AIM: Write a program to implement the bidirectional associative memory network

SOURCE CODE:

# Import Python Libraries

import numpy as np

# Take two sets of patterns:

# Set A: Input Pattern

x1 = np.array([1, 1, 1, 1, 1, 1]).reshape(6, 1)

x2 = np.array([-1, -1, -1, -1, -1, -1]).reshape(6, 1)

x3 = np.array([1, 1, -1, -1, 1, 1]).reshape(6, 1)

x4 = np.array([-1, -1, 1, 1, -1, -1]).reshape(6, 1)

# Set B: Target Pattern

y1 = np.array([1, 1, 1]).reshape(3, 1)

y2 = np.array([-1, -1, -1]).reshape(3, 1)

y3 = np.array([1, -1, 1]).reshape(3, 1)

y4 = np.array([-1, 1, -1]).reshape(3, 1)

'''

print("Set A: Input Pattern, Set B: Target Pattern")

print("\nThe input for pattern 1 is")

print(x1)

print("\nThe target for pattern 1 is")

print(y1)

print("\nThe input for pattern 2 is")

print(x2)

print("\nThe target for pattern 2 is")

print(y2)

print("\nThe input for pattern 3 is")

print(x3)

print("\nThe target for pattern 3 is")

print(y3)

print("\nThe input for pattern 4 is")

print(x4)

print("\nThe target for pattern 4 is")

print(y4)

print("\n------------------------------")

'''

# Calculate weight Matrix: W

inputSet = np.concatenate((x1, x2, x3, x4), axis = 1)

targetSet = np.concatenate((y1.T, y2.T, y3.T, y4.T), axis = 0)

print("\nWeight matrix:")

weight = np.dot(inputSet, targetSet)

print(weight)

print("\n------------------------------")

# Testing Phase

# Test for Input Patterns: Set A

print("\nTesting for input patterns: Set A")

def testInputs(x, weight):

  # Multiply the input pattern with the weight matrix

  # (weight.T X x)

  y = np.dot(weight.T, x)

  y[y < 0] = -1

  y[y >= 0] = 1

  return np.array(y)

print("\nOutput of input pattern 1")

print(testInputs(x1, weight))

print("\nOutput of input pattern 2")

print(testInputs(x2, weight))

print("\nOutput of input pattern 3")

print(testInputs(x3, weight))

print("\nOutput of input pattern 4")

print(testInputs(x4, weight))

# Test for Target Patterns: Set B

print("\nTesting for target patterns: Set B")

def testTargets(y, weight):

  # Multiply the target pattern with the weight matrix

  # (weight X y)

  x = np.dot(weight, y)

  x[x <= 0] = -1

  x[x > 0] = 1

  return np.array(x)

print("\nOutput of target pattern 1")

print(testTargets(y1, weight))

print("\nOutput of target pattern 2")

print(testTargets(y2, weight))

print("\nOutput of target pattern 3")

print(testTargets(y3, weight))

print("\nOutput of target pattern 4")

print(testTargets(y4, weight))

OUTPUT:

Weight matrix:

[[4 0 4]

[4 0 4]

[0 4 0]

[0 4 0]

[4 0 4]

[4 0 4]]

------------------------------

Testing for input patterns: Set A

Output of input pattern 1

[[1]

[1]

[1]]

Output of input pattern 2

[[-1]

[-1]

[-1]]

Output of input pattern 3

[[ 1]

[-1]

[ 1]]

Output of input pattern 4

[[-1]

[ 1]

[-1]]

Testing for target patterns: Set B

Output of target pattern 1

[[1]

[1]

[1]

[1]

[1]

[1]]

Output of target pattern 2

[[-1]

[-1]

[-1]

[-1]

[-1]

[-1]]

Output of target pattern 3

[[ 1]

[ 1]

[-1]

[-1]

[ 1]

[ 1]]

Output of target pattern 4

[[-1]

[-1]

[ 1]

[ 1]

[-1]

[-1]]