Praneesh R V
CB.SC.U4CYS23036

AINN Lab - 6 Implementation of XOR using MLP

```
import numpy as np

# Sigmoid Activation and Derivative
def sigmoid(x):
   return 1 / (1 + np.exp(-x))

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

# XOR Truth Table
# X1 X2 | Y
# 0 0  | 0
# 0 1  | 1
# 1 0  | 1
# 1 1  | 0
X = np.array([[0,0],[0,1],[1,0],[1,1]])
y = np.array([[0],[1],[1],[0]])

# Parameters
input_neurons = 2
hidden_neurons = 2  # minimum needed for XOR
output_neurons = 1
learning_rate = 1.5
threshold = 1

# Initialize weights and biases (all = 1 as per requirement)
w_input_hidden = np.ones((input_neurons, hidden_neurons))
w_hidden_output = np.ones((hidden_neurons, output_neurons))
b_hidden = np.ones((1, hidden_neurons)) * threshold
b_output = np.ones((1, output_neurons)) * threshold

# Training
epochs = 200  # try 50, 100, 200
for epoch in range(epochs):
```

```
    # Forward Pass
    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)

    # Error
    error = y - final_output
    loss = np.mean(np.square(error))

    # Backpropagation
    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

    # Print loss every 10 epochs
    if epoch % 10 == 0:
        print(f"Epoch {epoch}, Loss: {loss:.4f}")

# Final Prediction
print("\nFinal Outputs after training:")
print("Raw outputs:\n", final_output.round(3))
print("Rounded (Predicted XOR):\n", np.round(final_output))
```

Output:

```
Epoch 0, Loss: 0.4388
Epoch 10, Loss: 0.2489
Epoch 20, Loss: 0.2483
Epoch 30, Loss: 0.2476
Epoch 40, Loss: 0.2468
Epoch 50, Loss: 0.2460
Epoch 60, Loss: 0.2449
Epoch 70, Loss: 0.2437
Epoch 80, Loss: 0.2423
Epoch 90, Loss: 0.2406
Epoch 100, Loss: 0.2385
Epoch 110, Loss: 0.2360
Epoch 120, Loss: 0.2331
Epoch 130, Loss: 0.2298
Epoch 140, Loss: 0.2261
Epoch 150, Loss: 0.2221
Epoch 160, Loss: 0.2179
Epoch 170, Loss: 0.2137
Epoch 180, Loss: 0.2097
Epoch 190, Loss: 0.2059

Final Outputs after training:
Raw outputs:
 [[0.267]
 [0.601]
 [0.601]
 [0.649]]
Rounded (Predicted XOR):
 [[0.]
 [1.]
 [1.]
 [1.]]
```