# *Assignment 3: Introduction to Neural Networks*

*UMaine COS 470/570*

*Fall, 2021*

---

Assigned: 10/22/2021 | Due: 11/2/2021

---

In this homework assignment, you will play around with a neural network tool, TensorFlow, developed by Google and now open source. In particular, you will be using the Keras interface provided now as part of TensorFlow. Sadly, the best interface to TensorFlow is not Lisp, but Python, so that's what you'll be using in this assignment. (Try to contain your disappointment; we'll be back to Lisp really soon!)

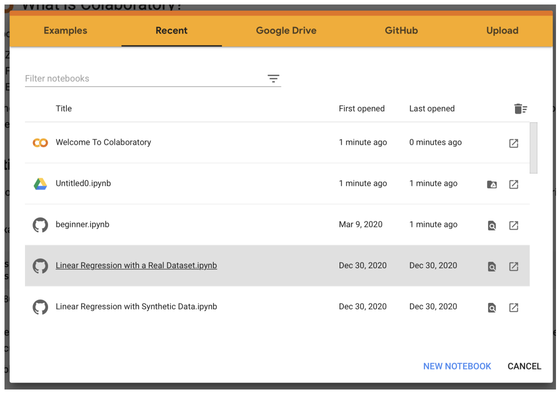
## *Part I: Setup*

There are two easy ways to access TensorFlow, either locally or using Google's own servers. Either is fine for this assignment, but you may want to download it so that you can more easily play with it. And, of course, Python is a very useful language for all sorts of things, and it's one of the major ones for data science and machine learning.

### *Using Google*

To use the Google servers, connect to the Google Colaboratory. When you connect in a browser, it will pop up a window that looks like:[1]

[1] If you have pop-ups blocked, you may have to change the settings, but since I do and I didn't, it may just work for you, too!



| Title | First opened | Last opened |
|---|---|---|
| Welcome To Colaboratory | 1 minute ago | 0 minutes ago |
| Untitled0.ipynb | 1 minute ago | 1 minute ago |
| beginner.ipynb | Mar 9, 2020 | 1 minute ago |
| Linear Regression with a Real Dataset.ipynb | Dec 30, 2020 | Dec 30, 2020 |
| Linear Regression with Synthetic Data.ipynb | Dec 30, 2020 | Dec 30, 2020 |

You can select the first thing, "Welcome To Colaboratory" to see how to use it, etc., or you can select the "New Notebook" button at the bottom to get a new Jupyter notebook.

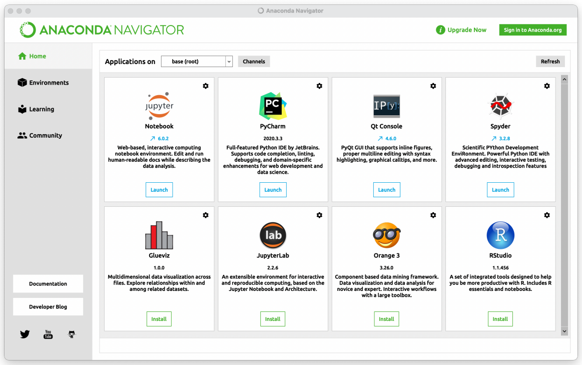
## *Setting things up on your computer*

I suspect that most of you have Python already installed on your computer. If not, the first thing you need to do is install it. You can find a wealth of information about Python on the Web, especially at Python.org. There, you can download a version of Python for your particular operating system. I suggest downloading Python 3 (I think 3.9 is the current version at this time). You should be able to install it using a package manager, such as `apt` on Linux or `homebrew` (see brew.sh) on macOS.[2]

[2] Homebrew works very well for lots of things, including not only Python, but also Emacs and SBCL ("Now he tells me!").

**However:** I would like to suggest that instead of doing the above, you consider installing Anaconda (anaconda.com).[3] This provides a very good installation experience and, if you use the included Anaconda Navigator (Fig. 1), a visual package manager, Jupyter, JupyterLab, PyCharm, and a ton of data science and machine learning tools. It can also manage multiple Python environments, which lets you have different setups for different purposes.

[3] Please, no Nicki Minaj or Sir Mix-A-Lot jokes.



Notebook
6.0.2
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

PyCharm
2020.3.3
Full-featured Python IDE by JetBrains. Supports code completion, linting, debugging, and domain-specific enhancements for web development and data science.

Qt Console
4.6.0
PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.

Spyder
3.2.8
Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features

Glueviz
1.0.0
Multidimensional data visualization across files. Explore relationships within and among related datasets.

JupyterLab
2.2.6
An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

Orange 3
3.26.0
Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox.

RStudio
1.1.456
A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks.

Figure 1: Anaconda Navigator

If you happen not to know Python, you'll need to learn at least the basics, although you won't need a lot to use Keras at first. A good place to start is "The Python Tutorial".[4]

[4] docs.python.org/3/tutorial/index.html

I'm not going to assign any particular problems for you to do; however, you might consider re-doing the Lisp practice problems in Python, where possible.

You will find that the tutorials at tensorflow.org all use Jupyter notebooks, which you can use on Google's servers, via Anaconda, or if you install Jupyter in Python some other way.

Once you have Python installed, you need to install TensorFlow on your computer. The current version includes Keras. If you're using Anaconda, you can use its Navigator or command-line package manager (`conda`) to install TensorFlow. If you have a good NVIDIA GPU, you might also want to explore installing NVIDIA's CUDA and the GPU variant of TensorFlow to speed up your work. (Unfortunately, CUDA only works on Windows and Linux.)

If you don't want to use Anaconda, you can install the programs by using Python's package manager, PIP.

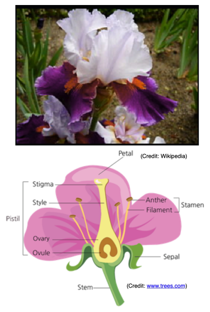


Figure 2: Top: Iris; Bottom: Flower anatomy.

## Part II: Some simple NNs

### Diabetes Among the Pima

There's a very easy tutorial to get your first neural network up and running at:

machinelearningmastery.com/tutorial-first-neural-network-python-keras

This tutorial uses a well-known dataset in machine learning, one that associates medical data for Pima Native Americans with their onset or not of diabetes within five years. You will use it to create a neural network that will predict the onset of diabetes given medical data.

Do the tutorial to create a network for the Pima dataset and run it.[5]

[5] There are also *many* other tutorials online, including ones at tensorflow.org that are for beginners. Feel free to play around with some!

**Note:** If you are using Python on your own computer, and when you do a line such as

```
from keras.models import Sequential
```

you get an error—and I don't think you will—it means that Python can't find Keras based on how TensorFlow was installed. This can be fixed if you first do this:

```
import tensor.keras as keras
```

before importing anything from Keras.

### *Identifying clothing*

This tutorial is directly from TensorFlow itself. Go to this link[6] and do the tutorial. You can either click the "Run in Google Colab" to use Google's servers, click "Download notebook" to run it using Jupyter on your machine, or just enter the example into a Python interpreter or notebook from the tutorial.

[6] Also available from the main TensorFlow page, then "Getting started with TensorFlow", then "See tutorials", then "Keras basics".

## *Part III: Create your own*

I'll put a version of the well-known "Iris" dataset on BrightSpace. This dataset, dating from 1936, relates four parameters of irises (petal length and width, sepal length and width) to their species (*Iris setosa*, *I. versicolor*, or *I. virginica*). (See Fig. 2.)I've manipulated this so that the dataset is shuffled and in a form amenable to a "softmax" output. A softmax output layer, which you learned about in the second tutorial, is one that contains $n$ neurons, one for each possible category of output, and whose activations sum to 1: thus, each neuron's activation can be viewed as the probability that the input is a member of the corresponding category.

The format of the iris data is a CSV file:

```
sepal-length, sepal-width, petal-length, petal-width, I.setosa, I.versicolor, I.virginica
```

So this data item:

```
6.6, 3.0, 4.4, 1.4, 0, 1, 0
```

would represent an *Iris versicolor* with sepal length and width of 6.6 cm and 3.0 cm, and with petal length and width of 4.4 cm and 1.4 cm.

Your job is to create **two** neural networks to classify irises based on values of the four input parameters. You can use as many layers and neurons per layer you like for each; a simple single-hidden-layer network is sufficient, although you can experiment with other kinds to see how your accuracy changes. Although you *could* use any type of layer, I would suggest sticking to the "dense" layers used in the tutorial. The output layer, however, *must* be a softmax layer with 3 neurons, one for each type of iris. Vary the number of layers and/or the number of nodes per layer between the two neural networks you construct.

I'll split the dataset into two files, `iris_train.csv` and `iris_test.csv` to use for training the net and testing it, respectively.

## *Part IV: Turn in...*

Turn in a file containing:

- A *brief* (1–2 page) discussion of your impressions of and experience with doing the tutorials.
- A *brief* (1–2 page) description of the networks you created and a discussion of what their differences meant for the accuracy of their answers. Relate any insights or hypotheses you may have about why what you observed happened.
- Your Python programs for the two networks. If you used Jupiter notebooks, you can just turn in the .ipynb file.
- Output from the two programs[7]
  - Your output must include a line with the accuracy of your for test data.
  - It must also include the results of predicting the output given the inputs for the test data.
  - If you use Jupyter, you can just turn in the notebooks (the .ipynb files), since they should have this information in them (make sure, though!).

[7] I'd strongly suggest running the program, then getting a screen shot of the tail-end of the "Epoch. . . " things, the accuracy, and the predictions; as you will note when you run it, lots of control characters are embedded in the output, which makes printing or viewing any captured output problematic.

For example, my output looks like this:

```
(keras) [rmt@rturner-2 NN]$ python iris.py
Using TensorFlow backend.
Epoch 1/150
132/132 [==============================] - 0s - loss: 0.6367 - acc: 0.6667
Epoch 2/150
132/132 [==============================] - 0s - loss: 0.6365 - acc: 0.6667
...
Epoch 150/150
132/132 [==============================] - 0s - loss: 0.3273 - acc: 0.7828
15/15 [==============================] - 0s

acc: 71.11%

Predictions (based on test dataset):
prediction=[ 0.11855328  0.44186708  0.43957967]
prediction=[ 0.98168379  0.01319122  0.00512492]
prediction=[ 0.09259017  0.44522405  0.46218583]
prediction=[ 0.10822026  0.44351041  0.44826928]
prediction=[ 0.09259017  0.44522405  0.46218583]
prediction=[ 0.09259017  0.44522405  0.46218583]
prediction=[ 0.10330817  0.44415826  0.45253354]
prediction=[ 0.09259017  0.44522405  0.46218583]
prediction=[ 0.99175453  0.00614701  0.00209844]
prediction=[ 0.09259017  0.44522405  0.46218583]
prediction=[ 0.09259017  0.44522405  0.46218583]
prediction=[ 0.09259017  0.44522405  0.46218583]
```

```
  prediction=[ 0.09259017  0.44522405  0.46218583]
  prediction=[ 0.12725127  0.44022453  0.43252417]
  prediction=[ 0.09259017  0.44522405  0.46218583]
```

In this case, the accuracy of the network was 71%, and (for example) the prediction for the last of the tests is that it is type 3, with a probability 0.46 (it actually is a type 3).