Experiment No:6

**Aim:** Implementation of unsupervised learning algorithm – Hebbian Learning.

**Theory:**

**Unsupervised Learning Algorithm:**

These types of model are not provided with the correct results during the training. It can be used to cluster the input data in classes on the basis of their statistical properties only. The labelling can be carried out even if the labels are only available for a small number of objects represented of the desired classes. All similar input patterns are grouped together as clusters. If matching pattern is not found, a new cluster is formed.

In contrast to supervised learning, unsupervised or self-organized learning does not require an external teacher. During the training session, the neural network receives a number of different patterns & learns how to classify input data into appropriate categories. Unsupervised learning tends to follow the neuro-biological organization of brain. It aims to learn rapidly and can be used in real-time.

**Hebbian Learning:**

In 1949, Donald Hebb proposed one of the key ideas in biological learning, commonly known as Hebb‘s Law. Hebb‘s Law states that if neuron i is near enough is excite enough to excite neuron j & repeatedly participates in its activation, the synaptic connection between these two neurons is strengthened & neuron j becomes more sensitive to stimuli from neuron i.

Hebb‘s Law can be represented in the form of two rules:

1. If two neurons on either side of a connection are activated synchronously, then the weight of that connection is increased.
2. If two neurons on either side of a connection are activated asynchronously, then the weight of that connection is decreased.

Hebb‘s law provide basis for learning without a teacher. Learning here is a local phenomenon occurring without feedback from the environment.

* Using Hebb‘s Law we can express the adjustment applied to weight at iteration p in the following form:



* As a special case, we can represent Hebb‘s Law as follows:



Where α is the learning rate parameter.

* Hebbian learning implies that weights can only increase. To resolve this problem, we might impose a limit on the growth of synaptic weights. It can be done by introducing non-linear forgetting factor into Hebb‘s Law:



Where φ is the forgetting factor.

**Hebbian Learning Algorithm:**

Step 1: Initialization

Set initial synaptic weights and thresholds to small random values, say in an interval [0,1].

Step 2: Activation

Compute the neuron output at iteration p



Where n is number of neuron inputs, & is the threshold value of neuron j.

Step 3: Learning

Update the weights in the network



Where is the weight correction at iteration p.



Step 4:Iteration

Increase iteration p by one, go back to step 2.

**Program:**

import java.util.Scanner;

public class Hebb{

public static void main(String [] args){

Scanner k = new Scanner(System.in);

System.out.println("Enter no. of weights: ");

int n = k.nextInt();

System.out.println("Enter no. of Training step: ");

int m = k.nextInt();

double w[] = new double[n];

double cw[] = new double[n];

double x[][] = new double[m][n];

System.out.println("Enter weight Matrix ");

for(int i =0;i<n;i++){

w[i] = k.nextDouble();

}

System.out.println("Enter Training Set: ");

for(int i =0;i<m;i++){

for(int j =0; j<n;j++){

x[i][j] = k.nextDouble();

}

}

int alpha =1;

for(int j =0;j<m;j++){

double net = 0;

for(int i=0;i<n;i++)

net = net+x[j][i]\*w[i];

for(int i=0;i<n;i++){

//cw[i]=alpha\*Math.signum(net)\*x[j][i];//for binary bipolar

cw[i]=alpha\*(2/(1+Math.exp(-net)) -1)\*x[j][i]; //for binary continuous

w[i] += cw[i];

}

System.out.print("W"+(j+2)+" = [");

for(int i=0;i<n;i++){

System.out.format("%.4f ",w[i]);

}

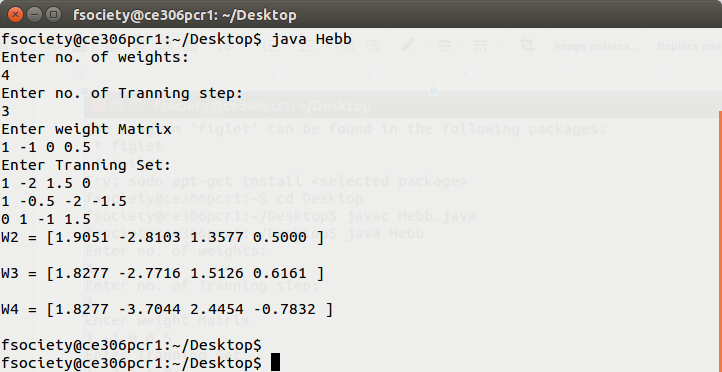
System.out.println("]\n");

}

}

}

**Output:**

****

**Conclusion:**

Unsupervised Hebbian learning algorithm is implemented which does not require supervisor. It update the weights the accordingly if error comes and train the network.