save the plot and provide it in your solution sheet with a brief interpretation.

(d) How much variance can be explained with each principal component (implement the function `computeVarianceExplained`)? Plot the variance explained for all your principal components in a cumulative plot (use the provided function `plotCumSumVariance`). What do you observe? How many principal components are necessary to explain at least 50%, 80%, and 95% of the variance?

(e) Perform a data preprocessing step, (1) whiten your data by subtracting from each feature the mean of the feature and (2) divide each feature by its standard deviation (implement the function `dataNormalisation`). Perform the analysis again using the transformed data! What do you observe? Plot a cumulative variance explained plot! How many principal components are necessary to explain at least 80% of the variance? Why might data normalisation be a necessary step?

## Command-line arguments

Your program should generate all the plots and should output all necessary information in a readable format.
The file `run_pca.py` must be executable using the following command:

```
$ python run_pca.py
```

Note: **Zero** points for Exercise 2 if your script is **not** executable with the above shown command line argument!

## Grading and submission guidelines

This homework is worth a total of 100 points. Table 1 shows the points assigned to each exercise/question.

Table 1: Grading key for homework 1

| | | |
|---|---|---|
| **35 pts.** | **Exercise 1** | |
| | 35 pts. | Exercise 1 |
| **65 pts.** | **Exercise 2** | |
| | 20 pts. | Exercise 2.a |
| | 10 pts. | Exercise 2.b |
| | 10 pts. | Exercise 2.c |
| | 10 pts. | Exercise 2.d |
| | 15 pts. | Exercise 2.e |

Follow the submission guidelines posted on the Moodle webpage. Refer to the document titled “General guidelines for homework sheets” (link named “General guidelines”).

# 1 Homework 1 specific submission format

From the "General guidelines" on Moodle:

1. Create a ZIP file containing your report and source code
2. Use the following naming convention: homework1-LAST NAME.zip
3. Upload the file in the homework assignment section of the class webpage

The ZIP file should contain the following files

- One pdf ("homework1-LAST NAME.pdf") document with all calculation and text answers.
- Three .py files provided in the homework ("utils.py", "pca.py" and "run pca.py"), with your implemented solutions.
- Four output plots ("exercise_2_1", "exercise_2_2", "cumvar_2_1", "cumvar_2_2") in an image format of your choice, e.g. pdf or png.

Providing the homework in a different format may result in point deduction. Providing additional files, e.g. pyc files is not necessary but will not cause point deduction. Please also note that missing axis labels have lead to point deductions in the past.

# Acknowledgments

This exercise was created by Karsten Borgwardt, Dean Bodenham, Dominik Grimm, Xiao He and Damian Roqueiro and most recently modified by Michael Moor, Caroline Weis and Leslie O'Bray.