

Practical work 01 – 21st of February 2023

Let's get started

The main objective of this Practical Work (PW) is to set up your computer environment and to bring you on track for this class on **Machine Learning and Deep Learning**.

We are going to use Python as programming language to perform the PW. This weeks' PW will include setting up Python on your computer and helping you to get familiar with this language.

For Python, many good libraries are available to work with data, to visualise data and for performing machine learning tasks. That's why we have chosen to use Python in this class.

For this class, we will work with iPython Notebooks which is also a good format to hand in your practical works. You can create, edit and run these notebooks from a web browser and write Python code or text (using Markdown) in so-called cells and also create nice output (e.g. tables, plots).

Warning : There are two "concurrent" versions of Python, versions 2 and 3 with some minor incompatible differences in language syntax. In this class, **we will use Python 3**.

Hand-In of Practical Work

You will have to submit your PW reports **by Tuesday, 3pm of the following week** (right before the beginning of the class).

For this first PW, we expect you to **submit a single zip-archive containing a solution for exercise 3** in form of a **iPython Notebook and a small report as pdf** (follow the instructions in the iPython Notebook).

Exercise 1 Python Installation on Your Computer

Skip this part if you are already set up with Python, Jupyter notebooks and your favorite IDE for Python. Otherwise, we recommend the following installations :

- Anaconda platform - a distribution of Python with popular data science packages that are pre-installed or easily installable. <https://www.continuum.io>. Select the 3.8 version of Python, as illustrated on Figure 1 below.




Windows 	MacOS 	Linux 
Python 3.8 64-Bit Graphical Installer (457 MB) 32-Bit Graphical Installer (403 MB)	Python 3.8 64-Bit Graphical Installer (435 MB) 64-Bit Command Line Installer (428 MB)	Python 3.8 64-Bit (x86) Installer (529 MB) 64-Bit (Power8 and Power9) Installer (279 MB)

FIGURE 1 – Installing the Anaconda platform

- PyCharm IDE - an integrated development environment for Python. <https://www.jetbrains.com/pycharm>. You may select Professional or Community edition - the Community edition should be enough for this class. Free licenses are typically given to students, use your school email address for the registration by JetBrains.

Now open an Anaconda prompt in order to set up a new conda environment (we name it TSM_DeLearn). During the creation process we directly install several required packages (jupyter notebook, matplotlib, pandas, scikit-learn, tensorflow) :

```
(base) C:\Users\Username>conda create -n TSM_DeLearn jupyter notebook matplotlib pandas scikit-learn tensorflow
```

Then activate the environment using :

```
(base) C:\Users\Username>conda activate TSM_DeLearn
```

Exercise 2 Python Language in a Nutshell

We assume here that students are knowledgeable in other programming languages such as Java or C and that basic data structure concepts are known. If you know already Python and the concept of notebooks, then you can skip this exercise.

- Open Jupyter from the Anaconda Navigator. Jupyter Python notebooks run in your browser so, after launching Jupyter from Anaconda, a browser should show up. Open a new notebook from menu File and follow the User Interface Tour as illustrated in Figure 2.
- Download the file `intro-python-3.ipynb` from moodle and open it from Jupyter in Anaconda. You need to navigate where you stored the ipynb file. See Figure 3 below.
- Go through the content of the `intro-python-3.ipynb` notebook and play with the cells. This document should give you a quick introduction to Python assuming that you are fluent with other programming languages. You may also want to get familiar with [Markdown syntax](#) if not known already.
- If you want a more fully fledged introduction to Python, read the official tutorial from <https://docs.python.org/3.8/tutorial/>.

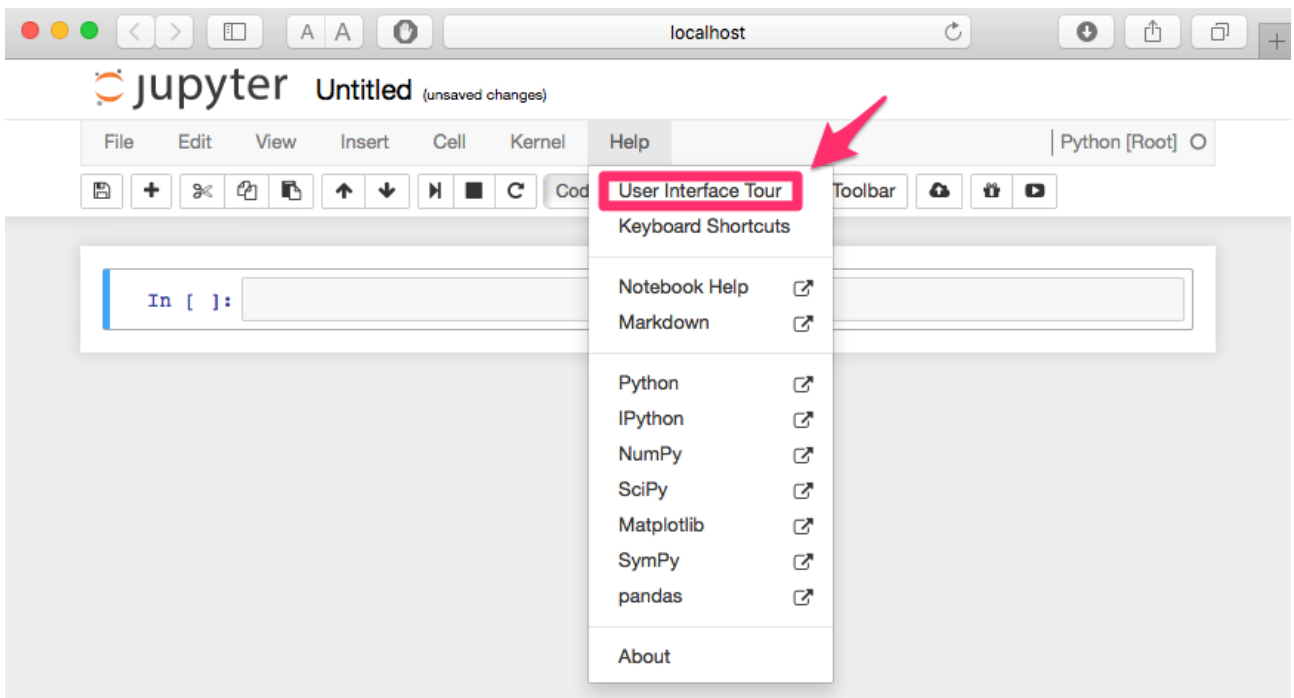


FIGURE 2 – First steps with Python notebooks.

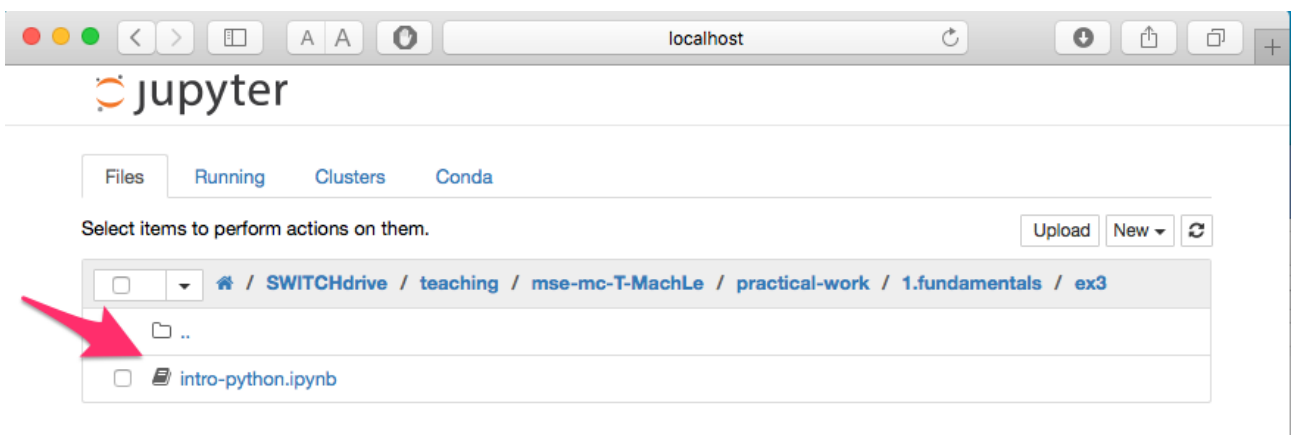


FIGURE 3 – Intro to Python language and Python notebooks.

Exercise 3 Perceptron Learning Algorithm

The goal of this exercise is to implement the perceptron learning algorithm and to apply it to given example data. You will program the learning algorithm so that it finds a decision boundary (such as as indicated by the green dashed line in the Figure below) that separates between the red and the blue crosses. The algorithm will perform updates on an initial (typically wrong) guess (such as represented by the magenta dashed line).

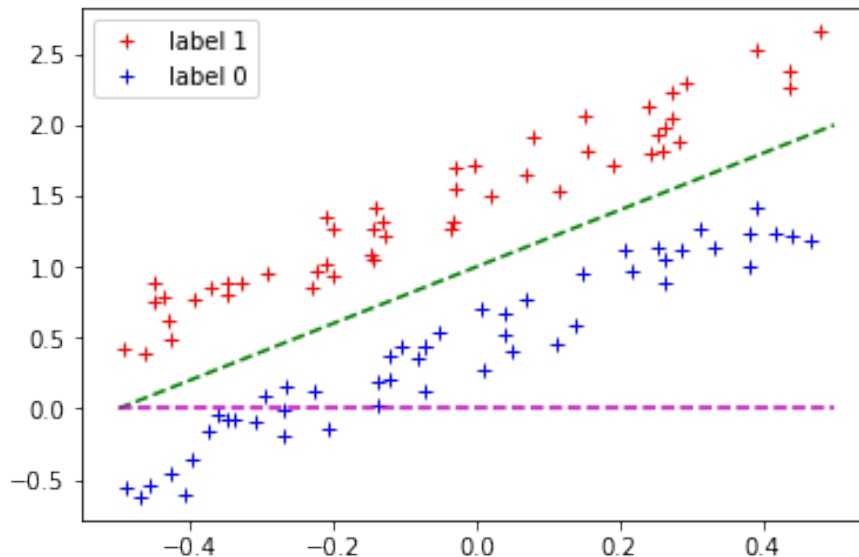


FIGURE 4 – The linearly separable example data used to train a classifier with the perceptron learning rule. The ideal decision boundary would be the green dashed line (the max margin classifier) - the magenta dashed line will be the first trivial guess which is bad since the number of mis-classifications is high.

- Open the notebook named `perceptron_learning_stud.ipynb` and inspect the functions defined therein. For better readability, you can use `help(function_name)` to inspect the documentation of what the functions should do. Carefully inspect the shapes of the numpy arrays used - you will benefit from it in later weeks!
- For some of the functions you should complete missing blocks of code that are marked as follows.

```
### START YOUR CODE ###
```

```
### END YOUR CODE ###
```

- If you manage to complete all the missing code properly the code in section 3/ *Test Your Implementation* should find the correct (but not necessarily ideal) decision boundary in a finite number of steps. Summarise your findings in an additional cell at the end of the notebook (as described in `perceptron_learning_stud.ipynb`).

- (Optional) Try to generate a dataset (by using the *prepare_data*-function), so that the Perceptron learning algorithm does not converge.

Exercise 4 Optional : Data Visualization

To train a bit yourself, we propose you to do a visualisation workout with the Iris dataset https://en.wikipedia.org/wiki/Iris_flower_data_set. Download the file `iris.txt` from moodle. Put it in a Python data structure and attempt to reproduce a plot close to the following one.

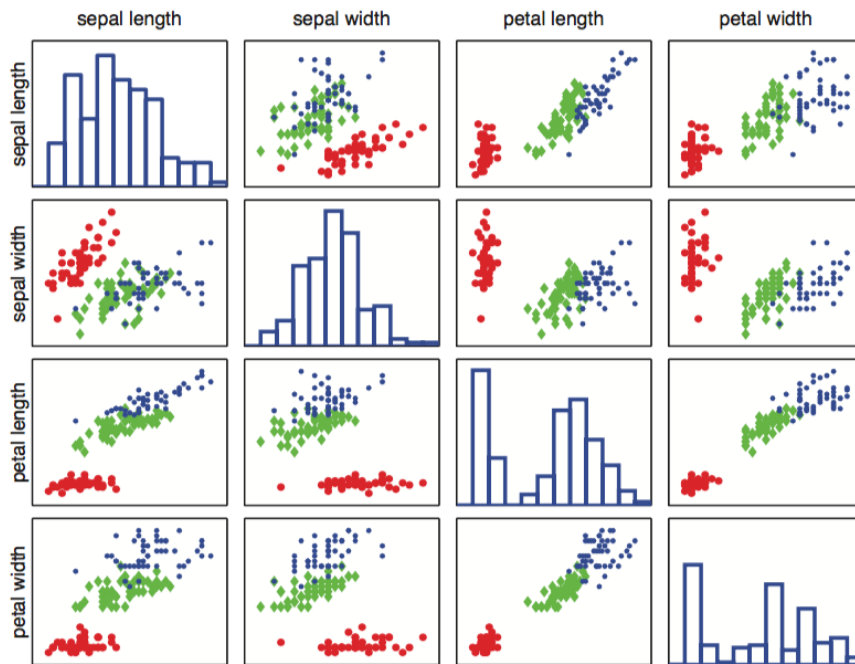


FIGURE 5 – Visualization of the Iris data as a pairwise scatter plot. The diagonal plots the marginal histograms of the 4 features. The other cells contain scatterplots of all possible pairs of features. Red circle = setosa, green diamond = versicolor, blue star = virginica.

The iris species classification task is a classical one. The goal is to distinguish three kinds of iris : setosa, versicolor and virginica. In the data set, a botanist has extracted 4 *features* : sepal length, sepal width, petal length and petal width.

What can you say regarding class separation ? One class seems easier to distinguish from the others, which one ? How would you separate the other two ?