Assignment 6

In [1]:

```
import numpy as np
```

In [2]:

```
class NeuralNetwork:
    def __init__(self, input_size, hidden_size, output_size):
        self.input_size = input_size
        self.hidden_size = hidden_size
        self.output_size = output_size

        #initialize weights
        self.W1 = np.random.randn(self.input_size, self.hidden_size)
        self.W2 = np.random.randn(self.hidden_size, self.output_size)

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

    def sigmoid_derivative(self, x):
        return x * (1 - x)

    def forward(self, X):
        #calculate output of hidden layer
        self.z = np.dot(X, self.W1)
        self.z2 = self.sigmoid(self.z)

        #calculate output of output layer
        self.z3 = np.dot(self.z2, self.W2)
        output = self.sigmoid(self.z3)
        return output

    def backward(self, X, y, output):
        #calculate the error and derivative of error for output layer
        self.output_error = y - output
        self.output_delta = self.output_error * self.sigmoid_derivative(output)

        #calculate error and derivative of error for hidden layer
        self.z2_error = self.output_delta.dot(self.W2.T)
        self.z2_delta = self.z2_error * self.sigmoid_derivative(self.z2)

        #update weights
        self.W1 += X.T.dot(self.z2_delta)
        self.W2 += self.z2.T.dot(self.output_delta)

    def train(self, X, y, epochs):
        for i in range(epochs):
            #forward propagation
            output = self.forward(X)

            #backward propagation
            self.backward(X, y, output)
```

In [3]:

```
#create a neural network object by specifying the number of inputs, hidden units,
nn = NeuralNetwork(input_size=2,hidden_size=3, output_size=1)
```

In [4]:

```
#specify train data 'X' and target output 'y'
X = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
y = np.array([[0], [1], [1], [0]])

#train the network
nn.train(X, y, epochs=10000)
```

In [5]:

```
#make predictions on new data
new_data = np.array([[0, 0.5], [0, 0.8], [1, 0.2], [1, 0.6]])
predictions = nn.forward(new_data)
print(predictions)
```

```
[[0.91723047]
 [0.97264792]
 [0.96824254]
 [0.33806858]]
```