Question 1 - 2 points

Consider the following multi-layer perceptron.

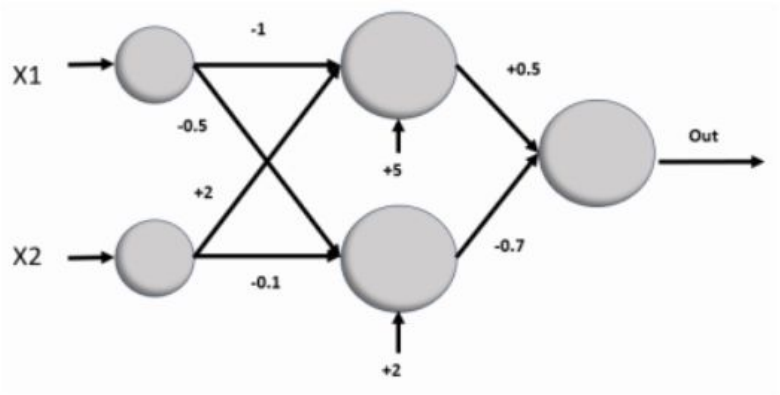


Figure 1: Two layer multi-layer perceptron

For the set of input data

$$\boldsymbol{X} = \left\{ \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \begin{bmatrix} -2 \\ -3 \end{bmatrix}, \begin{bmatrix} -5 \\ -1 \end{bmatrix}, \begin{bmatrix} -2 \\ 2 \end{bmatrix}, \begin{bmatrix} 6 \\ -2 \end{bmatrix}, \begin{bmatrix} 4 \\ 3 \end{bmatrix} \right\}$$

calculate the output vector $\boldsymbol{y} \in \mathbb{R}^6$. You can assume that all activation functions $\phi(\nu)$ are rectified linear units (ReLU), where

$$\phi(\nu) = \begin{cases} \nu & for\ \nu > 0 \\ 0 & otherwise \end{cases}$$

script used. (Done in Matlab)

solve_pair.m | solve_trig.m | solve_trig2.m | sincos.m | perceptfunct.m

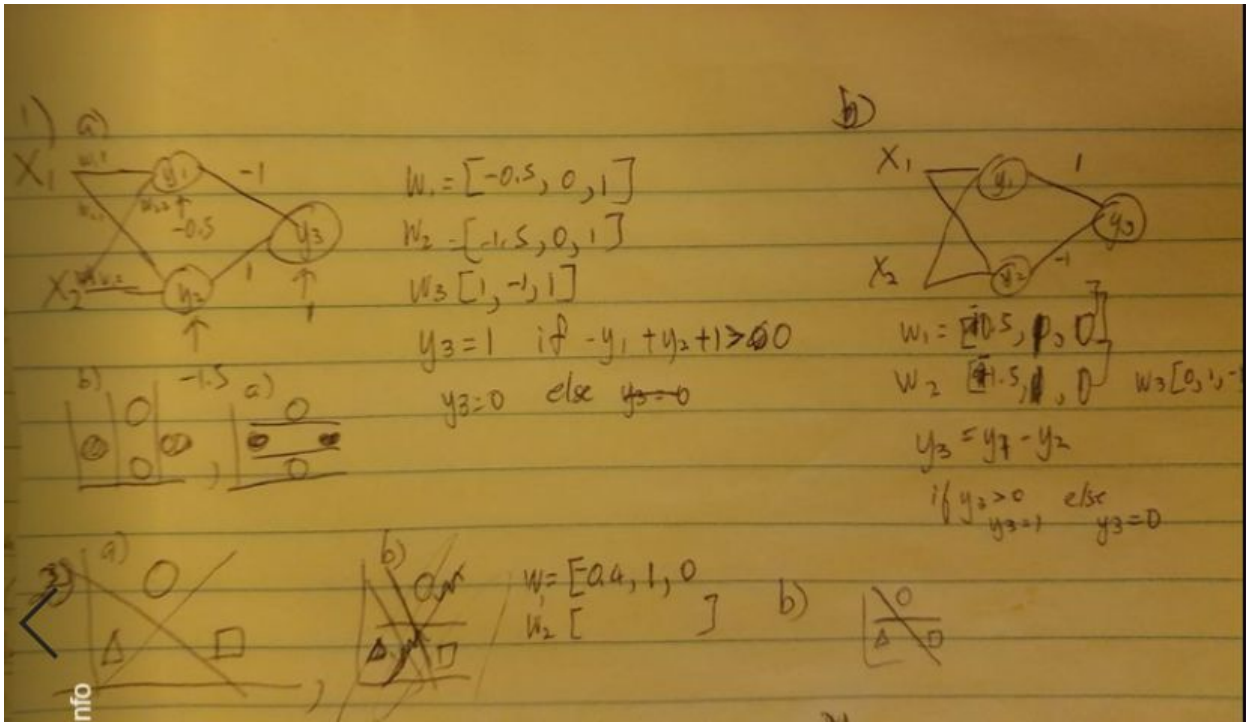
```
y1=-1*x1+2*x2+5;
y2=-0.5*x1+-0.1*x2+2;
if (y1<0)
    y1=0;
end
if (y2<0)
    y2=0;
end
output=y1*0.5+y2*-0.7;
if output <0
    output=0;
end
```

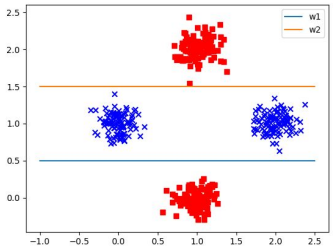
results obtained

y=[0, 0, 0.78, 3.54, 0, 3.5]

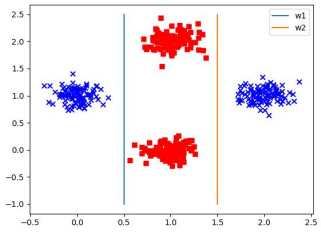
2)To my best ability, the neural networks presented below were designed by hand where all of the neuron uses a hard limit transfer function where y<=0 outputs results = 0, otherwise results =1. So far the weights are shown below, by drawing and by description, and via observation of the python code provided the comparison of the code results and the true label is indicated via boolean comparison.

2) image 1 a)Knowing 1 neuron outputs 2 values. I decided to create 2 hidden layer neurons to take a certain weight out of them and made a 3rd weight for the results with a bias of 1 which compliments the weights i have designed. b) the second neural network takes the inputs again and multiply them to some weights and the results of the hidden layer neurons subtracts each other to determine their labels.



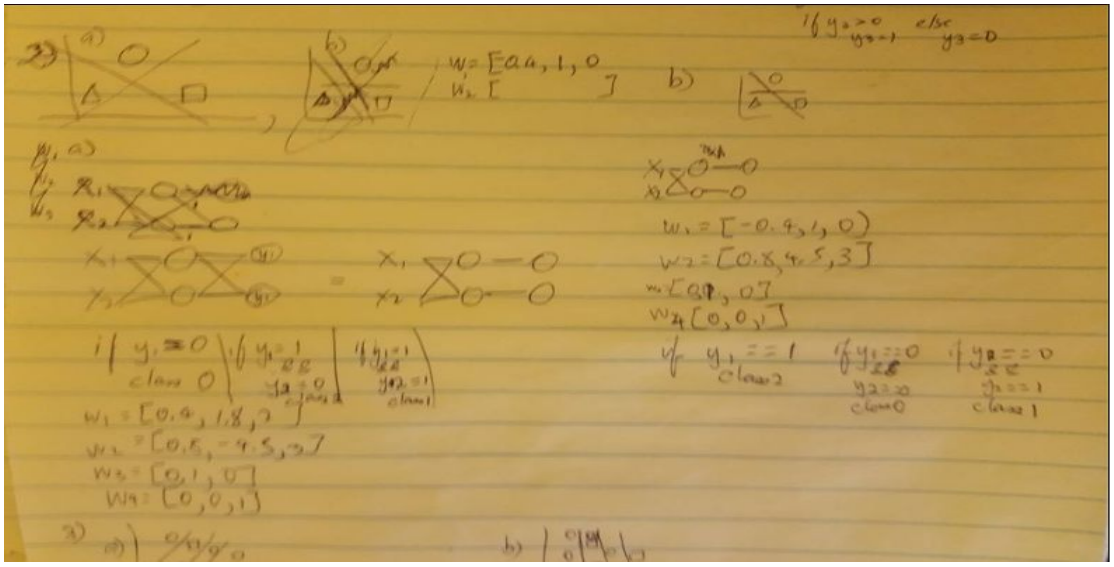


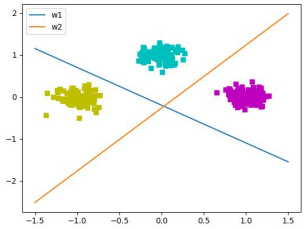
w1=[-0.5,0,1]
w2=[-3,0,2]
w3=[1,-1,1]



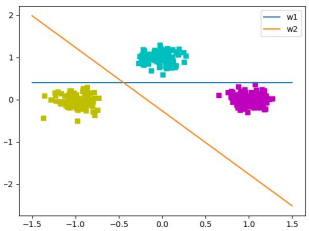
w1=[-0.5,1,0]
w2=[-1.5,1,0]
w3=[0,1,-1]

image 2 a) since this example outputs 3 classes [0,1,2] i made a hidden layer with 2 neurons developing 2 lines that could take in the inputs of x1 and x2. one of the lines represent dividing a section which can be classified as a certain class or otherwise while the second line discriminates the 2 remaining class , when classified to be "otherwise". b) The same statements applies from above, just made with different weights. The architecture is the same .



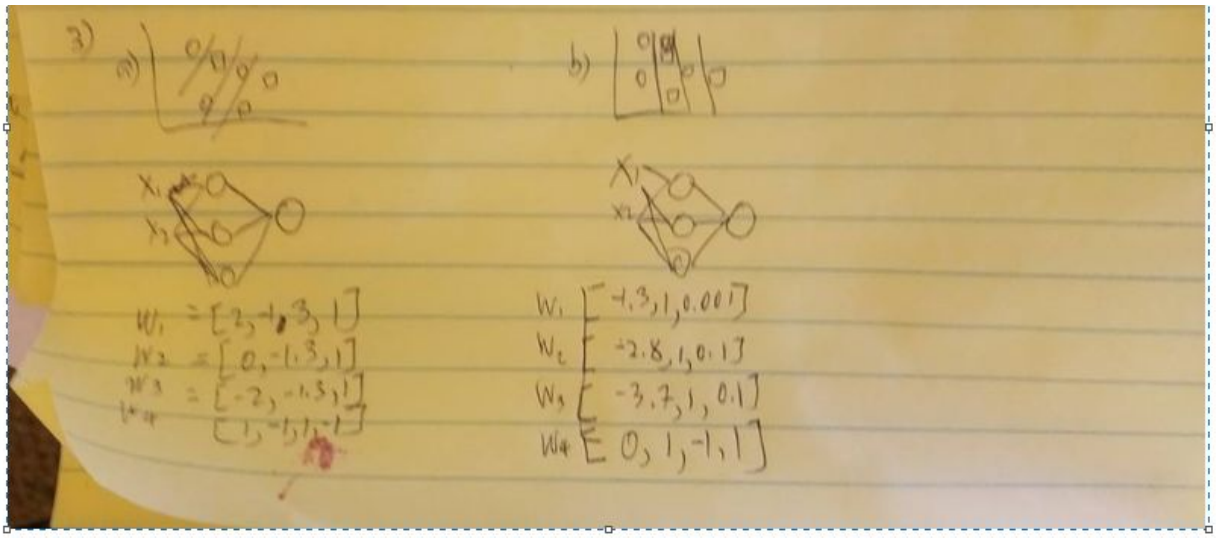


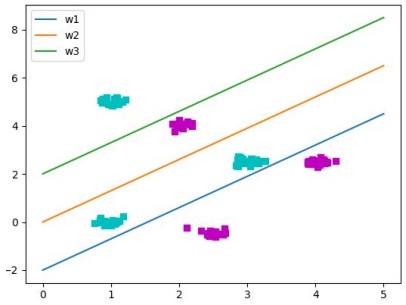
w1=[0.4,1.8,2]
w2=[0.8,-4.5,3]
w3=[0,1,0]
w4=[0,0,1]



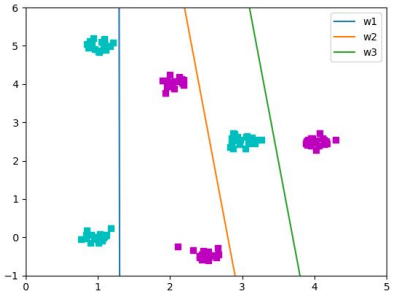
w1=[-1.3,1,0.001]
w2=[-2.8,1,0.1]
w3=[-3.7,1,0.1]
w4=[0,1,-1,1]

image3 a) This image only has 2 classes [0,1] but rather needs more discriminating or separating lines i made a hidden layer with 3 neurons developing 3 lines that could take in the inputs of x1 and x2. The lines separates the classes and the approach done was if one of the line results to be greater than 0 which results to 1 is either multiplied to a positive or a negative weight. Which summed up gives you a value of either 1 or 0. b) The same statements applies from above, just made with different weights. The architecture applied is the same





w1=[2,-1.3,1]
w2=[0,-1.3,1]
w3=[-2,-1.3,1]
w4=[1,-1,1,-1]



w1=[-1.3,1,0.001]
w2=[-2.8,1,0.1]

w3=[-3.7,1,0.1]
w4=[0,1,-1,1]
if last neuron =1 class =1 else 0

Reference:https://machinelearningmastery.com/implement-perceptron-algorithm-scratch-python/