Homework 5 **NSC 3270 / 6270**
Due Tue Oct 12 at start of class **Fall 2021**
20 points

For this assignment, I want you to implement some aspects of this web site that we looked at a couple of times during class: http://playground.tensorflow.org/

You will generate distributions of training patterns like some of those used on that web site (as discussed and demonstrated in class), train and test a simple neural network (with an input layer and an output layer) using Keras and Tensorflow (use the code reviewed in class as a starting point), and visualize the predictions of the network after training (using methods discussed in class).

Please refer to `Homework5.ipynb`, which is the starting point for this assignment, and the class slides where we went over details of the assignment. The coding you do should be added to `Homework5.ipynb`. Save it to a new file with your last name appended to "Homework5". Submit on Brightspace. There is no need to submit any figures with your code. Your Jupyter Notebook should create all the figures needed. Make sure you submit your notebook after it has been run (with figures and calculations shown).

As usual, try to use functions where appropriate, use variables rather than hard-coded values in your code, and use comments and/or markdown cells to document your code.

To complete this assignment, you should review the example Keras and Tensorflow code that we went over in class. All of the coding elements (functions and methods to call, parameters of functions and methods to pass) that you need to use (and adapt) are in various places throughout those examples.

Q1 (10 points). `Homework5.ipynb` includes code to generate examples for a simple classification problem, with two classes, with instances from each class drawn from a multivariate normal distribution (as discussed and demonstrated in class).

Write Keras code to learn this classification problem with a neural network with an input layer containing two nodes (for the x and y dimension) and one output node (no hidden layers). The network will be trained to produce a value of 0 on the output node for patterns from class 0 and a value of 1 on the output node for patterns from class 1.

You will need to use the `sample0` and `sample1` numpy arrays generated by the function `gensamples()` to produce a single numpy array containing the training patterns (see `Homework5.ipynb` for where these variables and functions are defined). You can do this explicitly with for loops; you can also research how to concatenate two numpy arrays using `np.concatenate()` (you can check out `NumpyArrays.ipynb` posted earlier in the semester).

You will also need to create a numpy array containing the teacher values associated with each training pattern.

You will need to pick the appropriate activation function for a classification network with these particular values of the output node. For this assignment, you should use stochastic gradient descent as the optimizer; follow the example code I gave you where SGD is configured so that `learning_rate=0.01`, `decay=1e-6`, and `momentum=0.9`. Use mean squared error (`mean_squared_error`) as the loss function. You should save `'accuracy'` as one of the metrics when you compile your network.

For now, assuming a batch size of 20.

You will need to select the number of training epochs so that the accuracy asymptotes (it changes by no more than .5% from epoch to epoch). Make sure you try training the network using several different randomizations of the training patterns (try training at least 10 times to hone in on a number of epochs that is in a sweet spot for most randomizations of the training patterns). In other words, the number of epochs should be large enough so that the network learns (to asymptote) but not so large that it runs unreasonably long. You will want to set `verbose=True` during training to see how the performance of the network evolves over training epochs.

Q2 (3 points). An object is returned when you call `history = network.fit(...)`. Recall that `history.history` is stored as a Python dictionary. Some of the sample Keras code demonstrated how to pull out part of the `history` dictionary. Create a plot of training accuracy as a function of epoch (with axes properly labeled). (You can also use this plot to help determine the number of epochs needed as part of Q1 instead of using `verbose=True`.)

Q3 (3 points). Explore what happens when you (a) set the batch size equal to 1 (so that weights get updated after every training pattern), and (b) set the batch size equal to the total number of training patterns (so that weights get updated once per epoch after all of the training patterns have been shown). Train a network with each method using the number of epochs you used in Q1.

In a markdown cell, describe what is happening and try to explain why it is happening (compared to the training in Q1).

Q4 (4 points). `Homework5.ipynb` provides code that generates an array of test patterns and provides a `plottest()` function that displays a shaded contour plot of network predictions on these test patterns (these were discussed in class). All you need to do is apply these test patterns to the trained network from Q1 and plot the results using the `plottest()` function.

Note that you will want to make sure that you are using the network trained in Q1 (using a batch size of 20), not the networks trained in Q3. You can ensure that you save each of the individual trained networks by recognizing that you do not need to name all the networks `network`; the initialized network object returned when you call `models.Sequential()` can be called any valid variable name (the do not need to be called `network`).

Hint: Don't overthink this – answering Q4 can be done in a couple of lines of code.

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.