**Simple Preparation for Numpy and Tensor Flow for Machine Learning Use – Setup and Test of the Basic Toolchain**

Note: This example was done in Windows, but it likely works better with Mac or (preferably) Linux. Windows does not have a number of the prerequisite files available to install via the package manager PIP, in this case you need to bring them in externally. Details of this are described. If Python is already setup and properly configured, many details in this description are irrelevant.

The following was done to setup the toolchain:

1. Python 3.6 was installed (x-64 version), all users and paths created. I like to put python in the following location: C:\Python36. I made a directory in here for files I created called “workspace”. Python version 3.6 is compatible with Tensor Flow in windows, yet 2.7 was tested and is not.
2. PIP was used to install Tensor Flow
3. PIP was used to install TFlearn
4. PIP was used to install h5py
5. [http://www.lfd.uci.edu/~gohlke/pythonlibs/](http://www.lfd.uci.edu/~gohlke/pythonlibs/#curses) was used to source numpy+mkl, curses, and scipy. The WHL (wheels) were downloaded and copied to the Python36 directory. Wheel (WHL) files were loaded via the following example command: pip install c:\Python36\scipy-0.19.1-cp36-cp27m-win\_amd64.whl (Note CP36 sets the version of python this works with – 3.6, respect if you are x64 as well – if you do not get this it will not load). Load each individually.
6. If you plan to make plots in the terminal (some examples use this), install matplotlib with the following “pip install matplotlib”

Test with the following examples:

1. At this point, you should be able to run the “linear\_regression.py” example without issue. Same should be true for the “multiple\_regression.py”. Download the files and place in the “workspace” directory, execute with “cd \Python36\workspace” followed by “python linear\_regression.py”. This assumes that PYTHON is properly set up with OS PATH.
2. Try a more complicated example: <http://cs231n.github.io/neural-networks-case-study/>. First verify if you can load and display the example data:

from \_\_future\_\_ import absolute\_import, division, print\_function

import tflearn

import numpy as np

import matplotlib.pyplot as plt

N = 100 # number of points per class

D = 2 # dimensionality

K = 3 # number of classes

X = np.zeros((N\*K,D)) # data matrix (each row = single example)

y = np.zeros(N\*K, dtype='uint8') # class labels

for j in range(K):

ix = range(N\*j,N\*(j+1))

r = np.linspace(0.0,1,N) # radius

t = np.linspace(j\*4,(j+1)\*4,N) + np.random.randn(N)\*0.2 # theta

X[ix] = np.c\_[r\*np.sin(t), r\*np.cos(t)]

y[ix] = j

# lets visualize the data:

plt.scatter(X[:, 0], X[:, 1], c=y, s=40, cmap=plt.cm.Spectral)

plt.show()

1. Try the Numpy Softmax classifier example – follow the example in the “softmax\_classifier\_partbypart.py” as you follow along the example on <http://cs231n.github.io/neural-networks-case-study/>. I put it together to work line by line of the explanation.
2. Try the Numpy Softmax classifier example that was set to work with the spiral data in a way that outputs convergence progress and view visualization.
3. Play with a Neural Network example: http://playground.tensorflow.org/
4. Try the Numpy NeuralBoundary visualization example: “neural\_decision\_boundary.py” This is a very simple example of plotting boundaries from neural network problems.
5. Try the “neural\_decision\_boundary.py” example and view visualization.
6. Try the TensorFlow “autoencoder.py” example. This program will download handwriting data to a generated folder called ”MNIST\_data” and perform recognition. This example is sourced from <https://github.com/aymericdamien/TensorFlow-Examples>