ECE 472/572 Getting Started

Dr. J. Gregor

January 17, 2023

1 Python and Anaconda

We will be using Python and packages like scikit-image to illustrate the algorithms covered in class. If you haven't already done so already for another course, you will need to install Python and Jupyter Notebook on your computer. There are several ways to do that, for example:

- Command line: pip install program-or-package
- Download and install Anaconda (comes preloaded)
- Download and install Miniconda + needed packages

Your friendly instructor once tried the command line approach and ran into all kinds of version control issues. The recommendation is to instead go the Anaconda route, even though it requires downloading and installing large amounts of data. For a MacBook Pro M1, the package was 500 MB which expanded to 3.5 GB when installed. Miniconda has a smaller footprint but doesn't come preloaded with everything you may need which may also result in version control issues.

2 Jupyter Notebook

The next step is to familiarize yourself with Jupyter Notebook. The web is full of tutorials such as https://realpython.com/jupyter-notebook-introduction/. Notebooks are a simple way to incorporate text, code and output. We will use them in class. You will also use them for your homework and project reports. We may even use them for the exams. The goal is not fancy formatting. Notebooks are merely a convenient way to keep comments, data, code, and output in a single file that can easily be shared.

3 Report Formatting

Jupyter Notebook comes with an HTML-like markup language called Markdown. LaTeX equations are also supported. You can for example produce inline math such as $f(x, \theta) = \sin(x/2 + \theta)$ or so-called math blocks like

$$(f * h)(x) \triangleq \sum_k f(k)\, h(x - k)$$

Again, the web is full of tutorials such as https://towardsdatascience.com/write-markdown-latex-in-the-jupyter-notebook-10985edb91fd which describes Markdown as well as LaTeX formatting.

This works for simple write-ups but not when sophisticated formatting is needed. In those cases, it may be more convenient to create a separate LaTeX document with the write-up and use notebooks just for data, code and results. If you are not familar with LaTeX, feel free to use MS Word in its place.

4 Displaying Images

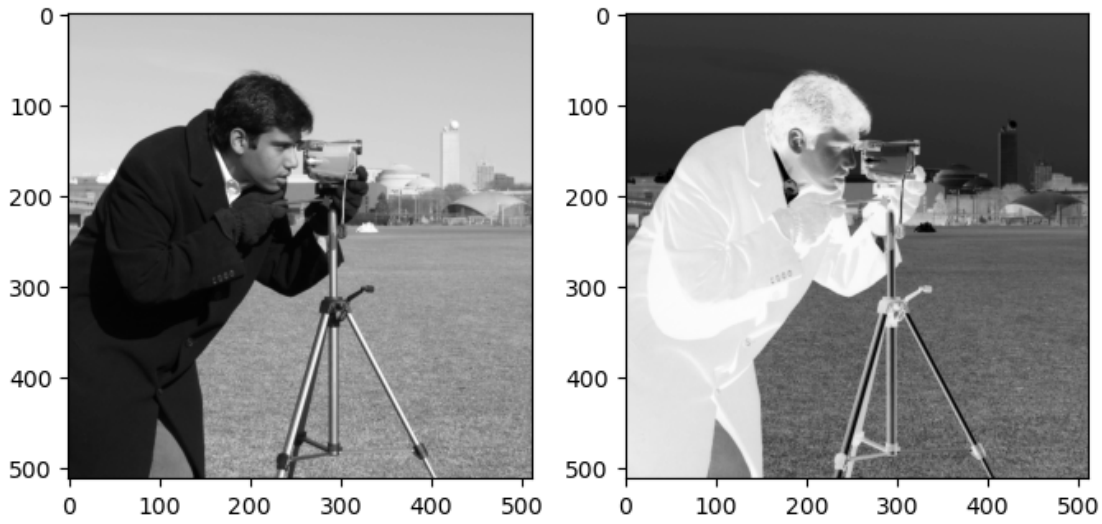
You will find that there are many ways to display images. Hands-on Image Processing with Python shows how to use PIL which has been deprecated and replaced by Pillow. The scikit-image package has an imshow functionality that does the job in a straightforward manner but doesn't give much control. Below, matplotlib is used which is a more powerful visualization tool. The display canvas is set to 8x4 inches, and the color maps are 'gray' and 'reverse gray' which corresponds to inverting the intensities (255-I).

[1]:
```
import matplotlib.pyplot as plt
from skimage import data

I = data.camera()

fig, ax = plt.subplots(1, 2, figsize=(8,4))
ax[0].imshow(I,cmap='gray')
ax[1].imshow(I,cmap='gray_r')
```

[1]:
```
<matplotlib.image.AxesImage at 0x10fc2af70>
```



5 Converting to PDF

This document was created as a notebook and saved to a file called class01.ipynb which in turn was converted to PDF using the command

```
unix> jupyter nbconvert hw1_setup.ipynb --to pdf
```

This required `nbextensions` to be installed. Going forward, you will likely find packages and tools to be missing when you try code from the textbook or some random webpage. Typically, you will be told what is missing and how to install it. On a Mac, homebrew can be used to install and update packages. A similar tool presumably exists for Windows.

Jupyter Notebook can also do the conversion in the browser. See File/Print Preview. However, the command line generated PDF looks nicer and is preferable.

NOTE: Markdown supports HTML formatting like linebreaks in the form of `<br>` and text color changes such as `<span style="color:blue">blue text</span>` only that doesn't seem to be understood by the conversion program as configured. Stay tuned for an update.

6 Report Title

By default, the PDF produced will have a title that corresponds to the name of the notebook. That may not be the report title you want, and it won't list you as the author. You can change that by editing the metadata. For example, the present notebook is called hw1_setup.ipynb but the title seen at the top of the PDF says "ECE472/572 Getting Started" and includes a fictional author. That was done by selecting Edit/Edit Notebook Metadata and adding title and author JSON objects:

```
{
  "kernelspec": { ... },
  "language\_info": { ... },
  "title": "ECE 472/572 Getting Started",
  "authors": [ { "name": "Dr. J. Gregor" } ]
}
```

There are other ways to add this information. Use whichever method works best for you and your set-up. Note that the above doesn't add the title and author info to the displayed notebook.

7 Submitting Reports

Notebook reports should be submitted as a PDF file that you create *after* you have done Kernel/Restart & Run All to produce results including plots and images. When submitting notebook code, be sure to the clean it up first, so all output has been removed. You do that using Kernel/Restart & Clear Output.