labX_intro

August 16, 2017

1 Lab 1: Introduction

1.1 Background

This course includes a set of computer laboratories that explore topics related to signals and systems. These will be made available to you as the course progresses, and you are expected to engage with the content and produce and submit a set of results related to some defined tasks. Optional tasks will be included keep it interesting for students that are proficient with programming or are more comfortable with mathematics. Assistance will be provided during tutorial sessions if required, but it is really expected that you'll work in your own time and at your own pace: it's important right from the start for you to take responsibility for your own learning. You shouldn't be working in groups, but helping one another out on specific issues is fine.

The computer labs are based on the Python programming language, and require a kernel that includes all the standard numerical libraries. They have been created as worksheets using the *Jupyter notebook* system, which works in a browser and allows you to interleave text and code snippets that can be run interactively. However, there should be no problem with you implementing the required tasks in any other Python IDE if you prefer.

Jupyter notebook has been installed in the EBE green and red computer labs, via a package called *Anaconda*. If you want to install it on your own machine you can download it from https://www.continuum.io/downloads. You should probably choose the distribution based on the Python 3.6 kernel, since that's the one installed in the labs. The download is a few hundred megabytes, so if you rather want us to provide it locally on the campus then just make a request to the teaching assistant.

1.2 Getting started

The following procedure works in the EBE computer labs. There are two tasks to perform: fetching the notebook files, and running the notebook server. The initial setup takes a little work, but once done subsequent usage is simpler.

Instructions for how to run Jupyter on your own PC are similar, and there's lots of help on the web. You'll also need to install git if you don't have it.

1.2.1 First-time setup

From the Windows start menu select "Run", and in the box that pops up get a command prompt by typing `cmd` and pressing enter. Change to your home directory, which in my case is drive "F": type `f:` and press enter. You can list the files by typing `dir`.

Make a new folder, called for example "labs", using the command `mkdir labs`, then change into this folder using `cd labs`.

The notebooks are being actively developed and are hosted on github at address https://bitbucket.org/fnicolls/notebooks.git. While you could go to this web address and use the "download" menu item, it will be better to make a local copy of the repository. In the command prompt use

```
git clone https://bitbucket.org/fnicolls/notebooks.git
```

to do this. If successful, typing `dir` should show you a new "notebooks" directory. If a login prompt pops up then you typed the web address incorrectly.

Finally we can start the Jupyter notebook server. From the command prompt change to the "notebooks" directory using `cd notebooks`, and type and run `jupyter notebook`. A web browser should open up, showing a file view of the current directory. You can open the notebook corresponding to this document by selecting "lab1_intro.ipynb".

Note that you must keep the server running in the command prompt to use the notebooks. When complete you can shut it down by typing "control-c".

It is strongly recommended that you don't modify the files distributed in the git repository. Rather copy a file to a new filename before modifying it for your own purposes. That way you'll be able to pull updates from the repository without having to deal with conflicts. Note that your renamed files can still be in the same directory as the original without causing any problems.

1.2.2 Subsequent usage

If you've already created a local labs repository then you don't need to do it again. However, before working on a new task you should probably try to pull any updates that might have been made. Get a command prompt as before, change drive, and change into "notebooks" directory. Then execute `git pull` to update local copies of labs.

As mentioned, you should make copies of files before working on them, so that updates can be pushed without conflicts. If `git pull` gives an error then you've edited one of the distribution files. Move or rename the problem files and run again. If you want to just overwrite the existing ones, losing all local changes, use `git reset --hard` before repeating the `git pull` instruction.

Once updated, run `jupyter notebook` from the "notebooks" directory in the command prompt.

1.3 Gaining familiarity

One of the nice things about Python is that there's an active community of technical people that provide useful information and resources. If you don't know how to do something, then a simple web search can quickly help.

The remainder of this section points you towards some resources that might help in getting you familiar with basic mathematical functionality. Take a look at what's available, or find any other resources you prefer. Youtube videos are also an option.

There's a basic numerical Python tutorial at http://cs231n.github.io/python-numpy-tutorial, and a Python notebook version at https://github.com/kuleshov/cs228-material/blob/master/tutorials/python/cs228-python-tutorial.ipynb. A copy of this notebook that has been updated for Python 3 is included in the "deliver" directory of your "notebooks" folder. Load this notebook and work through the cells. Make changes to the contents, and press "shift-enter" to run and see the corresponding outputs. Pay particular attention to how to use numpy arrays, and how to plot using matplotlib.

Possibly the best place to find resources is the Scipy homege: https://www.scipy.org/getting-started.html.

You can also access function help in Jupyter notebook. If you run the cell below, for example, a help window for the matplotlib.pyplot.plot() function should pop up.

```
In [2]: import matplotlib.pyplot as plt
        ?plt.plot()
```

1.4 Tasks

- Linear regression
- Polynomial (or other kernel) regression
- Image processing
- Plotting of functions
- Orthogonality of functions.

```
In [ ]:
```