Q.1. a) Implement AND gate using Neural network with backpropagation.

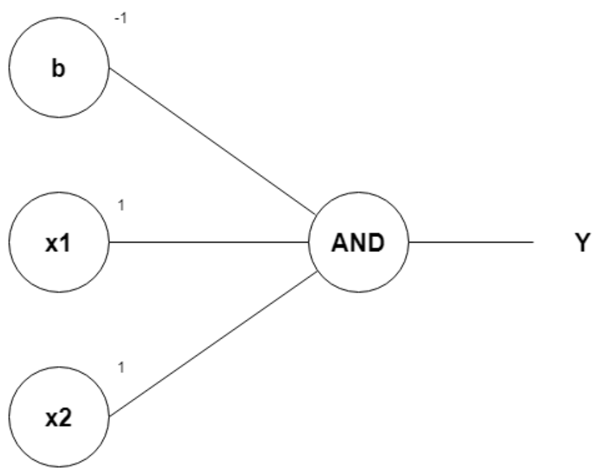
AND gate truth table:

X_1	X_2	Y
0	0	0
0	1	0
1	0	0
1	1	1

1. Inputs and outputs:

X = [[0, 0], [0, 1], [1, 0], [1, 1]] (shape= (4,2))

Y = [[0], [0], [0], [1]] (shape= (4,1))



NN representation of AND gate

2. Initialize the weight and bias parameters randomly such that:
   $W = [w_1, w_2]$ (shape = (2,1))
   B = [b] (shape = (1,1))

3. Activation function: Sigmoid function

```
def sigmoid(h):
        return 1/(1+exp(-h))
```

4. Forward propagation

```
a1 = X  #(4, 2)
z2 = dot( a1, W) + B  #(4, 1)
a2 = sigmoid(z2)  #(4, 1)
```

5. Backward propagation

- Modify sigmoid function for derivative:

```
def sigmoid_der(h):
    return h*(1-h)
```

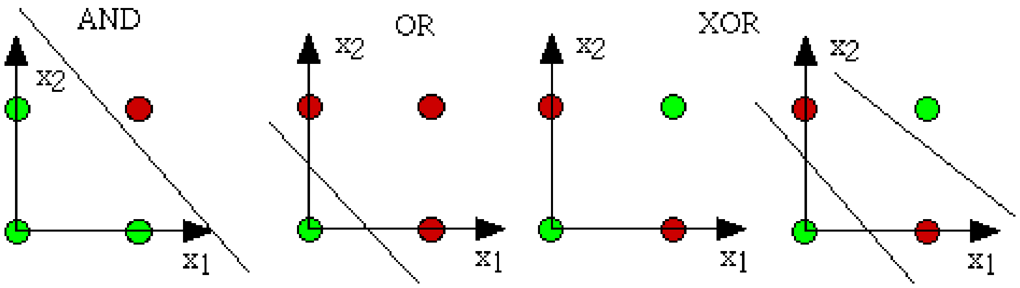
- Update weights and bias

```
error = (Y - a2)  #(4, 1)
delta_output = error * sigmoid_der(a2 ) # (4, 1)
update = dot(a1.T, delta _output)  #(2, 1)
W = W + update #(2,1)
B = B + sum(delta _output) #(1,1)
```

6. Repeat step3 till step 5 for 500 epochs.
7. Test it for example inputs
   X_{t1} = [0,0]
   X_{t2} = [0,1]
   X_{t3} = [1,0]
   X_{t4} = [1,1]

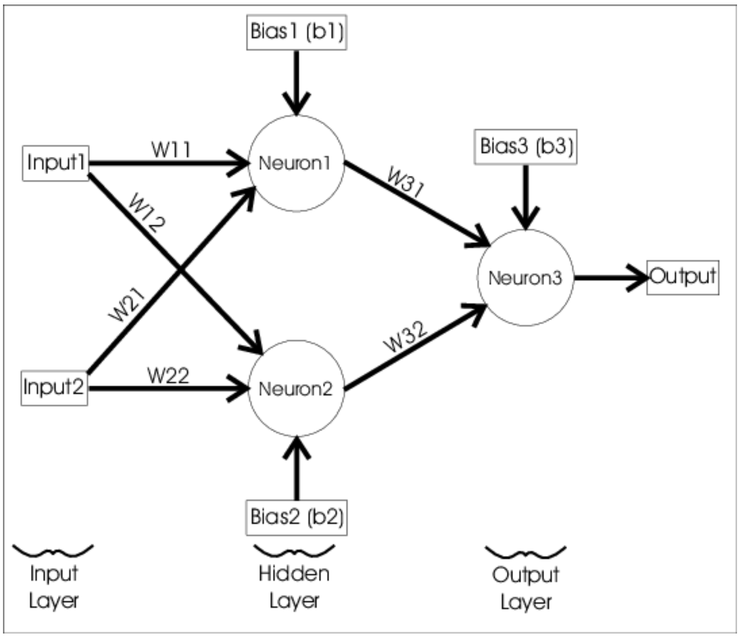
b) Implement OR, NAND and NOR gate in a similar way.

Q.2. a) Implement XOR gate using Neural network with backpropagation.



XOR gate truth table:

X_1	X_2	Y
0	0	0
0	1	1
1	0	1
1	1	0



NN representation of XOR gate

1. Inputs and outputs:

X = [[0, 0], [0, 1], [1, 0], [1, 1]]

Y = [[0], [0], [0], [1]]

2. Prepare input layer, hidden layer and output layer weights and bias parameters
   $W_h = [[w_{11}, w_{12}], [w_{21}, w_{22}]]$ shape = (2,2)
   $B_h = [b_1, b_2]$ shape = (1,2)
   $W_o = [w_{31}, w_{32}]$ shape = (2,1)
   $B_o = [b_3]$ shape = (1,1)

3. Activation function: Sigmoid function

```
def sigmoid(h):
        return 1/(1+exp(-h))
```

4. Forward propagation

```
a1 = X #(4,2)
z2 = dot(a1, Wh) + Bh  #(4,2)
a2 = sigmoid(z2) #(4,2)
z3 = dot(a2, Wo) + Bo #(4,1)
a3 = sigmoid(z3) #(4,1)
```

5. Backward propagation
   - Modify sigmoid function for derivative:

```
def sigmoid_der(h):
        return h*(1-h)
```

   - Error of output layer

```
error = (Y – a3) #(4,1)
```

delta_output = error * sigmoid_der(a3) #(4,1)
output_update = dot(a2.T, delta_output) #(2,1)

- Error of hidden layer
  error_h = np.dot(output_update, Wo.T) #(4,2)
  delta_hidden = error_h * sigmoid_der(a2) #(4,2)
  hidden_update = dot(a1.T, delta_hidden) #(2,2)

- Update weights and bias of output layer
  $W_o = W_o$ + output_update #learning ratio #(2,1)
  $B_o = B_o$ + node-wise-sum(delta_output) #learning ratio #(1,1)

- Update weights and bias of hidden layer
  $W_h = W_h$ + hidden_update #learning ratio #(2,2)
  $B_h = B_h$ + node-wise-sum(delta_hidden) #learning ratio #(1,2)

6. Repeat step3 till step 5 for 500 epochs.
7. Test it for example inputs
   X_{t1} = [0,0]
   X_{t2} = [0,1]
   X_{t3} = [1,0]
   X_{t4} = [1,1]

b) Implement XNOR gate using similar neural network model.

X_1	X_2	Y
0	0	1
0	1	0
1	0	0
1	1	1

Outputs expected: Testing with four cases, error vs. epoch plot