INDEX

Soft Computing Techniques

Sr. No	Experiment Name	Date	Page No	Remark
1	Write a program to perform Union, Intersection and Complement operation.	23/04/22		
2	Write a program to implement De-Morgan's Law.	23/04/22		
3	Write a program to plot various membership functions.	25/04/22		
4	Write a program to generate ANDNOT function using McCulloch-Pitts neural net.	25/04/22		
5	Write a program to calculate the weights for the following patterns using hetero associative neural net for mapping four input vectors to two output vectors.	30/04/22		
6	Generate XOR function using McCulloch-Pitts neural net by Scilab program.	30/04/22		
7	Write a Scilab program for Perceptron net for an AND function with bipolar inputs and targets.	23/06/22		
8	Write a Scilab program to store vector [-1 -1 -1 -1] and [-1 -1 1 1] in an auto associative net. Find weight matrix. Test the net with [1 1 1 1] as input.	23/06/22		
9	Write a Scilab Program To plot the surface z=2x!+y! (elliptic paraboloid).	20/06/22		
10	Write a Scilab program to store the vector (1 1 1-1).Find the weight matrix with no Self-connection. Test this using a discrete Hopfield net with mistakes in first and second component of stored vector i.e (0 0 1 0). Also the given pattern in binary form is [1 1 1 0].	20/06/22		

1. Write a program to perform Union, Intersection and Complement operation.

```
// Enter Data
 u=input(''Enter First Matrix');
 v=input(''Enter Second Matrix');
// To Perform Operations
w=max(u,v);
p=min(u,v);
q1=1-u;
q2=1-v;
// Display Output
disp(Union Of Two Matrices');
disp(w);
disp('Intersection Of Two Matrices');
disp(p);
disp('Complement Of First Matrix');
disp(q1);
disp('Complement Of Second Matrix');
disp(q2);
```

Output:

```
Enter First Matrix [0.3 0.4]
Enter Second Matrix [0.1 0.7]
Union of Two Matrices
w=0.3000   0.7000
Intersection of Two Matrices
p=0.1000   0.4000
Complement of First Matrix
q1=0.7000   0.6000
Complement of Second Matrix
q2= 0.9000   0.3000
```

2. Write a program to implement De-Morgan's Law.

```
// Enter Data
u-input('Enter First Matrix');
v=input('Enter Second Matrix');
//To Perform Operations
w=max(u,v);
p=min(u,v):
q1=1-u;
q2=1-v;
x1=1-w;
x2=min(q1,q2);
y1=1-p;
y2=max(q1.42):
// Display Output
disp('Union Of Two Matrices');
disp(w):
disp('Intersection Of Two Matrices');
disp(p):
disp(Complement Of First Matrix'');
disp(q1);
disp('Complement Of Second Matrix');
disp(q2);
disp(De-Morgans Law');
disp(‘LHS');
disp(x1);
disp('RHS');
disp(x2);
disp(‘LHS1’);
disp(y2);
disp(‘RHS1’);
disp(y1);
```

Output:

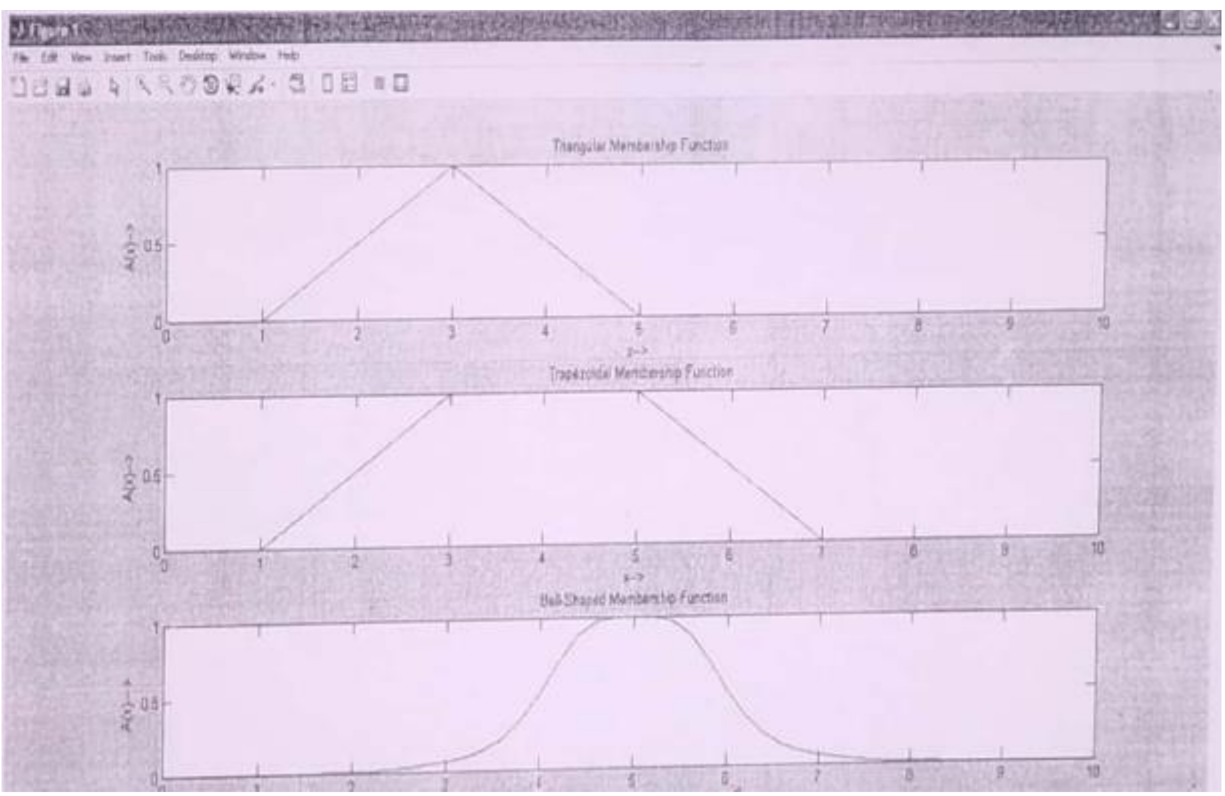
```
Enter First Matrix [0.3.0.4]
Enter Second Matrix [0.2 0.5]
Union Of Two Matrices
w=0.3000   0.5000
Intersection Of Two Matrices
p=0.2000   0.4000
Complement Of First Matrix
q1= 0.7000   0.6000
Complement Of Second Matrix
q2= 0.8000   0.5000
```

De-Morgans Law
LHS
x1= 0.7000   0.5000
RHS
x2=0.7000   0.5000
LHS1
yl= 0.8000   0.6000
RHS1
y2= 0.8000   0.6000

3. Write a program to plot various membership functions.

```
//Triangular Membership Function
x=(0.0:1.0:10.0):
yl=trim(x, [135]):
subplot(311);
plot(x,[y1]);
// Trapezoidal Membership Function
x=(0.0:1.0:10.0);
y1=trapmf(x, [1 3 5 7]);
subplot(312)
// Bell-Shaped Membership Function
x=(0.0:0.2:10.0);
yl=gbellmf(x, [1 2 5]);
subplot(313);
plot(x,[y1]);
```

Output:



4. Write a program to generate ANDNOT function using McCulloch-Pitts neural net.

```
//ANDNOT function using McCulloch-Pitts neuron
clear;
clc;
//Getting weights and threshold value
disp("Enter the weights');
wl=input('Weight wl=');
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];
con=1;
while con
zin= x1 * wl+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');
wl=input("Weight w1='),
w2=input('Weight w2=');
thete=input('theta='),
end
end
disp('McCulloch Pitts Net for ANDNOT function');
disp('Weights of neuron');
disp(w1);
disp(w2):
disp('Threshold value=');
disp(theta);
```

Output:

Enter the weights
Weight w1=1
Weight w2=1
Enter threshold value
theta 1
Output of net=0   1   1   1
Net is not learning Enter another set of weights and threshold value
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

5. Write a program to calculate the weights for the following patterns using hetero associative neural net for mapping four input vectors to two output vectors.

S1	S2	S3	S4	t1	t2
1	1	0	0	1	0
1	0	1	0	1	0
1	1	1	0	0	1
0	1	1	0	0	1

```
//Hetero-associative neural net for mapping input vectors to output vectors.
 clear;
clc:
x=[1 1 0 0;1 0 1 0;1 1 1 0;0 1 1 0];
t=[ 1 0;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);
```

Output:

```
Weight Matrix

  2    1

 1    2

 1    2

 0    0
```

6. Generate XOR function using McCulloch-Pitts neural net by Scilab program.

```
//XOR function using McCulloch-Pitts neuron
clear;
clc;
// Getting weights and threshold value
disp(''Enter the weights');
w11=input('Weight w11=');
w12=input('Weight w12=');
w21=input('Weight w21=');
w22=input(''Weight w22=');
vl=input('Weight v1=');
v2=input(Weight v2=');
disp('Enter threshold value');
x1=[0 0 1 1];
x2=[0 1 0 1];
z=[0 1 1 0];
con=1;
while con
zinl=x1* w11+x2 *w21;
zin2=x1* w21+x2 *w22;
for i=1:4
if zin1(i)>= theta
yl(i)=1;
else yl(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='),
vl=input('Weight v1=');
v2=input('Weight v2=’);
theta= input(''theta='');
end
end
disp('McCulloch Pitts Net for XOR function');
disp('Weights of neuron ZI');
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);
```

Output

```
Enter the 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.
0.

Net is not learning Enter another set of weights and threshold value
```

7. Write a Scilab program for Perceptron net for an AND function with bipolar inputs and targets.

```
//Perceptron for AND Function
clear;
clc;
x=[1  1  -1  -1;1  -1  1  -1];
t=[1  -1  -1  -1];
w=[0  0];
 b= 0;
alpha= input('Enter Learning rate=');
theta =input('Enter 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
y=1;
end
if yin<= theta & yin>= -theta
y= 0;
end
if yin<-theta
y= -1;
end
if y-t(i)
con=1;
for j =1:2
w(j)= w(j)+ alpha *t(i)*x(j,i);
end
b= b+alpha*t(i)
end
end
epoch=epoch+1;
end
disp('Perceptron for AND Function');
disp('Final Weight Matrix');
disp(w);
disp('Final Bias');
disp(b);
```

Output:

```
Enter Learning rate=1
Enter Threshold Value= 0.5
Perceptron for AND Function
Final Weight Matrix
  1.    1.
  Final Bias
-1
```

8. Write a Scilab program to store vector [-1 -1 -1 -1] and [-1 -1 1 1] in an auto associative net. Find weight matrix. Test the net with [1 1 1 1] as input.

```
//Auto-association problem
clc;
clear;
x=[-1 -1 -1 -1;-1 -1 1 1];
t=[1  1  1  1 ];
w =zeros(4,4);
for i=1:2
w= w+x(i, 1:4)'*x(i, 1:4);
 end
yin=t*w;
for i=1:4
 if yin(i)>0
 y(i)-1;
else
y(i)=-1;
end
end
disp("The calculated Weight Matrix');
disp(w);
 if x(1,1:4)==y(1:4)| x(2,1:4)==y(1:4)
disp('The Vector is a Known vector');
else
disp('The Vector is a Unknown vector');
end
```

Output

The calculated Weight Matrix

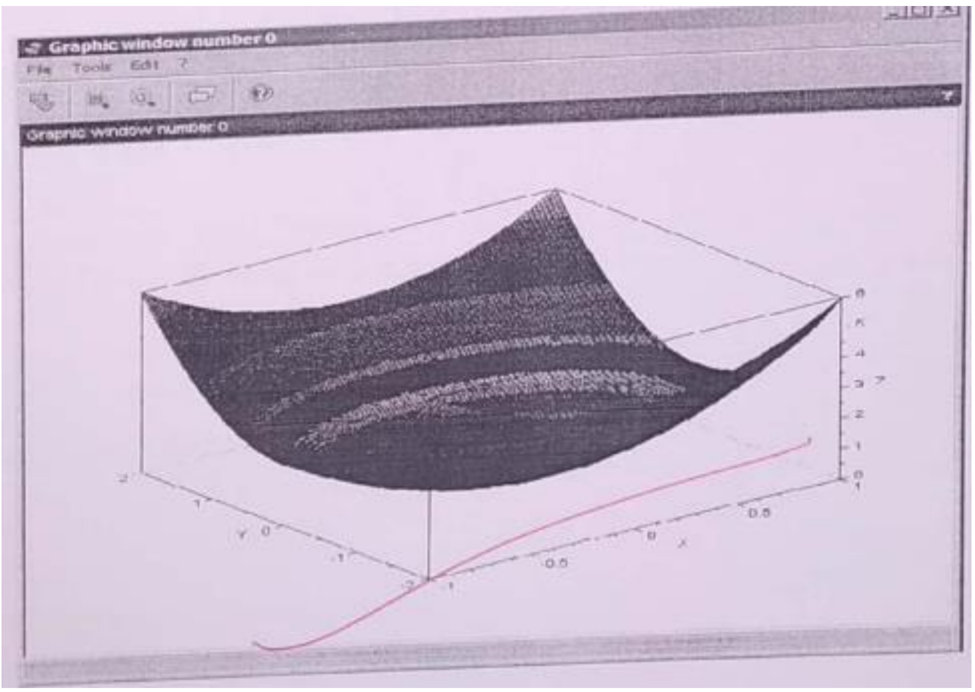
```
2.   2.   0.   0.
2.   2.   0.   0.
0.   0.   2.   2.
0.   0.   2.   2.
```

The Vector is a UnKnown vector

9. Write a Scilab Program To plot the surface z=2x!+y! (elliptic paraboloid).

```
function z=f(x,y)
z=2*x^2+y^2;
end function
 x=linspace(-1,1,100)
y=linspace(-2,2,200);
z=feval(x,y,f);
clf
surf(x,y,z)
```

Output



10. Write a Scilab program to store the vector (1 1 1-1).Find the weight matrix with no Self-connection. Test this using a discrete Hopfield net with mistakes in first and second component of stored vector i.e (0 0 1 0). Also the given pattern in binary form is [1 1 1 0].

```
//Discrete Hopfield Net
clc;
clear:
x=[1 1 1 0];
 tx=[0 0 1 0];
w=(2*x’-1)*(2*x-1);
for i=1:4
 w(1,1)=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  y(up(i)) > 0
y(up(i))=1;
end
end
if  y==x
disp('Convergence has been obtained');
disp(''The Converged Output');
disp(y);
con= 0;
end
end
```

Output:

```
up =  4  2  1   3

Convergence has been obtained

 The Converged Output

   1  1  1
```