CS 4210 – Assignment #4

Maximum Points: 100 pts.

Bronco ID: 013-789-022

Last Name: Nash

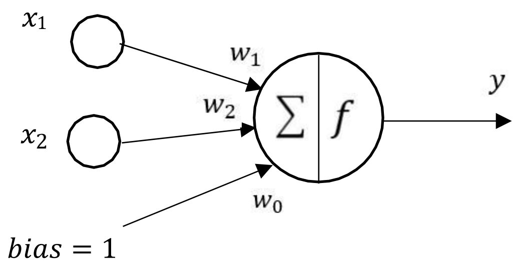
First Name: Mason

Note 1: Your submission header must have the format as shown in the above-enclosed rounded rectangle.
Note 2: Homework is to be done individually. You may discuss the homework problems with your fellow students, but you are NOT allowed to copy – either in part or in whole – anyone else's answers.
Note 3: Your deliverable should be a .pdf file submitted through Gradescope until the deadline. Do not forget to assign a page to each of your answers when making a submission. In addition, source code (.py files) should be added to an online repository (e.g., github) to be downloaded and executed later.
Note 4: All submitted materials must be legible. Figures/diagrams must have good quality.
Note 5: Please use and check the Canvas discussion for further instructions, questions, answers, and hints. The bold words/sentences provide information for a complete or accurate answer.

1. [10 points] Train the perceptron networks below to solve the logical AND problem correctly classifying the dataset instances. Use the parameters: learning rate η=0.4, initial weights = 1, and activation function = heaviside.

Heaviside: 0 if z<=0, 1 if z>0

Training formula: Deltawi = n(t-y)xi where t is target output, and y is actual output, and n is learning rate.



Datset

x_1	x_2	$x_1\ AND\ x_2$ y
0	0	0
0	1	0
1	0	0
1	1	1

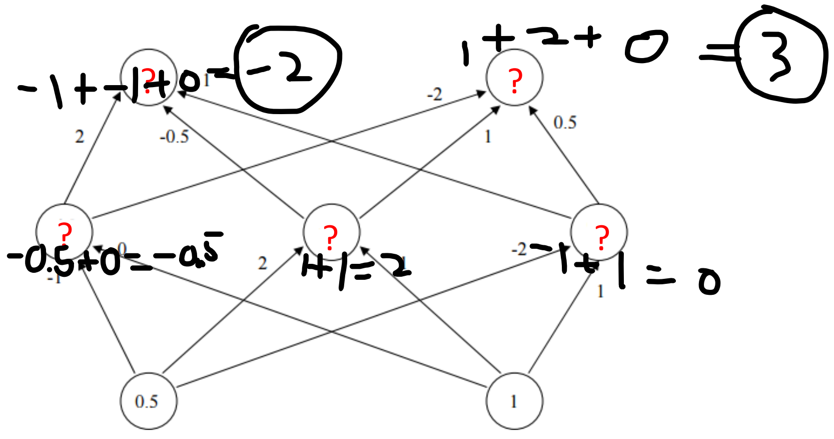
Solution format: Include the solution table (as illustrated in the lecture) with all variables and values calculated for each iteration.

	x_1	x_2	x_0	w_1	w_2	w_0	t	z(net)	y	t-y	Delta w_1	Delta w_2	Delta w_0
1	0	0	1	1	1	1	0	1	1	-1	0	0	-0.4
2	0	1	1	1	1	0.6	0	1.6	1	-1	0	-0.4	-0.4
3	1	0	1	1	0.6	0.2	0	1.2	1	-1	-0.4	0	-0.4
4	1	1	1	0.6	0.6	-0.2	1	1	1	0	0	0	0
5	0	0	1	0.6	0.6	-0.2	0	-0.2	0	0	0	0	0
6	0	1	1	0.6	0.6	-0.2	0	0.4	1	-1	0	-0.4	-0.4
7	1	0	1	0.6	0.2	-0.6	0	0	0	0	0	0	0
8	1	1	1	0.6	0.2	-0.6	1	0.2	1	0	0	0	0
1	0	0	1	0.6	0.2	-0.6	0	-0.6	0	0	0	0	0
2	0	1	1	0.6	0.2	-0.6	0	-0.4	0	0	0	0	0
3	1	0	1	0.6	0.2	-0.6	0	0	0	0	0	0	0
4	1	1	1	0.6	0.2	-0.6	1	0.2	1	0	0	0	0

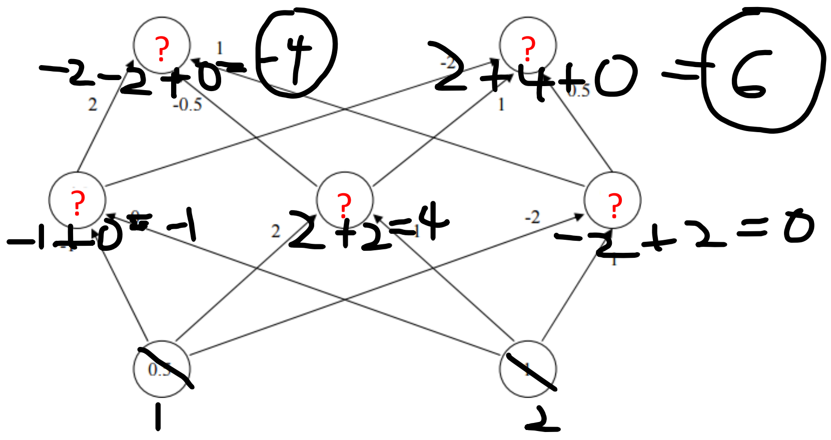
2. [15 points] Complete the Python program (perceptron.py) that will read the file optdigits.tra to build a Single Layer Perceptron and a Multi-Layer Perceptron classifiers. You will compare their performances and test which combination of two hyperparameters (learning rate and shuffle) leads you to the best prediction performance for each classifier. To test the accuracy of those distinct models, you will use the file optdigits.tes. You should update and print the accuracy of each classifier, together with the hyperparameters when it is getting higher.
   CPP-ML/perceptron.py at master · SkeltalFlamingo/CPP-ML (github.com)

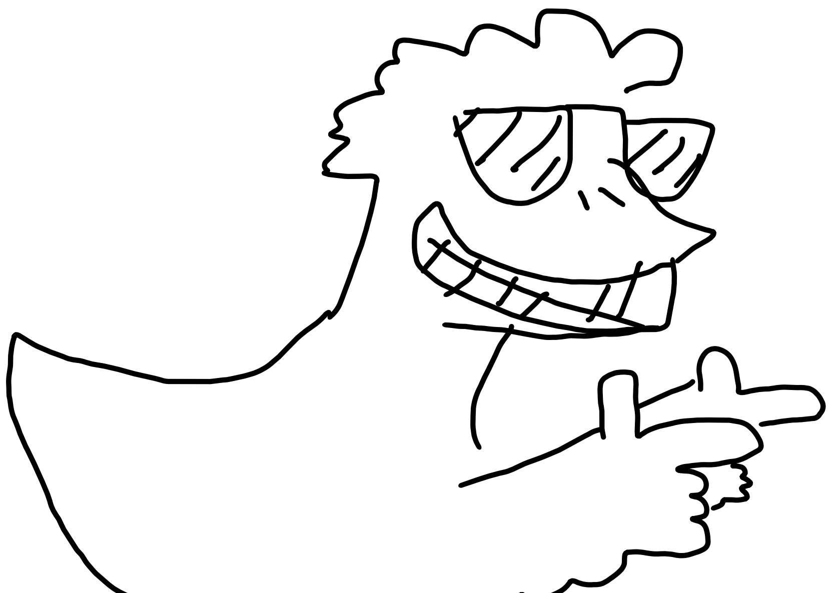
3. [15 points] The figure below is a network of linear neurons, that is the output of each neuron is identical to its input (linear activation function). The numbers at the connections indicate the weights of the links.

   a. [8 points] Find the value at the output nodes of the network for the given input (0.5,1).



   b. [5 points] Find the value at the output nodes of the network for the input (1,2). [2 points] Do you need to repeat the computations all over again? Why?
   No. Since it's a network of linear neurons, and this new input is just double the previous, the output should also be double the previous output. The new outputs should be -4, 6. I'm going to redo it to test just to be safe though.





4. [15 points] Deep Learning. Complete the Python program (deep_learning.py) that will learn how to classify fashion items. You will use the dataset Fashion MNIST, which includes 70,000 grayscale images of 28×28 pixels each, with 10 classes, each class representing a fashion item as illustrated below. You will use Keras to load the dataset which includes 60,000 images for training and 10,000 for test. Your goal is to train and test multiple deep neural networks and check their corresponding performances, always updating the highest accuracy found. This time you will use a separate function named build_model() to define the architectures of your neural networks. Finally, the weights of the best model will be printed, together with the architecture and the learning curves. To install TensorFlow use: python -m pip install --upgrade tensorflow.

https://github.com/SkeltalFlamingo/CPP-ML/blob/master/Assignment%204/deep_learning.py

5. [15 points] Considering the 1-D dataset below and the following bootstrap samples (bagging rounds 1 to 5) randomly generated during the bagging process. Show how a bagging algorithm can perfectly classify this data by **drawing** and **writing** the decision stumps for each round, the summary table of the trained decision stumps, and the combination table of your base classifiers with the final predictions. Hint: you might need to test alternative but equally accurate decision stumps on your training set to get maximum accuracy.

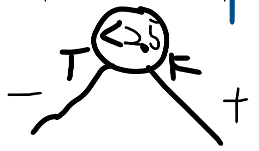
Dataset

x	1	2	3	4	5
y	-1	-1	1	1	-1

Round 1

x	1	1	2	4	5
Y	-1	-1	-1	1	-1

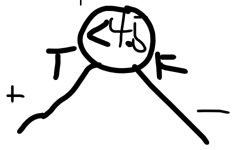
Decision Boundary at 2.5
Left negative, Right positive (80%)



Round 2

x	3	3	4	4	5
y	1	1	1	1	-1

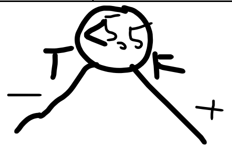
Decision Boundary at 4.5
Left positive, Right negative (80%)



Round 3

x	1	2	2	5	5
Y	-1	-1	-1	-1	-1

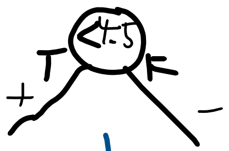
Decision Boundary at 5.5
Left negative, right positive (100%)



Round 4

x	1	3	4	4	5
Y	-1	1	1	1	-1

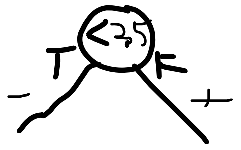
Decision Boundary at 4.5
Left positive, right negative (80%)



Round 5

x	1	2	3	3	4
Y	-1	-1	1	1	1

Decision Boundary at 2.5
Left negative, Right positive (100%)



x	1	2	3	4	5
R1	-1	-1	1	1	1
R2	1	1	1	1	-1
R3	-1	-1	-1	-1	-1
R4	1	1	1	1	-1
R5	-1	-1	1	1	1
Sum	-1	-1	3	3	-2
Sign	-1	-1	1	1	-1
y	-1	-1	1	1	-1

6. [15 points] Considering the different 1-D dataset below and the following rounds from 1 to 3 randomly generated during the boosting process. Show how a boosting algorithm can perfectly classify this data by **drawing** and **writing** the decision stumps and weights for each round, the summary table of the trained decision stumps, and the combination table of your base classifiers with the weighted final predictions. Hint: there is a single best decision stump (more accurate) for each round.

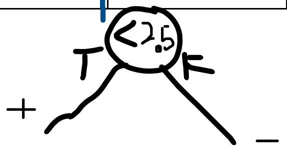
Dataset

x	1	2	3	4	5
y	1	1	-1	-1	1

Round 1

x	1	2	3	4	4
y	1	1	-1	-1	-1

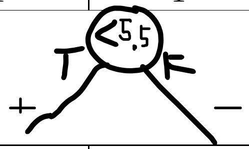
Decision Boundary at 2.5
Left positive, Right negative



Round 2

x	5	5	5	5	5
y	1	1	1	1	1

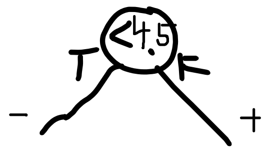
Decision Boundary at 5.5
Left positive, Right negative



Round 3

x	3	3	4	4	5
y	-1	-1	-1	-1	1

Decision Boundary at 4.5
Left negative, Right positive



Weight table

X	X = 1	X = 2	X = 3	X = 4	X = 5
Round 1 weights	0.2	0.2	0.2	0.2	0.2
Round 2 weights	0.0357	0.0357	0.0357	0.0357	0.8572
Round 3 weights	0.0061	0.0061	0.4206	0.4206	0.1466

Summary table

Round	Split Point	Left Class	Right Class	alpha
1	2.5	+	-	1.589
2	5.5	+	-	2.1165
3	4.5	-	+	3.0146

Classification Table

Round	X = 1	X= 2	X = 3	X = 4	X = 5
1	1	1	-1	-1	-1
2	1	1	1	1	1
3	-1	-1	-1	-1	1
Sum	0.6909	0.6909	-2.4871	-2.4871	3.5421
sign	1	1	-1	-1	1
y	1	1	-1	-1	1

Calculations for weights and alphas

Round 1:

e1 = (1/5)[(.2*0)+(.2*0)+(.2*0)+(.2*0)+(.2*1)] = 0.04

a1 = (1/2)ln[(1-0.04)/0.04] = 1.589

weights

w12 = w22 = w32 = w42 = 0.2/z1 *e ^(-1.589) = 0.0408/1.143 = 0.0357

w52 = 0.2/z1 * e ^(1.589) = .9798/1.143 = 0.8572

Round 2:

e2 = (1/5) [(0.0357*1) + (0.0357*1)] = 0.0143

a2 = (1/2)ln[(1-0.0143)/0.0143] = 2.1165

weights

w13 = w23 = 0.0357 / z2 * e^(-2.1165) = 0.0043/0.7045 = 0.0061

w33 = w43 = 0.0357 / z2 * e^(2.1165) = 0.2963/0.7045 = 0.4206

w53 = 0.8572 / z2 * e^(-2.1165) = 0.1033/0.7045 = 0.1466

Round 3:

e2 = (1/5) [(0.0061*1) + (0.0061*1)] = 0.0024

a2 = (1/2)ln[(1-0.0024)/ 0.0024] = 3.0146

weights

N/A

7. [15 points] Complete the Python program (bagging_random_forest.py) that will read the file optdigits.tra (3,823 samples) that includes training instances of handwritten digits (optically recognized). Your goal is to build a base classifier by using a single decision tree, an ensemble classifier that combines multiple decision trees, and a Random Forest classifier to recognize those digits. To test the accuracy of those distinct models, you will use the file optdigits.tes (1,797 samples).

https://github.com/SkeltalFlamingo/CPP-ML/blob/master/Assignment%204/bagging_random_forest.py
I'm not sure if this is the link to an outdated commit. You can find the most recent version of this code in the Assignment 4 folder in this repository.

Important Note: Answers to all questions should be written clearly, concisely, and unmistakably delineated. You may resubmit multiple times until the deadline (the last submission will be considered).

NO LATE ASSIGNMENTS WILL BE ACCEPTED. ALWAYS SUBMIT WHATEVER YOU HAVE COMPLETED FOR PARTIAL CREDIT BEFORE THE DEADLINE!