ROB313: Python Setup[*]

January 11, 2023

In ROB313 we will be using Python 3 for all assignments and in-class coding examples. Python is a powerful, clean, and well supported language that is popular for data analytics. Also, there are a huge number of powerful machine learning libraries available in Python, most of which we will not use in the course, however, by beginning in Python, you will easily be able to leverage these libraries for future work beyond this course.

Installation & Setup

This course requires a Python 3 installation with the packages `numpy, scipy, matplotlib, scikit-learn, pandas`. We strongly the use of the Anaconda distribution of Python which comes with all of the required packages for the course already installed.

If you do not already have Anaconda: Download the installer for your OS here. You can find download instructions at the same link as well.

If you already have Anaconda: If you have Anaconda then no action is required, otherwise you will need to upgrade the Python distribution. In a terminal (Mac or Linux), or Anaconda prompt (Windows) execute the following

```
conda install -c anaconda python=3.9
conda update --all
```

Programming in Python

Checkout this helpful Python refresher which reviews basic programming skills if it has been a while.

For program development, we suggest the use of Jupyter notebook/lab (a quick tutorial can be found here). To start Jupyter notebook, execute the following in a terminal (Mac or Linux), or Anaconda prompt (Windows)

```
jupyter notebook
```

To debug code using Jupyter notebook, we recommend using `pdb` from the Python standard library (see here for a quick tutorial on using `pdb` in a Jupyter notebook).

If you prefer a more traditional IDE, we suggest using Visual Studio Code with the Python extension.

Datasets

For the course assignments, we will use the list of datasets outlined in this document. Note that all multi-class classification datasets use a one-hot encoding, and all continuous features and responses are transformed to have zero mean and unit variance.

[*]originally written by Trefor Evans and Kevin Course, updated by Andrew Ilersich

Download the data .zip file from Quercus, within which is a `data_utils.py` module. The following code demonstrates how to load each of the datasets (note that `rosenbrock` is loaded differently since the number of training points and dimensionality must be specified manually).

```
from data_utils import load_dataset
x_train, x_valid, x_test, y_train, y_valid, y_test = load_dataset('mauna_loa')
x_train, x_valid, x_test, y_train, y_valid, y_test = load_dataset('rosenbrock', n_train=5000, d=2)
x_train, x_valid, x_test, y_train, y_valid, y_test = load_dataset('pumadyn32nm')
x_train, x_valid, x_test, y_train, y_valid, y_test = load_dataset('iris')
x_train, x_valid, x_test, y_train, y_valid, y_test = load_dataset('mnist_small')
```

Each dataset has designated training, validation and testing splits (of course, the testing splits should never be viewed during model training or selection of hyperparameters, it is only for reporting your final generalization performance). See the `data_utils.load_dataset` docstring for further details, if required. A description of all datasets is provided below.

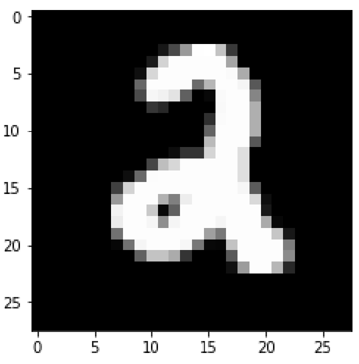
`mauna_loa`
($d = 1,\ n_{\text{train}} = 511,\ n_{\text{valid}} = 145,\ n_{\text{test}} = 73$, regression, source)
Temporal carbon dioxide measurements at an observatory near the summit of Mauna Loa. Measurements are monthly from March 1958 to November 2018.

`rosenbrock`
($d \geq 2,\ n_{\text{train}} \geq 1,\ n_{\text{valid}} = 1000,\ n_{\text{test}} = 1000$, regression, source)
An analytical function commonly used for non-convex optimization benchmarking. The user must specify the size of the dataset and the dimensionality. This dataset has no noise.

`pumadyn32nm`
($d = 32,\ n_{\text{train}} = 5733,\ n_{\text{valid}} = 1639,\ n_{\text{test}} = 820$, regression, source)
Prediction of the angular acceleration of one of the linkages in a Puma 560 robot arm.

`iris`
($d = 4,\ n_{\text{train}} = 104,\ n_{\text{valid}} = 31,\ n_{\text{test}} = 15$, 3-class classification, source)
Classification of iris flower species. The 3 columns of the binary response arrays specify whether the flower is *Iris Setosa*, *Iris Versicolour*, or *Iris Virginica*, respectively.

`mnist_small`
($d = 784,\ n_{\text{train}} = 10000,\ n_{\text{valid}} = 1000,\ n_{\text{test}} = 1000$, 10-class classification, source)
Classification of images of the hand-written digits 0-9. The original database is a mix of 70000 digits written by US census bureau employees and high-school students, and has become a popular classification benchmarking dataset. The inputs are vectors of length 784 and can be reshaped into a 28×28 image using `np.reshape`. Images can be plotted using `data_utils.plot_digit`.


0
5
10
15
20
25
0
5
10
15
20
25