Deadline is: Thursday, February 13, 2020 by 4:30pm. Submit a hardcopy directly to me (do not email to me). Also, the assignment can be submitted by groups of no more than 4 students.

Question #1 In this question you are to create some simulated data sets and then use the Adaline neuron and the Sigmoid to perform some prediction. Use whatever programming language you want to use.

Generate 5000 synthetic data points (x, y) as follows:

a. Using the `rnorm()` function in R (or equivalent in Matlab or Python or etc), create a vector, $\boldsymbol{x}$, containing 5000 observations drawn from a Gaussian distribution $N(0, 1)$ [ie, a normal distribution with mean 0 and variance 1]. This vector $\boldsymbol{x}$ represents your set of inputs x.

b. Using the `rnorm()` function in R (or equivalent in Matlab or Python or etc), create a vector, ***eps***, containing 5000 observation drawn from a $N(0, 0.25)$ distribution; ie, a normal distribution with mean 0 and variance 0.25.

c. Using vectors $\boldsymbol{x}$ and ***eps***, generate a vector $\boldsymbol{y}$ according to the model

$$y = -1 + 0.5x - 2x_2 + 0.3x_3 + eps.$$

Your 5000 data-points (x, y) are generated upon completion of this Part-c. Note that the true function is a cubic function with true weight vector being $\boldsymbol{w}_{true} = (-1, +0.5, -2, +0.3)$.

d. Implement the Adaline neuron learning algorithm using (i) batch gradient descent and (ii) stochastic gradient descent, and test and compare them on linear regression over your synthetic data-points. You need not to perform a cross-validation scheme here; only use the whole data set as your training set. Though if you wish, you may perform cross-validation (LOOCV or 10-fold-cv) but without a test set. The initializations, the learning rate, and the stopping criterion are left for you to explore. Think about the reasons why you use a particular strategy.

e. Repeat Part-d, but with a Sigmoid neuron of your choice (logistic or tanh).

f. Repeat Part-d and Part-e with cross-validation method of your own choice (LOOCV or 10-fold-cv) to find the best degree d. You must first randomly create a test set of size between 20% and 30% drawn from your original full data set. If this is correctly done, your methods should not only find $d = 3$ to be the degree, but they should also find the best weight vector, $\boldsymbol{w}_{best}$, to be as close as possible to $\boldsymbol{w}_{true} = (-1, +0.5, -2, +0.3)$.

g. Use your creativity and do whatever experiments you want to test, and then tell me whatever story your experiments told you. ☺

Question #2 In this question you are to create some simulated data sets and then use the Perceptron neuron to perform some prediction.

a. Randomly create 2500 data-points (x, y)'s of **class -1** to lie one side of the function f above and 2500 data-points (x, y)'s of **class +1** to lie on the other side of the function. *Indeed, here, you are not required to create your data using the function f above; you can use any function you want, as long as it is a simple linearly separable function of your choice to be used to separate 5000 data points into two classes (I have mentioned the function above simply because you have it already).*

b. Implement the Perceptron learning algorithm and run it on your synthetic data set. Obtain the best Perceptron model via any cross-validation method of your choice. Use your creativity to tell me anything about your Perceptron: for example, how does the performance (speed and accuracy) vary when changing the size of the training set from 50% to 80%? Also, how does Perceptron compare with the Adaline and Sigmoid neurons on the same training sets? [note here that you are using Adaline and Sigmoid on a classification problem, not a regression problem].

c. Implement the Pocket algorithm (or any improved Perceptron algorithm; there are many you can find online) and run it on your synthetic data set which you have modified in such a way that it is not linearly separable anymore. Compare the Pocket with Perceptron, Adaline, and Sigmoid on the data and discuss your results.

Question #3: We now that the XOR or XNOR functions are not linearly separable functions. How would you modify the Perceptron learning algorithm in order to learn the XOR or XNOR function?