Intro to Python, Kaggle, and Google Colab

Ron Nachum, Irfan Nafi, Vinay Bhaip

September 2021

1 Python

Python is a high-level language. It's very easy to read and interpret but this also has a trade-off: it's slower for computers to run in comparison to C++, Java, and other languages. The language is determined by indentations, so there is no need for semicolons or many other characters - enters, tabs, and spaces hold the significance in structuring code. Python is rich in libraries and open source code that is easy to use and integrate into your programs. Python is very popular for ML use with popular libraries like Keras, Tensorflow, Sci-Kit Learn, PyTorch, and more.

1.1 General Good Practices

Variable names must not start with a number, rather a character or underscore. Generally, people type variables in a few main ways, depending on context and style:

- Method headers and variable names: neural_network
- Class and exception names: NeuralNetwork

Classes are essential to keep code organized and enable you to create objects and run methods on various data. This is very similar to the object-oriented programming you learn in the CS courses at TJ (albeit Java vs. Python differences). ML libraries we'll use this year are structured this way so you can instantiate and develop various models.

VSCode is a popular development environment by Microsoft that works great for Windows, Mac, and Linux. For local development, we recommend using VSCode. We'll get into alternatives later that you can try out (namely Jupyter Notebook and Google Colab).

Package management is essential as well, with the great number of libraries for Python that you will surely use many of in your projects, it is essential to be able to manage their versions and location on your computer. Those collections of python packages are called environments, which are isolated and don't affect one another. We recommend using Conda, a python package manager that makes it very easy to switch between different environments. Here are common Conda commands to get you started:

- Create a new environment: `conda create -n nameofenv python=3.7`
- Delete environment: `conda env remove -n nameofenv`
- List environments: `conda info --envs`

2 Kaggle

2.1 Introduction

Kaggle Classroom is the platform we will be using this year to host machine learning competitions. To ease the learning curve and encourage as many people as possible to compete, this lecture will help set you up with how to use Kaggle Classroom.

2.2 Setting Up

The steps for getting set up are pretty simple, and we make it easy to start coding right away.

1. Create a Kaggle account if you don't already have one by clicking "sign up" in the top right.
2. Click on the competition link, which will be posted in the lecture table on this page. Just to make sure you're able to submit something for practice, we've created a sample Kaggle Classroom competition at https://www.kaggle.com/c/tj-ml-sample-competition.
3. Download the training and testing data.
4. Download any shell code from the website. For nearly all competitions, we'll provide you shell code so that you can focus primarily on making the model.

2.3 Creating a Model

After downloading the training and testing data, the next step is to actually write your algorithm and train it on the training data. The shell code we provide will handle reading in the training and testing data so all you have to do is make the model and run it on the testing data.

When you run it on the testing data, you'll need to create a submission file with the predictions in the format shown in the sample submission file. It's important to note here that the format has to match exactly, otherwise, Kaggle won't accept your file. Generally, we ask for the output file to be in the CSV format with the first column corresponding to the test data number and the remaining columns corresponding to your predictions.

2.4 Submitting Solutions

All you have to do is upload the CSV file you generated to Kaggle and you're done! You'll be able to submit multiple submission files if you want to try to improve your algorithm. After a few seconds, you'll see how you rank on the public leaderboard. It's important to note that the public leaderboard displays how well you performed on a subset of the testing data. The reason this is done is to prevent people from randomly changing the output file to maximize their score. This encourages people to create the best generalized algorithm. Once the competition ends, the private leaderboard will display the scores for the remaining testing data.

Your ranking for the public leaderboard and the private leaderboard might change since your score is dependent on different data. While it's disappointing to see how you might jump from the top of the leaderboard to way below, it's important to understand that this happens all the time in real Kaggle competitions with thousands of dollars the line.

3 Google Colab

3.1 Introduction

As machine learning algorithms require more and more computational power to be successful, it is important to find a way to overcome this barrier in order to create successful models. To combat this, Google has released Google Colab, an online coding environment with the ability to harness the power of GPUs, which are advanced processing units that vastly accelerate the time needed for machine learning models to learn.

3.2 Setting Up

After logging into your Google account, go to colab.research.google.com. Right off the bat, you'll have the ability to either import code from Google Drive, Github, or your own computer. You can also just create a new Python3 or Python2 notebook, though we recommend Python3 as Python2 will lose support in 2020.

It's important to note that when creating a file, it will be made with the .ipynb extension, which indicates its a Jupyter Notebook, rather than just a file. A Jupyter Notebook allows for some more nice features than a normal Python file. For the purposes of what we'll do, you can treat both as the same, however.

The most powerful feature of Google Colab is the ability to run your code with a lot of computational power. Under the "Runtime" tab, you'll see an option to "Change runtime type." When you click that, you'll have the ability to change your hardware accelerator to a GPU, which is ideal for running machine learning algorithms. Additionally, you'll see an option to use the very powerful TPU, a processing unit made by Google specifically tailored for AI and ML. It's harder to configure most ML libraries for TPUs than it is to configure them for GPUs, so we recommend just sticking with GPUs. Also keep in mind that Colab will time out eventually when you use the TPUs, so it is not ideal in all scenarios. At the beginning of the year, you probably will not require the use of a GPU for our competitions, but when we delve into neural networks and beyond, it will definitely be helpful.

3.3 Running Code

It's pretty simple to run code in Colab. It's structured just like a Jupyter Notebook, which means that you'd write code in "blocks" and execute those blocks of code. This is helpful with debugging and reading your code.

Most libraries you need such as Numpy, Keras, and Tensorflow are already installed into Google Colab so you don't need to worry about that. For reading in files, such as CSV files from our in-house Kaggle competitions, we recommend you check out https://colab.research.google.com/notebooks/io.ipynb, which covers this topic.

We encourage you to explore all the tools Google Colab has to offer as some may be helpful to whatever you are working on.