Machine learning for practical projects

Ita cirovic Donev (FinMetrika d.o.o.)

2024-02-01

Table of contents

Chapter 1

Introduction

The `finmetrika-ml` library is a machine learning library for practical projects, predominantly for the financial industry.

The library is organized as follows:

```
finmetrika_ml
├── data
    ├── data_processing.py
    └── vizualization.py
├── model
    ├── training.py
    ├── evaluation.py
    └── metrics.py
└── utils.py
```

To install use the `pip` command in your virtual environment:

```
pip install finmetrika_ml
```

Chapter 2

Setting up the machine learning project

Any data analysis and machine learning project requires lots of data exploration and experimentation with different model types along with different model arguments. Things can get pretty messy fast. To make life a bit easier it is best to organize at the start of the project by creating a new repository localy and/or in the cloud. This will ensure version control of your project.

Inevitably, there will be many arguments that we will need to use in the course of our experimentation. Best is to store them in the `config.py` file either as `ArgParse` or `dataclass` objects. Here is an example:

```
from dataclasses import dataclass, asdict, field
from pathlib import Path


# Get the absolute path to the directory where config.py is located
BASE_DIR = Path(__file__).resolve().parent.parent

@dataclass
class ProjectConfig:
    # Documenting experiments
    experiment_version: str = field(
        default="v0",
        metadata={'description': "Name of the training experiment"})

    experiment_description: str = field(
        default="This is test run",
        metadata={'description': "Describe the experiment in couple of sentences"})
```

Chapter 3

Data

Data module consists of:
- `data_read.py`: loading the local txt or csv files
- `create_datasets`: creating HuggingFace datasets from local files
-

Chapter 4

data_processing

Chapter 5

utils

Various utility functions for checking and defining compute engine, logging and creating the experimentation documentation.

Reproducibility is one of the most important aspects of proper project development and management, for ourselves, as well as for other people to whom we will share the project and possibly need to make decisions based on the results.

set_all_seeds

set_all_seeds(seed: int)

Set the seed for all packages: python, numpy, torch, torch.cuda, and mps.

Arguments:

	type	default	description
seed	int	None	Description not available

We can set the seed for most of the libraries that we use in machine learning like: `numpy`, `torch`, `torch.cuda`, `mps` as well as for Python in general.

```
set_all_seeds(seed=42)
```

If you are using `FLAGS` then simply replace the value of the seed for the data class defined for the reproducibility. For example, if my data class is called `seed` then I would use:

```
set_all_seeds(seed=FLAGS.seed)
```

check_device

check_device(verbose: bool)

Check which compute device is available on the machine.

Arguments:

	type	default	description
verbose	bool	True	Description not available

We can use the function as follows, which if the argument `verbose` is `True` it will print out the compute device currently available.

```
device = check_device()
```

```
Using mps device!
```

Creating experiment information document