Assignment 4

Nabin Chapagain
1001551151

October 15, 2020

1 TASK 1

1.1 Task 1a

Program is attached along with this file in the same folder.

The last line printed by my program as output was:

classification accuracy=0.8659

classification accuracy=0.9197

1.2 Task 1b

I experimented with different values for number of layers, number of units per layer (excluding first and last layer), and number of rounds which is tabulated in table 1.1.

layers	units per layer	rounds	accuracy
3	10	20	0.5248
3	20	20	0.5248
3	30	20	0.5269
3	50	20	0.5289
2	30	20	0.5248
3	10	20	0.5248
4	10	20	0.3595
5	10	20	0.3017
5	20	20	0.3017
3	20	20	0.5248
3	20	50	0.5310
3	**20**	**75**	**0.5661**
3	10	20	0.5186
3	10	50	0.5186
3	10	75	0.5579

Table 1.1: Table with changing accuracy for changing variable

For the sake of this problem, I used yeast_test.txt and yeast_training.txt for testing and training the data set.
As we can see from the table, **the most accuracy came out of combination of 3 layers, 20 units per layer and 75 rounds** The accuracy was 0.56 every time when rounded to two decimal places.

I checked different numbers of layers ranging from 2 to 5 and as we went up in number of layers, the accuracy went down. Although 2 number of layers was almost as accurate as 3 number of layers, it could not take advantage of multiple units in a layer to solve the problem. So, 3 layers was the right number of layers. Then, I experimented with different number of units per layer even for 3 layers and although accuracy was increasing, the change was very minuscule while the computational cost was getting higher and higher. Then, I checked different number of rounds, but as I went north of 75 rounds, it started taking really long time which would just increase the computational cost and would not give any significant change in accuracy.

1.3 Task 1c

I tried playing around with different activating functions like tanh or relu but the accuracy went down instead. I changed the value of η from 0.98 to 0.99 and it got slightly better. The accuracy percentage went up by a magnitude of about 1% but that was not consistent with each run.

I have attached the program along with this file.

2 Task 2

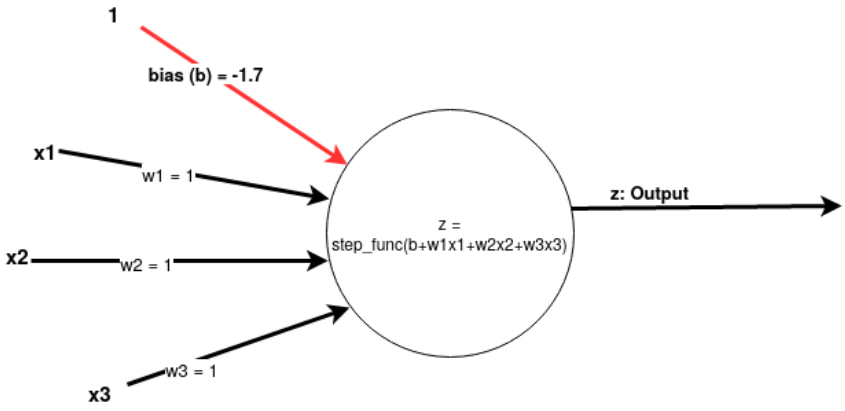

Figure 2.1: Perceptron for Task 2

Assuming the activation function for this question is the step function, I have come up with design for the perceptron as shown in Figure 2.1.

As we can see from the perceptron in figure 2.1, I have set bias as -1.7, and individual weights for each x_i as 1.

Check:

set of $\{T, T, F\} = \{1, 1, 0\}$ will give us:

$$z = 1 + 1 + 0 + (-1.7)$$

$$z = 0.3$$

which is True. But, for two lies and a truth we get,

$$z = 0 + 0 + 1 + (-1.7)$$

$$z = -0.7$$

which is False. So, the perceptron satisfies the requirement of the problem. And, just like that we can see with a set of three true values $z = 1 + 1 + 1 - 1.7$ would be $z = 1.3$ so, True and for three false values $z = 0 + 0 + 0 - 1.7$ which means $z = -1.7$ which is a False.

3 TASK 3

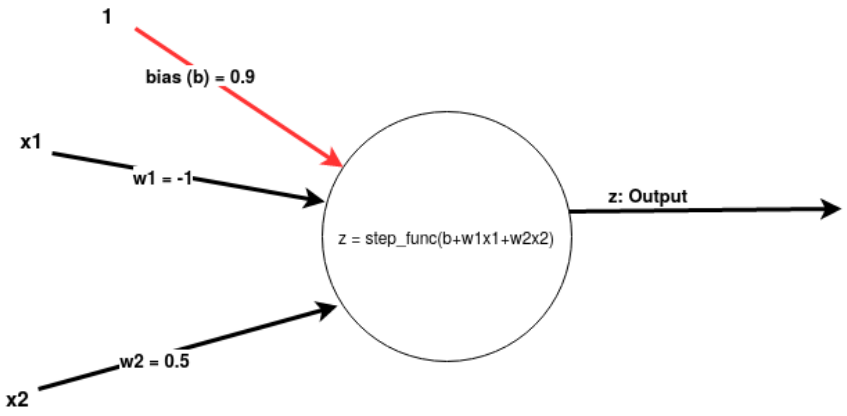

Figure 3.1: Perceptron for Task 3

Now, for Task 3, the approach is similar but there is a conditional element which makes it tricky. "If $\mathbb{X}$ then $\mathbb{Y}$ " statement gives True as output all the times except when $\mathbb{X}$ is true but $\mathbb{Y}$ is false. So, I have designed the perceptron as shown in figure 3.1 as a solution to this problem.

As we can see from the perceptron in figure 3.1, I have set bias as 0.9, and w_1 for x_1 as -1 and w_2 for x_2 as 0.5.

Check:

set of $\{T, T\} = \{1, 1\}$ will give us:

$$z = -1 + 0.5 + 0.9$$

$$z = 0.4$$

which is True. But, for $\mathbb{X} = True$ and a $\mathbb{Y} = False$ we get,

$$z = -1 + 0 + 0.9$$

$$z = -0.1$$

which is False. So, the perceptron satisfies the requirement of the problem. And, just like that we can see with both false values $z = 0 + 0 + 0.9$ would be $z = 0.9$ so, True and for $\mathbb{X} = False$ and $\mathbb{Y} = True$, we get $z = 0 + 0.5 + 0.9$ which means $z = 1.4$ which is True too.

4 TASK 4

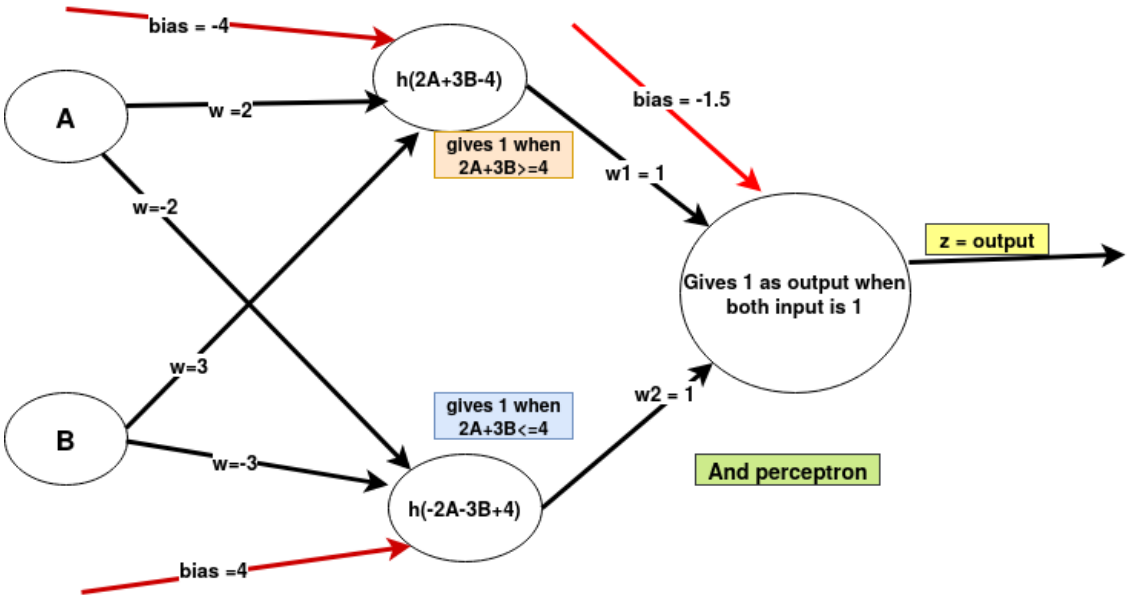

Figure 4.1: Neural Network for Task 4

This was a little bit tricky problem. The approach I took for this problem is heavily based on $\frac{sin(\theta)}{\theta}$ for $\theta = 0$ problem in calculus.

If we can get solutions for which, $2A + 3B \geq 4$ and $2A + 3B \leq 4$, both conditions hold true then the only logical solution would be $2A + 3B = 4$.

As we can see from the Figure 4.1, middle layer gets A and B as inputs and gives out 1 or 0 as output that fulfill the condition as given in the ellipse. And, final solution is filtered in the last layer where we get as an output 1, for those set of solutions which satisfy both conditions: $2A + 3B \geq 4$ and $2A + 3B \leq 4$, 0 otherwise.

5 TASK 5

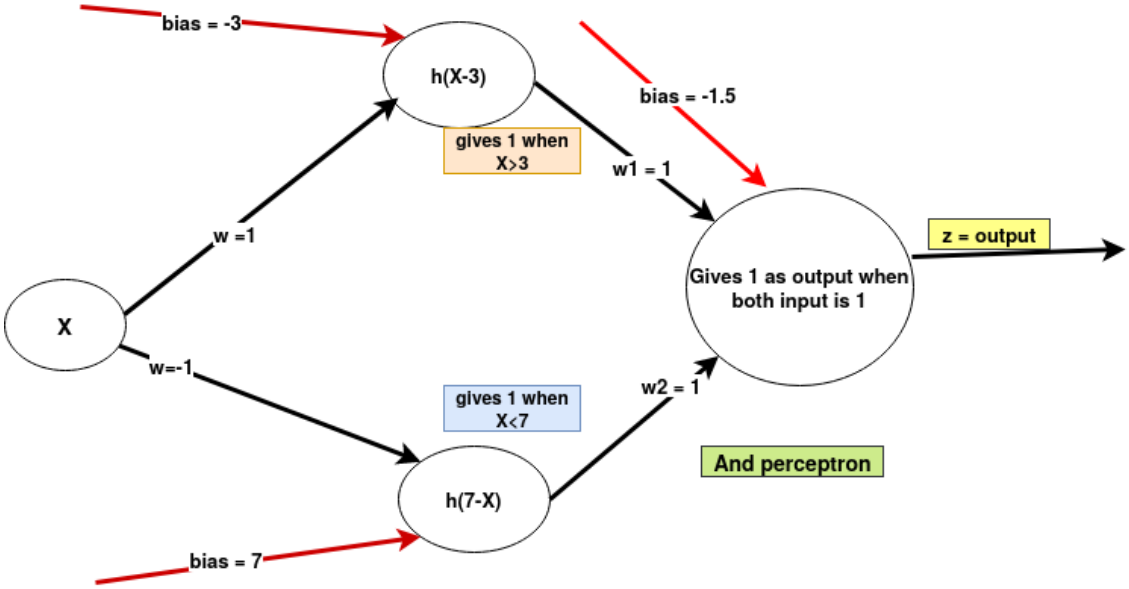

Figure 5.1: Neural Network for Task 5

So, as we can see from the figure 5.1, it is possible to design a neural network that outputs 1 for $3 < \mathbb{X} < 7$ and 0 for any other value. I designed this neural network using the idea from Task 4 where only values greater than 3 (from top perceptron) and values less than 7 (from bottom perceptron) would yield output 1 in second layer. Then, I used AND perceptron to combine their values which gives us our desired result.