Scilab Manual for Neural Networks

by Dr G V Maha Lakshmi
Electronics Engineering
Sreenidhi Institute Of Science And Technology[1]

Solutions provided by
Dr G V Maha Lakshmi
Electronics Engineering
Sreenidhi Institute Of Science And Technology

June 3, 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 AND NOT function using McCulloch-Pitts neural net 4

2 McCulloch-Pitts Net for XOR function 7

3 Hebb Net to classify two dimensional input patterns 11

4 Hetro associative neural net 13

5 Discrete Hopfield net 15

6 Kohonen self organizing maps 17

7 Learning Vector Quantisation 19

8 Full Counter Propagation Network for given input pair 22

9 ART1 Neural Net 24

10 MLP 26

List of Experiments

Solution 1.1 1 4
Solution 2.2 2 7
Solution 3.3 3 11
Solution 4.4 4 13
Solution 5.5 5 15
Solution 6.6 6 17
Solution 7.7 7 19
Solution 8.8 8 22
Solution 9.9 9 24
Solution 10.10 10 26

Experiment: 1

Generate AND NOT function using McCulloch-Pitts neural net

Scilab code Solution 1.1 1

```
//Generate AND NOT function using McCulloch−Pitts
   neural net
//Windows 10
//Scilab 5.4.1
clear;
clc;

//Generate weights and threshold value
disp('Enter the weights');
w1=input('Weight w1=');
w2=input('Weight w2=');
disp('Enter Threshold Value');
theta=input('theta=');
y=[0 0 0 0];
x1=[0 0 1 1];
x2=[0 1 0 1];
z=[0 0 1 0];
```

```
con=1;
while con
    zin=x1*w1+x2*w2;
    for i=1:4
        if zin(i)>=theta
            y(i)=1;
        else
            y(i)=0;
        end
    end
  disp('Output of Net');
  disp(y);
  if y==z
      con=0;
  else
      disp('Net is not learning enter another set of
          weights and Threshold value');
      w1=input('weight w1=');
      w2=input('weight w2=');
      theta=input('theta=');
  end
end
disp('Mcculloch-Pitts Net for ANDNOT function');
disp('Weights of Neuron');
disp(w1);
disp(w2);
disp('Threshold value');
disp(theta);

//Truth Table
//X1  X2  Y
//0   0   0
//0   1   0
//1   0   1
//1   1   0

//Output
//Enter the weights
```

```
//Weight w1=1
//Weight w2=1
//
// Enter Threshold Value
//theta=0.1
//
// Output of Net
//
//     0.     1.     1.     1.
//
// Net is not learning enter another set of weights
   and Threshold v
//       alue
//weight w1=1
//weight w2=-1
//theta=1
//
// Output of Net
//
//     0.     0.     1.     0.
//
// Mcculloch-Pitts Net for ANDNOT function
//
// Weights of Neuron
//
//     1.
//
//  - 1.
//
// Threshold value
//
//     1.
```

Experiment: 2

McCulloch-Pitts Net for XOR function

Scilab code Solution 2.2 2

```
//McCulloch-Pitts for XOR function
//Windows 10
//Scilab 5.4.1
clear;
clc;

//Getting weights and threshold value
disp('Enter weights');
w11=input('Weight w11=');
w12=input('weight w12=');
w21=input('Weight w21=');
w22=input('weight w22=');
v1=input('weight v1=');
v2=input('weight v2=');
disp('Enter Threshold Value');
theta=input('theta=');
x1=[0 0 1 1];
x2=[0 1 0 1];
z=[0;1;1;0];
```

```
con=1;
while con
  zin1=x1*w11+x2*w21;
  zin2=x1*w21+x2*w22;
  for i=1:4
      if zin1(i)>=theta
          y1(i)=1;
      else
          y1(i)=0;
      end
      if zin2(i)>=theta
          y2(i)=1;
      else
          y2(i)=0;
      end
  end
  yin=y1*v1+y2*v2;
  for i=1:4
      if yin(i)>=theta;
          y(i)=1;
      else
          y(i)=0;
      end
  end
  disp('Output of Net');
  disp(y);
  if y == z
      con=0;
  else
      disp('Net is not learning enter another set of
          weights and Threshold value');
      w11=input('Weight w11=');
      w12=input('weight w12=');
       w21=input('Weight w21=');
      w22=input('weight w22=');
      v1=input('weight v1=');
      v2=input('weight v2=');
      theta=input('theta=');
  end
end
disp('McCulloch-Pitts Net for XOR function');
disp('Weights of Neuron Z1');
disp(w11);
disp(w21);
disp('weights of Neuron Z2');
disp(w12);
disp(w22);
disp('weights of Neuron Y');
disp(v1);
disp(v2);
disp('Threshold value');
disp(theta);

//Truth Table
//X1  X2  Y
//0   0   0
//0   1   1
//1   0   1
//1   1   0

//Output

// Enter weights
//Weight w11=1
//weight w12=-1
//Weight w21=-1
//weight w22=1
//weight v1=1
//weight v2=1
//
// Enter Threshold Value
//theta=1
//
// Output of Net
//
//     0.
//     1.
//     1.
//     0.
//
// McCulloch-Pitts Net for XOR function
//
// Weights of Neuron Z1
//
//     1.
//
//   - 1.
//
// weights of Neuron Z2
//
//   - 1.
//
//     1.
//
// weights of Neuron Y
//
//     1.
//
//     1.
//
// Threshold value
//
//     1.
```

Experiment: 3

Hebb Net to classify two dimensional input patterns

Scilab code Solution 3.3 3

```
//Hebb Net to classify two dimensional input
   patterns
//Windows 10
//Scilab 5.4.1
clear;
clc;

//Input Patterns
E=[1 1 1 1 1 -1 -1 -1 1 1 1 1 1 -1 -1 -1 1 1 1 1];
F=[1 1 1 1 1 -1 -1 -1 1 1 1 1 1 -1 -1 -1 1 -1 -1
   -1];
x(1,1:20)=E;
x(2,1:20)=F;
w(1:20)=0;
w=w'
t=[1 -1];
b=0;
for i=1:2
    w=w+x(i,1:20)*t(i);
    b=b+t(i);
end
disp('Weight matrix');
disp(w);
disp('Bias');
disp(b);

//Output
//
// Weight matrix
//
//
//          column 1 to 18
//
//    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.
      0.    0.    0.    0.    0.//    0.    0.    2.
//
//          column 19 to 20
//
//    2.    2.
//
// Bias
//
//    0.
```

Experiment: 4

Hetro associative neural net

Scilab code Solution 4.4 4

```
//Hetro associative neural net
//Windows 10
//Scilab 5.4.1
clear;
clc;

x=[1 1 0 0;1 0 1 0;1 1 1 0;0 1 1 0];
t=[1 0;1 0;0 1;0 1];
w=zeros(4,2);
for i=1:4
    w=w+x(i,1:4)'*t(i,1:2);
end
disp('Weight matrix');
disp(w);


//Auotassociative net to store the vector

x=[1 1 -1 -1];
xv=[1;1;-1;-1];
w=zeros(4,4);
```

```
w=x'*x;
yin=x*w;
for i=1:4
    if yin(i)>0
        y(i)=1;
    else
        y(i)=-1;
    end
end
disp('Weight matrix');
disp(w);
if xv==y
    disp('The vector is a Known Vector');
else
    disp('The vector is an Unknown Vector');
end

//Output
// Weight matrix
//
//    2.    1.
//    1.    2.
//    1.    2.
//    0.    0.
//
// Weight matrix
//
//    1.    1.   -1.   -1.
//    1.    1.   -1.   -1.
//   -1.   -1.    1.    1.
//   -1.   -1.    1.    1.
//
// The vector is a Known Vector
```

Experiment: 5

Discrete Hopfield net

Scilab code Solution 5.5 5

```
//Discrete Hopfield net
//Windows 10
//Scilab 5.4.1
clear;
clc;

x=[1 1 1 0];
tx=[0 0 1 0];
w1=(2*x'-1);
w2=(2*x-1);
w=w1*w2;

for i=1:4
    w(i,i)=0;
end
con=1;
y=[0 0 1 0];
while con
    up=[4 2 1 3];
    for i=1:4
        yin(up(i))=tx(up(i))+y*w(1:4,up(i));
        if yin(up(i))>0
            y(up(i))=1;
        end
    end
    if y==x
        disp('Convergence has been obtained');
        disp('The Converged Ouput');
        disp(y);
        con=0;
    end
end

//Output
// Convergence has been obtained
//
// The Converged Ouput
//
//    1.    1.    1.    0.
```

Experiment: 6

Kohonen self organizing maps

Scilab code Solution 6.6 6

```
//Kohonen self organizing maps
//Windows 10
//Scilab 5.4.1
clear;
clc;

x=[1 1 0 0;0 0 0 1;1 0 0 0;0 0 1 1];
alpha=0.6;

//initial weight matrix
w=rand(4,2);
disp('Initial weight matrix');
disp(w);
con=1;
epoch=0;
while con
    for i=1:4
        for j=1:2
            D(j)=0;
            for k=1:4
                D(j)=D(j)+(w(k,j)-x(i,k))^2;
            end
        end
        for j=1:2
            if D(j)==min(D)
                J=j;
            end
        end
        w(:,J)=w(:,J)+alpha*(x(i,:)'-w(:,J));
    end
    alpha=0.5*alpha;
    epoch=epoch+1;
    if epoch==300
        con=0;
    end
end
disp('Weight Matrix after 300 epoch');
disp(w);

//Output
// Initial weight matrix
//
//    0.2113249    0.6653811
//    0.7560439    0.6283918
//    0.0002211    0.8497452
//    0.3303271    0.685731
//
// Weight Matrix after 300 epoch
//
//    0.9671633    0.0277033
//    0.4283588    0.0261632
//    0.0000092    0.5968633
//    0.0137532    0.9869153
```

Experiment: 7

Learning Vector Quantisation

Scilab code Solution 7.7 7

```
//Learning Vector Quantization
//Windows 10
//Scilab 5.4.1
clear;
clc;

s=[1 1 0 0;0 0 0 1;0 0 1 1;1 0 0 0;0 1 1 0];
st=[1 2 2 1 2];
alpha=0.6;

//initial weight matrix first two vectors of input
   patterns
w=[s(1,:);s(2,:)]';
disp('Initial weight matrix');
disp(w);

//set remaining as input vector
x=[s(3,:);s(4,:);s(5,:)];
t=[st(3);st(4);st(5)];
con=1;
epoch=0;
```

```
while con
    for i=1:3
        for j=1:2
            D(j)=0;
            for k=1:4
                D(j)=D(j)+(w(k,j)-x(i,k))^2;
            end
        end
        for j=1:2
            if D(j)==min(D)
                J=j;
            end
        end
        if J==t(i)
            w(:,J)=w(:,J)+alpha*(x(i,:)'-w(:,J));
        else
            w(:,J)=w(:,J)-alpha*(x(i,:)'-w(:,J));
        end
        end
    alpha=0.5*alpha;
    epoch=epoch+1;
    if epoch==100
        con=0;
    end
end
disp('Weight Matrix after 100 epochs');
disp(w);

//Output
// Initial weight matrix
//
//    1.    0.
//    1.    0.
//    0.    0.
//    0.    1.
//
// Weight Matrix after 100 epochs
//
```

```
//      1.              0.
//      0.2040471       0.561484
//      0.              0.9583648
//      0.              0.438516
```

Experiment: 8

Full Counter Propagation Network for given input pair

Scilab code Solution 8.8 8

```
//Full counter propagation network for given input
   pair
//Windows 10
//Scilab 5.4.1
clear;
clc;

//set initial weights
 v=[0.6 0.2;0.6 0.2;0.2 0.6; 0.2 0.6];
 w=[0.4 0.3;0.4 0.3];
 x=[0 1 1 0];
 y=[1 0];
 alpha=0.3;
 for j=1:2
     D(j)=0;
     for i=1:4
         D(j)=D(j)+(x(i)-v(i,j))^2;
     end
     for k=1:2
```

```
        D(j)=D(j)+(y(k)-w(k,j))^2;
     end
 end
 for j=1:2
     if D(j)==min(D)
         J=j;
     end
 end
 disp('After one step the weight matrix are');
 v(:,J)=v(:,J)+alpha*(x'-v(:,J))
 w(:,J)=w(:,J)+alpha*(y'-w(:,J))
 disp('v')
 disp(v)
 disp('w')
 disp(w)

//Output
//
// After one step the weight matrix are
//
// v
//
//    0.42    0.2
//    0.72    0.2
//    0.44    0.6
//    0.14    0.6
//
// w
//
//    0.58    0.3
//    0.28    0.3
```

Experiment: 9

ART1 Neural Net

Scilab code Solution 9.9 9

```
//ART1 Neural Net
//Windows 10
//Scilab 5.4.1
clear;
clc;

b=[0.57 0.0 0.3;0.0 0.0 0.3;0.0 0.57 0.3;0.0 0.47
   0.3];
t=[1 1 0 0;1 0 0 1;1 1 1 1];
vp=0.4;
L=2;
x=[1 0 1 1];
s=x;
ns=sum(s);
y=x*b;
con=1;
while con
    for i=1:3
        if y(i)==max(y)
            J=i;
        end
    end
    x=s.*t(J,:);
    nx=sum(x);
    if nx/ns >= vp
        b(:,J)=L*x(:)/(L-1+nx);
        t(J,:)=x(1,:);
        con=0;
    else
        y(J)=-1;
        con=1;
    end
    if y+1==0
        con=0;
    end
end
disp('Top Down Weights');
disp(t);
disp('Bottom up Weights');
disp(b);

//Output
// Top Down Weights
//
//    1.    1.    0.    0.
//    1.    0.    0.    1.
//    1.    1.    1.    1.
//
// Bottom up Weights
//
//    0.57    0.6666667    0.3
//    0.      0.           0.3
//    0.      0.           0.3
//    0.      0.6666667    0.3
```

Experiment: 10

MLP

Scilab code Solution 10.10 10

```
//MLP Algorithm and implementation
//Windows 10
//Scilab 5.4.1
clear;
clc;

deff('y=f(x)','y=1/(1+exp(-x))')
Wih=[0.1,-0.3;0.3,0.4];
Who=[0.4;0.5]
i=[0.2,0.6];
t=0.7;
a=10;
for k=1:3
    printf('\n\n\nAfter Iteration %i :\n\n',k)
    disp(Wih,'Wih = ')
    disp(Who,'Who = ')
a1=i*Wih;
disp(a1,'a = ')
h=[f(a1(1)),f(a1(2))]
disp(h,'h = ')
b1=h*Who
```

```
disp(b1,'b1 =')
o=f(b1)
disp(o,'o = ')
d=o*(1-o)*(t-o)
disp(d,'d =')
for j=1:2
    e(1,j)=h(j)*(1-h(j))*d*Who(j)
end
disp(e,'e =')
dWho=a*h'*d;
disp(dWho,'dWho =')
Who=Who+dWho;
dWih=a*i'*e;
disp(dWih,'dWih =')
Wih=Wih+dWih;
end

//Output
//After Iteration 1 :
//
//
// Wih =
//
//    0.1   -0.3
//    0.3    0.4
//
// Who =
//
//    0.4
//    0.5
//
// a =
//
//    0.2    0.18
//
// h =
//
//    0.549834    0.5448789
```

```
//
// b1 =
//
//    0.492373
//
// o =
//
//    0.6206653
//
// d =
//
//    0.0186786
//
// e =
//
//    0.0018493    0.002316
//
// dWho =
//
//    0.102701
//    0.1017755
//
// dWih =
//
//    0.0036986    0.004632
//    0.0110958    0.0138961
//
//
//
//After Iteration 2 :
//
//
// Wih =
//
//    0.1036986   -0.295368
//    0.3110958    0.4138961
//
// Who =
//
//     0.502701
//     0.6017755
//
// a =
//
//     0.2073972    0.189264
//
// h =
//
//     0.5516642    0.5471753
//
// b1 =
//
//     0.6065989
//
// o =
//
//     0.6471646
//
// d =
//
//     0.0120646
//
// e =
//
//     0.0015    0.0017989
//
// dWho =
//
//     0.066556
//     0.0660144
//
// dWih =
//
//     0.0030001    0.0035978
//     0.0090002    0.0107933
//
```

```
//
//
//After Iteration 3 :
//
//
// Wih =
//
//    0.1066987   -0.2917702
//    0.320096     0.4246894
//
// Who =
//
//    0.569257
//    0.6677899
//
// a =
//
//    0.2133973    0.1964596
//
// h =
//
//    0.5531478    0.5489575
//
// b1 =
//
//    0.6814715
//
// o =
//
//    0.6640671
//
// d =
//
//    0.008016
//
// e =
//
//    0.0011279    0.0013254
```

```
//
// dWho =
//
//    0.0443403
//    0.0440044
//
// dWih =
//
//    0.0022558    0.0026508
//    0.0067674    0.0079525
```