**Homework 2** **NSC 3270 / 6270**
**Due Tue Sep 13 by start of class** **Fall 2022**
**10 points**

The main goal of this assignment is to make sure everyone has 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.

<u>This assignment should be completed as a Jupyter Notebook</u>.
Before submission: Reset the kernel, clear the output, and conduct a final complete run of your notebook. Then submit the notebook on Brightspace with the Jupyter Notebook outputs displayed.

**Q0.** Style! Demonstrate that you can annotate your Notebook with descriptive formatted text (following some of the examples from class and on Brightspace). Make sure your code is broken up into individual cells. Don't answer multiple sub-questions in the same cell. Each sub-question should have its own cell, but sometimes you'll want to use more than one cell for a given sub-question. Your cells should either be commented or prefaced by a formatted markdown cell to make clear what each code cell is doing (in relation to the assignment). When we grade Jupyter Notebooks, we will be looking for you to use markdown cells and in-code comments sensibly, and to separate your code cells at reasonable places. You'll get feedback on style as we go along!

**Q1.** Create Python functions implementing two common activation functions and create plots of them. Each function should take a numpy array *n* as an input variable, and return a numpy array *a* as an output variable.

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

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

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

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

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

Again, $a$ is the activation value corresponding to 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 $a(n)$. As above, the other function should use "vectorized" operations (without a for loop) to fill in values of $a(n)$. This one is a little tricky! It might take a bit of searching through the course's example Python code to find something that works in a "vectorized" manner (though 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. Create a function that creates the plot, and use this function for this part of the assignment and the next part, as you'll be making several versions of your figures. Make sure your plots are labeled fully.

**Q2.** Making a tiny neural network with one weight and one bias value, and then manipulating those values to affect your activation plots.

For this sub-question, instead of creating the numpy array $n$ directly, you will create a neural network function that has three input variables: a numpy array of input values $x$, a single weight value $w$, and a single bias value $b$. This function will return $n$, which can then be passed to your activation function.

The neural network function will define a network by the 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 version of each function to call), you will create two plots. The first figure will display the activation function with 3 different values of $w$ all on the same graph (the lines may overlap somewhat). The second figure will display the activation function with 3 different values of $b$ all on the same graph. You'll make 4 figures total: the logistic function adjusting w, the logistic function adjusting b, the relu function adjusting w, and the relu function adjusting b.

Since you'll have 3 lines on each graph, make sure the lines are different colors, so we can tell them apart! Make sure you include a legend that shows which color goes with which value – see Brightspace slides and code for examples. Pick values of $w$ and $b$ that illustrate how these scalar values change the shape of each function. Make sure your plots are labeled fully (title, axis labels, axis numbers, legend). A slide from class shows an example of what some of the plots might look like.

Neuroscientific Motivation: 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 given a certain value of 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 function, 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.*