```python
import numpy as np

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

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

# Training data (AND function)
X = np.array([[0, 0],
              [0, 1],
              [1, 0],
              [1, 1]])

y = np.array([[0], [0], [0], [1]])  # AND gate output

# Initialize weights and biases
np.random.seed(1)
input_neurons = 2
hidden_neurons = 2
output_neurons = 1


# Weights
w_input_hidden = np.random.uniform(-1, 1, (input_neurons,
hidden_neurons))
w_hidden_output = np.random.uniform(-1, 1, (hidden_neurons,
output_neurons))

# Biases
b_hidden = np.random.uniform(-1, 1, (1, hidden_neurons))
b_output = np.random.uniform(-1, 1, (1, output_neurons))

# Training loop
epochs = 100
learning_rate = 0.1

for epoch in range(epochs):
    # --- Forward Propagation ---
    hidden_input = np.dot(X, w_input_hidden) + b_hidden
    hidden_output = sigmoid(hidden_input)

    final_input = np.dot(hidden_output, w_hidden_output) + b_output
    final_output = sigmoid(final_input)

    # --- Backpropagation ---
    error = y - final_output
    d_output = error * sigmoid_derivative(final_output)

    error_hidden = d_output.dot(w_hidden_output.T)
    d_hidden = error_hidden * sigmoid_derivative(hidden_output)
```

```
    # --- Update Weights and Biases ---
    w_hidden_output += hidden_output.T.dot(d_output) * learning_rate
    b_output += np.sum(d_output, axis=0, keepdims=True) *
learning_rate

    w_input_hidden += X.T.dot(d_hidden) * learning_rate
    b_hidden += np.sum(d_hidden, axis=0, keepdims=True) *
learning_rate

    # Optional: Print loss every 1000 epochs
    if epoch % 1 == 0:
        loss = np.mean(np.square(error))
        print(f"Epoch {epoch} Loss: {loss:.4f}")
```

```
Epoch 0 Loss: 0.0004
Epoch 1 Loss: 0.0004
Epoch 2 Loss: 0.0004
Epoch 3 Loss: 0.0004
Epoch 4 Loss: 0.0004
Epoch 5 Loss: 0.0004
Epoch 6 Loss: 0.0004
Epoch 7 Loss: 0.0004
Epoch 8 Loss: 0.0004
Epoch 9 Loss: 0.0004
Epoch 10 Loss: 0.0004
Epoch 11 Loss: 0.0004
Epoch 12 Loss: 0.0004
Epoch 13 Loss: 0.0004
Epoch 14 Loss: 0.0004
Epoch 15 Loss: 0.0004
Epoch 16 Loss: 0.0004
Epoch 17 Loss: 0.0004
Epoch 18 Loss: 0.0004
Epoch 19 Loss: 0.0004
Epoch 20 Loss: 0.0004
Epoch 21 Loss: 0.0004
Epoch 22 Loss: 0.0004
Epoch 23 Loss: 0.0004
Epoch 24 Loss: 0.0004
Epoch 25 Loss: 0.0004
Epoch 26 Loss: 0.0004
Epoch 27 Loss: 0.0004
Epoch 28 Loss: 0.0004
Epoch 29 Loss: 0.0004
Epoch 30 Loss: 0.0004
Epoch 31 Loss: 0.0004
Epoch 32 Loss: 0.0004
Epoch 33 Loss: 0.0004
Epoch 34 Loss: 0.0004
Epoch 35 Loss: 0.0004
```

```
Epoch 36 Loss: 0.0004
Epoch 37 Loss: 0.0004
Epoch 38 Loss: 0.0004
Epoch 39 Loss: 0.0004
Epoch 40 Loss: 0.0004
Epoch 41 Loss: 0.0004
Epoch 42 Loss: 0.0004
Epoch 43 Loss: 0.0004
Epoch 44 Loss: 0.0004
Epoch 45 Loss: 0.0004
Epoch 46 Loss: 0.0004
Epoch 47 Loss: 0.0004
Epoch 48 Loss: 0.0004
Epoch 49 Loss: 0.0004
Epoch 50 Loss: 0.0004
Epoch 51 Loss: 0.0004
Epoch 52 Loss: 0.0004
Epoch 53 Loss: 0.0004
Epoch 54 Loss: 0.0004
Epoch 55 Loss: 0.0004
Epoch 56 Loss: 0.0004
Epoch 57 Loss: 0.0004
Epoch 58 Loss: 0.0004
Epoch 59 Loss: 0.0004
Epoch 60 Loss: 0.0004
Epoch 61 Loss: 0.0004
Epoch 62 Loss: 0.0004
Epoch 63 Loss: 0.0004
Epoch 64 Loss: 0.0004
Epoch 65 Loss: 0.0004
Epoch 66 Loss: 0.0004
Epoch 67 Loss: 0.0004
Epoch 68 Loss: 0.0004
Epoch 69 Loss: 0.0004
Epoch 70 Loss: 0.0004
Epoch 71 Loss: 0.0004
Epoch 72 Loss: 0.0004
Epoch 73 Loss: 0.0004
Epoch 74 Loss: 0.0004
Epoch 75 Loss: 0.0004
Epoch 76 Loss: 0.0004
Epoch 77 Loss: 0.0004
Epoch 78 Loss: 0.0004
Epoch 79 Loss: 0.0004
Epoch 80 Loss: 0.0004
Epoch 81 Loss: 0.0004
Epoch 82 Loss: 0.0004
Epoch 83 Loss: 0.0004
Epoch 84 Loss: 0.0004
```

```
Epoch 85 Loss: 0.0004
Epoch 86 Loss: 0.0004
Epoch 87 Loss: 0.0004
Epoch 88 Loss: 0.0004
Epoch 89 Loss: 0.0004
Epoch 90 Loss: 0.0004
Epoch 91 Loss: 0.0004
Epoch 92 Loss: 0.0004
Epoch 93 Loss: 0.0004
Epoch 94 Loss: 0.0004
Epoch 95 Loss: 0.0004
Epoch 96 Loss: 0.0004
Epoch 97 Loss: 0.0004
Epoch 98 Loss: 0.0004
Epoch 99 Loss: 0.0004
```

```python
# Final output
print("\nFinal predictions after training:")
print(final_output.round(3))
```

```
Final predictions after training:
[[0.001]
 [0.027]
 [0.027]
 [0.953]]
```