# NRI INSTITUTE OF INFORMATION SCIENCE & TECHNOLOGY BHOPAL

## DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

# LAB MANUAL

# SOFT COMPUTING
# (CS 801)

# BACHELOR OF ENGINEERING (B.E.)

| NIIST BHOPAL | | NRI INSTITUTE OF INFORMATION SCIENCE & TECHNOLOGY<br>DEPT NAME: Computer Science & Engineering | FORM NO | NIIST/A/10 |
|---|---|---|---|---|
| BRANCH | CSE | LIST OF EXPERIMENT | REV. NO | 0 |
| SEMESTER | VIII | | REV. DT | 30/06/2011 |
| SUBJECT/CODE :- SOFT COMPUTING /CS 801 | | | | |

| S. NO. | LIST OF EXPERIMENT |
|---|---|
| 1 | Write a program in MATLAB to perform Union, Intersection and Complement operations. |
| 2 | Write a program in MATLAB to implement De-Morgan's Law. |
| 3 | Write a program in MATLAB to plot various membership functions. |
| 4 | To implement FIS Editor. |
| 5 | Generate ANDNOT function using McCulloch-Pitts neural net by MATLAB program. |
| 6 | Write a MATLAB program for Perceptron net for an AND function with bipolar inputs and targets |
| 7 | Write a M-file to calculate the weights for the following patterns using hetero-associative neural net for mapping four input vectors to two output vectors. |
| 8 | Write an M-file 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. |

**Subject Name: Soft Computing (Practical)**

**Subject Code: CS-801**

**Course Outcomes:**

CO1. Students will be able to implementation of various search algorithms.

CO2. Student will be able to implement various control strategies.

CO3. Students will be able to implement neural network concepts using MATLAB.

CO4. Students will be able to understand fuzzy logic and their implementation and Study of various evolutionary technologies.

CO5. Student will be able to implement genetic algorithms for different modelling.

## EXPERIMENT NO. 1

**AIM-:** Write a program in MATLAB to perform Union, Intersection and Complement operations.

## INTRODUCTION

It is important to recognize that fuzzy sets are different from simple probabilities. This concept tends to get lost because we use tools from statistics and probability along with fuzzy sets. The operations that we use to manipulate fuzzy sets have little in common with probability theory. Fuzzy logic is a subset of predicate logic, and the operators for fuzzy sets and for non-fuzzy sets are much the same. When you apply fuzzy operators to non-fuzzy sets, you get the sames answers as if you had used the corresponding operators from predicate logic.

Logic operators are based on truth values that range from 0 (false) to 1 (true). When these logic operators are applied to non-fuzzy values, they give the same results as the familiar operators AND, OR, and NOT. A further direct parallel with conventional logic is the use of three basic set operations - intersection, union, and complement.

**Fuzzy intersection**
Consider the intersection of two fuzzy sets. An intersection element that has a degree of membership in Set A and a different degree of membership in Set B has the lesser value as its membership in the intersection set. For example, a day that is 55 per cent warm and 75 per cent humid has only 55 percent membership in the set warm_and_humid days. This is an application of f-AND.

**Fuzzy union**
For a union of fuzzy sets, the greater of the two degrees of membership is the degree of membership in the union set. The warm and humid day from above has a 75 percent membership in warm_or_humid days, and this applies f-OR.

**Fuzzy complement**
Degree of membership in a complement set is 1 minus the degree of membership in the original set. A day that is 55 percent warm is 1 minus .55 or 45 percent cold if we are consistent in our definitions for warm and cold days. There is no difference from the NOT of predicate logic and thus no need for a f-NOT.

Since fuzzy logic is a subset of predicate logic, reasoning is the same. As would be expected, probability is often needed to clarify fuzzy, uncertain information. A degree of truth is the probability that a statement is true. Even with precise information, real-world problems may include uncertainty. There is a range of values over which real specimens fail, and a bridge builder will overdesign or use safety factors to make collapse highly unlikely. Unreliable data magnify the chances for erroneous conclusions, and fuzzy algorithms commonly estimate the level of confidence of the results.

## ALGORITHM

1. Enter the first matrix “u”
2. Enter the second matrix “v”
3. W= max(u,v)
4. P = min (u, v).
5. Q1= 1-u
6. Q2=1-v
7. Print “union of two matrices”.
8. Print “Intersection of two matrices”
9. Print “complement of first matrix”
10. Print “ complement of second matrix”

11. Exit

**PROGRAM**

```
%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

display('Union Of  Two Matrices');

display(w);

display('Intersection Of  Two Matrices');

display(p);

display('Complement Of  First Matrix');

display(q1);

display('Complement Of  Second Matrix');
display (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

## VIVA QUESTIONS

1. What is fuzzy logic?
2. What do you mean by fuzzy operators?
3. Explain fuzzy union with example?
4. Explain fuzzy intersection with example?
5. Explain fuzzy complement with example?

## EXPERIMENT NO.2

**AIM-:** Write a program in MATLAB to implement De-Morgan's Law.

## INTRODUCTION

Fuzzy logic includes 0 and 1 as extreme cases of truth (or "the state of matters" or "fact") but also includes the various states of truth in between so that, for example, the result of a comparison between two things could be not "tall" or "short" but ".38 of tallness."

Fuzzy logic seems closer to the way our brains work. We aggregate data and form a number of partial truths which we aggregate further into higher truths which in turn, when certain thresholds are exceeded, cause certain further results such as motor reaction. A similar kind of process is used in artificial computer neural network and expert systems.

It may help to see fuzzy logic as the way reasoning really works and binary or Boolean logic is simply a special case of it.

Let union represent "or", intersection represent "and", and ^' represent "not." Then, for two logical units E and F,

(E union F)^'=E^' intersection F^'
(E intersection F)^'=E^' union F^'.

These laws also apply in the more general context of Boolean algebra and, in particular, in the Boolean algebra of set theory, in which case union would denote union, intersection intersection, and ^' complementation with respect to any superset of E and F.

## ALGORITHM

De-Morgan's Law c(i(u,v)) = max(c(u),c(v))
c(u(u,v)) = min(c(u),c(v))

1. Enter the first matrix "u"
2. Enter the second matrix"v"
3. W= max(u,v)
4. P= min(u,v)
5. Q1=1-u
6. Q2=1-v
7. X1=1-w
8. X2=min(Q1,Q2)
9. Y1=1-P
10. Y2=max(Q1,Q2)
11. Print "union of two matrices"
12. Print "intersection of two matrices"
13. Apply De-morgans law to the above result.
14. Exit.

## PROGRAM

De-Morgan's Law c(i(u,v)) = max(c(u),c(v))
c(u(u,v)) = min(c(u),c(v))

```
%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,q2);

%Display Output

display('Union Of  Two Matrices');

display(w);

display('Intersection  Of  Two Matrices');

display(p);

display('Complement Of  First Matrix');

display(q1);

display('Complement Of  Second Matrix');

display(q2);
display('De-Morgans Law');

display('LHS');

display(x1);

display('RHS');

display(x2);

display('LHS1');
```

```
display(y1);

display('RHS1');

display(y2);
```

**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
 y1 =0.8000    0.6000

RHS1
y2 = 0.8000    0.6000
```

**VIVA QUESTIONS**

1. What do you mean by fuzzy set and crisp set?
2. Define De-morgans law with example?
3. What are the various applications of fuzzy logic?
4. Define associativity rule in fuzzy logic?

## EXPERIMENT NO.3

**AIM-:** Write a program in MATLAB to plot various membership functions.

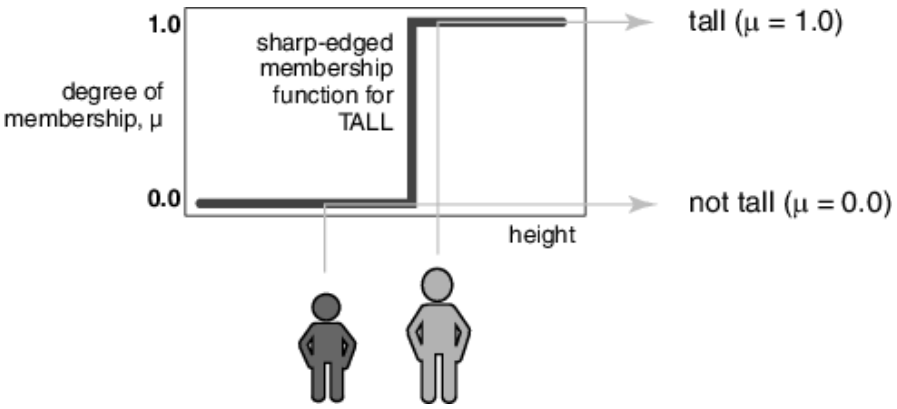
## INTRODUCTION

A membership function for a fuzzy set A on the universe of discourse X is defined as $\mu_A: X \rightarrow [0,1]$, where each element of X is mapped to a value between 0 and 1. This value, called membership value or degree of membership, quantifies the grade of membership of the element in X to the fuzzy set A.

Membership functions allow us to graphically represent a fuzzy set. The *x* axis represents the universe of discourse, whereas the *y* axis represents the degrees of membership in the [0,1] interval.

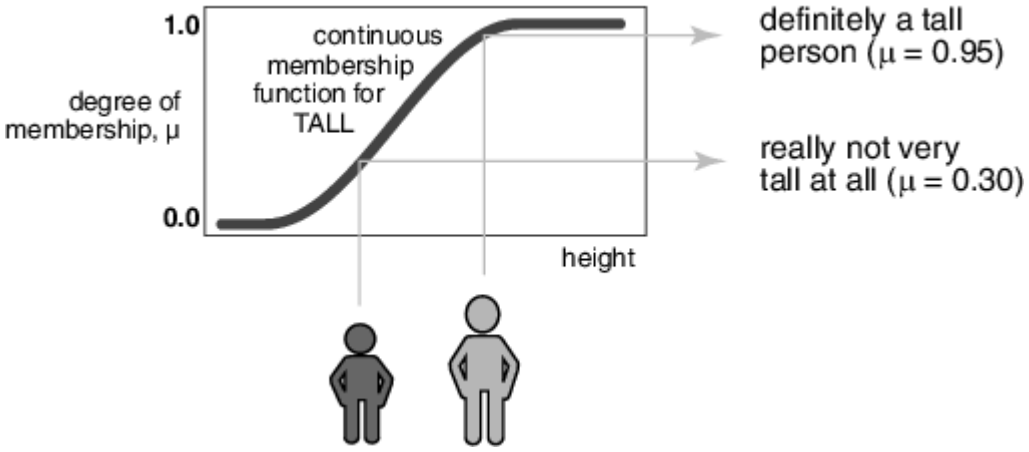
Simple functions are used to build membership functions. Because we are defining fuzzy concepts, using more complex functions does not add more precision.

In mathematics a set, by definition, is a collection of things that belong to some definition. Any item either belongs to that set or does not belong to that set. Let us look at another example; the set of tall men. We shall say that people taller than or equal to 6 feet are tall. This set can be represented graphically as follows:




The function shown above describes the membership of the 'tall' set, you are either in it or you are not in it. This sharp edged membership functions works nicely for binary operations and mathematics, but it does not work as nicely in describing the real world. The membership function makes no distinction between somebody who is 6'1" and someone who is 7'1", they are both simply tall. Clearly there is a significant difference between the two heights. The other side of this lack of disctinction is the difference between a 5'11" and 6' man. This is only a difference of one inch however this membership function just says one is tall and the other is not tall.

The fuzzy set approach to the set of tall men provides a much better representation of the tallness of a person. The set, shown below, is defined by a continuously




incliningfunction.

The membership function defines the fuzzy set for the possible values underneath of it on the horizontal axis. The vertical axis, on a scale of 0 to 1, provides the membership value of the height in the fuzzy set. So for the two people shown above the first person has a membership of 0.3 and so is not very tall. The second person has a membership of 0.95 and so he is definitely tall. He does not, however, belong to the set of tall men in the way that bivalent sets work; he has a high degree of membership in the fuzzy set of tall men.

## ALGORITHM

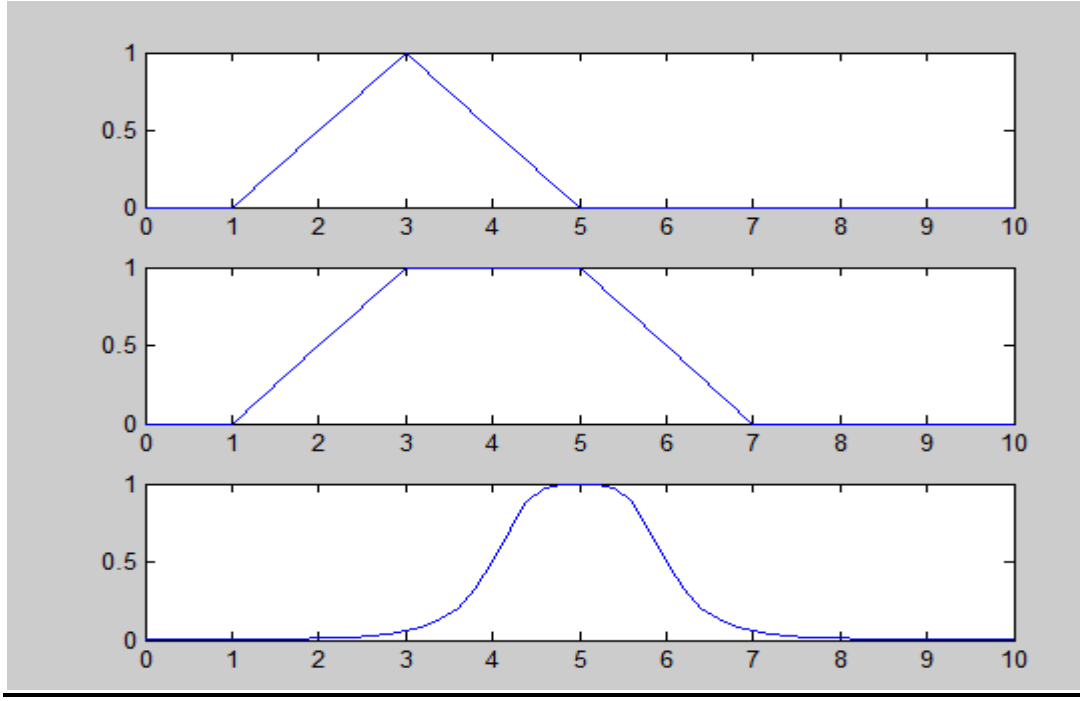
1. Enter the value of x
2. y1= trimf(x,[1357]) for triangular membership function
3. y1= trapmf(x,[1357]) for trapezodial membership function
4. y1= gbellmf(x,[1357]) for bell shaped membership function
5. Plot the various membership functions.
6. Exit

## PROGRAM

```
%Triangular Membership Function

x=(0.0:1.0:10.0)';

y1=trimf(x, [1 3 5]);

subplot(311)

plot(x,[y1]);

%Trapezoidal Membership Function

x=(0.0:1.0:10.0)';
y1=trapmf(x, [1 3 5 7]);

subplot(312)
```

```
plot(x,[y1]);

%Bell-Shaped Membership Function

x=(0.0:0.2:10.0)';

y1=gbellmf(x, [1 2 5]);

subplot(313)

plot(x,[y1]);
```

**OUTPUT**




**VIVA QUESTIONS**

1. Define membership functions with example?
2. Define trapezodial membership function with example?
3. What is triangular membership function?
4. Define bell shaped membership function with example?

## EXPERIMENT NO.4

**AIM-:** To implement FIS Editor.

## INTRODUCTION

The syntax for fuzzy inference system is as follows:

fuzzyLogicDesigner
fuzzyLogicDesigner(fis)

An example of a Mamdani inference system is shown in Figure given below. To compute the output of this FIS given the inputs, one must go through six steps:

1. Determining a set of fuzzy rules

2. Fuzzifying the inputs using the input membership functions,

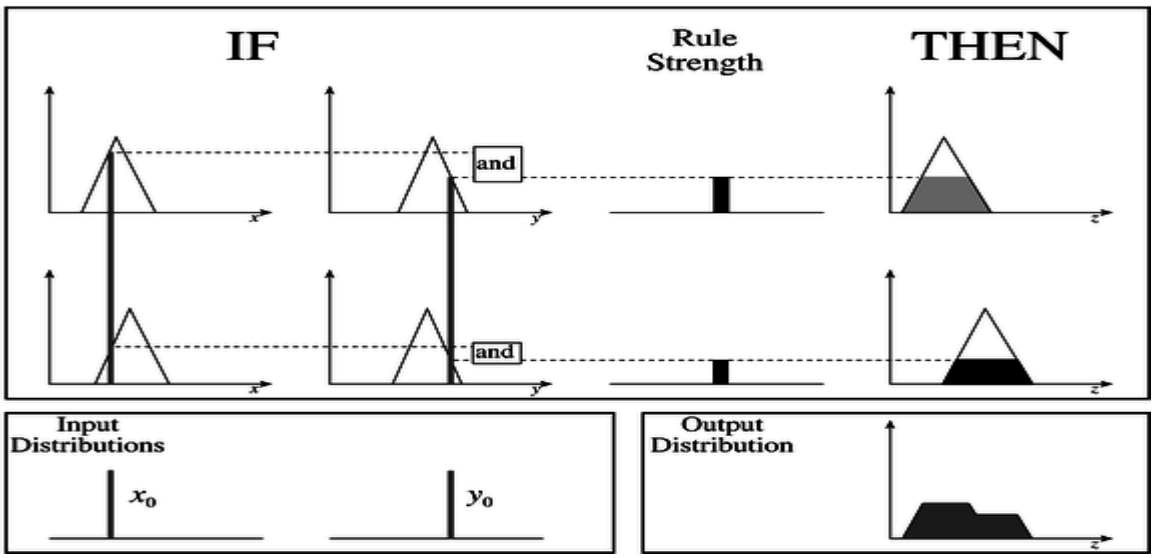
3. Combining the fuzzified inputs according to the fuzzy rules to establish a rule strength,

4. Finding the consequence of the rule by combining the rule strength and the output membership function,

5. Combining the consequences to get an output distribution, and

6. Defuzzifying the output distribution (this step is only if a crisp output (class) is needed).

The following is a more detailed description of this process.




A two input, two rule Mamdani FIS with crisp inputs

## PROGRAM

FIS stands for Fuzzy Inference System.In FIS fuzzy rules are used for approximate reasoning.It is the logical framework that allows us to design reasoning systems based on fuzzy set theory.

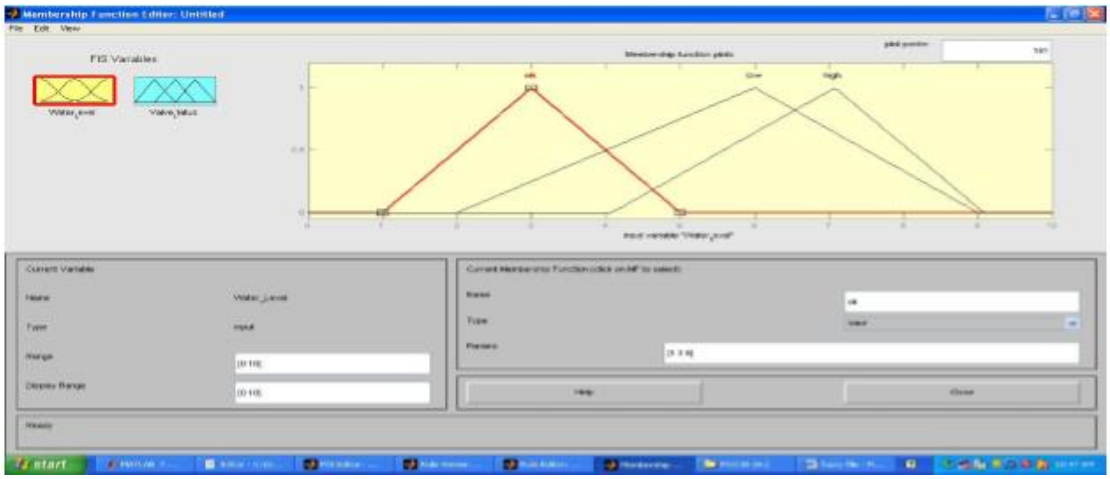
To illustrate these concepts we use example of Water Tank:-

FIS editor consists of following units:-

i) Input
ii) Inference System
iii) Output

The Water Level is considered as the Input variable and Valve status is taken as Output Variable.

The Input-Output Variable's Membership functions should be plotted along with their ranges:-



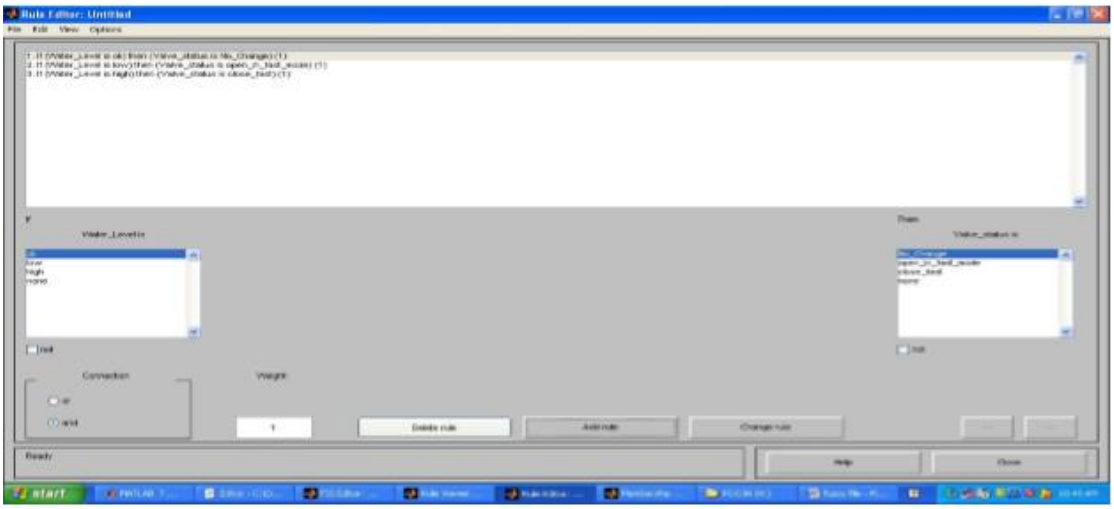
The following screen appearance is obtained by clicking on the FIS Rule system indicator:-
Rules are added by selecting variable's values and clicking on add rule menu each time a new rule is added.

The fuzzy Rules defined for water tank are:-

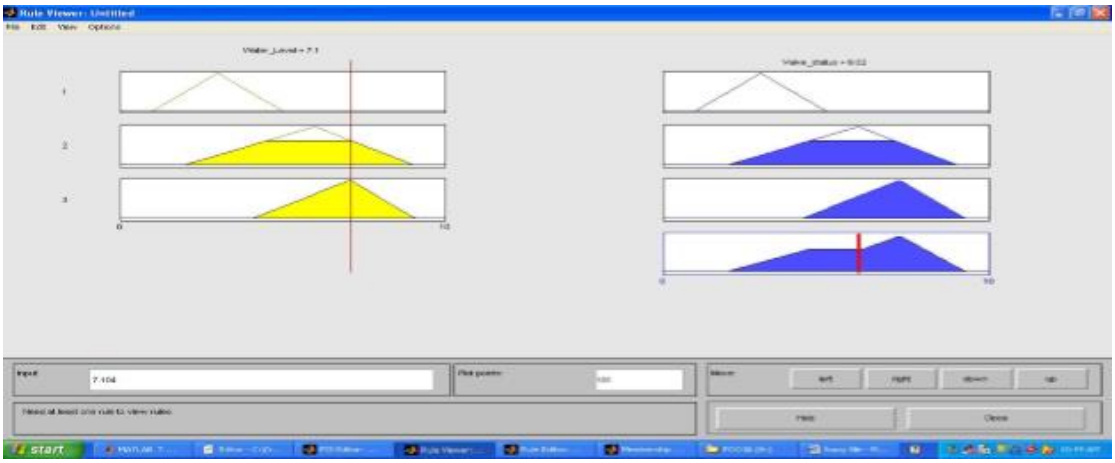
IF level is ok,THEN there is no change in valve.

IF level is low,THEN valve is open in fast mode.

IF level is high,THEN valve is closed in fast mode.



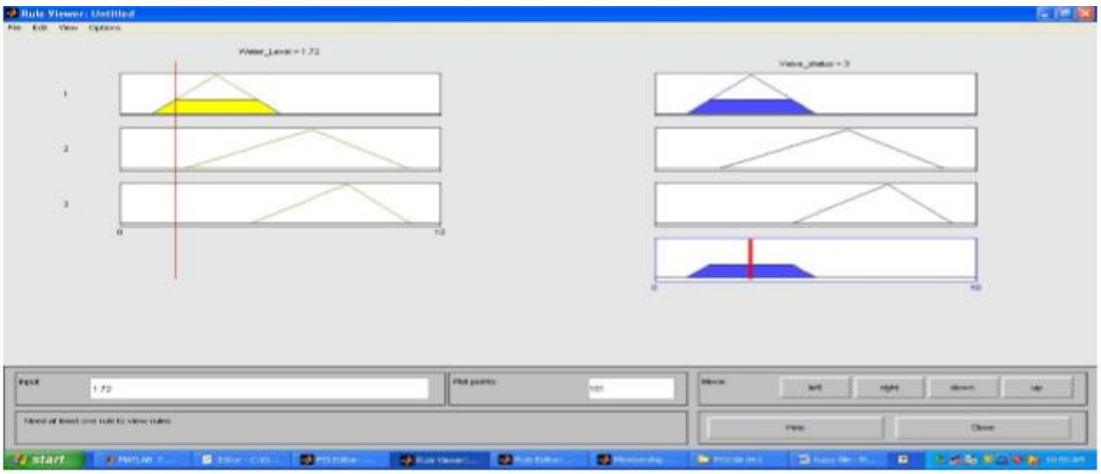
The result is displayed as plots of input-output membership functions:

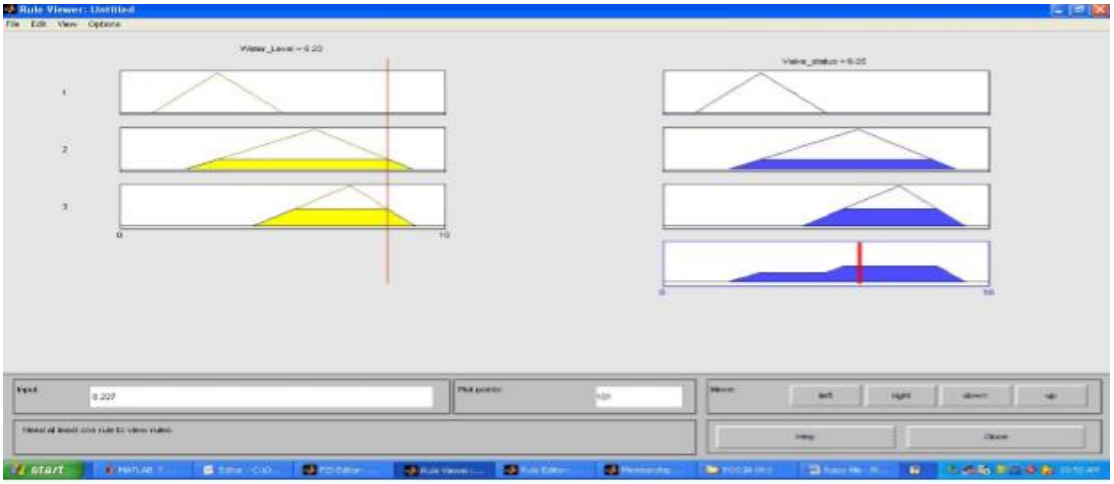


Water Level(ok,low,high)

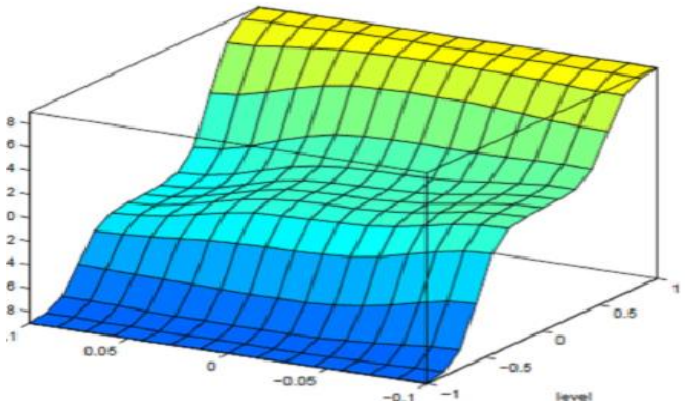
Valve Status(no change,open fast,closed fast)

The output in accordance with the input and rules provided by user is shown as (view-rule viewer):-





**OUTPUT**




**VIVA QUESTIONS**

1. Define fuzzy inference system with example?
2. What do you mean by fuzzification?
3. Define defuzzification process?

## EXPERIMENT NO. 5

**AIM:**Generate ANDNOT function using McCulloch-Pitts neural net by MATLAB program

## INTRODUCTION

McCulloch and Pitts tried to understand how the brain could produce highly complex patterns by using many basic cells that are connected together. These basic brain cells are called neurons, and McCulloch and Pitts gave a highly simplified model of a neuron in their paper. The McCulloch and Pitts model of a neuron, which we will call an MCP neuron for short, has been very important in computer science.

A group of MCP neurons that are connected together is called an artificial neural network. In a sense, the brain is a very large neural network. It has billions of neurons, and each neuron is connected to thousands of other neurons. McCulloch and Pitts showed how to encode any logical proposition by an appropriate network of MCP neurons. And so in theory anything that can be done with a computer can also be done with a network of MCP neurons. McCulloch and Pitts also showed that every network of MCP neurons encodes some logical proposition. So if the brain were a neural network, then it would encode some complicated computer program. But the MCP neuron is not a real neuron; it's only a highly simplified model. We must be very careful in drawing conclusions about real neurons based on properties of MCP neurons.

A study of the McCullough Pitts neuron is enlightening for several reasons. An artificial neuron is a mathematical function conceived as a model of biological neurons. Artificial neurons are the constitutive units in an artificial neural network. Depending on the specific model used they may be called a semi-linear unit, Nv neuron, binary neuron, linear threshold function, or McCulloch–Pitts (MCP) neuron.

The artificial neuron receives one or more inputs (representing dendrites) and sums them to produce an output (representing a neuron's axon). Usually the sums of each node are weighted, and the sum is passed through a non-linear function known as an activation function or transfer function. The transfer functions usually have a sigmoid shape, but they may also take the form of other non-linear functions, piecewise linear functions, or step functions. They are also often monotonically increasing, continuous, differentiable and bounded.

The artificial neuron transfer function should not be confused with a linear system's transfer function.

## ALGORITHM

1. Enter the weights w1.
2. Enter the weights w2.
3. Enter the threshold value “theta”.
4. While con=1, then zin= x1*w1+x2*w2.
5. If zin(i)>=theta, then y(i)=1 else y(i)=0.
6. Print “output of net”.
7. Exit.

## PROGRAM

```
%ANDNOT function using McCulloch-Pitts neuron

clear;
```

```
clc;
% Getting 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=');

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
```

**VIVA QUESTIONS**

1. Define neural network with example?
2. What is neuron?
3. Differentiate between artifical neural network and biological neural network?
4. Define McCulloch-Pitts neural net with example?

## EXPERIMENT NO. 6

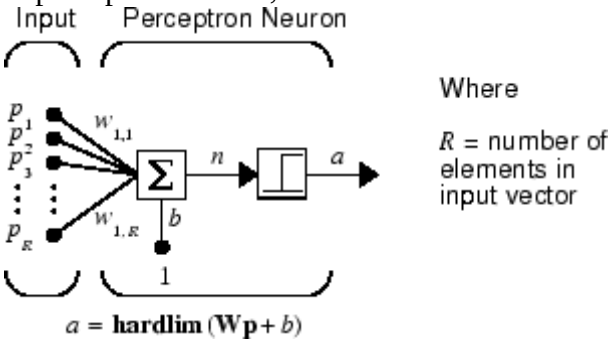
**AIM-:** Write a MATLAB program for Perceptron net for an AND function with bipolar inputs and targets.

## INTRODUCTION

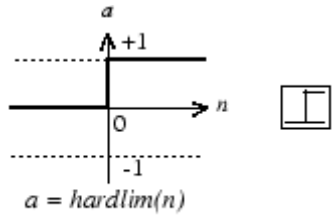
One of the simplest was a single-layer network whose weights and biases could be trained to produce a correct target vector when presented with the corresponding input vector. The training technique used is called the perceptron learning rule. The perceptron generated great interest due to its ability to generalize from its training vectors and learn from initially randomly distributed connections. Perceptrons are especially suited for simple problems in pattern classification. They are fast and reliable networks for the problems they can solve. In addition, an understanding of the operations of the perceptron provides a good basis for understanding more complex networks.

## NEURON MODEL

A perceptron neuron, which uses the hard-limit transfer function hardlim, is shown below.




Each external input is weighted with an appropriate weight $w_{1j}$, and the sum of the weighted inputs is sent to the hard-limit transfer function, which also has an input of 1 transmitted to it through the bias. The hard-limit transfer function, which returns a 0 or a 1, is shown below.




Hard-Limit Transfer Function

The perceptron neuron produces a 1 if the net input into the transfer function is equal to or greater than 0; otherwise it produces a 0.

The hard-limit transfer function gives a perceptron the ability to classify input vectors by dividing the input space into two regions. Specifically, outputs will be 0 if the net input *n* is less than 0, or 1 if the net input *n* is 0 or greater. The following figure show the input space of a two-input hard limit neuron with the weights $w_{1,1} = -1$, $w_{1,2} = 1$ and a bias $b = 1$.

## ALGORITHM

1. Enter the learning rate "alpha".
2. Enter the threshold value "theta".
3. yin=b+x(1,i)*w(1)+x(2,i)*w(2), where b=0
4. If yin>theta, then y=1.
5. If yin<=theta & yin>=-theta, then y=0.
6. If yin<-theta, then y=-1;
7. Exit

**PROGRAM**

```
% 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
```

**VIVA QUESTIONS**

1. Define perceptron with example?
2. What do you mean by threshold value?
3. Differentiate between ADALINE and MADALINE?
4. Explain perceptron learning algorithm?

## EXPERIMENT NO. 7

**AIM-:** Write a M-file 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 |

## INTRODUCTION

Autoassociative memory, also known as auto-association memory or an autoassociation network is often misunderstood to be only a form of backpropagation or other neural networks. It is actually a more generic term that refers to all memories that enable one to retrieve a piece of data from only a tiny sample of itself.

Traditional memory stores data at a unique address and can recall the data upon presentation of the complete unique address. Autoassociative memories are capable of retrieving a piece of data upon presentation of only partial information from that piece of data. Heteroassociative memories, on the other hand, can recall an associated piece of datum from one category upon presentation of data from another category. Hopfield networks have been shown to act as autoassociative memory since they are capable of remembering data by observing a portion of that data. Bidirectional Associative Memories (BAM) is Artificial Neural Networks that have long been used for performing heteroassociative recall.

An efficient neural network computing technique capable of synthesizing two sets of output signal data from a single input signal data set. The method and device of the invention involves a unique integration of autoassociative and heteroassociative neural network mappings, the autoassociative neural network mapping enabling a quality metric for assessing the generalization or prediction accuracy of the heteroassociative neural network mapping.

## ALGORITHM

1. Enter the four input vectors.
2. Enter the two ouput vectors.
3. Specify the number of zeros in input vector “w”.
4. For i=1:4, w=w+x (i,1:4)'*t(i,1:2).
5. Print weight matrix “w”.
6. Exit.

## PROGRAM

```
% 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;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
```

**VIVA QUESTIONS**

1. Define hetro-associative neural network with example?
2. What is associative nural network?
3. What are the applications of hetro-associative neural network?
4. What are the limitations of hetro-associative neural network?

## EXPERIMENT NO. 8

**AIM-:** Write an M-file 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.

## INTRODUCTION

Traditional memory stores data at a unique address and can recall the data upon presentation of the complete unique address. Autoassociative memories are capable of retrieving a piece of data upon presentation of only partial information from that piece of data. Heteroassociative memories, on the other hand, can recall an associated piece of datum from one category upon presentation of data from another category. Hopfield networks have been shown to act as autoassociative memory since they are capable of remembering data by observing a portion of that data. Bidirectional Associative Memories (BAM) is artificial Neural Networks that have long been used for performing heteroassociative recall.

The autoassociative neural network is also known as feedback network.Feedback networks can have signals travelling in both directions by introducing loops in the network. Feedback networks are very powerful and can get extremely complicated. Feedback networks are dynamic; their 'state' is changing continuously until they reach an equilibrium point. They remain at the equilibrium point until the input changes and a new equilibrium needs to be found. Feedback architectures are also referred to as interactive or recurrent, although the latter term is often used to denote feedback connections in single-layer organisations.

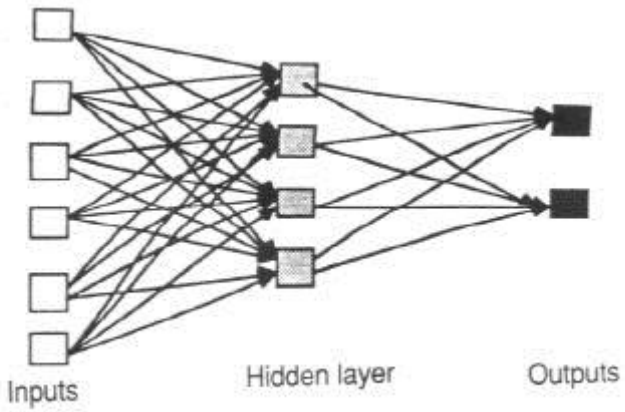



Figure 4.1 An example of a simple feedforward network

## ALGORITHM

1. Enter the two input vectors “x”
2. Enter the one ouput vector “t”
3. Specify the number of zeros in input vector “w”.
4. For i=1:2, w=w+x (i,1:4)'*t(i,1:4).
5. Print weight matrix “w.
6. yin=t*w
7. If yin(i)>0 then y(i)=1 else y(i)=-1
8. Print the calculated weight matrix.

**PROGRAM**

```
% 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 an UnKnown vector

## VIVA QUESTIONS

1. Define auto associative net with example?
2. Define feed forward network with example?
3. What is neural network?
4. Why do we use neural network?