Due date: Nov. 9, 2018, 11:59 PM (Arlington time). You have **two** late days — use it at as you wish. Once you run out of this quota, the penalty for late submission will be applied. You can either use your late days quota (or let the penalty be applied). **Clearly indicate** in your submission if you seek to use the quota.

What to turn in:

1. Your submission should include your complete code base in an archive file (`zip, tar.gz`) and `q1/, q2/,` and so on), and a very very clear README describing how to run it.
2. A brief report (typed up, submit as a PDF file, NO handwritten scanned copies) describing what you solved and implemented and known failure cases. The report is **important** since we will be evaluating the grades mostly based on the report.
3. Submit your entire code and report to Blackboard.

Notes from instructor:

- Start early!
- You may ask the TA or instructor for suggestions, and discuss the problem with others (minimally). But **all parts of the submitted code must be your own**.
- Use Matlab or Python for your implementation.
- Make sure that the TA can easily run the code by plugging in our test data.

Problem 1

(Logistic regression, **40pts**) Implement a perceptron for logistic regression. For your training data, generate 2000 training instances in two sets of random data points (1000 in each) from multi-variate normal distribution with

$$\mu_1 = [1, 0],\ \mu_2 = [0, 1.5],\ \Sigma_1 = \begin{bmatrix} 1 & 0.75 \\ 0.75 & 1 \end{bmatrix},\ \Sigma_2 = \begin{bmatrix} 1 & 0.75 \\ 0.75 & 1 \end{bmatrix} \tag{1}$$

and label them 0 and 1. Generate testing data in the same manner but include 500 instances for each class, i.e., 1000 in total. Use sigmoid function for your activation function and cross entropy for your objective function. You will implement a logistic regression for the following questions. Initialize the starting weight as $w = [1, 1, 1]$. During training, stop your loop when the objective function (i.e., cross entropy) does not decrease any more (below certain threshold) or when the gradient is close to 0 or the iteration reaches 10000. Set your thresholds properly so that the iteration doesn't reach 10000 for all the learning rate that you will be using.

1. Perform batch training using gradient descent. Divide the derivative with the total number of training dataset as you go through iteration (it is very likely that you will get NaN if you don't do this.). Set your learning rate to be $\eta = \{1, 0.1, 0.01\}$. How many iterations did you go through the training dataset? What is the accuracy that you have? What are the edge weights that were learned?
2. Perform online training using gradient descent. Set your learning rate to be $\eta = \{1, 0.1, 0.01\}$. Set your maximum number of iterations to 10000. How many iterations did you go through your training dataset? What is the accuracy that you have? What are the edge weights that were learned? Compare the learned parameters and accuracy to the ones that you got from batch training. Are they the same? Explain in your report.

3. Write your own code to draw ROC curve and computing area under the curve(AUC). Evaluate the performance of your LR classifier with batch training with $\eta = \{1, 0.1, 0.01\}$.