

Deerwalk Institute of Technology

School of Computer Science and Information Technology

3rd Year, 5th Semester

Neural Network

LAB SHEET - 3

Submitted By: Submitted To:

Sumit Shrestha Sarbin Sayami

Roll No: 0209

Section: A

**1) Hopfield Implementation**

public class Main {

public static void main(String [] args){

int patrn1[]= {1,0,1,0}, patrn2[] ={0,1,0,1},i;

Weight weight = new Weight();

Network network = new Network(weight.getWt1(),weight.getWt2(),weight.getWt3(),weight.getWt4());

network.activation(patrn1);

for(i=0;i<4;i++){

if(network.output[i] == patrn1[i]){

System.out.println("\n Pattern = "+patrn1[i]);

System.out.println("\n Output = "+network.output[i]);

System.out.println("Component Matches");

}

else{

System.out.println("\n Pattern = "+patrn1[i]);

System.out.println("\n Output = "+network.output[i]);

System.out.println("Discrepancy Occured");

}

}

network.activation(patrn2);

for(i=0;i<4;i++)

{ if(network.output[i]== patrn2[i]) {

System.out.println("\n Pattern = " + patrn2[i]);

System.out.println("\n Output = " + network.output[i]);

System.out.println("Component Matches");

}

else {

System.out.println("\n Pattern = "+patrn2[i]);

System.out.println("Output = "+network.output[i]);

System.out.println(" Discrepancy Occured");

}

}

}

}

import java.util.ArrayList;

public class Network{

ArrayList<Neuron> nrn = new ArrayList<Neuron>(4);

public int[] output = new int[4];

public int threshold(int k){

if(k>=0)

return (1);

else

return (0);

}

public void activation(int[] patrn){

int i,j;

for(i=0;i<4;i++)

{

for (j=0;j<4;j++)

{

System.out.printf("\nnrn[%d].weighttv[%d} is %d", i, j,(nrn.get(i).weightv[j]));

}

nrn.get(i).activation=nrn.get(i).act(4, patrn);

System.out.printf("\nactivation is %d", nrn.get(0).activation);

output[i]=threshold(nrn.get(i).activation);

System.out.printf("\noutput value is %d \n", output[i]);

}

}

public Network(int[]a, int[]b, int[] c, int[] d) {

nrn.add(new Neuron(a));

nrn.add(new Neuron(b));

nrn.add(new Neuron(c));

nrn.add(new Neuron(d));

}

}

public class Neuron {

protected int activation;

public int[] weightv = new int[4];

public Neuron(){}

public Neuron(int[] j){

int i;

for(i=0;i<4;i++)

{

weightv[i]=j[i];

}

}

int act(int m ,int[] x){

int i;

int a=0;

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

{

a+=x[i]\*weightv[i];

}

return a;

}

}

public class Weight {

private int wt1[]= {0,-3,3,-3};

private int wt2[]= {-3,0,-3,3};

private int wt3[]= {3,-3,0,-3};

private int wt4[]= {-3,3,-3,0};

public int[] getWt1(){return wt1;}

public int[] getWt2(){return wt2;}

public int[] getWt3(){return wt3;}

public int[] getWt4(){return wt4;}

}

**2) Hebbian Learning Implementation**

public class HebbianLearning {

public static void main(String[] args) {

int [] input1=new int[]{1,1,-1,-1};

int [] input2=new int[]{1,-1,1,-1};

int [] taget=new int[]{1,-1,-1,-1};

int bias=1;

double weight1=0.0,weight2=0.0,Bweight=0.0;

for(int i=0;i<input1.length;i++){

weight1=weight1+(input1[i]\*taget[i]);

weight2=weight2+(input2[i]\*taget[i]);

Bweight=Bweight+taget[i];

}

int []output=new int[input1.length];

System.out.println("Weight1 "+weight1+"\nWeight2 "+weight2+"\nBias Weight "+Bweight);

System.out.println("Input1 "+"Input2 "+"Bias "+"Output "+"Target");

for(int i=0;i<input1.length;i++){

if(input1[i]\*weight1+input2[i]\*weight2+bias\*Bweight>0){

output[i]=1;

}

else{

output[i]=-1;

}

System.out.println(input1[i]+"\t\t"+input2[i]+"\t\t"+bias+"\t\t"+output[i]+"\t\t"+taget[i]);

}

}

}

**3) HeteroAssociative Implementation.**

public class Mcculoch\_pitts {

public static void main(String[] args) {

int []input1=new int[]{0,0,1,1};

int []input2=new int[]{0,1,0,1};

double weight1=0.7,weight2=0.7;

double threshold=0.5;

int output[]= new int[input1.length];

for(int i=0;i<input1.length;i++){

if((input1[i]\*weight1+input2[i]\*weight2)>threshold){

output[i]=1;

}

else{

output[i]=0;

}

}

System.out.println("Input1 "+"Input2 "+"Output");

for(int i=0;i<output.length;i++){

System.out.println(input1[i]+"\t\t"+input2[i]+"\t\t"+output[i]);

}

}

}

**4) HeteroAssociative Implementation.**

public class HeteroAssociative {

public static void main(String[] args) {

int []pattern1=new int[]{1,-1,-1,-1};

int []pattern2=new int[]{-1,1,-1,-1};

int []pattern3=new int[]{-1,-1,1,-1};

int []pattern4=new int[]{-1,-1,-1,1};

int [][]pattern5=new int[][]{{1,-1,-1},{1,-1,1},{-1,1,-1},{-1,1,1}};

int [][]pair1=new int[4][3];

int [][]pair2=new int[4][3];

int [][]pair3=new int[4][3];

int [][]pair4=new int[4][3];

int [][]sum=new int[4][3];

int []firstPattern=new int[]{1,-1,-1,-1};

for(int i=0;i<4;i++){

for(int j=0;j<3;j++){

pair1[i][j]=pattern1[i]\*pattern5[0][j];

pair2[i][j]=pattern2[i]\*pattern5[1][j];

pair3[i][j]=pattern3[i]\*pattern5[2][j];

pair4[i][j]=pattern4[i]\*pattern5[3][j];

sum[i][j]=pair1[i][j]+pair2[i][j]+pair3[i][j]+pair4[i][j];

}

}

int []T=new int[4];

int output[]=new int[4];

for(int j=0;j<3;j++){

for(int i=0;i<4;i++){

T[j]+=firstPattern[i]\*sum[i][j];

}

if(T[j]>0){

output[j]=1;

}

else if(T[j]==0){

output[j]=0;

}

else{

output[j]=-1;

}

}

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

for(int i=0;i<4;i++){

System.out.print("\t"+firstPattern[i] + " ");

}

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

System.out.println("\tWeight Matrix\n");

for(int i=0;i<4;i++){

for(int j=0;j<3;j++){

System.out.print("\t"+sum[i][j]);

}

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

}

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

for(int j=0;j<3;j++){

System.out.print("\t"+output[j]);

}

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

}

}