Batch: A2 **Roll No.: 16010121045**

Experiment No. 05

Grade: A / AB / BB / BC / CC / CD /DD

Signature of the Staff In-charge with date

Title: To Study and implement HEBBS learning rule

Objective: To write a program to implement HEBBS learning rule for the given data.

Expected Outcome of Experiment:

CO3: Understand perceptron's and counter propagation networks

Books/ Journals/ Websites referred:

Pre Lab/ Prior Concepts:

Hebbian learning is a concept in neuroscience and artificial neural networks that describes a learning rule based on the idea that "cells that fire together wire together." In simpler terms, it suggests that the strength of a synaptic connection between two neurons should be increased if both neurons are activated simultaneously.

Here's a more detailed explanation of the Hebbian learning rule and algorithm:

Hebbian Learning Rule: The Hebbian learning rule is often summarized as follows:

"When an axon of cell A is near enough to excite cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased."

In other words, if neuron A and neuron B are both active at the same time, the synaptic connection from neuron A to neuron B should be strengthened. This rule is often expressed mathematically.

Hebbian Learning Algorithm: The Hebbian learning algorithm can be expressed mathematically as follows:

$\Delta W = \eta * (x * y)$
Where:

ΔW represents the change in synaptic weight.
η (eta) is the learning rate, controlling the magnitude of weight updates.
x represents the output (activity) of the presynaptic neuron (neuron A).
y represents the output (activity) of the postsynaptic neuron (neuron B).
The key idea is that if both x and y are positive (both neurons are active together), the weight ΔW will be positive, causing the synaptic connection to strengthen. If either x or y is negative, the weight change will be negative, potentially weakening the connection.

Biological Basis: Hebbian learning is a simplified model of how synaptic plasticity might occur in biological neural networks. It suggests that repeated co-activation of neurons can lead to physical changes in the synapses, such as the formation of new synaptic connections or the strengthening of existing ones.

Limitations and Modern Variations: While Hebbian learning captures some aspects of learning in the brain, it is overly simplistic and does not account for various important factors, such as the need for stability in learning and the potential for runaway positive feedback. Modern neural networks often use more sophisticated learning rules like backpropagation for supervised learning and variants of Hebbian learning for unsupervised learning.

Implementation Details:

```
import numpy as np

x1 = np.array([1, -2, 1.5, 0])
x2 = np.array([1, -0.5, -2, -1.5])
x3 = np.array([0, 1, -1, 1.5])

def signum(x):
    if x > 0:
        return 1
    elif x == 0:
        return 0
    else:
        return -1

def sig1(net):
    sgn = []
    for i in net:
        sgn.append(signum(i))
    sgn = np.array(sgn)
    return sgn

w1 = np.array([1, -1, 0, 0.5])
print("Input 1: ", x1)
print("Weight 1: ", w1)
net1 = np.dot(w1, x1)
```

```
print("Net 1: ", net1)

w2 = w1 + np.dot(sig1(net1), x1)
print("\nInput 2: ", x2)
print("Weight 2: ", w2)

net2 = np.dot(w2, x2)
print("Net 2: ", net2)

w3 = w2 + sig1(net2) * x2
print("\nInput 3: ", x3)
print("Weight 3: ", w3)

net3 = np.dot(w3, x3)
print("Net 3: ", net3)

w4 = w3 + sig1(net3) * x3
print("\nWeight 4: ", w4)
```

Output:

```
❯ python3 -u "/Users/pargatsinghdhanjal/Desktop/Soft Computing/exp5.py"
Input 1:  [ 1.  -2.   1.5  0. ]
Weight 1:  [ 1.  -1.   0.   0.5]
Net 1:  [1. 2. 0. 0.]
Input 2:  [ 1.  -0.5 -2.  -1.5]
Weight 2:  [ 2.  -3.   0.   0.5]
Net 2:  [ 2.    1.5  -0.   -0.75]
Input 3:  [ 0.   1.  -1.   1.5]
Weight 3:  [ 3.  -3.5  0.   2. ]
Net 3:  [ 0.  -3.5 -0.   3. ]
Weight 4:  [ 3.  -4.5  0.   3.5]
```

Conclusion: Thus, we have successfully implemented Hebbian learning algorithm of Neural Network.

Post Lab Descriptive Questions :

1. Compare the Hebbian learning and competitive learning. .

Hebbian Learning:- Information is stored in the connections between neurons in neural networks, in the form of weights. Weight change between neurons is proportional to the product of activation values for neurons. Competitive Learning:- Competitive learning is a form of unsupervised learning in artificial neural networks, in which nodes compete for the right to respond to a subset of the input data. A variant of Hebbian learning, competitive learning works by increasing the specialization of each node in the network

2. Find the weights after one iteration for hebbian learning of a single neuron network. Start with initial weights W=[1,-1] and Inputs as X1=[1,-2] X2=[2,3], x3[1,-1] and C=1.
   i. Use bipolar binary activation function
   ii. Use continuous activation function

```
Discrete Hebb Rule
Input :-  [ 1 -2]
net :-  3
Weights  [ 2 -3]

Input :-  [2 3]
net :-  -1
Weights  [-1 -4]

Input :-  [ 1 -1]
net :-  2
Weights  [ 2 -2]

Continuous Hebb Rule
Input :-  [ 1 -2]
net :-  3
Weights  [ 1.90514825 -2.81029651]

Input :-  [2 3]
net :-  -1
Weights  [ 0.07576569 -2.38635147]

Input :-  [ 1 -1]
net :-  2
Weights  [ 1.76159416 -1.76159416]
```

Date: **Signature of faculty in-charge**