CS/ECE 3501: Foundations of Data Analysis **Due Tu 4/22**

Homework 4: PCA & 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.

*****Note that you have an extended deadline till April 22nd for this homework assignment. No further extension will be granted.*****

1. (55%) In this problem, we will use **Principal Component Analysis** to find the most relevant dimensions of variance in a set of hand shapes. Load the matrix `all-hands.dat`. Each row of this matrix is an entire set of hand points as a list of x, y coordinates: $(x_1, y_1, x_2, y_2, \ldots, x_{72}, y_{72})$. Do the following:

   (a) Compute the mean hand (this should be a $72 \times 2 = 144$ vector, consisting of the means of each column in your matrix). Plot this mean as a hand shape.

   (b) Compute the covariance matrix Σ for this data. **Use the formulas we covered in class, not a covariance function!** What is the total variance of the data?

   (c) What is the covariance between the x_1 coordinate and the x_2 coordinate? What is the correlation between these two coordinates. These are adjacent points on the hand, can you explain why the correlation comes out to this value?

   (d) Compute the PCA of the hands. (Use Python function to get eigenvalues and eigenvectors of Σ).

   (e) Plot a scree plot of the eigenvalues. How many eigenvalues are nonzero? What does this tell you about the dimensionality of your data?

   (f) Plot a sequence (as a strip of 5 side-by-side figures) of hand shapes along the first principal component at $s = -3\sqrt{\lambda_1}, -1.5\sqrt{\lambda_1}, 0, 1.5\sqrt{\lambda_1}, 3\sqrt{\lambda_1}$, where λ_1 is the first eigenvalue. So, you will plot hands corresponding to:

   $$\mu + s e_1,$$

   where e_1 is the first eigenvector. What does this dimension in the data correspond to, in terms of hand shape changes? Repeat this process for the second and third principal component.

   (g) How many dimensions do you need to represent 95% of the variance in the hand data?

   (h) Using your PCA results with the reduced number of dimensions you found in the previous answer, project the first hand (row 1 of the matrix) onto this reduced dimensional subspace. What is the vector of weights needed to represent this hand? Plot the reconstructed hand shape on top of the original hand shape (again, use two different colors). Is the reconstructed hand similar to the original?

2. (45%) Implement a function to fit a logistic regression using gradient descent to minimize the negative log 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 descent or logistic regression for you.

   Use your function to fit a logistic regression to the OASIS hippocampus data from HW1, 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 step size, δ. Pick an initial δ. Run your algorithm for a few iterations, watching the value of the negative log-likelihood. If it ever goes up, your step size may be too big. Decrease it. If it decreases 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 negative log-likelihood values each iteration. Plot the negative log-likelihood (vertical axis) as a function of the iteration number (horizontal axis). This plot should be decreasing 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? Note that you should achieve higher than 60% accuracy. How does it compare to the accuracy you got in HW1 with naïeve Bayes?

3. (Bonus - 20%) 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. Your goal is to build a classifier that can recognize the difference between digits '3' and '5' images. The images are 28×28, for a total dimension of $d = 784$ (number of pixels).

   (a) Reshape the images from 28×28 arrays into 784 vectors. Scale each pixel value to the range $[0, 1]$. Put your training data (with '3' and '5' labels) 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 on your training data. Note that you should achieve higher than 90% accuracy. 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 '3' and '5' testing data. Again, report your accuracy.

   (e) From your results, randomly pick five '3' and five '5' examples that were classified correctly. Display these as images. Do the same for five '3' and ten '5' 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)?