Practical Work 01 – 22/02/2024
Let's get started

Objectives. The main objective of this Practical Work (PW) is to set up your computer environment and to bring you on track for the **Deep Learning** class. We are going to use Python as it offers many good libraries to handle data and to run machine learning tasks. This weeks' PW will also include tutorials and exercises to get familiar with this language. Finally, an introduction to Pytorch tensors will be given. We will also work with iPython Notebooks which is also a good format to hand in your PW. 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).

Moodle. If not yet done, register on Moodle at `https://moodle.msengineering.ch`. Navigate to *Home* → *Region D* → *TSM - Technical Scientific Modules* → *SPR24* → *TSM_DeLearn Zurich* and use subscription key : `moodlemsekey`.

Hand-In of PW. Most of the time, you'll need to submit your PW reports by Wednesday, 10am of the following week. However, we may override this rule. In any cases, the dates indicated on Moodle for each PW are the ones you should follow. For this first PW, we expect you to submit a solution for exercise 2 on `numpy`, exercise 4 about Pytorch tensor manipulation, and exercise 7 on review questions. The other exercises are optional. Submission is in the form of a zip archive including your notebooks.

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, read the following options for you to install the tools.

We encourage you to use *virtual environments* that allow you to insulate your Python version and dependencies in a given directory, without polluting or disturbing other Python projects you may have on your computer (see Fig. 1). With *virtual environments*, you may also easily switch Python and its libraries from one version to another by installing multiple *virtual environments*.

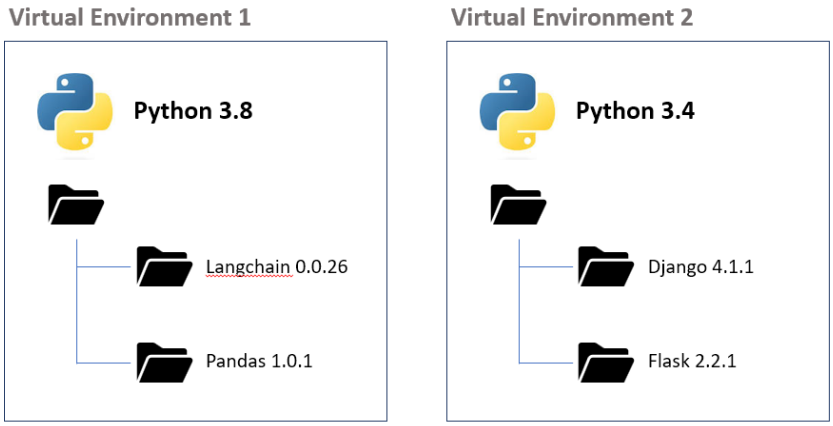

FIGURE 1 – In Python data science projects, virtual environments serve to segregate package dependencies utilized in various projects, preventing conflicts between them. For instance, you might have two projects : one requiring pandas 1.0.1 and the other pandas 1.5. While installing both versions system-wide isn't feasible, you can establish separate, isolated environments for each project, enabling you to activate them individually and start coding.

Jupyter notebooks with manual installation in virtual environments

This is our **recommended** way to go. First install Python 3.10 from https://www.python.org. Versions 3.8, 3.9 and 3.11 should also work for the class but be aware that 3.11 is still in *bugfix* status.

Create a working directory and open a terminal in that directory :

a) Install the virtual environment : `python3.10 -m venv venv`
b) Activate the virtual environment on Mac : `source venv/bin/activate`
c) Activate the virtual environment on Windows : `C:\> <venv>\Scripts\activate.bat`
d) Install Jupyter : `pip install jupyter jupyterlab-tour`
e) Install needed libs : `pip install numpy matplotlib pandas scikit-learn imageio`
f) Install tensorflow : `pip install tensorflow`
g) Install pytorch : `pip install torch torchvision`
h) Launch Jupyter : `venv/bin/jupyter lab`

Whenever you need to re-launch the notebooks, repeat steps b) and h). If you need to install other libraries, use `pip install LIB_NAME` once the virtual environment is setup with step b). If you need to deactivate the virtual environment, use the command `deactivate`.

Anaconda distribution or miniconda

Anaconda is a popular Python distribution of data science packages. Have a look at their documentation before installing. You can chose between the full Anaconda distribution or the smaller sized miniconda. If you opt for the full Anaconda distribution, over 250 data science and machine learning packages are going to be installed together with the *Anaconda Navigator* (see Fig. 2).

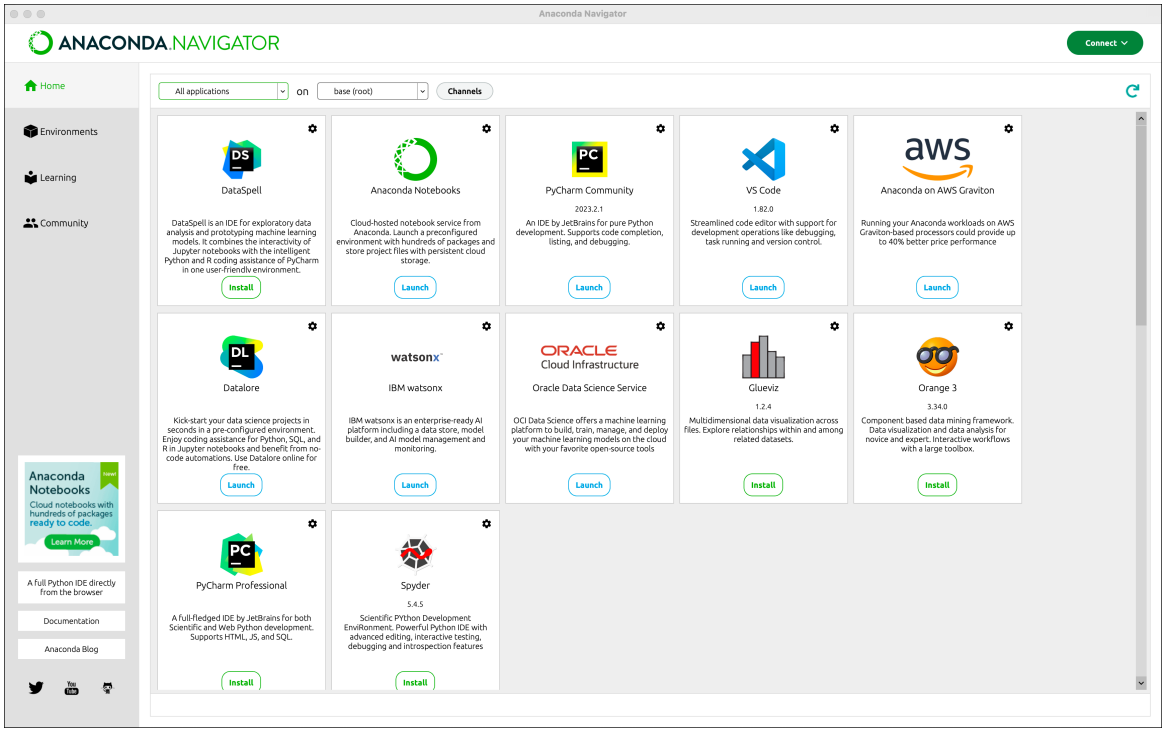

FIGURE 2 – Anaconda Navigator after installing the Anaconda distribution.

Once Anaconda or miniconda installed, the `conda` command is available from the terminal to manage your virtual environement, in a similar way as described above. Read the doc if this is what you want to do.

Other options

You may run iPython notebooks directly from integrated development environments (IDE).

- PyCharm - an IDE for Python from JetBrains. You may select Professional or Community edition - the Community edition should be enough for this class. Free licenses are usually given to students, use your `SCHOOL.ch` address for the registration by JetBrains.
- DataSpell - an IDE for Data Science supporting Python and notebooks, also from JetBrains. Same remark as above for licenses.
- Visual Studio code - a general purpose IDE that supports Python and notebooks from Microsoft.

If you do not want to install anything on your computer, you may use Google Colab at https://colab.research.google.com/.

Exercise 1 OPTIONAL – 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 in your working directory launching `venv/bin/jupyter lab` in the terminal or from the Anaconda Navigator. Jupyter Python notebooks run in your browser so, after launching Jupyter lab, a browser should show up.
— Open a new notebook from menu File and follow the User Interface Tour as illustrated in Figure 3.

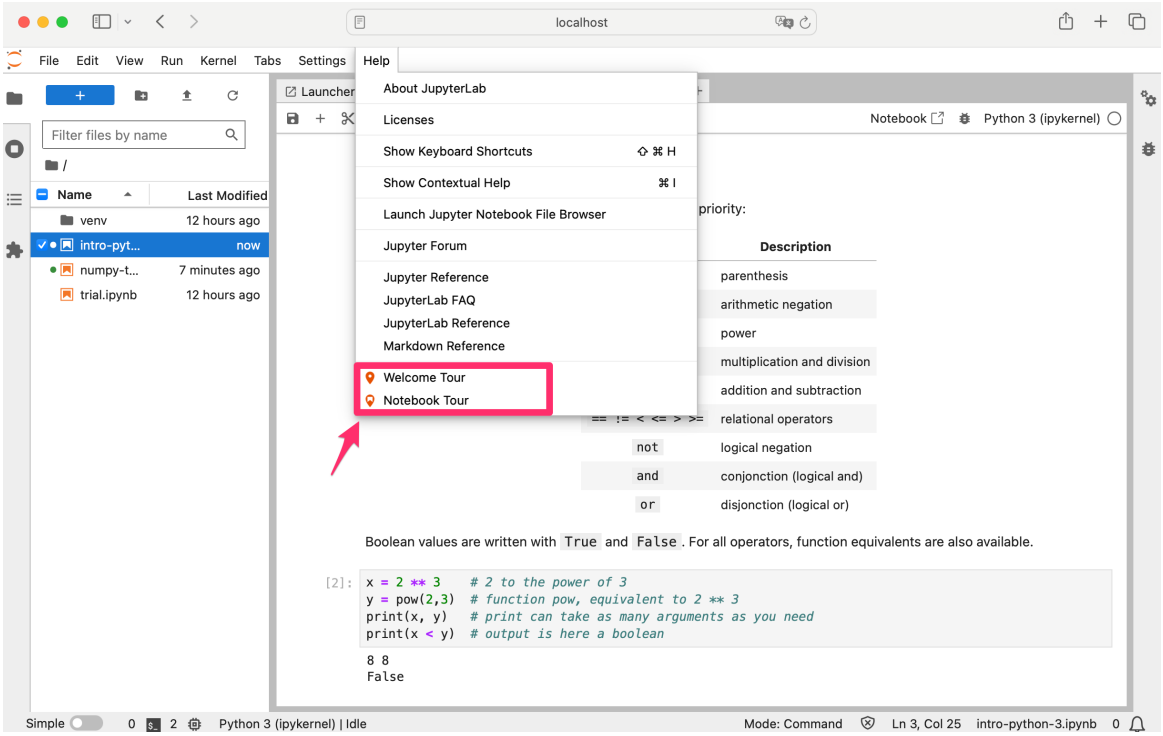

FIGURE 3 – First steps with Jupyter Lab and Python notebooks.

— 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 4 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 slowlier and fully fledged introduction to Python, read the official tutorial from https://docs.python.org/3/tutorial/.

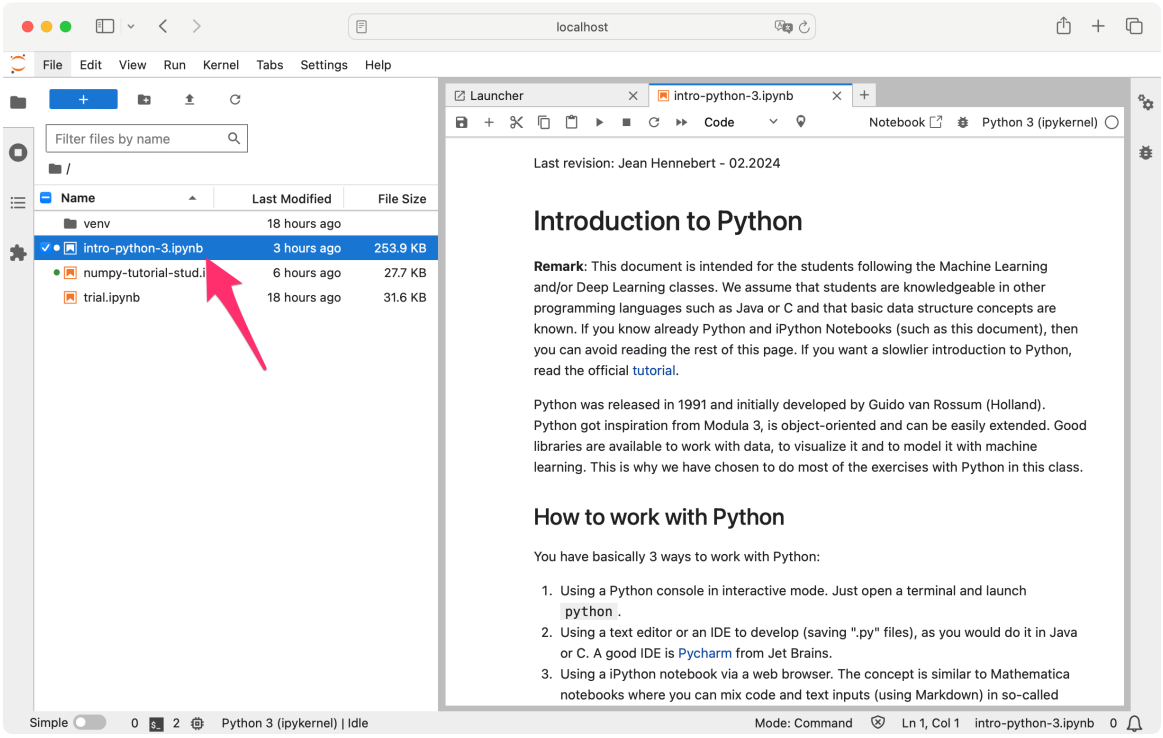

FIGURE 4 – Introduction to Python language and Python notebooks.

Exercise 2 Numpy in a Nutshell

This exercise is to become more familiar with `numpy`. Read the content of the Jupyter notebook `numpy-tutorial-stud.ipynb` that you will find on Moodle. Pay special attention to the *broadcasting* section that allows to gain significant speedup when processing large numpy arrays. Regarding this, it is usually more efficient to use *broadcasting* instead of `for` loops in your code.

At the end of the tutorial, you have to complete some manipulations of images stored by arrays to obtain a result as in 5.

Exercise 3 First hands on Pytorch tensors

a) Read and listen to F. Fleuret class 1.4 *Tensor basics and linear regression* and class 1.5 *High dimension tensors* on https://fleuret.org/dlc/.

b) Read the Pytorch tutorial on https://pytorch.org/tutorials/beginner/basics/tensorqs_tutorial.html and get familiar with basic tensor manipulations.

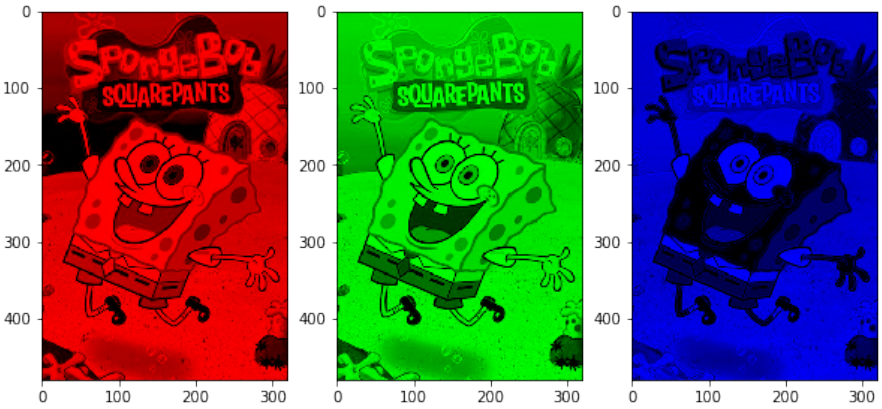

FIGURE 5 – Tri-color planes of Sponge Bob image.

Exercise 4 Tensors – Linear regression

Start from the notebook `linear-regression-stud.ipynb` available on Moodle. We will use a small data set of apartment rent prices, composed of N samples. The objective is to build a linear model to predict the rent price from the surface (m2) of the apartment. By linear model we mean here the equation of a line.

We are then in the case of a monovariable linear regression where we want to discover the parameters θ of a linear model defined with

$$h_\theta(\mathbf{x}) = \theta_0 + \theta_1 x \tag{1}$$

The "best" θ's are the ones that will minimise the squared difference between the gotten outputs and the *target* outputs defined with

$$J(\theta) = \frac{1}{2N} \sum_{n=1}^{N} (h_\theta(\mathbf{x}_n) - y_n)^2 \tag{2}$$

Equation 2 is a **loss** function, in this case the Mean Squared Error or MSE loss. A closed form solution to this problem is given by the following normal equation :

$$\theta = (X^T X)^{-1} X^T \vec{y} \tag{3}$$

With X and $\vec{y}$ defined as

$$X = \begin{pmatrix} 1 & x_1 \\ 1 & \\ 1 & \vdots \\ 1 & \\ 1 & x_N \end{pmatrix} \tag{4}$$

$$\vec{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_N \end{pmatrix} \tag{5}$$

a) Following the instructions in the notebook, implement Equation 3 assuming that x is the living area and y is the renting price. Use `numpy` for the vector operations. Plot the computed line on top of the scatter plot.

b) Compute the overall cost value according to Equation 2.

From the numpy solution, provide a solution to the problem using pytorch.

Exercise 5 OPTIONAL – Data Visualization

To train a bit yourself, we propose you to do a visualization workout with the Iris data set https://en.wikipedia.org/wiki/Iris_flower_data_set. Download the file `iris.txt` from moodle or from http://www.statlab.uni-heidelberg.de/data/iris/. Put it in a Python data structure and attempt to reproduce a plot close to the one in Figure 6. Advice : start by working with individual plots and then move to a grid of plots with `subplot` **or** take the shortcut of using the library `pandas`.

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 ?

Exercise 6 OPTIONAL – Reading Assignments

— Read the first chapter of Murphy's book "Machine Learning".

Build your own summary of these chapters by doing a taxonomy of machine learning problems. You can find the pdf on Moodle.

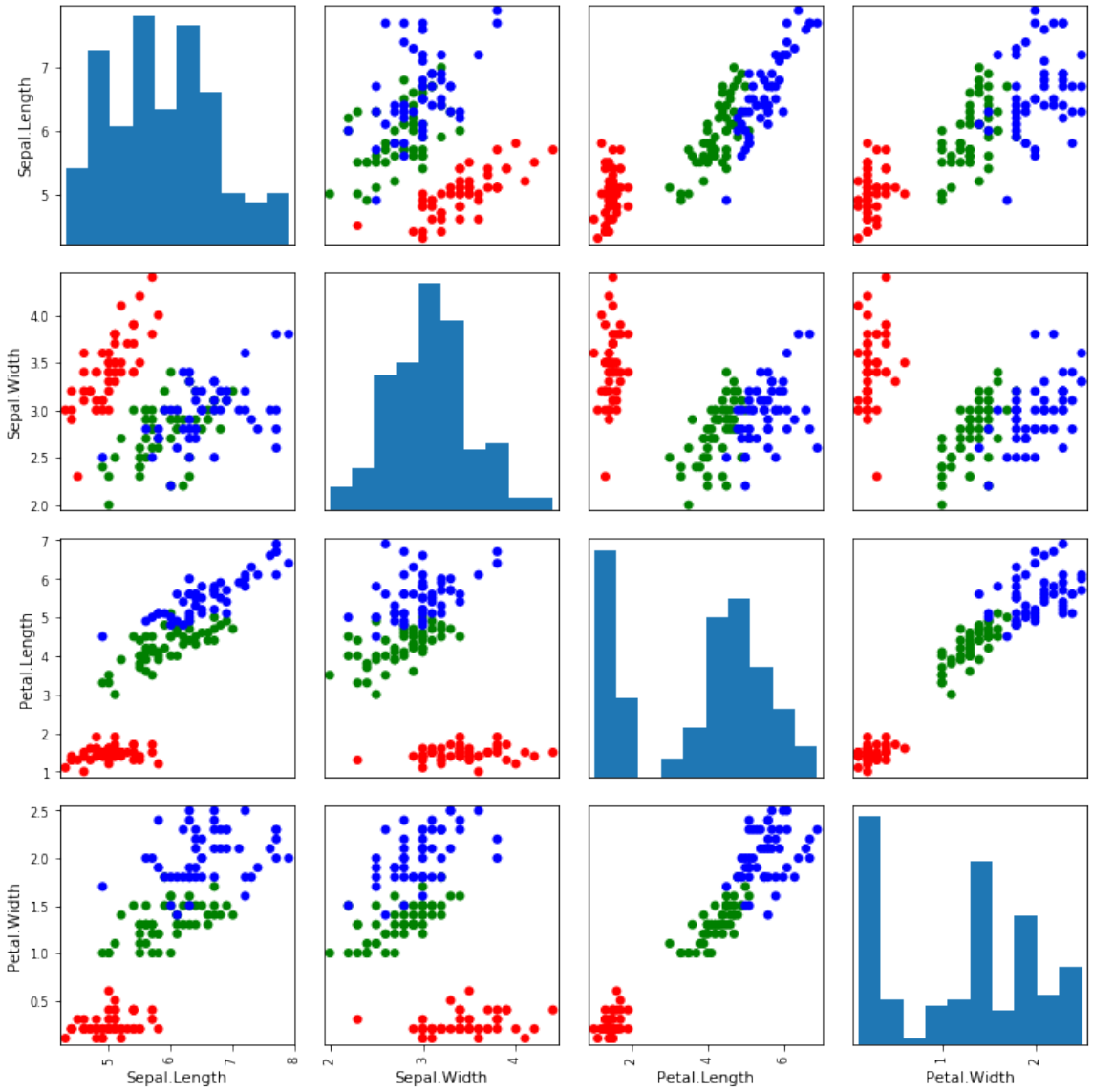

FIGURE 6 – Visualization of the Iris data as a pairwise scatter plot. The diagonal plots the marginal histograms of the 4 features. The other cells contain scatterplots of all possible pairs of features. Red circle = setosa, green diamond = versicolor, blue star = virginica.

Exercise 7 Review Questions

Insert your answers in a Python notebook and submit it with the other notebooks.

1) Supervised vs. unsupervised systems

Of the following examples, which one would you address using a supervised or an unsupervised learning algorithm ? Give some explanations for your answers.

a) Given email labeled as spam/not spam, learn a **spam filter**.
b) Given a set of news articles found on the web, group them into sets of **related articles**.
c) Given a database of customer data, automatically discover **market segments** and group customers into different market segments.
d) Given a dataset of patients diagnosed as either having **glaucoma** or not, learn to classify new patients as having glaucoma or not.

2) Classification vs. regression systems

Can we transform a regression problem into a classification problem ? What would be the benefits of doing so ?

3) Numpy arrays

a) What are the main differences between a Python *list* and a numpy *array* ?
b) Can we mix different types of data in a numpy array ?
c) Define what is the *rank* in a numpy array.
d) If we load a gray-level image in a numpy array, what is the rank of this array ? Explain the different dimensions. Give a potential value of the `shape` attribute for a square images.
e) Regarding the previous point, what would be the difference with a color image ?