```
import numpy as np
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 and biases
        self.weights_input_hidden = np.random.randn(self.input_size, self.hidden_size)
        self.bias_hidden = np.zeros((1, self.hidden_size))
        self.weights_hidden_output = np.random.randn(self.hidden_size, self.output_size)
        self.bias_output = np.zeros((1, 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):
        # Forward pass
        self.hidden_layer_input = np.dot(X, self.weights_input_hidden) + self.bias_hidden
        self.hidden_layer_output = self.sigmoid(self.hidden_layer_input)

        self.output_layer_input = np.dot(self.hidden_layer_output, self.weights_hidden_output) +
self.bias_output
        self.output = self.sigmoid(self.output_layer_input)

        return self.output
```

```
    def backward(self, X, y, output, learning_rate):
        # Backpropagation
        error = y - output
        output_delta = error * self.sigmoid_derivative(output)

        hidden_error = output_delta.dot(self.weights_hidden_output.T)
        hidden_delta = hidden_error * self.sigmoid_derivative(self.hidden_layer_output)

        # Update weights and biases
        self.weights_hidden_output += self.hidden_layer_output.T.dot(output_delta) * learning_rate
        self.bias_output += np.sum(output_delta, axis=0, keepdims=True) * learning_rate
        self.weights_input_hidden += X.T.dot(hidden_delta) * learning_rate
        self.bias_hidden += np.sum(hidden_delta, axis=0, keepdims=True) * learning_rate

    def train(self, X, y, epochs, learning_rate):
        for epoch in range(epochs):
            output = self.forward(X)
            self.backward(X, y, output, learning_rate)
            if epoch % 100 == 0:
                print(f'Epoch {epoch}: Error {np.mean(np.square(y - output))}')



# Example usage:

# Initialize neural network
input_size = 1
hidden_size = 1
output_size = 1
```

```
nn = NeuralNetwork(input_size, hidden_size, output_size)
# Example training data
X = np.array([[5]])  # Input
y = np.array([[1]])  # Target output

# Train the neural network
nn.train(X, y, epochs=1000, learning_rate=0.1)

# Make predictions
predictions = nn.forward(X)
print("Predictions:", predictions)

# Example usage:
# Initialize neural network
input_size = 2
hidden_size = 3
output_size = 1

nn = NeuralNetwork(input_size, hidden_size, output_size)
# Example training data
X = np.array([[0, 0],[1, 0],[0, 1], [1, 1]])
y = np.array([[1],[0],[1], [0]])

# Train the neural network
nn.train(X, y, epochs=987, learning_rate=0.1)

# Make predictions
predictions = nn.forward(X)
print("Predictions:", predictions)
```