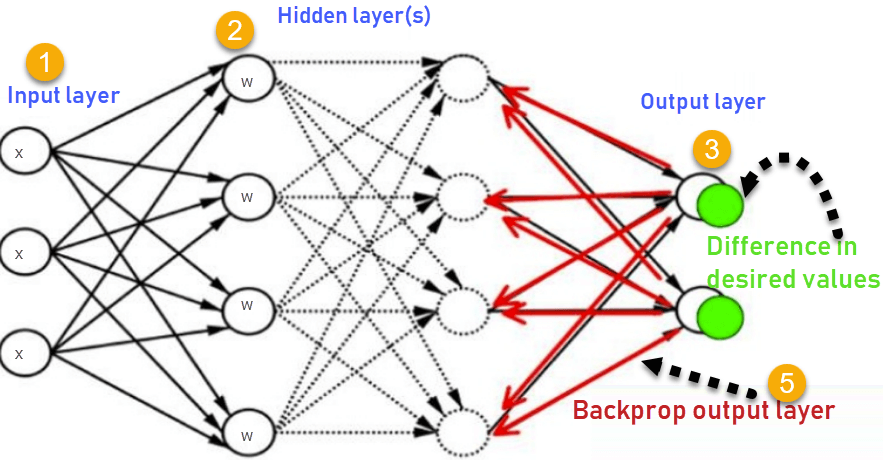
**Experiment No.: 08**

**Aim:** Write a program to implement a Backpropagation network



import numpy as np

def sigmoid(x, deriv=False):

if deriv:

return x \* (1 - x)

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

# Input data

X = np.array([[0, 0, 1],

[0, 1, 1],

[1, 0, 1],

[1, 1, 1]])

print("Input:")

print(X)

# Output labels

y = np.array([[0],

[1],

[1],

[0]])

# Seed for reproducibility

np.random.seed(1)

# Initialize weights randomly with mean 0

syn0 = 2 \* np.random.random((3, 4)) - 1

syn1 = 2 \* np.random.random((4, 1)) - 1

# Training loop

for j in range(10001):

# Forward propagation

l0 = X

l1 = sigmoid(np.dot(l0, syn0))

l2 = sigmoid(np.dot(l1, syn1))

# Calculate error

l2\_error = y - l2

if (j % 10000) == 0:

print(f"\nEpoch {j}, Error: {np.mean(np.abs(l2\_error))}")

print("Expected Output:")

print(y.flatten()) # Flatten the array to print as numbers

print("Output:")

print(np.round(l2, 3).tolist()) # Flatten the array to print as numbers

# Backpropagation

l2\_delta = l2\_error \* sigmoid(l2, deriv=True)

l1\_error = l2\_delta.dot(syn1.T)

l1\_delta = l1\_error \* sigmoid(l1, deriv=True)

# Update weights

syn1 += l1.T.dot(l2\_delta)

syn0 += l0.T.dot(l1\_delta)

**Output:**

