CAP 6619 Deep Learning

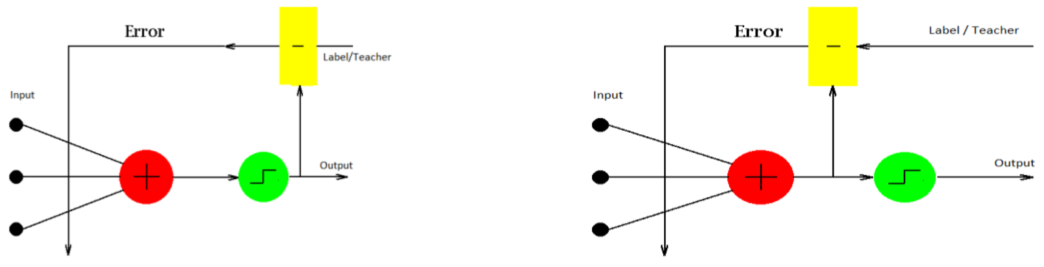
2020 Fall

Homework 2 (**12 pts, Due: Oct. 3 2020, Late Penalty: -1/day**)

[If two homework submissions are found to be similar to each other, both submissions will receive 0 grade]

[Homework solutions must be submitted through Canvas. No email submission is accepted. If you have multiple files, please include all files as one zip file, and submit zip file online (only zip, pdf, or word files are allowed). You can always update your submissions. I will only grade the latest version.]

Question 1 [0.5 pt] Figure 1 shows neuron architectures used for perceptron learning rule and gradient descent learning rule training. (1) Explain ~~why~~ which one is used for perceptron learning rule, and which one is used for gradient descent learning rule? (2) what is the minimization objective of (a) and (b), respectively? (3) what does the "Error" in (a) and (b) represent, respectively?



(a) Architecture 1 (b) Architecture 2

Figure 1: Neuron architecture

Question 2 [1 pt] Figure 2 shows a single layer neural network with three weight values (including bias). Given a training instance x(n), assume desired label of the instance is d(n), please define the squared error of the instance with respect to the network [0.5 pt]. Please use gradient descent learning to derive weight updating rules for w_0, w_1, and w_2, respectively [0.5 pt]

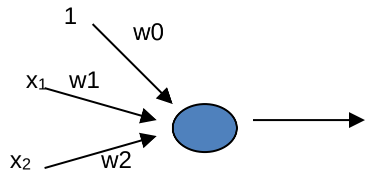


Figure 2: Single layer neural network

Question 3 [1 pt]: The following figure shows a quadratic function $y=2x^2-4x+1$. Assume we are at the point (2,1), and is searching for the next movement to find the minimum value of the quadratic function using gradient descent (the learning rate is 0.1).

- What is the gradient at point (2,1)? (Show your solutions) [0.5 pt]
- Following gradient decent principle, find the next movement towards the global minimum [0.5 pt]

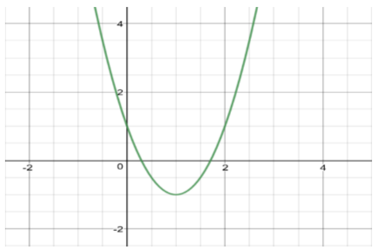


Figure 3

Question 4 [2 pts] Assuming we have two sets of instances, which belong to two classes, with each class containing three instances. C_1={(1, 0), (1, 1), (0, -1)}; C_2={(0, 1), (-1, 0), (-1, -1)}. The class label of C_1 is 1, and the class label of C_2 is 0. Assuming η=1, and the initial weights are w_0=1, w_1=-1, and w_2=1. Please use perceptron learning rule to learn a linear decision surface for these two classes. List the results in the first two rounds by using tables in the following form.

Assume the activation is defined as follow.

$$\phi(v) = \begin{cases} 1 & if\ v \geq 0.5 \\ 0 & otherwise \end{cases}$$

Input	Weight	v	Desired	Actual	Update?	New Weight
(1,1,0)						
(1,1,1)						
(1,0,-1)						
(1,0,1)						
(1,-1,0)						
(1,-1,-1)						

The Second Pass

Input	Weight	v	Desired	Actual	Update?	New Weight
(1,1,0)						
(1,1,1)						
(1,0,-1)						
(1,0,1)						
(1,-1,0)						
(1,-1,-1)						

Question 5 [2 pts] Assuming we have two sets of instances, which belong to two classes, with each class containing three instances. C1={(1, 0), (1, 1), (0, -1)}; C2={(0, 1), (-1, 0), (-1, -1)}. Assuming η=0.1, and the initial weights are w_0=1, w_1=1, and w_2=1. Please use gradient learning rule to learn a linear decision surface for these two classes. List the results in the first two rounds by using tables in the following form (Please report the mean squared errors of all instances with respect to the initial weight values, and also report the mean squared errors E(W) AFTER the weight updating for each round).

Mean squared errors E(W) before the first-round weight updating:

First Round

Input	Weight	v	Desired	Output	Δw
(1,1,0)					
(1,-1,0)					
(1,0,-1)					
(1,0,1)					
(1,1,1)					
(1,-1,-1)					

New weight after first round:

Mean squared errors E(W) after the first-round weight updating:

Second Round

Input	Weight	v	Desired	Output	Δw
(1,1,0)					
(1,-1,0)					
(1,0,-1)					
(1,0,1)					
(1,1,1)					
(1,-1,-1)					

New weight after second round:

Mean squared errors E(W) after the second-round weight updating:

Question 6 [2 pts] Assuming we have two sets of instances, which belong to two classes, with each class containing three instances. C1={(1, 0), (1, 1), (0, -1)}; C2={(0, 1), (-1, 0), (-1, -1)}. Assuming η=0.1, and the initial weights are w_0=1, w_1=1, and w_2=1. Please use Delta rule (AdaLine) to learn a linear decision surface for these two classes. List the results in the first round by using tables in the following form (Please report the mean squared errors of all instances with respect to the initial weight values, and also report the mean squared errors E(W) AFTER the weight updating of the last instance).

Input	Weight	v	Desired	Output	Δw	New Weight
(1,1,0)						
(1,-1,0)						
(1,0,-1)						
(1,0,1)						
(1,1,1)						
(1,-1,-1)						

Mean squared errors E(W) after the weight updating of the last instance:

Question 7 [1 pt]

Class1Linear.txt and Class2Linear.txt files in the Canvas contain two-dimensional instances in two classes (C_1 and C_2 respectively, with 70 instances in each class). In these two files, the first column denotes the *x* values and the second column represents the *y* values (separated by comma). Please use the Perceptron learning rule [Notebook, pdf] in the Canvas to report the following results:

- Combine instances in the Class1Linear.txt and Class2Linear.txt files as one dataframe, and label instances in Class1 as 1, and instances in Class2 as 0. Report the scatter plot of all 140 instances in the same plot, using different color to show instance in different class [0.25 pt]
- Use learning rate 0.01 and report the error rates of the perceptron learning with respect to different iterations (using a plot where the x-axis denotes the iterations, and the y-axis shows the error rate. [0.25 pt]
- Report final perceptron weight values and the slope and y-intercept of the decision surface [0.25 pt]
- Report the final decision surface on the same scatter plot which shows the 140 instances [0.25 pt]

Question 8 [2.5 pts]

Class1.txt and Class2.txt files in the Canvas contain two-dimensional instances in two classes (C_1 and C_2 respectively, with 100 instances in each class). The instances are not linearly separable. Revise code in the Gradient Descent Learning Notebook to implement following tasks.

- Randomly select 80% instances from class1.txt and 80% instances from class2.txt to train a perceptron classifier (using gradient descent learning rule), and use the classifier to classify remaining 20% instances in class1.txt and class2.txt. Please report the classification accuracy of the perceptron classifier on the 20% test instances (using learning rate=1/(# of training samples), error threshold 0.01, and iteration numbers 2,000) [0.5 pt]
- Please report the training errors and test errors of the perceptron classifier with respect to each iteration. Please show the two error rates on the same chart, where the x-axis denotes the iteration and the y-axis denotes the mean classification errors [0.5 pt]
- Report the final decision surface on the same scatter plot which shows the 200 instances [0.5 pt]
- Revise Delta learning rule in the Gradient Descent Learning Notebook to learn from the same 80% of training samples (using learning rate =0.01, error threshold 0.01, and iteration numbers 2,000).
    - Report the training errors as a plot, where the x-axis denotes the iteration and the y-axis denotes the mean classification errors [0.5 pt]
- Compare Delta learning rule and Gradient Descent Learning rule, explain advantage and disadvantage of each of them, respectively [0.5 pt]