Homework 2 **NSC 3270 / 6270**
Due Tue Sep 14 at start of class **Fall 2021**
10 points

The main goal of this assignment is to make sure everyone has some of the basic Python programming skills required to be successful in this course, using examples relevant to the course. Feel free to reference the example code we went over in class and that has been posted on Brightspace.

This assignment should be completed as a Jupyter Notebook. Remember to reset the kernel and clear the output and conduct a final complete run of your code before turning it in on Brightspace with the Jupyter Notebook outputs displayed.

Q0. Demonstrate that you can add some formatted text in your Notebook (following some of the examples from class and on Brightspace). Make sure your code is broken up into individual cells that each implement one and only one subquestion asked below. The cells should either be commented or prefaced by a markdown cell to make clear what each code cell is doing (in relation to the assignment). We will be looking for you to use markdown cells and comments, and to separate your code cells at logical breaking points when grading Jupyter Notebooks throughout the semester.

Q1. This question asks you to code up two common activation functions (as Python functions) and create plots of them.

a) Create two different Python functions that implements the **logistic** (also called sigmoidal) activation function given by the following formula:

$$a(n) = \frac{1}{1 + \exp(-n)} = \frac{1}{1 + e^{-n}}$$

where a is the activation at each value of n and n is a numpy array of net input values. One of the functions should use a for loop to fill in values of $a(n)$. The other function should use "vectorized" operations (without a for loop) to fill in values of $a(n)$.

b) Create two different functions that implements a basic **relu** (rectified linear unit) activation function given by the following formula:

$$a(n) = \begin{cases} n, & n \geq 0 \\ 0, & n < 0 \end{cases}$$

where a is the activation at each value of n and n is a numpy array of net input values. One of the functions should use a for loop to fill in values of $a(n)$. The other function should use "vectorized" operations (without a for loop) to fill in values of $a(n)$ – this might take a bit of searching on the example Python code I shared to find something that works in a "vectorized" manner (thought there's not just one way to do it – just don't use for loops here).

c) Choose a range of values for n and increments for n to create a smooth plot (using matplotlib) that shows the full shape of each activation function. You should create a function that completes the plot since you will be creating four separate plots here. Make sure your plots are labeled fully.

d) Instead of calling the activation functions using *n*, I want you to call them using values defined by a function *n* = *wx*+*b*, where *w* is a scalar weight value and *b* is a scalar bias value, and *x* is a numpy array with an appropriate range of values and increments. For each of the two activation functions, logistic and relu (you can pick which two version of the function to call), create a plot of each function for 3 different values of *m* all on the same graph (overlapping) and create a separate plot of each function for 3 different values of *b* all on the same graph (overlapping) (make sure you show a legend of the values) – see Brightspace slides for examples. Pick different values of *m* and *b* that illustrate how these scalar values change the shape of each function. Again, you should create a function that plots the three curves on the same graph (overlapping) since you will be creating four separate plots here. Make sure your plots are labeled fully. A slide from class shows an example of what some of the plots might look like.

<u>Neuroscientific Motivation:</u> We will be working with simplified models of neurons in this course. The logistic and relu functions are commonly used to model a neuron's activation to the incoming net input. A key point is that the neuron's response is nonlinear. Low amounts of input don't affect the neuron's activity much, but as the net input increases, there is a point where a small change in net input has a large effect on the neuron's activation; for the logistic, at some point the neuron saturates, it is as active as it is going to get, and small additional increases in net input don't have much of an effect.

Unexcused late assignments will be penalized 10% for every 24 hours late, starting from the time class ends, for a maximum of two days, after which they will earn a 0.