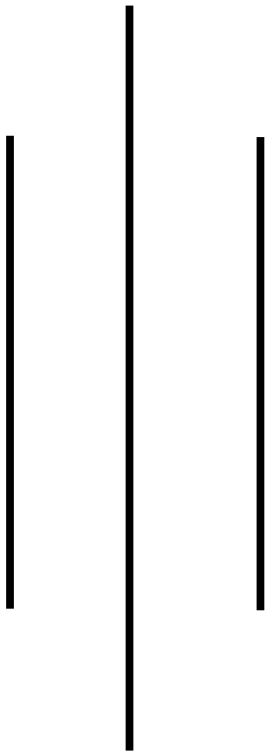
Deerwalk Institute of Technology



Neural Network

LAB SHEET - 3

Submitted By:
Sagar Giri
Roll No: 0205
Section: A

Submitted To:
Sarbin Sayami

1) Hopfield Implementation

Neuron.java

```
package com.hoppfield;

public class Neuron {
    protected int neti;
    public int[] weightv = new int[4];

    Neuron(){}
    Neuron(int wt[]){
        for (int i = 0; i < wt.length; i++) {
            weightv[i] = wt[i];
        }
    }


    protected int act(int m,int pattern1[]){
        int i;
        int a=0;
        for(i=0;i<m;i++){
            a+=pattern1[i]*weightv[i];
        }
        return a;
    }
}
```

Network.java

```
package com.hoppfield;

public class Network {
    Neuron[] nrn = new Neuron[4];
    int[] output = new int[4];

    protected Network(int[] w1, int[] w2, int[] w3, int[] w4){
        nrn[0] = new Neuron(w1);
        nrn[1] = new Neuron(w2);
        nrn[2] = new Neuron(w3);
        nrn[3] = new Neuron(w4);
    }

    private int threshold(int neti){
        return ((neti>0)?1:0);
    }
```

```
    protected void activition(int pattern[]){
        int i,j;
        for(i=0;i<4;i++) {
            for (j=0;j<4;j++) {
                System.out.println("\n nrn[" + i + "].weightv[" + j +
"] is" + nrn[i].weightv[j]);
            }
            nrn[i].neti=nrn[i].act(4,pattern);
            System.out.println("\nactivation is "+nrn[i].neti);
            output[i]=threshold(nrn[i].neti);
            System.out.println("\noutput value is "+output[i]+"\n");
        }
    }

}
```

HoppMain.java

```
package com.hoppfield;

public class HoppMain {
    public static void main(String[] args){
        int patrn1[]= {0,1,0,1},i;
        int wt1[]= {0,-10,10,-10};
        int wt2[]= {-10,0,-10,10};
        int wt3[]= {10,-10,0,-10};
        int wt4[]= {-10,10,-10,0};

        Network network = new Network(wt1,wt2,wt3,wt4);
        network.activition(patrn1);
        for(i=0;i<4;i++)
        {
            if(network.output[i]== patrn1[i])
                System.out.println("\n pattern = "+patrn1[i]+" | output =
"+network.output[i]+"   component matches");
            else
                System.out.println("\n pattern= "+patrn1[i]+" | output=
"+network.output[i]+" discrepancy occured");
        }
    }

}
```

2) McCulloch/Pitts Neuron

Mnp.java

```
package com.mnp;

public class Mnp {
    private double neti;
    private double[] weights;
    public int output;
    double sum=0;

    public Mnp(double[] wts){
        this.weights = wts;
    }
    private double act(int[] input){
        for (int i: input){
            sum+=input[i]*weights[i];
        }
        return sum;
    }

    private int threshold(double neti){
        return ((neti>=0)?1:0);
    }

    public void activation(int[] inputs) {
        neti = act(inputs);
        output=threshold(neti);
    }
}
```

MnpMain.java

```
package com.mnp;

public class MnpMain {
    public static void main(String[] args) {
        int[] inputs = {0,0,1,1};
        double[] weights = {0.5,0.5,0.75,0.25};
        Mnp mnp = new Mnp(weights);
        mnp.activation(inputs);

        if (mnp.output==1){
            System.out.println("The neuron is fired.");
        }
        else {
```

```
            System.out.println("The neuron is not fired.");
        }
    }
}
```

3.Hebbian Learning

HebbianLearning.java

```
package com.hebbian;

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 [] target=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]*target[i]);
            weight2=weight2+(input2[i]*target[i]);
            Bweight=Bweight+target[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|"+target[i]);
        }
    }
}
```