In [1]:

```python
import numpy as np
```

In [2]:

```python
def sigmoid (x):
    return 1/(1 + np.exp(-x))
```

In [3]:

```python
def sigmoid_derivative(x):
    return x * (1 - x)
```

In [5]:

```python
#Input datasets

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

In [6]:

```python
epochs = 10000
lr = 0.1
inputLayerNeurons, hiddenLayerNeurons, outputLayerNeurons = 2,2,1

```

In [7]:

```python
#Random weights and bias initialization

hidden_weights = np.random.uniform(size=(inputLayerNeurons,hiddenLayerN
hidden_bias =np.random.uniform(size=(1,hiddenLayerNeurons))
output_weights = np.random.uniform(size=(hiddenLayerNeurons,outputLayer
output_bias = np.random.uniform(size=(1,outputLayerNeurons))

```

In [8]:

```python
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: [0.84730363 0.76859947] [0.91037732 0.31547999]
Initial hidden biases: [0.21788549 0.33373447]
Initial output weights: [0.41528087] [0.32743761]
Initial output biases: [0.0497275]
```

In [10]:

```
#Training algorithm

for _ in range(epochs):

#Forward Propagation
    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_weights
    output_layer_activation += output_bias
    predicted_output = sigmoid(output_layer_activation)

#Backpropagation
    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_lay


```

In [11]:

```
#Updating Weights and Biases

output_weights += hidden_layer_output.T.dot(d_predicted_output) * lr
output_bias += np.sum(d_predicted_output,axis=0,keepdims=True) * lr
hidden_weights += inputs.T.dot(d_hidden_layer) * lr
hidden_bias += np.sum(d_hidden_layer,axis=0,keepdims=True) * lr

```

In [12]:

```
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: [0.84728475 0.7683328 ] [0.9103537  0.31532894]
Final hidden bias: [0.21700646 0.33291834]
Final output weights: [0.40645002] [0.31851796]
Final output bias: [0.0366907]
```

In [13]:

```
print("\nOutput from neural network after 10,000 epochs: ",end='')
print(*predicted_output)
```

```

Output from neural network after 10,000 epochs: [0.61554638] [0.64073409]
[0.64665556] [0.66329578]
```

In [ ]:

```

```