In [1]:

```
import numpy as np
```

In [2]:

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

X = data[:, :-1]
X = np.insert(X, 0, 1, axis=1)

Y_true = data[:, len(data)-2:]
```

In [3]:

```
W1 = np.random.rand(2, 3)
W2 = np.random.rand(1, 3)
```

In [4]:

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

def diff_sigmoid(x):
    temp = sigmoid(x)
    return temp * (1 - temp)
```

In [5]:

```
def feed_forward(x):

    net1 = np.matmul(W1, x)
    x1 = sigmoid(net1)
    x1 = np.insert(x1, 0, 1, axis=0)

    net2 = np.matmul(W2, x1)
    y = sigmoid(net2)

    return net1, x1, net2, y

def backprop(x, y_true, net1, x1, net2, y):

    del2 = (y - y_true) * diff_sigmoid(net2)
    Delta_W2 = np.matmul(del2, np.transpose(x1))

    del1 = np.matmul(np.transpose(np.delete(W2, 0, axis=1)), del2) * diff_sigmoi
d(net1)
    Delta_W1 = np.matmul(del1, np.transpose(x))

    return Delta_W1, Delta_W2

def get_gradients(x, y_true):

    net1, x1, net2, y = feed_forward(x)

    Delta_W1, Delta_W2 = backprop(x, y_true, net1, x1, net2, y)

    return Delta_W1, Delta_W2
```

In [6]:

```
def display_results():
    _, _, _, Y = feed_forward(np.transpose(X))

    print('Predicted Values - ', Y[0])

    error = np.average(np.square(Y_true - np.transpose(Y)), axis=0) [0]
    print('Error - ', error, end='\n\n')

    return error
```

In [7]:

```
def train(epochs, learning_rate, threshold):

    global W1, W2

    for epoch in range(epochs):
        for i in range(len(data)):

            x = np.reshape(X[i], [len(X[i]), 1])
            y_true = Y_true[i]

            Delta_W1, Delta_W2 = get_gradients(x, y_true)

            W1 = W1 - learning_rate*Delta_W1
            W2 = W2 - learning_rate*Delta_W2

        if epoch % (epochs/100) == 0:
            print('Epoch Number - ', epoch + 1)
            error = display_results()

            if(error < threshold):
                print('Error is less than threshold, threshold -> ' + str(thresh
old) + ' Error -> ' + str(error))
                break
```

In [8]:

```
train(epochs=100000, learning_rate=0.4, threshold=0.001)
```

```
Epoch Number -  1
Predicted Values -  [0.69025653 0.69919366 0.72664528 0.7338418 ]
Error -  0.2950462805454739

Epoch Number -  1001
Predicted Values -  [0.50431393 0.49209272 0.50744261 0.49462669]
Error -  0.24989267223478268

Epoch Number -  2001
Predicted Values -  [0.38436826 0.36647015 0.7477677  0.41307007]
Error -  0.19583676250596063

Epoch Number -  3001
Predicted Values -  [0.0880732  0.89505655 0.92653987 0.07448663]
Error -  0.007428666389726656

Epoch Number -  4001
Predicted Values -  [0.05046363 0.94164201 0.95526258 0.04343271]
Error -  0.002460017503708493

Epoch Number -  5001
Predicted Values -  [0.03841644 0.9558938  0.96533385 0.03331813]
Error -  0.0014332549395157951

Epoch Number -  6001
Predicted Values -  [0.03206888 0.9632937  0.97080275 0.02793644]
Error -  0.001002172489218904

Epoch Number -  7001
Predicted Values -  [0.02802425 0.96797392 0.97434406 0.02448599]
Error -  0.000767204870576844

Error is less than threshold, threshold -> 0.001 Error -> 0.00076720
4870576844
```

In [ ]:

```

```