ELEN0062 - Introduction to machine learning
A few words about Python

Jean-Michel Begon and Antonio Sutera

University of Liege

September 26, 2018

Spoiler

Take home message

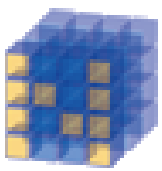
Python is great (especially for machine learning).

Pythonsphere

Python

Python is a general purpose, dynamically typed, interpreted programming language, which comes with tons of useful libraries for scientific computing. Use Python 3.5+.

NumPy



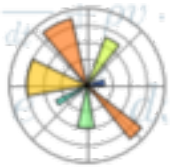
NumPy offers a rich N-dimensional array data structure, together with some important mathematical capabilities (linear algebra, Fourier transform, random number generation, etc.)

SciPy

Python's scientific library: integration, optimization, signal processing, statistics, etc.

Pythonsphere

Matplotlib



Python's plotting library. Line plot, scatter plot, histogram, pie chart, boxplot, it is in here. Together with NumPy and SciPy, covers everything that Matlab does (but it is free, and Python is richer, and indexing starts at 0).

Pandas

pandas

$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$

A library to analyze data. When doing machine learning, it is most useful for loading data and computing summary statistics about it.

Scikit-learn

scikit learn

The most widely used multipurpose machine learning library (in Python). It is built on top of NumPy, SciPy and Matplotlib.

Pythonsphere – and more

XGBoost

A library for fast boosting.

Deep learning

TensorFlow

A library to do (linear algebra) on GPU.

Keras

A framework on top of TensorFlow to do machine learning.

PyTorch

A library to do machine learning on GPU. More dynamic and pythonic than TensorFlow + Keras.

How to use Python

Interactive mode

1. Start Python shell
   ```
   $ ipython
   ```
2. Write Python code
   ```
   >>> print("Hello World")!
   Hello World!
   ```

Script mode

1. hello.py
   ```
   print("Hello World!")
   ```
2. Launch the script
   ```
   $ python hello.py
   Hello World!
   ```

Scikit-learn exercise

Run the following code:

```
from sklearn.datasets import load_digits
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

digits = load_digits()
X, y = digits.data, digits.target
```

1. Using numpy, compute the class proportions.
2. Divide the dataset into train/test sets. Use 70% of the samples as training set.
3. Instanciate a decision tree classifier.
4. Using the `fit` method, fit the decision tree on the training set.
5. Using the `predict` method, classify the test set.
6. Using the `accuracy_score` function, compute the accuracy.

Too fast, too quick?

Go through the crash course:
`http://www.montefiore.ulg.ac.be/~jmbegon/?pytutorial`