Assignment No:6

Title: ANN training process of forward propogation, back propogation.

Name: Tavhare Ruchita Sharad

Roll No: 58

In [17]:

```
import numpy as np
```

In [18]:

```
class NeuralNetwork:
    def __init__(self, input_size, hidden_size, output_size,activation):
        # Initialize weights and biases
        self.W1 = np.random.randn(input_size, hidden_size)
        self.b1 = np.zeros((1, hidden_size))
        self.W2 = np.random.randn(hidden_size, output_size)
        self.b2 = np.zeros((1, output_size))

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

    # Derivative of sigmoid activation function
    def sigmoid_derivative(self, x):
        return x * (1 - x)

    # Forward propagation
    def forward_propagation(self, X):
        self.z1 = np.dot(X, self.W1) + self.b1
        self.a1 = self.sigmoid(self.z1)
        self.z2 = np.dot(self.a1, self.W2) + self.b2
        self.y_hat = self.sigmoid(self.z2)
        return self.y_hat

    # Backward propagation
    def backward_propagation(self, X, y, y_hat):
        self.error = y - y_hat
        self.delta2 = self.error * self.sigmoid_derivative(y_hat)
        self.a1_error = self.delta2.dot(self.W2.T)
        self.delta1 = self.a1_error * self.sigmoid_derivative(self.a1)

        # Gradient descent weight and bias updates
        self.W2 += self.a1.T.dot(self.delta2)
        self.b2 += np.sum(self.delta2, axis=0, keepdims=True)
        self.W1 += X.T.dot(self.delta1)
        self.b1 += np.sum(self.delta1, axis=0 )

    # Training function
    def train(self, X, y,lr , epochs):
        for i in range(epochs):
            y_hat = self.forward_propagation(X)
            self.backward_propagation(X, y, y_hat)

            if i % 100 == 0:
                print("Error at epoch",i,":",np.mean(np.abs(self.error)))

    def predict(self, X):
        return self.forward_propagation(X)
```

In [19]:

```
# Define the input and output datasets (XOR problem)
X = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])  # Example input dataset for XOR problem
y = np.array([[0], [1], [1], [0]])  # XOR output dataset

# Initialize the neural network with 2 input neurons, 4 hidden neurons, and 1 output neuron, using ReLU activat
nn = NeuralNetwork(input_size=2, hidden_size=4, output_size=1, activation='relu')

# Train the neural network on the input and output datasets for 10,000 epochs with a learning rate of 0.1
nn.train(X, y, lr=0.1, epochs=100)

predictions = nn.predict(X)

# Print the predictions

print(predictions)
```

```
Error at epoch 0 : 0.5002984967937288
[[0.30370449]
 [0.6375522 ]
 [0.51053682]
 [0.57836019]]
```

In [ ]: