ECE 3502 / CS 3501: Foundations of Data Analysis **Due Fr 5/10**

Final: Neural Networks

Instructions: Submit a single Jupyter notebook (.ipynb) of your work **and the saved HTML of your formatted notebook after running it** to Collab by 11:59pm on the due date. All code should be written in Python. **Be sure to show all the work involved in deriving your answers! If you just give a final answer without explanation, you may not receive credit for that question.**

You may discuss the concepts with your classmates, but write up the answers entirely on your own. Do not look at another student's answers, do not use answers from the internet, and do not show your answers to anyone.

This is due on the last day of finals. There will be no late days and no extensions!

Overview: In this final project, you will implement the two layer neural network that we covered in class. Your network should take a vector input $x \in \mathbb{R}^d$, have one hidden layer (fully connected) to $z \in \mathbb{R}^m$, and then a logistic regression output layer to produce an output $\hat{y} = p(y = 1 \mid x)$. Note: the output layer is what you already implemented in HW 5. Reuse that code!

1. Train your logistic regression model to classify betwen the MNIST digits '3' and '5'. You will need to do the same preprocessing steps that you did in HW 5 (for example, scale the data by 1/255). Run your trained model on the test data. What classification accuracy do you get? (We are going to use this as a baseline and compare this result to your two-layer neural network.)

2. Implement a function that takes an input data matrix `X` of size $n \times d$ and feeds it forward through your network to generate a vector of n outputs. Use the logistic function σ as your activation function in the hidden layer.

   Your function will need to take two sets of weights: `w1` for the first layer will be a $d \times m$ matrix, `w2` for the second layer will be a m vector (equivalent to what we called β in the logistic regression).

3. Implement a function to compute the graidents of your weights `w1` and `w2` using backpropagation.

4. Train your network for multiple sizes of the hidden units, $m = 2, 10, 40$, on the MNIST digits '3' and '5'. What is the effect of the number of hidden units on the running time of an iteration of your training? Plot your loss function as a function of training iteration (as you did in HW 5). What is the effect of m on the convergence rate? Report the testing accuracy of these three models. How is accuracy effected by m? How did your two-layer models compare to the result you got from logistic regression?

5. Pick a value for m. You may use one of the values from the previous part, or if you are feeling adventurous, push m even higher. Now change the activation function in your hidden layer. You may use the rectified linear unit (ReLU) we discussed in class, or if you are feeling bold, use a different one such as leaky ReLU or exponential linear unit (ELU). How does the activation function effect your runtime, convergence rate, and your accuracy?