**Assignment 5 – Linear Perceptron and Adaline**

For this assignment you are going to compare the results from Excel to those of Python using the small dataset that we worked with in the class. We are going to do this for both the Linear Perceptron and Adaline learning algorithms.

The goal of this assignment is to ensure that you understand exactly what is going on when we use both Excel AND Python. This will help you cross-check all of your work, because the results will be identical.

The Python code that I am using is from the author of the following book…

**Python Machine Learning**

By Sebastian Raschka and Vahid Mirjalili

The code is from Chapter 2 and is freely available from github. I am giving you the relevant code for this assignment after removing the unimportant pieces. You simply have to cut and paste the code at the beginning of your .py file to use it. You will also have to include numpy with

**from** numpy **import** \*

placed as the first line in your file. Note that this is not a course in using Python. We are using it because it is simple and effective for doing numerical computation – particularly when working with linear algebra. You may have plenty of questions about using Python and I would encourage you to ask me as you go through the assignment. However, I will most often return to our focus, which is solving numeric problems, when answering them. Python is a rich language with a very specific way of doing things and now is not the time to try to sort out all of that – i.e. the “Pythonic” way.

This assignment is designed to tie the work we are doing in Excel with using Python. My first inclination was to set this assignment up so that you would do a simple example for both the Linear Perceptron and for Adaline in Excel and then a bigger one in Python. We will do a bigger example using the Support Vector Machine in Python next week and that will set things up nicely for your project. For now, given that this is the first time many of you have worked with Python, I felt it was best that we start of small AND have Excel available to check the results.

As an aside, when doing numerical work of this kind, it is absolutely **imperative** that you are confident that the results you are getting are correct. One small error somewhere can render the results meaningless and you will have no way of knowing this has happened! Further, I have found it unwise to trust anyone’s code unless I can verify that it does what it claims. As a result, I compared the author’s code to the Excel results iteration by iteration and line by line. That is, in essence, what you can do with this assignment.

In any case, your task is to read in the small file that I have supplied, called data1.txt. You can simply place that file in the same folder as your .py file.

Importing the data file couldn’t be simpler – it is a comma delimited file with 3 columns. The first two contain the two attribute values and the third contains the class, which I have labeled as -1 or 1. The command to do this and read the comma delimited file into an array called rawTrainingData is

rawTrainingData = loadtxt('data1.txt', delimiter = ',')

The Perceptron and Adaline algorithms expects the data to be separated into 2 parts – the attribute values in one array (which I called X) and the class label in a vector (which I called y). This is also easy to do in Python WITHOUT using loops:

X = rawTrainingData[:,0:2]

y = rawTrainingData[:,2]

I am not going to tell you how the array indexing works in Python in this assignment description – I want you to look that up so that you remember it!

The author wrote object-oriented code. So, what you need to do is create two objects from the classes that I provided – Perceptron and AdalineGD. There is a constructor that is used to initialize the objects with the learning rate (eta) and the number of iterations (n\_iter). There are then 2 member functions that you will use – fit and predict – for each of the classes. Fit is for training and predict is for getting the predicted label. Again, I am not going to tell you exactly how to use these – I want you to have a look at the code and try to reason it out! Use common sense to tell you what data should be supplied to these member functions.

In Python, you should output the prediction of the class of the training data at the end of the iteration for both the Perceptron and for Adaline. In both cases, the results will be perfect – there will be no errors. Most importantly, though, you should also output the 3 weights EACH iteration! To do this, you will have to modify the code within the classes to send the output to the screen (or an array for use later). Another option (if you cannot figure out how to do that) would be to set break points at the end of each iteration and check the values of the weights.

You should use a learning rate of 0.1 for the Perceptron and 3 iterations is sufficient. For the Adaline learning rule, use a learning rate of 0.001 and 8 iterations.

Now, in Excel, implement both the Perceptron and the Adaline algorithms as I showed you in the class and in the sample videos (which are almost ready for you). Again, run through 3 iterations for the Perceptron and 8 iterations for Adaline with the same learning rates as for the Python example. Verify that the weights after each iteration for both algorithms are exactly the same as they are from the Python code. To do this, you will have to get the random numbers that are used to initialize the weights from the first iteration of your Python code. This is easier than it sounds. Each time you run the code, you will end up with the SAME random numbers – in fact, the weights will be initialized the same for both the Perceptron and Adaline algorithms! This is simply because random number generators aren’t random unless they are SEEDED with a different value each time they are called. Sometimes, the current time (in seconds since 1970 or some such thing) is used. But the author didn’t do this – the random number generators are seeded with the same value. So, you get the same random number the first time you call the generator function. If you called it again, you would get a different number, but it is only called ONCE at the beginning of the code!

**Note: this is all going to take some work and some thought. Do not give up on this until you are convinced that what you have in Excel and Python are the same!**

Lastly, if you have time, for Adaline, calculate the exact value of the weights using the pseudoinverse as I showed you in the class. Determine roughly how many iterations it takes to arrive at the values calculated by the pseudoinverse with the learning rate of 0.001. If you wish, try changing the learning rate and the number of iterations and see if you can arrive at the exact solution more quickly.

Above all, have fun with this assignment! With a small dataset and verifiable results, you should be able to gain confidence that you understand how these algorithms work. That will also give you confidence that you have developed skills for working with machine learning algorithms. These assignments are not meant to be an academic exercise, they are supposed to be a learning opportunity!

Please EMAIL or set up a WebEx meeting with me if things are not going well early on – either in setting up Python or getting started with the assignment. Nothing can be gained out of this if you are frustrated by the setup or are having difficulty understanding what is being asked. These things can be resolved very easily which will allow you to move on to the important part – the algorithms.