Assignment 4 Introduction to Deep Learning – CAP 4613

Due: Sunday, March 20

Instructions [Failure to follow instructions will be penalized at least by 1 point]

- All the solutions must be based on the methods taught in class. Otherwise, it will not be graded. This is an important note to the students who do not study the lectures and learn from other online resources.

- The Colab link must be shared.

- Make sure that it is possible to copy the Colab link from your pdf.

- Make sure that your pdf is not cutting the text.

- Provide the full answers in your pdf file with all the outputs of your code.

- Only one file must be submitted electronically.

- Combine the handwritten and Python parts before submission.

- The paper-based problems must be with no coding/python etc.

- The paper-based problems/parts can be typed, but if handwritten, it must be clean and legible.

- Show your detailed work to each problem.

Problem 1) [Paper-based] Gradient descent learning: Consider the following set of data points:

input		desired
x_1	x_2	label
1	1	1
1	0	1
0	1	0
-1	-1	0
-1	0	0
-1	1	0

As the above table shows, the data points are categorized (labeled) in two groups specified by the labels "1" and "0".

a) Use the gradient descent learning algorithm to train a single neuron on the data samples given in the table above.
<u>Repeat the gradient descent algorithm for only 3 epochs.</u>
Assume the learning rate is 0.1 with initial weight vector of (1 1 1).

a) Calculate the loss function (i.e., average square error) at each epoch. Plot the learning curve. Is the neuron learning? Explain your answer.
<u>Repeat these tasks using the classifier derived in the last epoch.</u>

b) Draw the schematics of the trained neuron.

c) Provide the equation of the trained model.

d) Plot the given data points with two different markers for each group.
e) Plot the classifier line in figure (d). Label your plots.
f) Provide the confusion matrix. Calculate the accuracy, sensitivity, and specificity.

Problem 2) [Python] Gradient descent learning in Python:

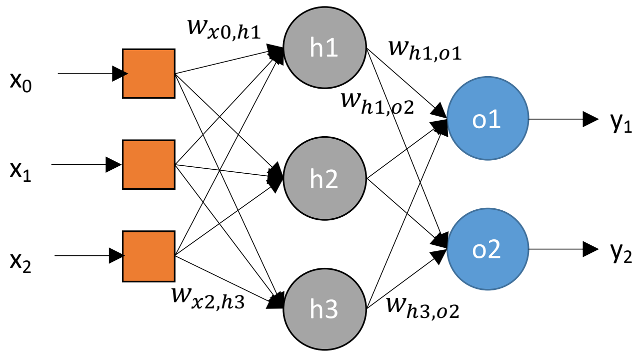
a) Create class NeuralNetwork(): that creates a single neuron with a linear activation, train it using gradient descent learning. This class should have the following function:
   i. def __init__(self, learning_r): that initializes a 3x1 weight vector randomly and initializes the learning rate to learning_r. Also, it creates a history variable that saves the weights and the training cost after each epoch (i.e., iteration).
   ii. def forward_propagation(self, inputs): that performs the forward propagation by multiplying the inputs by the neuron weights and then generating the output.
   iii. def train(self, inputs_train, labels_train, num_train_epochs=10): that performs the gradient descent learning rule for num_train_epochs times using the inputs and labels. This function also saves the weights and costs at every epoch to the history variable.
b) Use the gradient descent rule to train a single neuron on the datapoints given below:
   i. Create an np array of a shape 10x2 that contains the inputs, and another array with a shape 10x1 that contains the labels.
   ii. Add the bias to the inputs array to have a 10x3 shape.
   iii. Create the network with one neuron using the class NeuralNetwork() with learning rate of 1 then train it using train(inputs, labels, 50) function.

input		desired
x_1	x_2	label
1	1	1
1	0	1
0	1	-1
-1	-1	-1
0.5	3	1
0.7	2	1
-1	0	-1
-1	1	-1
2	0	1
-2	-1	-1

c) Plot the given data points with two different markers for each group.
d) Use the trained weights at each epoch and plot the classifier lines of every 5 epochs (i.e., 1,5,10, …, 50) to the plot in (c).
e) Use the trained weights and plot the final classifier line to the plot in (c).

f) Plot the training cost (i.e., the learning curve) for all the epochs.
g) Repeat step (b) with the learning rates of 0.5 and 0.05. Create a subplot with the learning curves of learning rates 1, 0.5, and 0.05. Add titles.
h) What behavior do you observe from the learning curves with the different learning rates? Explain your observations. Which learning rate is more suitable? Explain.

Problem 3) [Paper-based] Forward-backward propagation: Consider the following forward feed neural network:



Initial weights are given as below. Learn rate is 0.5. The activation function is sigmoid function with $\alpha = 1$ for all the neurons.

$$w_{x_0,h_1} = 0.18, \quad w_{x_1,h_1} = 0.32, \quad w_{x_2,h_1} = 0.42$$

$$w_{x_0,h_2} = 0.51, \quad w_{x_1,h_2} = 0.64, \quad w_{x_2,h_2} = 0.12$$

$$w_{x_0,h_3} = 0.43, \quad w_{x_1,h_3} = 0.72, \quad w_{x_2,h_3} = 0.33$$

$$w_{h_0,O_1} = 0.53, \quad w_{h_1,O_1} = 0.22, \quad w_{h_2,O_1} = 0.19, w_{h_3,O_1} = 0.61$$

$$w_{h_0,O_2} = 0.61, \quad w_{h_1,O_2} = 0.38, \quad w_{h_2,O_2} = 0.21, w_{h_3,O_2} = 0.15$$

For a training, the inputs of the features (x_1=1, x_2=0) and the label of class y_1, perform the forward-backward steps as instructed below.

HINT: The data sample belongs to class y_1. This means that for this data sample, y_1=1 and y_2=0.

<u>NOTE: If you are not typing your answer, you need to submit your clean and neat handwritten solution with all the details. Otherwise, the assignment will not be graded.</u>

a) Perform the forward propagation from the left to the right side of the network. Show the detailed calculations in each step. Report the computed outputs in the table below:

y_{h_1}	y_{h_2}	y_{h_3}	y_1	y_2

b) Compute the local gradients of each node from the right to the left side of the network. Show the detailed calculations in each step. Report the computed local gradients in the table below:

δ_{h_1}	δ_{h_2}	δ_{h_3}	δ_{O_1}	δ_{O_2}

c) <u>Only for the bias weights of each neuron</u>: compute the change in the weight and the updated weight. Show the detailed calculations in each step. Report the computed values in the table below:

$\Delta w_{x_0,h_1}$	$\Delta w_{x_0,h_2}$	$\Delta w_{x_0,h_3}$	$\Delta w_{h_0,O_1}$	$\Delta w_{h_0,O_2}$

w_{x_0,h_1}	w_{x_0,h_2}	w_{x_0,h_3}	w_{h_0,O_1}	w_{h_0,O_2}