ECE 3502 / CS 3501: Foundations of Data Analysis **Due Tu 4/30**

Homework 5: Logistic Regression

Instructions: Submit a single Jupyter notebook (.ipynb) of your work 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.

1. Implement a function to fit a logistic regression using gradient ascent to maximize the likelihood of the parameter β. Your implementation should take an $n \times d$ data matrix X and corresponding binary array y of n labels. As usual, you should implement this yourself, not using a library that does gradient ascent or logistic regression for you.

   Use your function to fit a logistic regression to the OASIS hippocampus data from HW 1, with $d = 2$ features (left/right hippocampus volume) for x, and the `Dementia` diagnosis as the y labels. Use the same training and testing data split as before. Do the following:

   (a) Scale all of your x data so that both features are in the range $[0, 1]$. Put these into a data matrix X adding a column of 1's for the intercept. Randomly initialize your β estimate.

   (b) You will have to **tune** the gradient ascent step size, δ. Pick an initial δ. Run your algorithm for a few iterations, watching the value of the log-likelihood. If it ever goes down, your step size is too big! Decrease it. If it goes up too slowly, your step size is too small! Increase it. Eventually you should settle on a step size that works well, and the algorithm will converge to gradient close to zero.

   (c) Your function should save the log-likelihood values each iteration. Plot the log-likelihood (vertical axis) as a function of the iteration number (horizontal axis). This plot should be increasing and flatten out if your algorithm converges correctly.

   (d) Plot a 2D scatterplot of the x training data points, with two different colors for healthy and dementia subjects. Then, draw a line corresponding to your classifier prediction $p(y = 1 \mid x) = 0.5$. (This is your classifier's estimated separating line between the two classes that we discussed in class.)

   (e) Use your estimated logistic regression model to predict the `Dementia` diagnosis from the testing data x features. What is your final accuracy? How does it compare to the accuracy you got in HW 1 with naïeve Bayes?

2. In this part you will run your logistic regression model on a higher-dimensional example. Use the MNIST data from the HW2. These are grayscale images of hand-written digits (0-9). Your goal is to build a classifier that can recognize the difference between '0' and '1' images. The images are 28×28, for a total dimension of $d = 784$ (number of pixels).

   (a) Flatten out the images from 28×28 arrays into 784 vectors. Scale each pixel value by $1/255$ so that they will be in the range $[0, 1]$. Put these into a data matrix X adding a column of 1's for the intercept. Randomly initialize your β estimate.

(b) Run your logistic regression fit using just the data with '0' and '1' labels. Repeat the same process as before to find a step size that gets your gradient ascent to converge.

(c) Make the same plot of your log-likelihood function vs. the iteration number.

(d) Use your estimated logistic regression model to predict the the labels of the '0' and '1' test data. Again, report your accuracy.

(e) From your results, randomly pick ten '0' and ten '1' examples that were classified correctly. Display these as a grid of images. Do the same for ten '0' and ten '1' examples that were classified incorrectly. Do the mistakes that your classifier made seem reasonable (in other words, do you think these cases were more difficult)?