Scilab Manual for Neural Network by Dr Nadir N. Charniya Electronics Engineering VESIT[1]

Solutions provided by
Nandan Hegde
Electronics Engineering
V.E.S.I.T

April 4, 2026

[1]Funded by a grant from the National Mission on Education through ICT, http://spoken-tutorial.org/NMEICT-Intro. This Scilab Manual and Scilab codes written in it can be downloaded from the "Migrated Labs" section at the website http://scilab.in

Contents

List of Experiments

List of Figures

Experiment: 1

Generate the activation functions- Logistic,Hyperbolic,Identity that are used in Neural networks

Scilab code Solution 1.1 Activation functions in Neural network

```
1 //Illustrations of various activation fumctions used
      in Neural networks
2
3 x= -10:0.1:10;
4 tmp=exp(-x);
5 y1=1 ./(1+tmp);     //Logistic function
6 y2=(1-tmp)./(1+tmp);       //Hyperbolic Tangent
     function
7 y3=x;          //Identity function
8 subplot(2,3,1);
9 plot(x,y1);
```

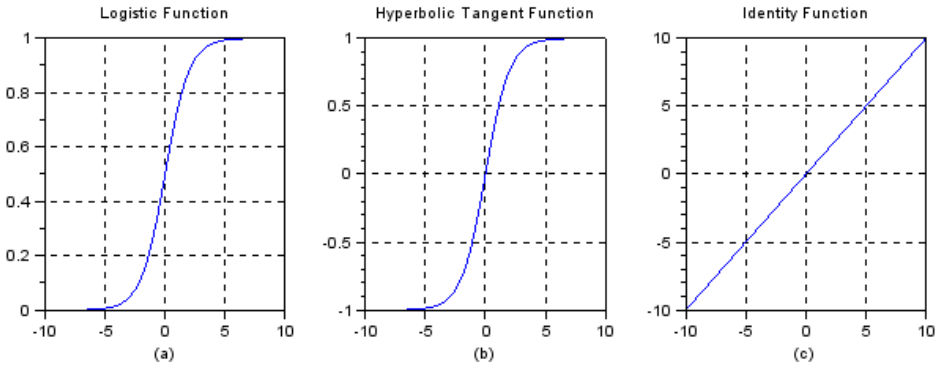

Figure 1.1: Activation functions in Neural network

```
10 set(gca(),"grid",[1 1]);
11 ([min(x) max(x) -2 2]);
12 title('Logistic Function');
13 xlabel('(a)');
14 ('square');
15 subplot(2,3,2);
16 plot(x,y2);
17 set(gca(),"grid",[1 1]);
18 ([min(x) max(x) -2 2]);
19 title('Hyperbolic Tangent Function');
20 xlabel('(b)');
21 ('square');
22 subplot(2,3,3);
23 plot(x,y3);
24 set(gca(),"grid",[1 1]);
25 ([min(x) max(x) -2 2]);
26 title('Identity Function');
27 xlabel('(c)');
28 ('square');
```

Experiment: 2

program for perceptron net for an AND function with bipolar inputs and targets

Scilab code Solution 2.1 Bipolar AND function

```
1 // Generate perceptron net for an AND function with
     bipolar inputs and targets
2 //      Truth table for AND gate
3 //       X1     X2     Y
4 //       -1     -1     -1
5 //       -1      1     -1
6 //        1     -1     -1
7 //        1      1      1            (Bipolar   (1,-1))
8
9 clc;
10 clear;
11 x=[1 1 -1 -1;1 -1 1 -1];       //input
12 t=[1 -1 -1 -1];          //target
13 w=[0 0];         //Weights
14 b=0;            //bias
15 alpha=input('Enter learning rate=');       //learning
     rate
```

```
16 theta=input('Enter threshold value=');     //
      threshold value
17 con=1;
18 epoch=0;
19 while con
20     con=0;
21     for i=1:4
22         yin=b+x(1,i)*w(1)+x(2,i)*w(2);
23         if yin>theta then
24             y=1;
25         end
26         if yin<=theta & yin>= -(theta) then
27             y=0;
28         end
29         if yin<-(theta) then
30             y=-1;
31         end
32         if y-t(i) then
33             con=1;
34             for j=1:2
35                 w(j)=w(j)+alpha*t(i)*x(j,i);
                       //upgrading of weight
36             end
37             b=b+alpha*t(i)        //upgrading of bias
38         end
39         end
40         epoch=epoch+1;
41     end
42     disp('perceptron for AND function');
43     disp('Final Weight matrix');
44     disp(w);
45     disp('Final Bias');
46     disp(b);
```

Experiment: 3

Generate Or function with bipolar inputs and targets using Adaline network

Scilab code Solution 3.1 Bipolar OR function

```
1 //Generate OR function with bipolar inputs and
     targets using Adaline network
2 //      Truth table for OR gate
3 //       X1     X2     Y
4 //       -1     -1     -1
5 //       -1      1      1
6 //        1     -1      1
7 //        1      1      1            (Bipolar   (1,-1))
8
9
10 clc;
11 clear;
12 disp('Adaline network for OR function Bipolar inputs
       and targets');
13 //input pattern
14 x1=[1 1 -1 -1];
15 x2=[1 -1 1 -1];
```

```
16 //bias pattern
17 x3=[1 1 1 1];
18 //target vector
19 t=[1 1 1 -1];
20 //initial weights and bias
21 w1=0.1;w2=0.1;b=0.1;
22 //initialize learning rate
23 alpha=0.1;
24 //error convergence
25 e=2;
26 //change in weights and bias
27 delw1=0;delw2=0;delb=0;
28 epoch=0;
29 while(e>1.018)
30     epoch=epoch+1;
31     e=0;
32     for i=1:4
33         nety(i)=w1*x1(i)+w2*x2(i)+b;
34         //net input calculated and target
35         nt=[nety(i) t(i)];
36         delw1=alpha*(t(i)-nety(i))*x1(i);
37         delw2=alpha*(t(i)-nety(i))*x2(i);
38         delb=alpha*(t(i)-nety(i))*x3(i);
39         //weight changes
40         wc=[delw1 delw2 delb]
41         //updating of weights
42         w1=w1+delw1;
43         w2=w2+delw2;
44         b=b+delb;
45         //new weights
46         w=[w1 w2 b];
47         //input pattern
48         x=[x1(i) x2(i) x3(i)];
49         //printing the results obtained
50         disp([x nt wc w]);
51     end
52     for i=1:4
53         nety(i)=w1*x1(i)+w2*x2(i)+b;
```

```
54            e=e+(t(i)-nety(i))^2;
55        end
56        end
```

Experiment: 4

Generate XOR function for bipolar inputs and targets using Madaline network

Scilab code Solution 4.1 Bipolar XOR function

```
1 //Generate XOR funcion for bipolar  inputs and
     targets using madaline network
2 //      Truth table for XOR gate
3 //       X1    X2    Y
4 //       -1    -1    -1
5 //       -1     1     1
6 //        1    -1     1
7 //        1     1    -1            (Bipolar   (1,-1))
8
9
10 clc;
11 clear;
12 x=[1 1 -1 -1;1 -1 1 -1];    //input
13 t=[-1 1 1 -1];           //target
14 //assuming initial weight matrix and bias
15 w=[0.05 0.1;0.2 0.2];
16 b1=[0.3 0.15];
```

```
v=[0.5 0.5];
b2=0.5;
con=1;
alpha=0.5;
epoch=0;
while con
    con=0;
    for i=1:4
        for j=1:2
            zin(j)=b1(j)+x(1,i)*w(1,j)+x(2,i)*w(2,j)
               ;       //neural functin output
            if zin(j)>=0 then
                z(j)=1;
            else
                z(j)=-1;
            end
        end
        yin=b2+z(1)*v(1)+z(2)*v(2);
        if yin>=0 then
            y=1;
        else
            y=-1;
        end
        if y~=t(i) then
            con=1;
            if t(i)==1 then
                if abs(zin(1))>abs(zin(2)) then
                    k=2;
                else
                    k=1;
                end
                b1(k)=b1(k)+alpha*(1-zin(k));
                   //upgrading bias
                w(1:2,k)=w(1:2,k)+alpha*(1-zin(k))*x
                   (1:2,i);      //upgrading weight
            else
                for k=1:2
                    if zin(k)>0 then
```

```
52                              b1(k)=b1(k)+alpha*(-1-zin(k)
                                   );      //upgrading bias
53                              w(1:2,k)=w(1:2,k)+alpha*(-1-
                                   zin(k))*x(1:2,i);     //
                                   upgrading weight
54                              end
55                          end
56                      end
57                                  end
58          end
59          epoch=epoch+1;
60      end
61
62  disp('Weight matrix of hidden layer');
63  disp(w);
64  disp('Bias');
65  disp(b1);
66  disp('Total epoch');
67  disp(epoch);
```

Experiment: 5

Find the weight matrix of an auto associative net to store the vector(1 1 -1 -1).Test the response by presenting same pattern.

Scilab code Solution 5.1 Auto associative net

```
1 //Find the weight matrix of an Auto associative net
    to store the vector (1 1 -1 -1).Test the respnse
    of the network by presenting the same pattern and
     recognize whether it is a known vector  or
    unknown vector
2 //Auto associative net has the same inputs and
    targets
3 clc;
4 clear;
5 x=[1 1 -1 -1];       //Given vector
6 w=zeros(4,4);
7 w=x'*x;
8 yin=x*w;
```

```
for i=1:4
    if yin(i)>0 then
        y(i)=1;
    else
         y(i)=-1;
    end
end
disp('weight matrix');
disp(w);
if x==y then
    disp('The vector is a known vector');
else
    disp('The vector is unknown vector');
end
```

Experiment: 6

Find weight matrix in Bipolar form for BAM network on binary i/p o/p pairs

Scilab code Solution 6.1 BAM network

```
1 //Find the weight  matrix in bipolar form for Bi-
    directional Associative Memory (BAM) network
    based on the following binary input output pairs
2 // s(1)=(1 1 0)        t(1)=(1 0)
3 // s(2)=(1 0 1)        t(2)=(0 1)
4
5 clc;
6 clear;
7 s=[1 1 0;1 0 1];       // s(1),s(2)
8 t=[1 0;0 1];          //t(1),t(2)
9 x=2*s-1;
10 y=2*t-1;
11 w=zeros(3,2);
12 for i=1:2
13     w=w+x(i,:)'*y(i,:);
14 end
15 disp('The calculated weight matrix');
```

```
16 disp(w);
```