Tools for machine learning

Tools for machine learning

- In recent years, the machine learning community had **settled** on a set of standard software tools that are used.
- These are not necessarily the best possible solutions for ML problems.
- They were chosen because they reached maturity at the right moment, and they work well together.
- Also, they are open source, free to use and supported by very large communities of users.
- New systems might emerge in the future. For the time being **make sure that you understand them well**
    - and only experiment with other tools once you know the standard ones well.

Objective of this lecture

- Present the various tools.
- Explain the logic behind their use.
- Explain how they work together.

Not an objective:

- Teach you the details of each tool.
- Extensive resources available on the internet.
- The tools' own tutorial the best starting point.

Python

- The **programming language** of choice for Machine Learning.
  - Interpreted language (no compilation, execution of individual lines possible)
  - Favors compactness and readability
  - **Extensive** standard library
- **Slow!!!** when compared to compiled languages like C/C++
  - But normally, you will only execute several dozen lines of python
  - The rest of the code will happen inside the libraries
  - Don't do loops over datasets in Python!
  - Looping over training epochs, or parameter alternatives is fine.

Jupyter notebook

- A software / data format that allows you to combine into a notebook a number of cells
  - Python code (other languages also supported)
  - Output of the code cells, text and visualization
  - Text, using the markdown format.
- You can use it by:
  - Running jupyter on your computer and connecting through a web-browser
  - In vscode
  - In Google Colab, Microsoft Azure Notebook or Amazon Sagemaker Notebook.

How to use Jupyter notebook

- Enables a workflow called **interactive computing**
- Individual cells can be executed in any order, or re-executed
- Static variables assigned are kept between cells
- imports are not re-imported unless you restart the notebook

Python programming in Jupyter and outside

- Python programming **outside** Jupyter
  - Proper programming techniques still apply
  - Functions, encapsulation, avoidance of global variables etc.
  - Object oriented programming is supported but not enforced
  - Functional programming is supported, but not enforced
- **Inside** Jupyter
  - Short scripts, usually not encapsulated in functions
  - Global variables are the typical way of communicating between cells.
  - Cells should be short and individually debuggable.

Numpy: vector and matrix operations in Python

- It is a library for matrix and vector computing in Python.
- Primarily: it provides support for a data structure called ndarray
- Why? Python already has an array data structure?
  - The Python array is an indexable list, collecting arbitrary data types.
    - `a = ["apple", 7, 8.1, {"b" : 2}]`
  - ndarray is a rectangular array, of a fixed data type (typically double or integer). It is an array
    - `a=np.array([[1,2,3,4],[3,4,6,7],[5,9,0,5]])`

Numpy for machine learning

- The numpy array is used as the underlying datastructure by **every** library for data science, ML, deep learning, etc. in Python
- When you have some data, you should convert it ASAP into a numpy array, and from then, use numpy operations to manipulate it.
- numpy has a useful collection of math algorithms. For instance: solving system of linear equations:

```
x = np.linalg.solve(a, b)
```

SciPy: scientific computing in Python

- Python library for scientific computing
- Operates on numpy arrays
- Extensive library of numerical algorithms. Some of them duplicate the ones in numpy while being more general etc.
- Interpolation, optimization, image processing, signal processing etc.
- In ML practice, it is the first place where you go shopping for an implementation of an algorithm. :-)

Pandas: manipulating datasets

- pandas: python library for data manipulation operations.
- Allows importing data from spreadsheets, databases, comma-separated text files etc.
- Primary data structure: DataFrame (approximately: a database table)
  - Convenient, python-style way to select data
  - **You should use pandas functions for work with it!!!**
  - Do not treat it as a huge array - do not iterate over it!
- In machine learning, typically it is used for:
  - **Data cleaning**: filling in missing values, eliminating corrupt data
  - **Data wrangling**: converting formats to those appropriate to downstream tasks (in this case, machine learning)

Pandas workflow:

- Interactive:
  - Load the dataset
  - Look / visualize
  - Perform changes and corrections
  - If you are a "data scientist" you can continue in pandas to analyze etc.
- Pandas workflow for ML:
  i. Load your data in pandas / convert into appropriate data types
  ii. Select the features you are interested
  iii. Fill in defaults / Filter rows that are bad
  iv. Convert to numpy
  v. The rest of the work will happen in ML algorithms.

Matplotlib: plotting data

- Matplotlib: python library for plotting and visualization
- Almost universally used by ML and data science practitioners to investigate data and present results
- Uses numpy arrays for data storage
- Provides a wide range of data plots: https://matplotlib.org/stable/plot_types/index.html
- Two ways to use:
  - **function-based** Matlab style interface (pyplot)
  - **object-oriented** Build the plot step by step.
- In this class: you should learn how to use at least scatter plots (for data) and line plots (for training progress)

scikit-learn (aka sklearn)

- Scikit-learn: python-based library for machine learning algorithms
- Uses numpy as underlying data structure
- It is the first place you look for most classical machine learning algorithms
    - It can also be used for neural networks, but if you want to do deep learning, more specialized libraries exist.
- If you try to develop new ML algorithms, it is usually a good idea to see whether you can beat scikit-learn

Deep learning libraries

- Libraries supporting deep learning: tensorflow, pytorch, jax etc.
- Almost always python, almost always building on numpy
- Main functionality
  - Automatic differentiation - ability to find the gradient of functions
  - Libraries for commonly used neural network components
  - Parallelization support for different hardware types (GPUs, TPUs)