In [1]:
```
import numpy as np
```

In [2]:
```
def abs(x):
      return x if x>0 else -x
```

In [3]:
```
def sigmoid (x):
  return 1/(1 + np.exp(-x))
```

In [4]:
```
def sigmoid_derivative(x):
  return x * (1 - x)
```

In [5]:
```
def checkError(predicted_output):
      expected_output = [[0],[1],[1],[0]]
      for i,j in zip(expected_output , predicted_output):
               if abs(i[0]-j[0]) > 0.001:
                        return True
      return False
```

In [6]:
```
inputs = np.array([[0,0],[0,1],[1,0],[1,1]])
expected_output = np.array([[0],[1],[1],[0]])
```

In [7]:
```
epoch = 0
lr = 0.1
```

In [8]:
```
inputLayerNeurons = int(input("enter no of inputLayer"))
hiddenLayerNeurons = int(input("enter no of hiddenlayer "))
outputLayerNeurons = int(input("enter no of outputlayer "))
```

```
enter no of inputLayer2
enter no of hiddenlayer 2
enter no of outputlayer 1
```

In [9]:
```
hidden_weights = []
```

In [10]:
```
for i in range(1,inputLayerNeurons+1):
        hidden_weights_ind = []
        for j in range(inputLayerNeurons+1,inputLayerNeurons+hiddenLa
                hidden_weights_ind.append(float(input('w'+str(i)+str(
        hidden_weights.append(hidden_weights_ind)
```

```
w133
w146
w234
w245
```

In [11]:
```
output_weights = []
```

In [12]:
```
for i in range(inputLayerNeurons+1,inputLayerNeurons+hiddenLayerNeuro
        output_weights_ind = []
        for j in range(inputLayerNeurons+hiddenLayerNeurons+1,inputLa
                output_weights_ind.append(float(input('w'+str(i)+str(
        output_weights.append(output_weights_ind)
```

```
w352
w454
```

In [13]:
```
hidden_bias = []
output_bias = []
```

In [14]:
```
for i in range(inputLayerNeurons+1,inputLayerNeurons+hiddenLayerNeuro
        if i > inputLayerNeurons+hiddenLayerNeurons:
                output_bias.append(float(input("o"+str(i))))
        else:
                hidden_bias.append(float(input("o"+str(i))))
```

```
o31
o4-6
o5-3.93
```

In [15]:
```
hidden_weights = np.asarray(hidden_weights)
hidden_bias = np.asarray([hidden_bias])
output_weights = np.asarray(output_weights)
output_bias = np.asarray([output_bias])
```

In [16]:
```
print("Initial hidden weights: ",end='')
print(*hidden_weights)
print("Initial hidden biases: ",end='')
print(*hidden_bias)
print("Initial output weights: ",end='')
print(*output_weights)
print("Initial output biases: ",end='')
print(*output_bias)
```

```
Initial hidden weights: [3. 6.] [4. 5.]
Initial hidden biases: [ 1. -6.]
Initial output weights: [2.] [4.]
Initial output biases: [-3.93]
```

In [17]:
```
predicted_output = [[0],[0],[0],[0]]
```

In [18]:
```
while checkError(predicted_output):
    epoch += 1
    hidden_layer_activation = np.dot(inputs,hidden_weights)
    hidden_layer_activation += hidden_bias
    hidden_layer_output = sigmoid(hidden_layer_activation)
    output_layer_activation =np.dot(hidden_layer_output,output_weight
    output_layer_activation += output_bias
    predicted_output = sigmoid(output_layer_activation)
    error = expected_output - predicted_output
    d_predicted_output = error * sigmoid_derivative(predicted_output)
    error_hidden_layer = d_predicted_output.dot(output_weights.T)
    d_hidden_layer = error_hidden_layer *sigmoid_derivative(hidden_la
    output_weights += hidden_layer_output.T.dot(d_predicted_output) *
    output_bias += np.sum(d_predicted_output,axis=0,keepdims=True) *
    hidden_weights += inputs.T.dot(d_hidden_layer) * lr
    hidden_bias += np.sum(d_hidden_layer,axis=0,keepdims=True) * lr
    if(epoch==100000):
        break;
```

In [19]:
```
print("Final hidden weights: ",end='')
print(*hidden_weights)
print("Final hidden bias: ",end='')
print(*hidden_bias)
print("Final output weights: ",end='')
print(*output_weights)
print("Final output bias: ",end='')
print(*output_bias)
```

```
Final hidden weights: [3.92794571 9.8071798 ] [3.9279819  9.8002004
1]
Final hidden bias: [-5.87200846 -2.43171093]
Final output weights: [-8.40883702] [8.42301753]
Final output bias: [-4.10619277]
```

In [20]:
```
print("\nOutput from neural network: ",end='')
print(*predicted_output)
print("\nNo of epochs")
print(epoch)
```

```
Output from neural network: [0.03078948] [0.96298127] [0.96298378]
[0.0441378]

No of epochs
100000
```

In [ ]: