

Fortgeschrittenenpraktikum for Machine Learning

Abstract

This document provides a brief overview of the Fortgeschrittenenpraktikum for Machine Learning, along with links to blogs and videos containing the necessary information to complete the exercises. Additionally, it includes instructions for setting up Python, the programming language used for solving the various exercises.

1 Description of the tasks

The aim of the Fortgeschrittenenpraktikum is that the students get familiar with various methods and practices in Machine Learning. To achieve this, we have designed four tasks that must be completed using the Python programming language, with each task presented in its own Jupyter notebook.

A key package for this practicum is PyTorch. We recommend this Pytorch introduction to familiarize yourself with it.

1.1 Jupyter Notebook 1: Data Visualization and k-NN

The aim of this task is to familiarize yourself with Python and the dataset that will be utilized in most subsequent tasks. To achieve this, students will implement the k-Nearest Neighbor classifier from scratch and evaluate its performance. The dataset for this task is the GalaxyMNIST, which consists of images of galaxies accompanied by their corresponding labels.

Relevant sources for solving this task can be found here:

- **K-Nearest Neighbors:** [geeks4geeks post](#)

1.2 Jupyter Notebook 2: Data Visualization using dimensionality reduction

Dimensionality reduction is a relevant tool in machine learning, as it aids the analysis of high dimensional data. It can be used to visualize the data in two or three dimensions and identify potential clusters or underlying structures. Additionally, dimensionality reduction can serve as a preprocessing step to reduce the number of features, lowering computational costs with minimal loss of information. In this task, you will explore various dimensionality reduction techniques: PCA (a linear method), t-SNE, and UMAP (non-linear methods).

These techniques will be applied to visualize the GalaxyMNIST dataset from the previous task.

Relevant resources for solving this task can be found here:

- **PCA:** stack exchange explanation, wikipedia
- **t-SNE:** blogpost
- **UMAP:** blogpost

1.3 Jupyter Notebook 3: Advanced Machine Learning

In this notebook, we will compare the performance of classical machine learning classification algorithms with fully connected and convolutional neural networks, using the GalaxyMNIST dataset once again.

Relevant resources for solving this task can be found here:

- **Decision Trees, Random Forests, and Gradient Boosting:** blogpost, sklearn comparison, decision tree wikipedia, random forest wikipedia, gradient boost wikipedia
- **Neural Networks:** Pytorch implementation tutorial, visual series on neural networks, more detailed explanation of convolutional neural networks

1.4 Jupyter Notebook 4: Tabular Data

In this Jupyter notebook, we will compare the performance of classical machine learning algorithms and neural networks, this time focusing on tabular data, which can include different data types such as numerical and categorical values. Specifically, we will be working with the MiniBoone dataset, which consists of particle identification data that aims to distinguish between signal (electron neutrinos) and background (muon neutrinos).

2 Setup

Once the FP starts, we will provide access to a github repository, where you will have access to the jupyter notebooks.

2.1 Environment set up

At first, you need a working python installation and package manager. Alternatively, if you don't have any python installation either package manager, we recommend you to install micromamba. To install it follow the instructions at micromamba installations

2.1.1 Method 1 (not recommended)

The simplest way to install the required Python packages is to use your current Python installation and execute the following command:

```
pip install -r requirements.txt
```

To do this, first create a file named `requirements.txt` containing the following list of packages:

Listing 1: `requirements.txt`

```
numpy
scipy
matplotlib
ipykernel
pandas
jupyter
h5py
tqdm
torch --index-url https://download.pytorch.org/whl/cpu
torchvision --index-url https://download.pytorch.org/whl/cpu
git+https://github.com/mwalmsley/galaxy_mnist.git
scikit-learn
umap-learn
```

However, installing packages in this way may cause conflicts with other packages now or in the future. It is generally recommended to use virtual environments to manage dependencies and avoid such conflicts (method 2). See this article for more details on why virtual environments are beneficial.

2.1.2 Method 2

If you'd like to avoid potential conflicts, you can create a virtual environment. To do so, create a file called `environment.yaml` with the following content:

Listing 2: `environment.yaml`

```
name: mlfp
channels:
  - defaults
  - conda-forge
  - pytorch
dependencies:
  - python
  - numpy
  - scipy
  - matplotlib
  - ipykernel
  - pandas
  - jupyter
  - h5py
  - tqdm
  - umap-learn
  - pip:
    - git+https://github.com/mwalmsley/galaxy_mnist.git
    - --index-url https://download.pytorch.org/whl/cpu
    - torch
    - torchvision
    - scikit-learn
```

In case you have access to a GPU, you may use the following variant, which will provide you an speed up when using neural networks

Listing 3: environment.yaml in case of access to GPU

```
name: mlfp
channels:
  - pytorch
  - nvidia
  - defaults
  - conda-forge
dependencies:
  - python
  - numpy
  - scipy
  - scikit-learn
  - pytorch
  - pytorch-cuda=12.1
  - torchvision
  - matplotlib
  - ipykernel
  - pandas
  - jupyter
  - h5py
  - tqdm
  - umap-learn
  - pip:
    - git+https://github.com/mwalmsley/galaxy_mnist.git
```

To create the environment with these dependencies, run the following command:

```
micromamba env create -f environment.yaml
```

After the environment is created, activate it using:

```
micromamba activate mlfp
```