Homework 0, submission deadline: 03.05.2023 Wednesday, 15:00

1. 1 Register as a Group in Ilias

For submission please form a group of three students. You have to assign to a group in Ilias. You can use the "Homework Group Finding Forum" (https://ilias.studium.kit.edu/ilias.php?ref_id=2094355&cmd=showThreads&cmdClass=ilrepositorygui&cmdNode=x0&baseClass=ilrepositorygui) to team up. Although you will submit the homework as a group, to avoid confusions we ask all students to submit the homework in Ilias. Please make sure that every group member submits the same homework version.

1.2 Introduction

This is a test exercise to get familiar with the way the exercises are presented in the Lecture.

We recommend you to use conda (https://docs.conda.io/en/latest/) to create a virtual environment before you do your homework. Virtual environment is a sandbox where you can feel free to install / uninstall packages without making damage to your system or other ongoing projects. If you are a Windows user, conda may be very useful for installing python packages.

The exercise sheets comes in form of Jupyter notebooks, which consists of different cells. These can be used for markdowns (like this cell) or Python code. Each cell can be run separately to present the text properly or to execute the code. It is also possible to use $\LaTeX$ equations inside the markdown environment. For more information about how to install and start Jupyter, visit https://jupyter-notebook.readthedocs.io/en/stable/

The given Jupyter notebook should be filled out by where it is marked. **The exercises have to be solved and submitted as a group of three students**.

There are either math derivation questions to answer or programming tasks to code. You should answer both inside this notebook. **Python for programming, Markdown (Latex) for math derivation.**

For the submission of the exercise please follow these steps:

1. Make sure that every cell is executed and the output is printed.
2. Create a PdF of the Jupyter notebook via: *File → ... → PDF via LaTeX (.pdf)* or *File → Print Preview → Use your favorit PDF printing program*
3. Or export to pdf using your browser, e.g. chrome, but **make sure you have all content and codes shown in the generated pdf**, e.g., use landscape layout or A3 paper!
4. Zip your created PdF file and your original notebook, i.e. the .ipynb file.

5. Rename your zip file with the following naming convention: group_??_uxxxx_uxxxx_uxxxx where ?? is your group number, uxxxx is the kit user from each group member.
6. Upload the zip file to Ilias. Please make sure that every group member uploads the group submission.
7. Please submit your homework before the deadline. As we will publish the solution right after the deadline, we do not accept any late submission.

For this exercise, please try to fill in the code where stated. It demonstrates some basic operations we will need for this lecture. **Then, submit this exercise following the instructions above. This submission is only for getting familiar with the submission procedure. It will not be graded.**

Exercise 1: Matrices and Vectors in Python

As mentioned, we will use Python for the coding exercise. We will use version 3.8. Additionally, for this and all following exercise we will use NumPy, one of the most fundamental python libraries, designed to efficiently create and operate on multi-dimensional arrays (, i.e., vectors, matrices and higher order tensors)

If you are new to NumPy we refer to [https://numpy.org/devdocs/user/quickstart.html (https://numpy.org/devdocs/user/quickstart.html)](https://numpy.org/devdocs/user/quickstart.html) .

1.1 Create Matrices and Vectors in Python

We first create a matrix

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 2 & 6 \\ 7 & 3 & 8 \end{pmatrix}$$

and two vectors

$$v = \begin{pmatrix} 3 \\ 1 \\ 2 \end{pmatrix}, \; w = \begin{pmatrix} 1 \\ 1 \\ 2 \end{pmatrix}$$

```
In [1]: import numpy as np

        # Let us create a matrix A and vectors v and w:
        A = np.array([[1, 2, 3],
                      [4, 2, 6],
                      [7, 3, 8]])
        v = np.array([3, 1, 2])
        w = np.array([1, 1, 2])
        print("Matrix A:")
        print(A)
        print("Vector v:")
        print(v)
        print("Vector w:")
        print(w)
```

```
Matrix A:
[[1 2 3]
 [4 2 6]
 [7 3 8]]
Vector v:
[3 1 2]
Vector w:
[1 1 2]
```

1.2 Basic Operations with Vectors and Matrices

Implement the v + w and 2 · A. Print out your results and verify that they are correct.

```
In [2]: # We can perform general element-wise operations:
        elem_wise_sum =  # do the element-wise sum v+w here
        print(elem_wise_sum)
        print("----------")
        elem_wise_mult =  # do the element-wise multiblication 2*A here
        print(elem_wise_mult)
```

```
  File "<ipython-input-2-ff092243e077>", line 2
    elem_wise_sum = # do the element-wise sum v+w here
                                                       ^
SyntaxError: invalid syntax
```

Do the matrix vector product A · v. Print out your result and verify that the result is correct.

```
In [3]: # To perform a matrix multiplication, use either np.dot or just the "@" symbol:
        matrix_vector_prod =  # do the matrix vector product A*v here
        print(matrix_vector_prod)
```

```
  File "<ipython-input-3-dd9faa1e1aa6>", line 2
    matrix_vector_prod = # do the matrix vector product A*v here
                                                                ^
SyntaxError: invalid syntax
```

Do the matrix matrix product A · A. Print out your result and verify that the result is correct.

```
In [4]: # Two matrices can be multiplied in the same fashion:
        matrix_matrix_prod =  # do the matrix matrix product A*A here
        print(matrix_matrix_prod)
```

```
  File "<ipython-input-4-7f0798fde133>", line 2
    matrix_matrix_prod = # do the matrix matrix product A*A here
                                                               ^
SyntaxError: invalid syntax
```

Exercise 2: Sampling from a Gaussian

Sample 1000 samples from a Gaussian distribution with mean $\mu = -2$ and standard deviation $\sigma = 0.5$.

We also visualize the samples. For this we use matplotlib, a python package for plotting

```
In [5]: # for jupyter notebooks you need the following line of jupyter magic command (that's literally the name)
        # to display matplotlib figures inside the notebook. Make sure it comes before the actual import!
        % matplotlib inline
        #actually import matplotlib
        import matplotlib.pyplot as plt

        # In Machine Learning, it is often necessary to sample from a probability distribution.
        # To sample from a Gaussian distribution (https://en.wikipedia.org/wiki/Normal_distribution):

        mu, sigma = -2, 0.5  # mean and standard deviation
        samples =  # do the sampling from the normal distribution here

        # plot the histogram of the samples
        count, bins, ignored = plt.hist(samples, 40, density=True)

        # plot the density function
        plt.plot(bins, 1 / (sigma * np.sqrt(2 * np.pi)) * np.exp(
            - (bins - mu) ** 2 / (2 * sigma ** 2)), linewidth=2, color='r')

        # show the plot
        plt.show()
```

```
  File "<ipython-input-5-207b58d377a6>", line 11
    samples = # do the sampling from the normal distribution here
                                                                 ^
SyntaxError: invalid syntax
```

Exercise 3: Solving a Linear Equation

You are given the linear equation

$$Ax = v.$$

We want to find the vector x. Assume that the matrix A and the vector v are given as in Exercise 1.1. Find the solution for x. Hint: Use the function 'linalg.solve' from numpy - this should always be preferred over direct matrix inversion.

In [6]:
```python
# To solve the linear equation system A x = v, you may use the np.linalg.solve function. In other words we can
# use np.linalg.solve for inverting any matrix which does not stand alone.

x =   # solve for x here

print("Solution: x =", x)
print("Test: A@x = {A@x} is the same as v = {v}")
```

```
  File "<ipython-input-6-fa89ef1b11db>", line 4
    x = # solve for x here
        ^
SyntaxError: invalid syntax
```

Closing Remarks

NumPy provides easy and efficient ways to deal with vectors, matrices and tensors. Linear algebra functionality like the function introduced above, together with more advanced techniques such as broadcasting (https://numpy.org/doc/stable/user/basics.broadcasting.html (https://numpy.org/doc/stable/user/basics.broadcasting.html)), are the key to efficient programming in Python. If you want to do any practical Machine Learning you need to familiarize yourself with those techniques. We will use and introduce more of this in the following exercises but also expect you to use it for your submission. So use them whenever possible and, most importantly, avoid unnecessary for loops.

In [ ]:

In [ ]: