**EXPERIMENT 16**

import math

import random

def sigmoid(x):

return 1 / (1 + math.exp(-x))

def sigmoid\_derivative(x):

return x \* (1 - x)

random.seed(42)

input\_size = 2

hidden\_size = 2

output\_size = 1

learning\_rate = 0.5

w1 = [[random.uniform(-1, 1) for \_ in range(hidden\_size)] for \_ in range(input\_size)]

w2 = [random.uniform(-1, 1) for \_ in range(hidden\_size)]

b1 = [random.uniform(-1, 1) for \_ in range(hidden\_size)]

b2 = random.uniform(-1, 1)

X = [[0,0], [0,1], [1,0], [1,1]]

y = [0, 1, 1, 0]

for epoch in range(10000):

for i in range(len(X)):

# Forward pass

input\_layer = X[i]

hidden\_input = [sum(input\_layer[j] \* w1[j][k] for j in range(input\_size)) + b1[k] for k in range(hidden\_size)]

hidden\_output = [sigmoid(h) for h in hidden\_input]

final\_input = sum(hidden\_output[k] \* w2[k] for k in range(hidden\_size)) + b2

final\_output = sigmoid(final\_input)

error = y[i] - final\_output

d\_output = error \* sigmoid\_derivative(final\_output)

d\_hidden = [d\_output \* w2[k] \* sigmoid\_derivative(hidden\_output[k]) for k in range(hidden\_size)]

for k in range(hidden\_size):

w2[k] += learning\_rate \* d\_output \* hidden\_output[k]

b2 += learning\_rate \* d\_output

for j in range(input\_size):

w1[j][k] += learning\_rate \* d\_hidden[k] \* input\_layer[j]

b1[k] += learning\_rate \* d\_hidden[k]

print("Trained XOR predictions:")

for i in range(len(X)):

input\_layer = X[i]

hidden\_input = [sum(input\_layer[j] \* w1[j][k] for j in range(input\_size)) + b1[k] for k in range(hidden\_size)]

hidden\_output = [sigmoid(h) for h in hidden\_input]

final\_input = sum(hidden\_output[k] \* w2[k] for k in range(hidden\_size)) + b2

final\_output = sigmoid(final\_input)

print(f"Input: {X[i]} → Output: {round(final\_output, 3)}")

