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

February 12, 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 Scilab Solutions 3

1 Generate the activation functions- Logistic,Hyperbolic,Identity that are used in Neural networks 5

2 program for perceptron net for an AND function with bipolar inputs and targets 8

3 Generate Or function with bipolar inputs and targets using Adaline network 10

4 Generate XOR function for bipolar inputs and targets using Madaline network 13

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. 16

6 Find weight matrix in Bipolar form for BAM network on binary i/p o/p pairs 18

List of Experiments

Solution 1.1 Activation functions in Neural network 5
Solution 2.1 Bipolar AND function 8
Solution 3.1 Bipolar OR function 10
Solution 4.1 Bipolar XOR function 13
Solution 5.1 Auto associative net 16
Solution 6.1 BAM network 18

List of Figures

1.1 Activation functions in Neural network 6

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

```
//Illustrations of various activation fumctions used in Neural networks

x= -10:0.1:10;
tmp=exp(-x);
y1=1 ./(1+tmp);     //Logistic function
y2=(1-tmp)./(1+tmp);       //Hyperbolic Tangent function
y3=x;         //Identity function
subplot(2,3,1);
plot(x,y1);
```

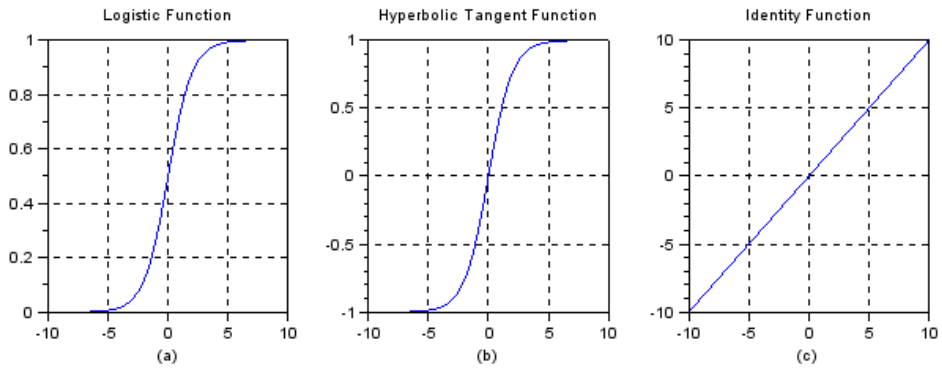

Figure 1.1: Activation functions in Neural network

```
set(gca(),"grid",[1 1]);
([min(x) max(x) -2 2]);
title('Logistic Function');
xlabel('(a)');
('square');
subplot(2,3,2);
plot(x,y2);
set(gca(),"grid",[1 1]);
([min(x) max(x) -2 2]);
title('Hyperbolic Tangent Function');
xlabel('(b)');
('square');
subplot(2,3,3);
plot(x,y3);
set(gca(),"grid",[1 1]);
([min(x) max(x) -2 2]);
title('Identity Function');
xlabel('(c)');
('square');
```

Experiment: 2

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

Scilab code Solution 2.1 Bipolar AND function

```
// Generate perceptron net for an AND function with bipolar inputs and targets
//      Truth table for AND gate
//       X1    X2    Y
//       -1    -1    -1
//       -1     1    -1
//        1    -1    -1
//        1     1     1            (Bipolar   (1,-1))

clc;
clear;
x=[1 1 -1 -1;1 -1 1 -1];      //input
t=[1 -1 -1 -1];         //target
w=[0 0];        //Weights
b=0;           //bias
alpha=input('Enter learning rate=');      //learning rate
```

```
theta=input('Enter threshold value=');    //threshold value
con=1;
epoch=0;
while con
    con=0;
    for i=1:4
        yin=b+x(1,i)*w(1)+x(2,i)*w(2);
        if yin>theta then
            y=1;
        end
        if yin<=theta & yin>= -(theta) then
            y=0;
        end
        if yin<-(theta) then
            y=-1;
        end
        if y-t(i) then
            con=1;
            for j=1:2
                w(j)=w(j)+alpha*t(i)*x(j,i);  //upgrading of weight
            end
            b=b+alpha*t(i)       //upgrading of bias
        end
        end
        epoch=epoch+1;
    end
    disp('perceptron for AND function');
    disp('Final Weight matrix');
    disp(w);
    disp('Final Bias');
    disp(b);
```

Experiment: 3

Generate Or function with bipolar inputs and targets using Adaline network

Scilab code Solution 3.1 Bipolar OR function

```
//Generate OR function with bipolar inputs and
   targets using Adaline network
//      Truth table for OR gate
//       X1    X2    Y
//       -1    -1    -1
//       -1     1     1
//        1    -1     1
//        1     1     1            (Bipolar   (1,-1))


clc;
clear;
disp('Adaline network for OR function Bipolar inputs
    and targets');
//input pattern
x1=[1 1 -1 -1];
x2=[1 -1 1 -1];
```

```
//bias pattern
x3=[1 1 1 1];
//target vector
t=[1 1 1 -1];
//initial weights and bias
w1=0.1;w2=0.1;b=0.1;
//initialize learning rate
alpha=0.1;
//error convergence
e=2;
//change in weights and bias
delw1=0;delw2=0;delb=0;
epoch=0;
while(e>1.018)
    epoch=epoch+1;
    e=0;
    for i=1:4
        nety(i)=w1*x1(i)+w2*x2(i)+b;
        //net input calculated and target
        nt=[nety(i) t(i)];
        delw1=alpha*(t(i)-nety(i))*x1(i);
        delw2=alpha*(t(i)-nety(i))*x2(i);
        delb=alpha*(t(i)-nety(i))*x3(i);
        //weight changes
        wc=[delw1 delw2 delb]
        //updating of weights
        w1=w1+delw1;
        w2=w2+delw2;
        b=b+delb;
        //new weights
        w=[w1 w2 b];
        //input pattern
        x=[x1(i) x2(i) x3(i)];
        //printing the results obtained
        disp([x nt wc w]);
    end
    for i=1:4
        nety(i)=w1*x1(i)+w2*x2(i)+b;
```

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

Experiment: 4

Generate XOR function for bipolar inputs and targets using Madaline network

Scilab code Solution 4.1 Bipolar XOR function

```
//Generate XOR funcion for bipolar  inputs and
   targets using madaline network
//      Truth table for XOR gate
//       X1    X2    Y
//       -1    -1    -1
//       -1     1     1
//        1    -1     1
//        1     1    -1            (Bipolar   (1,-1))


clc;
clear;
x=[1 1 -1 -1;1 -1 1 -1];     //input
t=[-1 1 1 -1];           //target
//assuming initial weight matrix and bias
w=[0.05 0.1;0.2 0.2];
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
```

```
                        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
                        end
                    end
                end
                            end
        end
        epoch=epoch+1;
    end

disp('Weight matrix of hidden layer');
disp(w);
disp('Bias');
disp(b1);
disp('Total epoch');
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

```
//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
//Auto associative net has the same inputs and targets
clc;
clear;
x=[1 1 -1 -1];        //Given vector
w=zeros(4,4);
w=x'*x;
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

```
//Find the weight  matrix in bipolar form for Bi-
   directional Associative Memory (BAM) network
   based on the following binary input output pairs
// s(1)=(1 1 0)        t(1)=(1 0)
// s(2)=(1 0 1)        t(2)=(0 1)

clc;
clear;
s=[1 1 0;1 0 1];       // s(1),s(2)
t=[1 0;0 1];          //t(1),t(2)
x=2*s-1;
y=2*t-1;
w=zeros(3,2);
for i=1:2
    w=w+x(i,:)'*y(i,:);
end
disp('The calculated weight matrix');
```

```
disp(w);
```