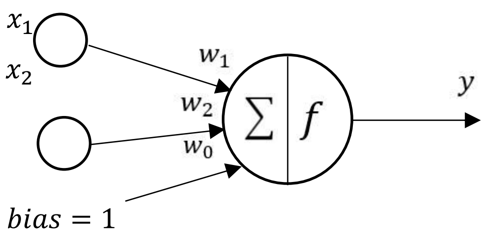
Bronco ID: 014429779
Last Name: KOEPKE
First Name: CHRISTOPHER

CS 4210.01 ASSIGNMENT #4

1. Train the perceptron networks below to solve the following logical problems:
   a. logical AND problem correctly classifying the dataset instances. Use the parameters: learning rate η=0.4, initial weights = 1, and activation function = Heaviside.



$$Heaviside(z) = \begin{cases} 0\ if\ z \le 0 \\ 1\ if\ z > 0 \end{cases}$$

Dataset

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

- FIRST INTERATION:
  - $(x_1 = 0,\ x_2 = 0,\ t = 0,\ x_0 = 1,\ w_0 = w_1 = w_2 = 1\ )$
    $z = (1 * 0) + (1 * 0) + (1 * 1) = 1$

  Since $y = 1$, $t = 0$, and $\Delta w_i = \eta(t - y)x_i$ we have:
  - $\Delta w_0 = (0.4)(0 - 1)(1) = -0.4,\ \Delta w_1 = (0.4)(0 - 1)(0) = 0,\ \Delta w_2 = (0.4)(0 - 1)(0) = 0$

  Since $w_i = w_i + \Delta w_i$, we have:
  - $w_0 = 1 - 0.4 = 0.6,\ w_1 = 1 - 0 = 1,\ w_2 = 1 - 0 = 1$

- SECOND INTERATION:
  - $(x_1 = 0,\ x_2 = 1,\ t = 0,\ x_0 = 1,\ w_0 = 0.6,\ w_1 = w_2 = 1)$
    $z = (1 * 0) + (1 * 1) + (0.6 * 1) = 1.6$

  Since $y = 1$, $t = 0$, and $\Delta w_i = \eta(t - y)x_i$ we have:
  - $\Delta w_0 = (0.4)(0 - 1)(1) = -0.4,\ \Delta w_1 = (0.4)(0 - 1)(0) = 0,\ \Delta w_2 = (0.4)(0 - 1)(1) = -0.4$

  Since $w_i = w_i + \Delta w_i$, we have:
  - $w_0 = 0.6 - 0.4 = 0.2,\ w_1 = 1 - 0 = 1,\ w_2 = 1 - 0.4 = 0.6$

- THIRD INTERATION:
  - $(x_1 = 1, x_2 = 0, t = 0,\ x_0 = 1,\ w_0 = 0.2,\ w_1 = 1,\ w_2 = 0.6)$
    $z = (1 * 1) + (0.6 * 0) + (0.2 * 1) = 1.2$

  Since $y = 1$, $t = 0$, and $\Delta w_i = \eta(t - y)x_i$ we have:
  - $\Delta w_0 = (0.4)(0 - 1)(1) = -0.4,\ \Delta w_1 = (0.4)(0 - 1)(1) = -0.4,\ \Delta w_2 = (0.4)(0 - 1)(0) = 0$

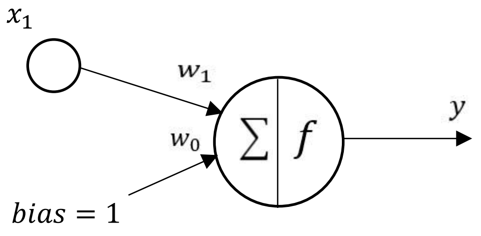
  Since $w_i = w_i + \Delta w_i$, we have:
  - $w_0 = 0.2 - 0.4 = -0.2,\ w_1 = 1 - 0.4 = 0.6,\ w_2 = 0.6 - 0 = 0.6$

- FOURTH INTERATION:
  - $(x_1 = 1, x_2 = 1, t = 1,\ x_0 = 1,\ w_0 = -0.2,\ w_1 = w_2 = 0.6)$
    $z = (0.6 * 1) + (0.6 * 1) + (-0.2 * 1) = 1$

  Since $y = 1$ and $t = 1$, there would be no change in weights because $(t - y) = 0$
  Since $w_i = w_i + 0$, we have:
  - $w_0 = -0.2 + 0 = -0.2,\ w_1 = 0.6 + 0 = 0.6,\ w_2 = 0.6 + 0 = 0.6$ C

	x_1	x_2	x_0	w_1	w_2	w_0	t	$z(net)$	y	$t-y$	Δw_1	Δw_2	Δw_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
9	0	0	1	0.6	0.2	-0.6	0	-0.6	0	0	0	0	0
10	0	1	1	0.6	0.2	-0.6	0	-0.4	0	0	0	0	0
11	1	0	1	0.6	0.2	-0.6	0	0	0	0	0	0	0
12	1	1	1	0.6	0.2	-0.6	1	0.2	1	0	0	0	0

b. logical NOT problem correctly classifying the dataset instances. Use the parameters: learning rate ɳ=0.1, initial weights = 0, and activation function = Heaviside.



Dataset

x_1	$NOT\ x_1$
0	1
1	0

- FIRST INTERATION:
  - $(x_1 = 0,\ t = 1,\ x_0 = 1,\ w_0 = w_1 = 0)$
    $z = (0 * 0) + (0 * 1) = 0$

  Since $y = 0$, $t = 1$, and $\Delta w_i = ɳ(t - y)x_i$ we have:
  - $\Delta w_0 = (0.1)(1-0)(1) = 0.1,\ \Delta w_1 = (0.1)(1-0)(0) = 0$

  Since $w_i = w_i + \Delta w_i$, we have:
  - $w_0 = 0 + 0.1 = 0.1,\ w_1 = 0 + 0 = 0$

- SECOND INTERATION:
  - $(x_1 = 1,\ t = 0,\ x_0 = 1,\ w_0 = 0.1,\ w_1 = 0)$
    $z = (0 * 1) + (0.1 * 1) = 0.1$

  Since $y = 1$, $t = 0$, and $\Delta w_i = ɳ(t - y)x_i$ we have:
  - $\Delta w_0 = (0.1)(0-1)(1) = -0.1,\ \Delta w_1 = (0.1)(0-1)(1) = -0.1$

  Since $w_i = w_i + \Delta w_i$, we have:
  - $w_0 = 0.1 - 0.1 = 0,\ w_1 = 0 - 0.1 = -0.1$

	x_1	x_0	w_1	w_0	t	$z(net)$	y	$t-y$	Δw_1	Δw_0
1	0	1	0	0	1	0	0	1	0	0.1
2	1	1	0	0.1	0	0.1	1	-1	-0.1	-0.1
3	0	1	-0.1	0	1	0	0	1	0	0.1
4	1	1	-0.1	0.1	0	0	0	0	0	0
5	0	1	-0.1	0.1	1	0.1	1	0	0	0
6	1	1	-0.1	0.1	0	0	0	0	0	0

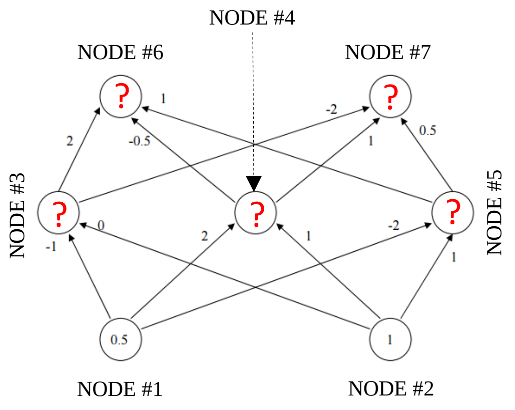
2. Complete the Python program (perceptron.py) that will read the file optdigits.tra to build a Single Layer Perceptron and a Multi-Layer Perceptron classifier. You will simulate a grid search, trying to find which combination of two hyperparameters (learning rate and shuffle) leads you to the best prediction 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.

   https://github.com/chris-k87/CS_4210.01/tree/main/Assignment_4/Perceptron

3. 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. Find the value at the output nodes of the network for the given input (0.5,1).

   - $Input\ Node\ \#3 = (0.5 * -1) + (1 * 0) = -0.5$
     $Output\ Node\ \#3 = -0.5$
   - $Input\ Node\ \#4 = (0.5 * 2) + (1 * 1) = 2$
     $Output\ Node\ \#4 = 2$
   - $Input\ Node\ \#5 = (0.5 * -2) + (1 * 1) = 0$
     $Output\ Node\ \#5 = 0$
   - $Input\ Node\ \#6 = (-0.5 * 2) + (2 * -0.5) + (0 * 1) = -2$
     $Output\ Node\ \#6 = -2$
   - $Input\ Node\ \#7 = (-0.5 * -2) + (2 * 1) + (0 * 0.5) = 3$
     $Output\ Node\ \#7 = 3$



   b. Find the value at the output nodes of the network for the input (1,2). Do you need to repeat the computations all over again? Why?

   - $Output\ Node\ \#3 = -1$
   - $Output\ Node\ \#4 = 4$
   - $Output\ Node\ \#5 = 0$
   - $Output\ Node\ \#6 = -4$
   - $Output\ Node\ \#7 = 6$
   - We do not need to repeat the calculations since the inputs in this part is double the values of the inputs from the previous part. From this observation, all we need to do to update the outputs of each node in this part is to double them from the previous part.

4. 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/chris-k87/CS_4210.01/tree/main/Assignment_4/Deep_Learning

5. Consider the dataset to the right. We will apply *steady state* Genetic Algorithms ($r = 0.5$) to solve this classification problem, by using a similar approach of the Find-S algorithm. If the instance matches the rule pre-condition of the chromosome, you predict according to its post-condition, otherwise you predict the opposite class defined by the chromosome (there must be a prediction for each instance then). Follow the planning below to build your solution.

- Representation: single if-then rule by using bit strings (binary encoding).
  - Outlook $<$*Sunny, Overcast, Rain*$>$
  - Temperature $<$*Hot, Mild, Cool*$>$

Outlook	Temperature	PlayTennis
Sunny	Hot	No
Overcast	Cool	Yes
Overcast	Hot	Yes
Rain	Cool	No
Overcast	Mild	Yes

Examples:

  - Outlook = *Overcast* $\rightarrow$ 010
  - Outlook = *Overcast* $\lor$ Rain $\rightarrow$ 011
  - Outlook = *Sunny* $\lor$ *Overcast* $\lor$ Rain $\rightarrow$ 111
  - Outlook = (*Overcast* $\lor$ *Rain*) $\land$ (Temperature = *Hot*) $\rightarrow$ 011100
  - Outlook = *Sunny* $\land$ Temperature = *Hot* then PlayTennis = *yes* $\rightarrow$ 1001001
- Initial population (Chromosomes) : (C_1=1001001, C_2=0100101, C_3=1011000, C_4=1101100). Population size should remain the same (4 individuals) over time.
- Fitness function: accuracy
- Penalty criterion: no penalty.
- Selection method: The best two chromosomes are carried over to the next generation. The other two are selected for crossover by using the roulette wheel (simulation).
- Crossover strategy: single-point crossover with mask 1110000 in the 1st generation, two-point crossover with mask 0001100 in the 2nd generation. Use the following chromosomes to perform crossover (simulating the process of spinning the roulette wheel) according to the relative fitness (sectors of a roulette wheel), generating two offspring for each crossover:
  - (1st and 3rd) in the 1st generation
  - (1st and 2nd) in the 2nd generation
- Mutation: on the 6th bit of the chromosome(s) 1011000 generated during the 2nd generation.
- Termination criteria: accuracy = 1.0. **Return the corresponding chromosome(s) – your model.**

Solution:

$$\text{Fitness}(C_1) = \frac{1}{5} = 0.2 \qquad \Pr(C_1) = \frac{0.2}{0.2+0.6+0.8+0.4} = 0.1 \ (4^{th})$$

$$\text{Fitness}(C_2) = \frac{3}{5} = 0.6 \qquad \Pr(C_4) = \frac{0.4}{0.2+0.6+0.8+0.4} = 0.2 \ (3^{rd})$$

$$\text{Fitness}(C_3) = \frac{4}{5} = 0.8 \qquad \Pr(C_2) = \frac{0.6}{0.2+0.6+0.8+0.4} = 0.3 \ (2^{nd})$$

$$\text{Fitness}(C_4) = \frac{2}{5} = 0.4 \qquad \Pr(C_3) = \frac{0.8}{0.2+0.6+0.8+0.4} = 0.4 \ (1^{st})$$

1st generation (C_1 = 1001001, C_2 = 0100101, C_3 = 1011000, C_4 = 1101100):

Applying single-point crossover with mask 1110000:

$C_3 = 1011000 \rightarrow C_5 = 1011100$

$C_4 = 1101100 \rightarrow C_6 = 1101000$

$$\text{Fitness}(C_2) = \frac{3}{5} = 0.6 \qquad \Pr(C_2) = \frac{0.6}{0.6+0.8+0.8+0.6} \approx 0.214286 \ (4^{th})$$

$$\text{Fitness}(C_3) = \frac{4}{5} = 0.8 \qquad \Pr(C_6) = \frac{0.6}{0.6+0.8+0.8+0.6} \approx 0.214286 \ (3^{rd})$$

$$\text{Fitness}(C_5) = \frac{4}{5} = 0.8 \qquad \Pr(C_3) = \frac{0.8}{0.6+0.8+0.8+0.6} \approx 0.285714 \ (2^{nd})$$

$$\text{Fitness}(C_6) = \frac{3}{5} = 0.6 \qquad \Pr(C_5) = \frac{0.8}{0.6+0.8+0.8+0.6} \approx 0.285714 \ (1^{st})$$

2nd generation (C_2 = 0100101, C_3 = 1011000, C_5 = 1011100, C_6 = 1101000)

Applying two-point crossover with mask 0001100:

$C_5 = 1011100 \rightarrow C_7 = 1011000$

$C_3 = 1011000 \rightarrow C_8 = 1011100$

Applying mutation on 1011000:

$C_5 = 1011000 \rightarrow C_5 = 1011010$

$$\text{Fitness}(C_3) = \frac{4}{5} = 0.8$$

$$\text{Fitness}(C_5) = \frac{5}{5} = 1.0$$

$$\text{Fitness}(C_7) = \frac{4}{5} = 0.8$$

$$\text{Fitness}(C_8) = \frac{4}{5} = 0.8$$

3rd generation (C_3 = 1011000, C_5 = 1011010, C_7 = 1011000, C_8 = 1011100)

Final answer: C_{3m} = 1011010