Machine Learning Project
WS20/21

Machine Learning Group
Fakulty IV, Technische Universität Berlin
Prof. Dr. Klaus-Robert Müller
Email: `klaus-robert.mueller@tu-berlin.de`

Machine Learning Project

Aim and scope

The aim of this module is to give students a good understanding of how Machine Learning techniques are applied to real world problems in an academic or industrial context. In contrast to other modules which focus primarily on methodology, this module covers the whole life span of a Machine Learning project, from the extraction of useful feature vectors from raw data to the final assessment of a prediction system from a practical point of view, which goes beyond reporting a single accuracy figure.

Students are not expected to invent or reimplement algorithms, but should use `PyTorch` and the `scikit-learn` toolbox. Even so, programming will be a major part: data logistics, feature extraction, model analysis and assessment. All code must be written in Python.

Project Machine Learning vs. Lab Course Machine Learning

There is a clear difference between the Lab Course and the Project:

- In the Lab Course, the tasks are very clear and you are told exactly what to program, what to plot and what to discuss.
- The Project is the more advanced class. Instructions are more general: it is up to you to decide how to approach the tasks, which aspects are most important and how to design meaningful figures.

Prerequisites

There are no formal entry requirements. However, we strongly recommend

- the module Machine Learning 1,
- Python Programming for Machine Learning and
- ideally, the Lab Course Machine Learning.

Students who do not have attended these lecture, should make sure they have the following skills:

Basic theory Students should be fluent in probability theory, linear algebra and understand how and why mainstream learning algorithms work.

Some practical ML experience Prior exposure to the practical application of ML algorithms is essential; students should know how to select hyperparameters and assess the performance of a trained predictor.

Python programming All code that is to be handed in must be written in Python; students should be able to program in Python using the packages `numpy` and `scipy`.

Dates and structure

The project is divided into the following three milestones:

1. Data processing and prototyping.
2. Calibration and assessment of a learning method (reproducing existing results).
3. Calibration of final model and in–depth evaluation.

For each milestone, each group must hand in

1. a written expose (not longer than ten pages) and
2. code in a standardized format.

Around two weeks before each deadline, we will meet to discuss problems and progress. After each milestone, we will hold a seminar where students presents their solutions in an informal short talk (< 10 minutes). The solutions (report and code) will be made available to all participants; we actively encourage students to reuse ideas and code from other groups in future milestones.

Registration and examinations

The maximum number of participants is 30, all course work must be handed in groups of two or three students. Registrations will be considered on a first come first serve basis, just send an email to the address below (Contact). Registration closes at the first meeting on October 16.

The grades are entirely based on the course work handed in for each milestone[1]. Every milestone will be assessed individually (from zero to ten points), and the final grade is based on the total number of points.

Programming hints and guidelines

You will need the packages `numpy` and `scipy` for numerical linear algebra. If you are new to Python, but experienced with Matlab, have a look at

`http://www.scipy.org/NumPy_for_Matlab_Users`.

Additional information and resources can be found under

`https://wiki.ml.tu-berlin.de/wiki/Main/PythonKurs`.

We highly recommend you to install the package `matplotlib`, which is a plotting library for `NumPy`. Additionally, it will be very helpful to install `ipython`, which is an improved Python shell, and especially useful for interactive data visualization. Once installed, the command

```
ipython --pylab
```

will start a Python environment, in which all numerical and plotting libraries are already imported.

In the Python Programming for Machine Learning course, `jupyter` notebooks are used as a comfortable development environment:

```
jupyter notebook
```

In the code that you submit, every single function must be documented, at least with a descriptive docstring. Moreover, you must adhere *exactly* to the prescribed filenames, function signatures and behaviour; we will provide simple test scripts to guard against some mistakes.

Contact

Jacob Kauffmann
Office MAR 4.058
`j.kauffmann@tu-berlin.de`

[1] *Prüfungsäquivalente Studienleistungen* in TU-parlance.