IntroML: Lab 1 - Setup and Data Loading

Kyle Swanson

January 8, 2018

0   Introduction

This lab will primarily consist of all the necessary setup for future labs. All labs in this class will be written in Python 3. Some labs will involve building machine learning algorithms from scratch in Python, while others will involve using pre-built Python machine learning packages. After you've finished all the setup steps, we'll use the rest of the time today to get used to working with data in Python.

1   Lab Setup

1.1   GitHub

All the materials in this class will be provided on the class GitHub: `https://github.com/swansonk14/IntroML`. The IntroML repository will be updated each day to include that day's lecture and lab material.

1. If you do not have a GitHub account already, set one up here: `https://github.com/`.

2. Log into `github.com` and create a new repository called "`IntroML`". Instructions for creating a GitHub repo can be found here: `https://help.github.com/articles/create-a-repo/`.

3. Clone the repo to your laptop. Instructions for cloning a repository can be found here: `https://help.github.com/articles/cloning-a-repository/`.

4. Open a terminal, navigate to the repo, and perform the following command:

   `git remote add class https://github.com/swansonk14/IntroML.git`

   From now on, you can get the latest class material with:

   `git pull class master`

   After working on each lab, add all the files you modified with:

   `git add <filename>`

Then, commit the changes using:

`git commit -m "<Commit message>"`

Finally, push the changes to your personal repo with:

`git push origin master`

If you're unfamiliar with git, please don't hesitate to ask me for help.

1.2   Python

In this class we will be using Python 3. I recommend installing Python 3 with Anaconda. Anaconda is a Python distribution which includes many useful Python packages, some of which we will be using in this class. To download Anaconda, follow the steps on the relevant page:

- Linux: `https://conda.io/docs/user-guide/install/linux.html`

- Mac: `https://conda.io/docs/user-guide/install/macos.html`

- Windows: `https://conda.io/docs/user-guide/install/windows.html`

Make sure to install Anaconda, *not* Miniconda, since Anaconda includes all the packages we want whereas Miniconda doesn't.

If you don't want to use Anaconda, you may install Python directly from `https://www.python.org/downloads/`

1.3   Python Packages

In addition to Python, we need to install some relevant Python packages. If you installed Python with Anaconda, you only need to install `keras`. If you installed Python without Anaconda, you need to install all of the following packages. Packages can be installed with the Python Package Index (`https://pypi.python.org/pypi/pip`) by running

`pip install <package-name>`

where `<package-name>` is replaced with each of the following package names:

- `numpy` - Array processing

- `scipy` - Scientific computing

- `scikit-learn` - Machine learning

- `pillow` - Image processing

- `matplotlib` - Plotting

- `keras` - Deep learning

You can test these installations by opening a Python shell (just run the command `python` in the terminal) and then running `import <package-name>`. Note that to import `scikit-learn`, you should run the command `import sklearn`, and to import pillow, you should run `import PIL`. All other import names are the same as the package names.

2 Data Loading and Plotting

This final section is intended to give you some practice working with data in Python. If you don't have time to complete this section, don't worry about it – I'll post solutions tomorrow.

2.1 Data Loading

In order to apply machine learning algorithms to our data, we need to load the data into Python in a form that makes sense for the problem we're trying to solve. One common data format is the Comma Separate Value (CSV) file. Data stored in spreadsheets (ex. Excel) can easily be converted to CSV format, so it's worth knowing how to load CSV files into Python.

To practice, we'll be loading a CSV file containing reviews of Amazon products. We'll run some machine learning algorithms on these reviews later in the week. All data in this class, including the Amazon reviews, can be found in the `Data` directory. Navigate to the `Data` directory and open up `reviews_train.csv` in a text editor to see the format you will have to process.

Now navigate to `Labs/Lab1`. In the `Lab1` directory, you'll see two Python scripts, `main.py` and `lab1.py`. `main.py` is the script you will run, and it imports functions from `lab1.py` which you need to implement. For this part of the lab, the function you need to implement is `load_reviews_data`. Read the function specification in `lab1.py` and implement the function. I recommend using the Python csv module: `https://docs.python.org/3/library/csv.html`. (Hint: Look up the `DictReader` csv function.)

Once your implementation is complete, run `python main.py`. If you see

```
Reviews data loaded correctly
```

printed to the terminal, then you have correctly implemented the function. Ignore any errors which may appear afterwards – those will be fixed when you implement the remaining functions.

2.2 Plotting

Another useful skill when working with data is the ability to visualize data and results. Visualizing data is incredibly important when trying to determine what machine learning methods are appropriate for the dataset. Visualizing results can help you understand what your machine learning model is doing well and where it can be improved.

We'll practice this skill with a very simple example: plotting two-dimensional data points. The data we'll be working with is in `toy_data.csv` in the `Data` directory. Open up the file with a text editor to see the format of the data. You'll see that the file consists of "labels", which are either $+1$ or -1, and x and y coordinates of data points.

Your job is to implement two functions in `lab1.py`: `load_toy_data` and `plot_toy_data`.

The function `load_toy_data` loads the text file and creates two arrays, a `data` array containing the two-dimensional points, and a `labels` array containing the label of the data point. To load this data, you can use the Python csv reader again if you'd like, but I found it easier to use the Numpy function `loadtxt` with `unpack = true` (`https://docs.scipy.org/doc/numpy-1.13.0/reference/generated/numpy.loadtxt.html`).

The function `plot_toy_data` should take these two arrays and plot the data as a scatterplot with the $+1$ points in blue and the -1 points in red. Plotting in Python is typically done using Matplotlib. To create a scatterplot, take a look at the `scatter` function of `matplotlib.pyplot.scatter` (`https://matplotlib.org/devdocs/api/_as_gen/matplotlib.pyplot.scatter.html`). Note that `matplotlib.pyplot` has already been imported for you as `plt`, so you can call the `scatter` function with `plt.scatter()`.

Once your implementation is complete, run `python main.py`. There should be no errors, and you should see

```
Reviews data loaded correctly
Toy data loaded correctly
```

Additionally, you should see a new window open up with the toy data plot. Your plot should look like this:
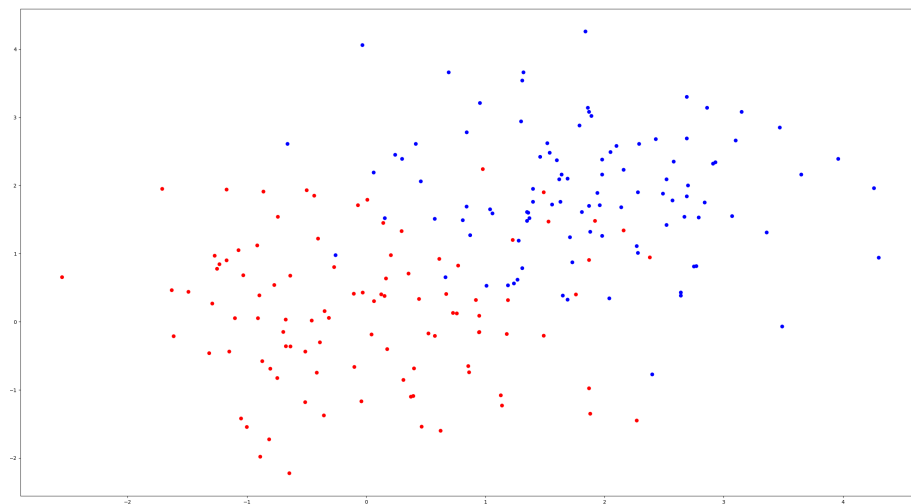
If your functions and plot are correct, you are done! Tomorrow we'll implement our first machine learning algorithm, the perceptron algorithm.