

# ***SOFT COMPUTING***

ETCS-456

Faculty Name: Mr. Sandeep Tayal

Name: Prince

Roll No: 10314802720

Semester: 8th

Group: 8C6



Maharaja Agrasen Institute of Technology, PSP Area,  
Sector – 22, Rohini, New Delhi – 110085

## SOFT COMPUTING LAB

### PRACTICAL RECORD

**PAPER CODE : ETCS-456**

Name of the student : Prince

University Roll No. 10314802720

Branch : CSE

Section/ Group : 8C6

### PRACTICAL DETAILS

a) Experiments according to the list provided by GGSIPU

Exp. no	Experiment Name	Date of performance	Date of checking	Remarks	Marks (10)
1.	Implementation of Fuzzy Operations				
2.	Implementation of Fuzzy Relations (Max-min Composition)				
3.	Implementation of Fuzzy Controller (Washing Machine)				
4.	Implementation of Simple Neural Network (McCullohPitts model)				
5.	Implementation of Perceptron Learning Algorithm				
6.	Implementation of Unsupervised Learning Algorithm				

[illegible]



## MAHARAJA AGRASEN INSTITUTE OF TECHNOLOGY

### VISION

To nurture young minds in a learning environment of high academic value and imbibe spiritual and ethical values with technological and management competence.

### MISSION

The Institute shall endeavour to incorporate the following basic missions in the teaching methodology:

- ❖ **Engineering Hardware – Software Symbiosis:** Practical exercises in all Engineering and Management disciplines shall be carried out by Hardware equipment as well as the related software enabling a deeper understanding of basic concepts and encouraging inquisitive nature.
- ❖ **Life-Long Learning:** The Institute strives to match technological advancements and encourage students to keep updating their knowledge for enhancing their skills and inculcating their habit of continuous learning
- ❖ **Liberalization and Globalization:** The Institute endeavors to enhance technical and management skills of students so that they are intellectually capable and competent professionals with Industrial Aptitude to face the challenges of globalization.
- ❖ **Diversification:** The Engineering, Technology and Management disciplines have diverse fields of studies with different attributes. The aim is to create a synergy of the above attributes by encouraging analytical thinking.
- ❖ **Digitization of Learning Processes:** The Institute provides seamless opportunities for innovative learning in all Engineering and Management disciplines through digitization of learning processes using analysis, synthesis, simulation, graphics, tutorials and related tools to create a platform for multi-disciplinary approach.
- ❖ **Entrepreneurship:** The Institute strives to develop potential Engineers and Managers by enhancing their skills and research capabilities so that they emerge as successful entrepreneurs and responsible citizens.



**MAHARAJA AGRASEN INSTITUTE OF TECHNOLOGY**

**COMPUTER SCIENCE & ENGINEERING DEPARTMENT**

**VISION**

To Produce “Critical thinkers of Innovative Technology”

**MISSION**

To provide an **excellent learning environment** across the computer science discipline to inculcate professional behaviour, strong ethical values, innovative **research capabilities** and leadership abilities which enable them to become successful **entrepreneurs** in this globalized world.

- ❖ To nurture an excellent learning environment that helps students to enhance their problem solving skills and to prepare students to be lifelong learners by offering a solid theoretical foundation with applied computing experiences and educating them about their professional, and ethical responsibilities.
- ❖ To establish Industry-Institute Interaction, making students ready for the industrial environment and be successful in their professional lives.
- ❖ To promote research activities in the emerging areas of technology convergence.
- ❖ To build engineers who can look into technical aspects of an engineering solution thereby setting a ground for producing successful entrepreneur

# EXPERIMENT-1

**Aim: Implementation of Fuzzy Operations.**

## Objectives:

- Understanding of the basic mathematical elements of fuzzy sets.
- Analyze concepts of fuzzy set.
- To use fuzzy set operations to implement current computing techniques used in fuzzy computing.

## Theory:

### Fuzzy Logic:

Fuzzy logic is an organized method for dealing with imprecise data. It is a multivalued logic that allows intermediate values to be defined between conventional solutions.

In classical set theory, the membership of elements in a set is assessed in binary terms according to a bivalent condition — an element either belongs or does not belong to the set. By contrast, fuzzy set theory permits the gradual assessment of the membership of elements in a set; this is described with the aid of a membership function valued in the real unit interval  $[0, 1]$ .

Bivalent Set Theory can be somewhat limiting if we wish to describe a 'humanistic' Illustration mathematically. For Illustration, Fig 1 below illustrates bivalent sets to characterize the temperature of a room. The most obvious limiting feature of bivalent sets that can be seen clearly from the diagram is that they are mutually exclusive - it is not possible to have membership of more than one set. Clearly, it is not accurate to define a transition from a quantity such as 'warm' to 'hot' by the application of one degree Fahrenheit of heat. In the real world a smooth (unnoticeable) drift from warm to hot would occur.

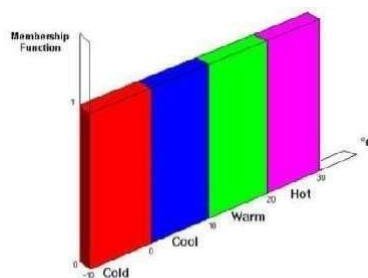


Fig. 1 : Bivalent Sets to Characterize the Temp. of a room.

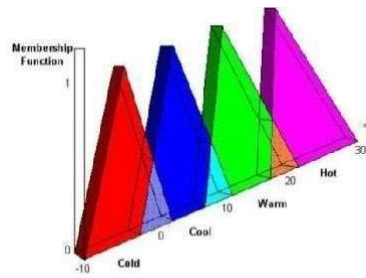


Fig. 2 - Fuzzy Sets to characterize the Temp. of a room.

## Fuzzy Sets:

A fuzzy set is a pair  $(U, m)$  where  $U$  is a set and  $m: U \rightarrow [0, 1]$ .

For each  $x \in U$ , the value  $m(x)$  is called the degree of membership of  $x$  in

Let  $x \in U$ . Then,

- $x$  is called not included in the fuzzy set  $(U, m)$  if  $m(x) = 0$
- $x$  is called fully included if  $m(x) = 1$
- $x$  is called a fuzzy member if  $0 < m(x) < 1$ .

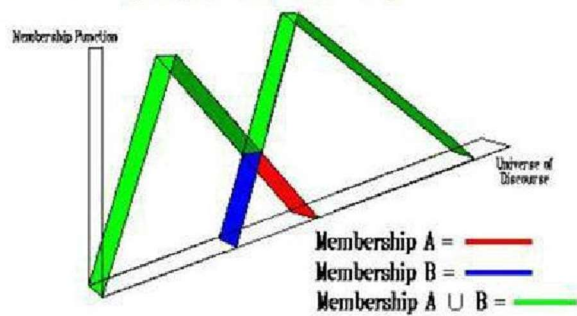
### Illustration:

$$\tilde{A} = \{(x_1, 0.1), (x_2, 0.7), (x_3, 1), (x_4, 0)\}$$

$$\tilde{B} = \{(x_1, 0.4), (x_2, 0.3), (x_3, 1), (x_4, 0.2)\}$$

#### 1. Union:

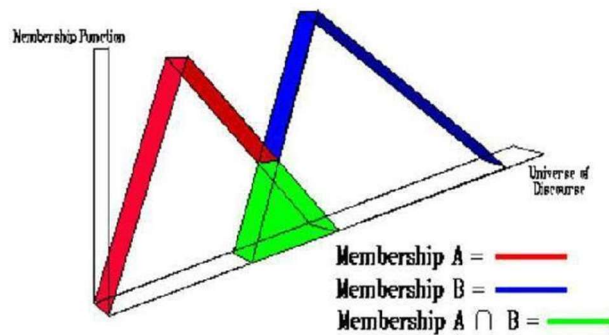
Union of two fuzzy sets  $\tilde{A}$  and  $\tilde{B}$  is denoted as  $\tilde{A} \cup \tilde{B}$  and is defined as,

$$\mu_{A \cup B}(x) = \max(\mu_A(x), \mu_B(x))$$


$$\tilde{A} \cup \tilde{B} = \{(x_1, 0.4), (x_2, 0.7), (x_3, 1), (x_4, 0.2)\}$$

#### 2. Intersection

$$\mu_{A \cap B}(x) = \min(\mu_A(x), \mu_B(x))$$



$$\tilde{A} \cap \tilde{B} = \{(x_1, 0.1), (x_2, 0.3), (x_3, 1), (x_4, 0)\}$$

#### 3. Complement

Complement of a fuzzy set denoted by  $\tilde{A}$  and  $\tilde{B}$  defined as,

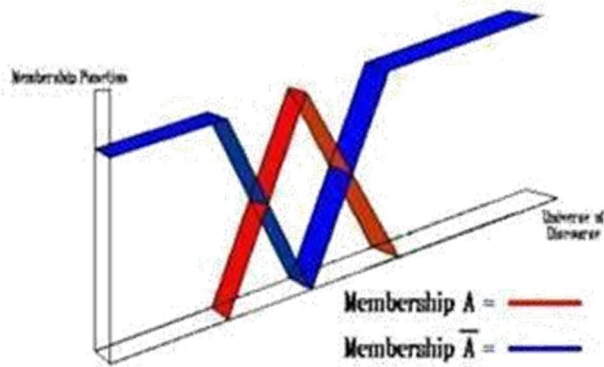


Illustration:  $\widetilde{A}^c = \{(x_1, 0.9), (x_2, 0.3), (x_3, 0), (x_4, 1)\}$

#### 4. Algebraic Sum

Algebraic sum of two fuzzy sets  $\widetilde{A}$  and  $\widetilde{B}$  is denoted by  $\widetilde{A} \dot{+} \widetilde{B}$  and is defined as,  
 $\mu_{\widetilde{A} \dot{+} \widetilde{B}}(x) = \mu_{\widetilde{A}}(x) + \mu_{\widetilde{B}}(x) - \mu_{\widetilde{A}}(x) \cdot \mu_{\widetilde{B}}(x)$

Illustration:  $\widetilde{A} \dot{+} \widetilde{B} = \{(x_1, 0.46), (x_2, 0.79), (x_3, 1), (x_4, 0.2)\}$

#### 5. Algebraic Product

Algebraic product of two fuzzy sets  $\widetilde{A}$  and  $\widetilde{B}$  is denoted by  $\widetilde{A} \dot{\cdot} \widetilde{B}$  and is defined as,  
 $\mu_{\widetilde{A} \dot{\cdot} \widetilde{B}}(x) = \mu_{\widetilde{A}}(x) \cdot \mu_{\widetilde{B}}(x)$

as, Illustration:  $\widetilde{A} \dot{\cdot} \widetilde{B} = \{(x_1, 0.04), (x_2, 0.21), (x_3, 1), (x_4, 0)\}$



## Source Code:

```
clc; clear all; clf;
u=input('Enter First Matrix');
v=input('Enter Second Matrix');
w=max(u,v);
p=min(u,v);
q1=1-u;
q2=1-v;
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/SCREENSHOTS:

```
Scilab 5.5.2 Console

Enter First Matrix [0.1 0.4]
Enter Second Matrix [0.2 0.3]

Union Of Two Matrices

    0.2    0.4

Intersection Of Two Matrices

    0.1    0.3

Complement Of First Matrix

    0.9    0.6

Complement Of Second Matrix

    0.8    0.7

-->|
```

## Result/Conclusion

The concepts of union, intersection and complement are implemented using fuzzy sets which helped to understand the differences and similarities between fuzzy set and classical set theories. It provides the basic mathematical foundations to use the fuzzy set operations.

## EXPERIMENT-2

### Aim: Implementation of fuzzy relations (Max-Min Composition)

#### Objectives:

- Understanding of the basic mathematical elements of the theory of fuzzy sets.
- To introduce the ideas of fuzzy sets, fuzzy logic.
- To implement applications using the fuzzy set operations.

#### Theory:

##### Max-Min Composition of fuzzy Relations

Fuzzy relation in different product space can be combined with each other by the operation called —Composition—. There are many composition methods in use , e.g. max- product method, max-average method and max-min method. But max-min composition method is best known in fuzzy logic applications.

#### Definition:

##### Composition of Fuzzy Relations

- Consider two fuzzy relation;  $R (X \times Y)$  and  $S (Y \times Z)$ , then a relation  $T (X \times Z)$ , can be expressed as max-min composition

$$\begin{aligned} T &= R \circ S \\ \mu_T (x, z) &= \max\text{-min} [\mu_R (x, y), \mu_S (y, z)] \\ &= V [\mu_R (x, y) \wedge \mu_S (y, z)] \end{aligned}$$

- If algebraic product is adopted, then max-product compositions adopted:

$$\begin{aligned} T &= R \circ S \\ \mu_T (x, z) &= \max [\mu_R (x, y) \cdot \mu_S (y, z)] \\ &= V [\mu_R (x, y) \cdot \mu_S (y, z)] \end{aligned}$$

- The max-min composition can be interpreted as indicating the strength of the existence of relation between the elements of  $X$  and  $Z$ .
- Calculations of  $(R \circ S)$  are almost similar to matrix multiplication.

**Crisp relation:**

Crisp relation is defined on the Cartesian product of two sets. Consider,  
 $X \times Y = \{(x,y) | x \in X, y \in Y\}$

The relation on this Cartesian product will be,

$$\mu_R = \begin{cases} 1, & (x,y) \in R \\ 0, & (x,y) \notin R \end{cases}$$

Illustration: Let  $X = \{1,4,5\}$  and  $Y = \{3,6,7\}$  then for relation  $R = x < y$ ,

$$R = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

### Fuzzy relation:

Let  $X, Y \subseteq R$  be universal sets then,

$$R = \{((x, y), \mu_R(x, y)) \mid (x, y) \in X \times Y\}$$

Is called a fuzzy relation in  $X \times Y \subseteq R$

**Illustration:** Let  $X = \{1,2,3\}$  and  $Y = \{1,2\}$

If  $\mu_R(x, y) = e^{-(x-y)^2}$ , then

$$R = \left\{ \frac{e^{-(1-1)^2}}{(1,1)}, \frac{e^{-(1-2)^2}}{(1,2)}, \frac{e^{-(2-1)^2}}{(2,1)}, \frac{e^{-(2-2)^2}}{(2,2)}, \frac{e^{-(3-1)^2}}{(3,1)}, \frac{e^{-(3-2)^2}}{(3,2)} \right\}$$

$$R = \begin{bmatrix} 1 & 0.37 \\ 0.37 & 1 \\ 0.02 & 0.37 \end{bmatrix}$$

### Max-Min Composition:

Let  $X, Y$  and  $Z$  be universal sets and let  $R$  and  $Q$  be relations that relate them as,

$$R = \{(x, y) \mid x \in X, y \in Y, R \subset X \times Y\}$$

$$Q = \{(y, z) \mid y \in Y, z \in Z, Q \subset Y \times Z\}$$

Then  $S$  will be a relation that relates elements of  $X$  with elements of  $Z$  as,

$$S = R \circ Q$$

$$S = \{(x, z) \mid x \in X, z \in Z, S \subset X \times Z\}$$

Max min composition is then defined as,

$$\mu_S(x, z) = \max \left( \min \left( \mu_R(x, y), \mu_Q(y, z) \right) \right)$$

**Illustration:**

$$\text{and } Q = \begin{bmatrix} 0.2 & 0.6 \\ 0.1 & 0.3 \\ 0.7 & 0.5 \end{bmatrix}$$

$$S = \begin{bmatrix} 0.4 & 0.6 \\ 0.2 & 0.2 \end{bmatrix}$$

## Source code:

```
clear;
clc;
R=input("enter the first relation ");
disp("R=",R);
S=input("enter the second relation ");
disp("S=",S);
[m,n]=size(R);
[a,b]=size(S);
if(n==a)
for i=1:m
for j=1:b
c=R(i,:);
d=S(:,j);
[f,g]=size(c);
[h,q]=size(d);
for l=1:g
e(1,l)=c(1,l)*d(l,1);
end
t(i,j)=max(e);
end
end
disp("the final max-product is ")
disp("t=",t);
else
disp("cannot find max-product");

end
if(n==a)
for i=1:m
for j=1:b
c=R(i,:);
d=S(:,j);
f=mtlb_t(d);
```

```
e=min(c,f);  
h(i,j)=max(e);  
end  
end  
disp("the final min-max output is ")  
disp("h=",h);  
else  
disp("cannot find min-max");  
end
```

## Output:

```
Scilab 5.5.2 Console
Scilab 5.5.2 Console

enter the first relation [1 0 1 0;0 0 0 1;0 0 0 0]

    1.    0.    1.    0.
    0.    0.    0.    1.
    0.    0.    0.    0.

R=
enter the second relation [0 1;0 0;0 1;0 0]

    0.    1.
    0.    0.
    0.    1.
    0.    0.

S=

the final max-product is

    0.    1.
    0.    0.
    0.    0.

t=

the final min-max output is

    0.    1.
    0.    0.
    0.    0.

h=

-->|
```

**Result/Conclusion:** With the use of fuzzy logic principles max min composition of fuzzy set is calculated which describes the relationship between two or more fuzzy sets.



## EXPERIMENT-3

### Aim: Implementation of fuzzy controller (Washing Machine)

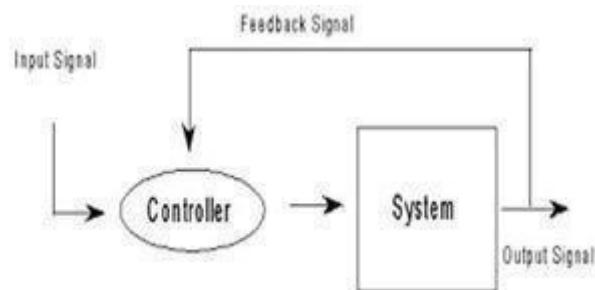
#### Objectives:

- Cover fuzzy logic inference with emphasis on their use in the design of intelligent or humanistic systems.
- Prepare the students for developing intelligent systems.

#### Theory:

##### Control System:

Any system whose outputs are controlled by some inputs to the system is called control system.



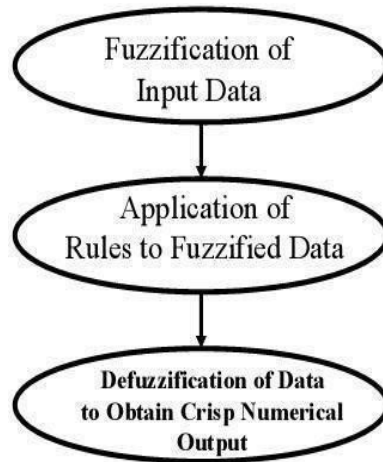
## **Fuzzy Controller:**

Fuzzy controllers are the most important applications of fuzzy theory. They work different than conventional controllers as:

Expert knowledge is used instead of differential equations to describe a system. This expert knowledge can be expressed in very natural way using linguistic variables, which are described by fuzzy sets.

The fuzzy controllers are constructed in following three stages:

1. Create the membership values (fuzzify).
2. Specify the rule table.
3. Determine your procedure for defuzzifying the result.



## **Washing Machine Controller:**

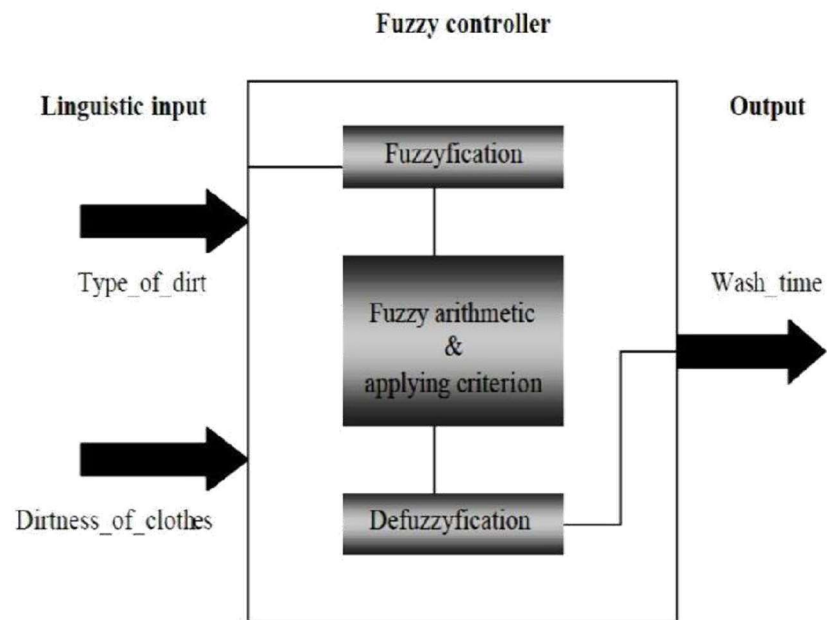
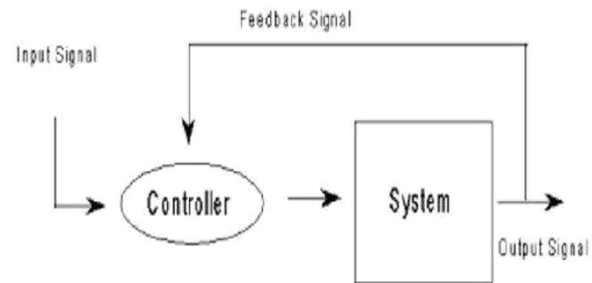
To design a system using fuzzy logic, input & output is necessary part of the system. Main function of the washing machine is to clean cloth without damaging the cloth. In order to achieve it, the output parameters of fuzzy logic, which are the washing parameters, must be given more importance. The identified input & output parameters are:

### **Input:**

1. Degree of dirt
2. Type of dirt

### **Output:**

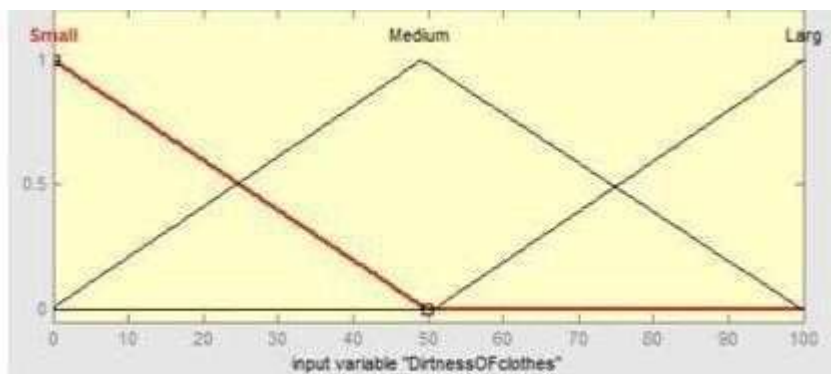
Wash time



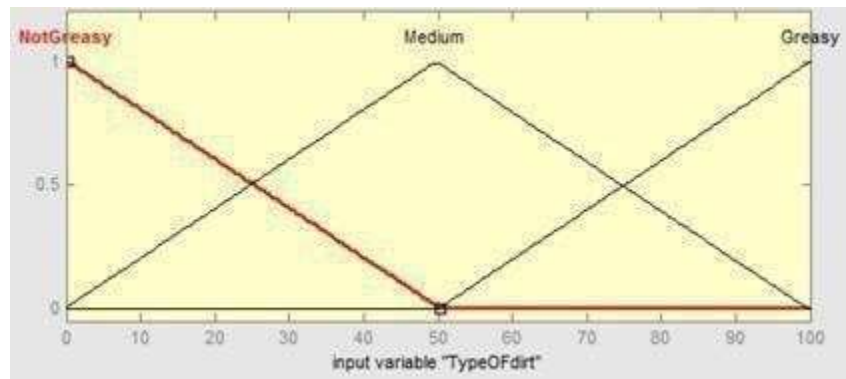
### Fuzzy sets:

The fuzzy sets which characterize the inputs & output are given as follows:

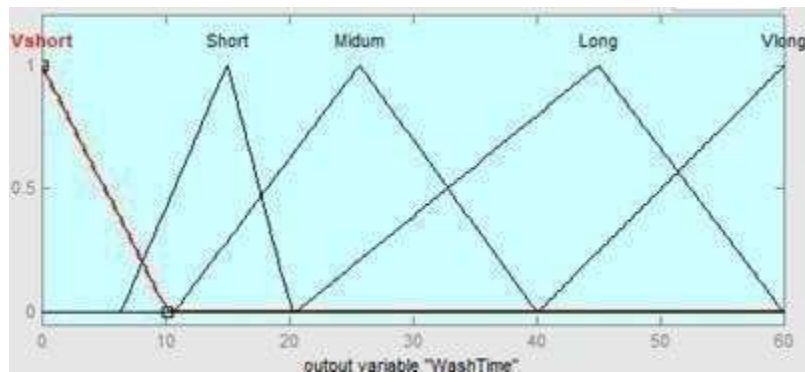
#### 1. Dirtiness of clothes



## 2. Type of dirt



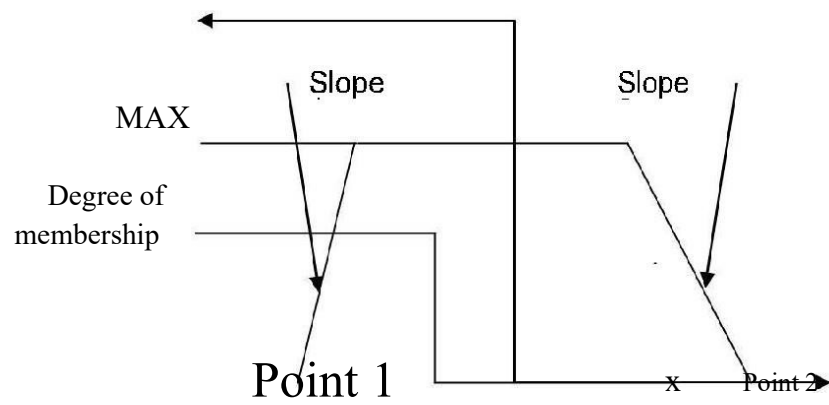
## 3. Wash time



## Procedure:

### Step1: Fuzzification of inputs

For the fuzzification of inputs, that is, to compute the membership for the antecedents, the formula used is,



$$\Delta 1 = x - \text{point1}$$

$$\Delta 2 = \text{point2} - x$$

If  $(\Delta 1 \leq 0)$  or  $(\Delta 2 \leq 0)$

then Degree of membership = 0

else

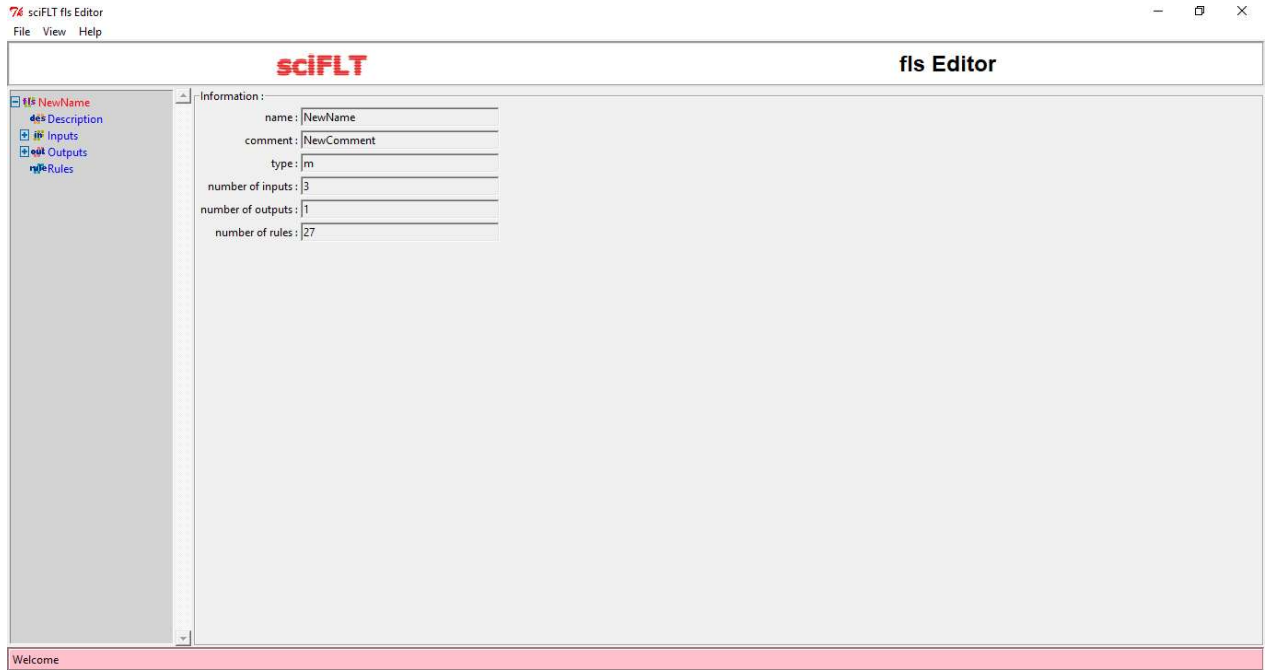
$$\text{Degree of membership} = \min \left( \frac{\Delta 1 * \text{Slope1}}{\Delta 2 * \text{Slope2}}, \text{Max} \right)$$

**Step 2:** Defining set of rules

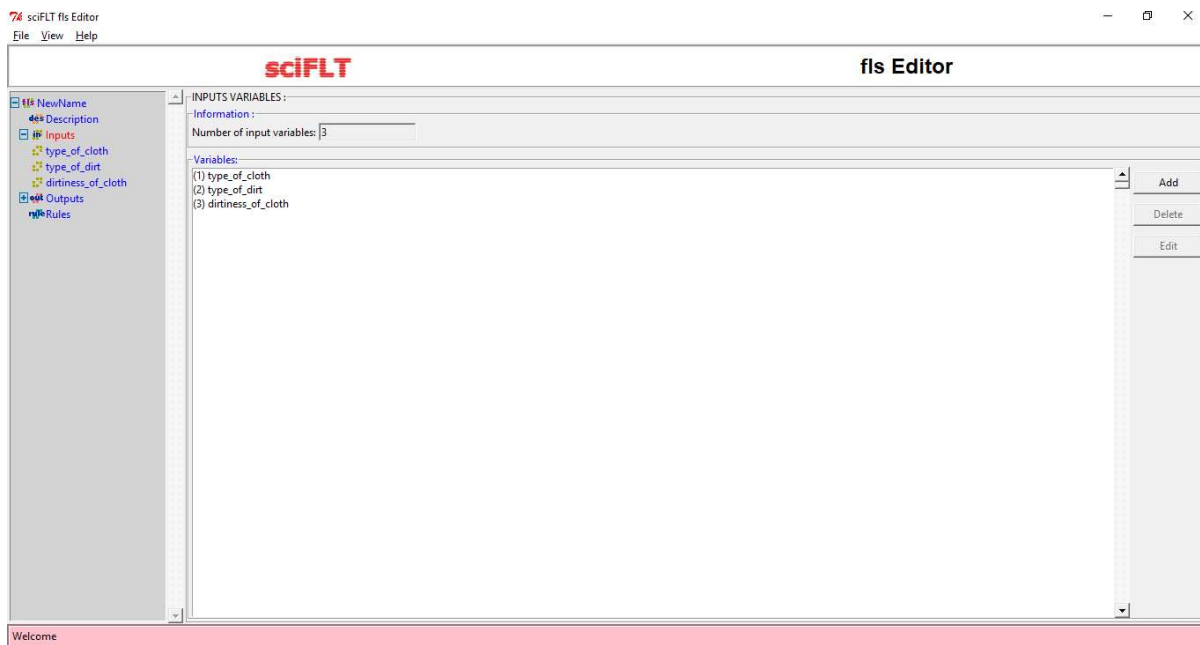
	S	M	L
NG	VS	S	M
M	M	M	L
G	L	L	VL

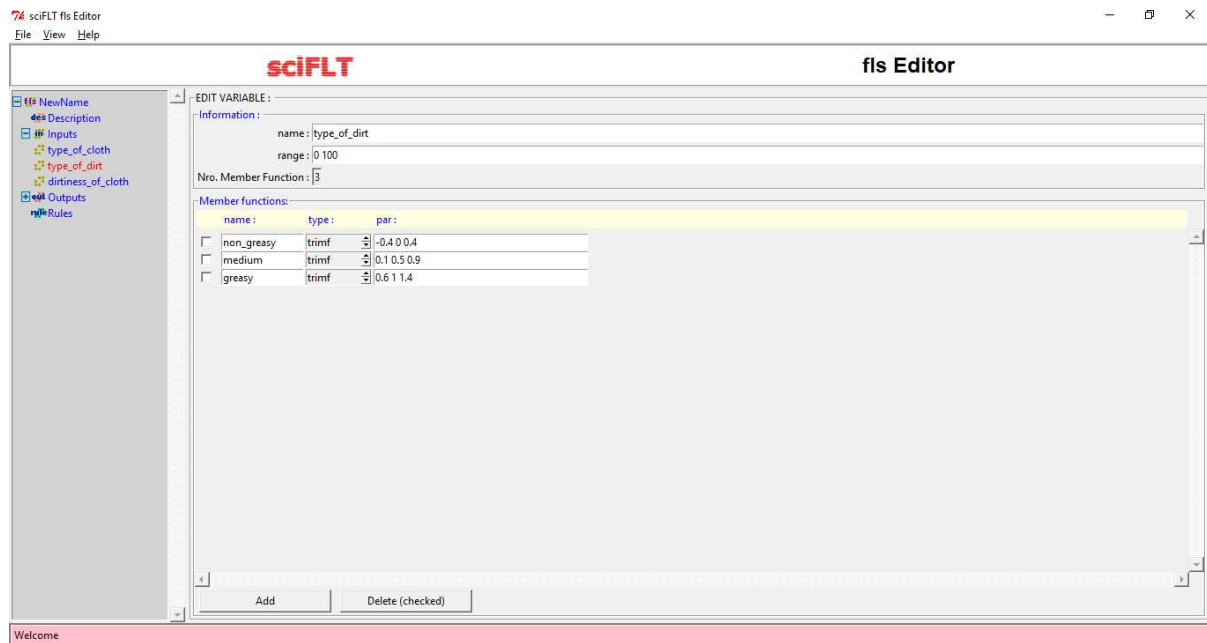
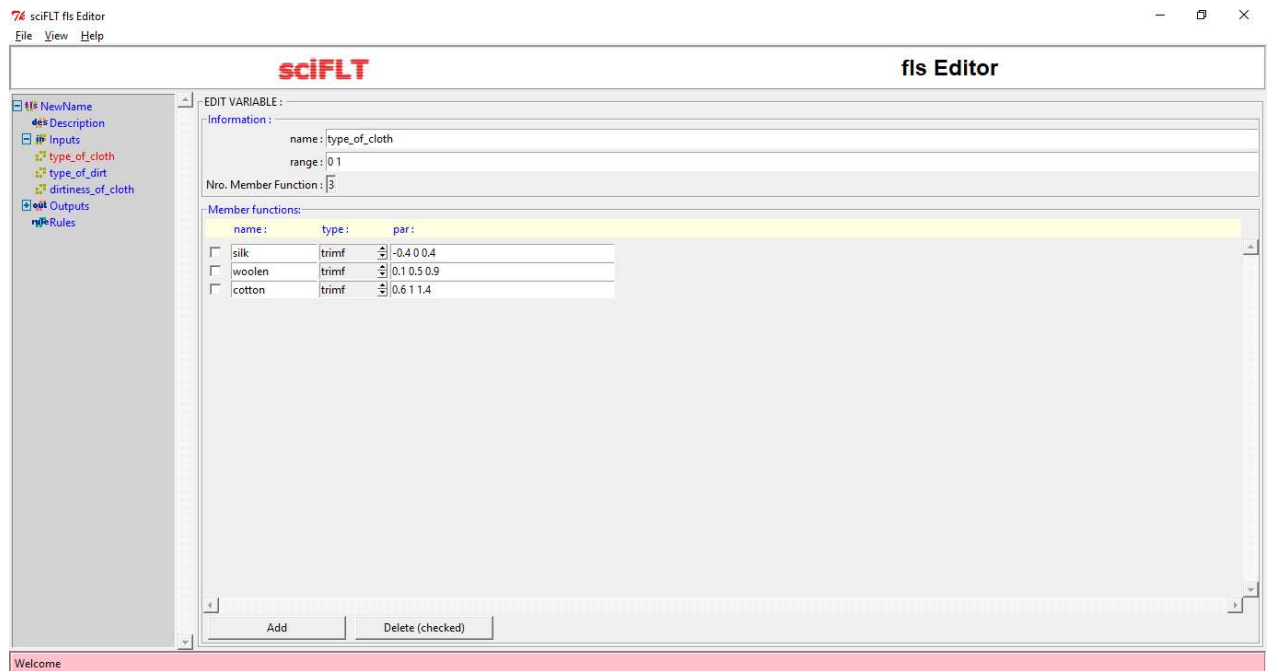
## Execution:

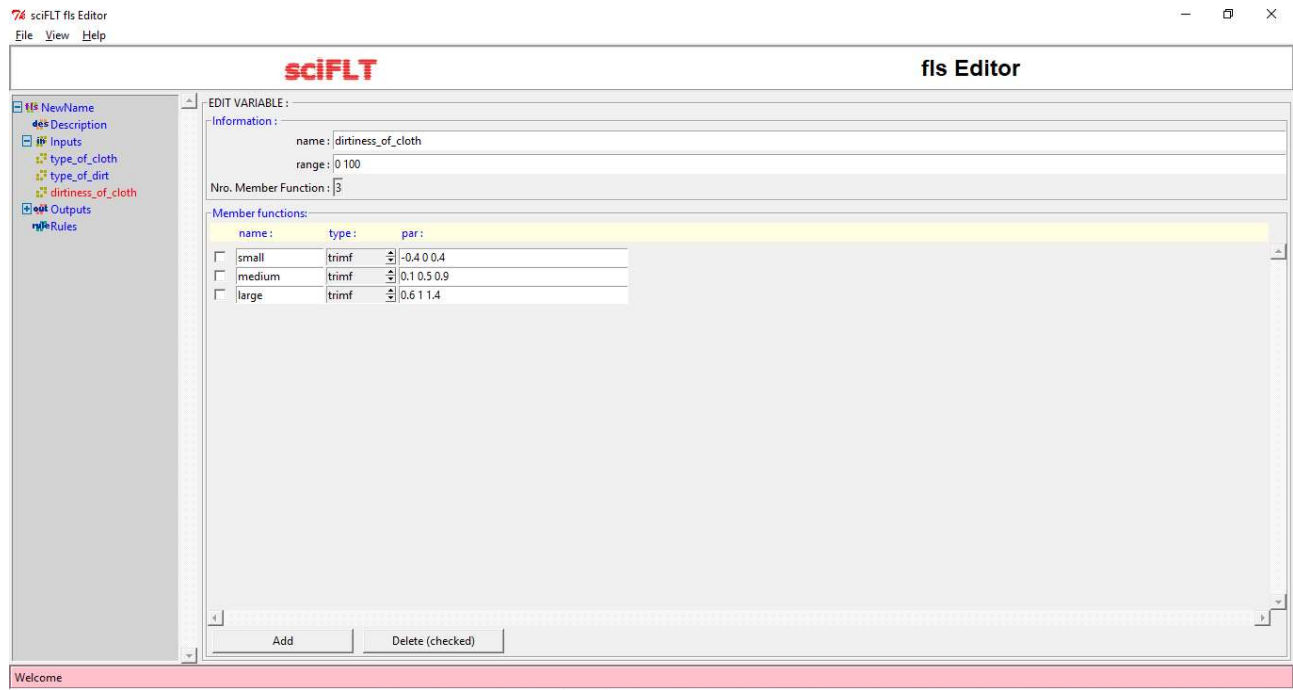
1. Run sciFLT Editor using `sciFLTEditor()` command.
2. Create New File.



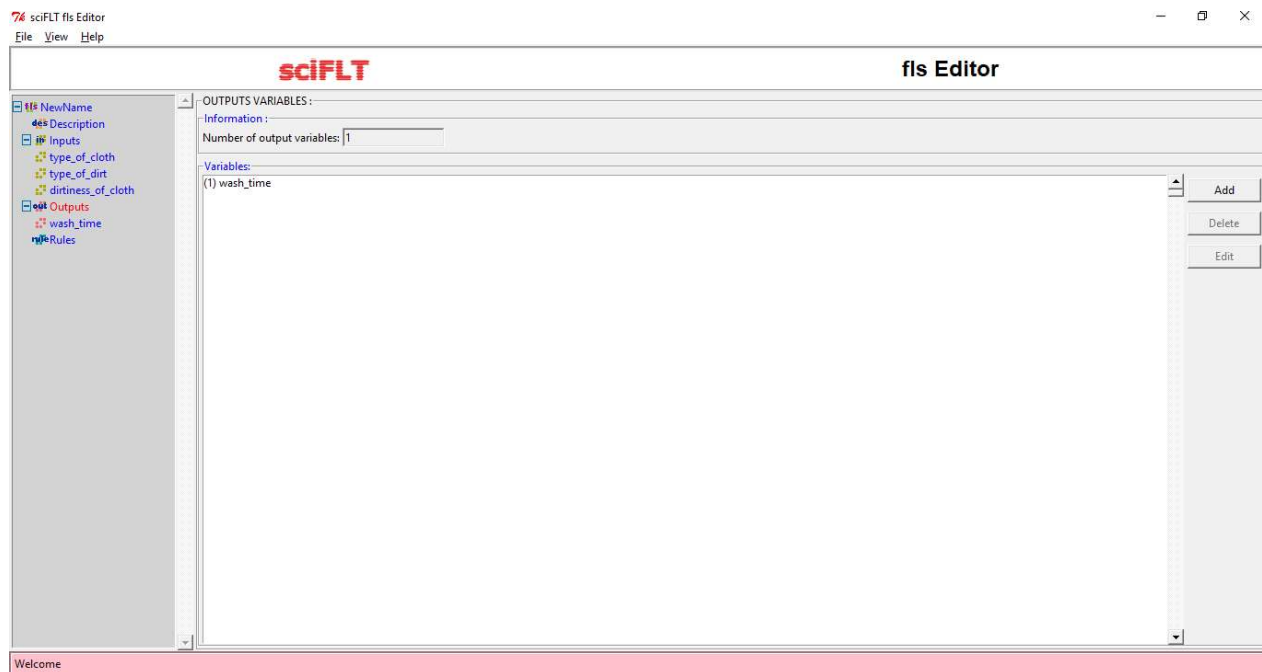
## 3. Design the input variables



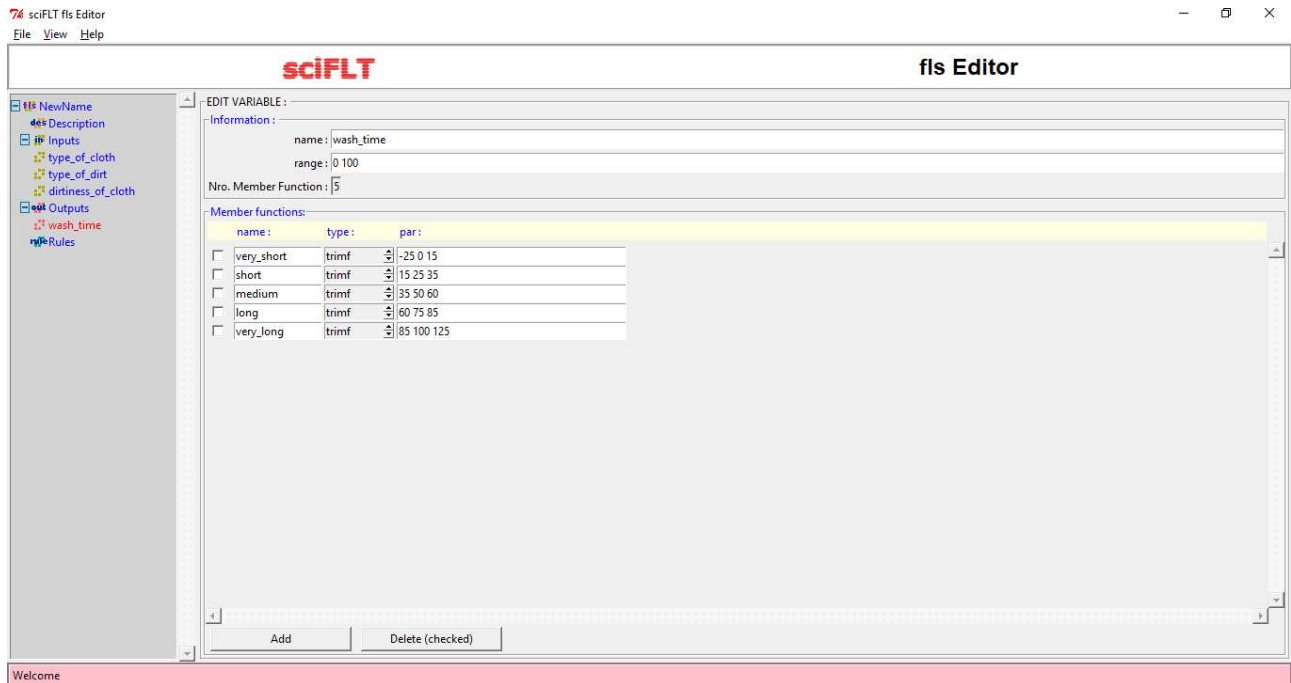




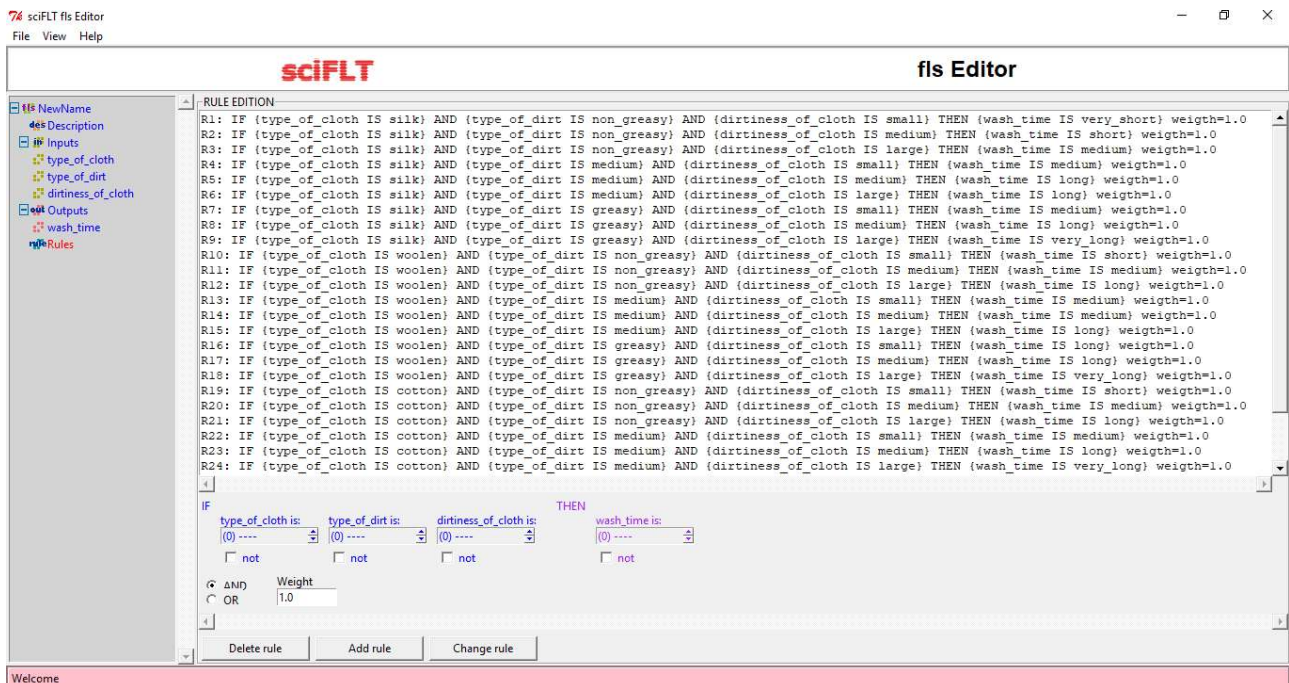
#### 4. Design output variable







## 5. Define the rules



## 6. Execute using loadfls and evaluatefls commands

## Output:

```
Scilab 5.5.2 Console

-->fls1=loadfls("washing.flc")
fls1 =

      name : 'NewName'
    comment : 'NewComment'
      type : 'm'
      SNorm : 'max'
    SNormPar : [0]
      TNorm : 'min'
    TNormPar : [0]
      Comp : 'one'
    CompPar : [0]
    ImpMethod : 'min'
    AggMethod : 'max'
defuzzMethod : 'centroide'
      input : 3 input(s)
      output : 1 output(s)
      rule : 27 rule(s)

-->evalfls([0.2 0.66 0.22],fls1)
ans =

      58.507743

-->plotsurf(flsl)

-->|
```

## EXPERIMENT-4

**Aim: To implement Mc-Culloch pits Model using XOR**

### Objectives:

- The student will be able to obtain the fundamentals and different architecture of neural networks.
- The student will have a broad knowledge in developing the different algorithms for neural networks.

### Theory:

Neural network was inspired by the design and functioning of human brain and components.

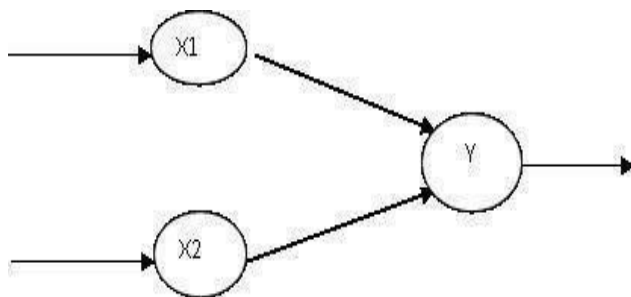
#### Definition:

—Information processing model that is inspired by the way biological nervous system (i.e the brain) process information, is called Neural Network. |

Neural Network has the ability to learn by Illustrations. It is not designed to perform fix /specific task, rather task which need thinking (e.g. Predictions).

ANN is composed of large number of highly interconnected processing elements(neurons) working in unison to solve Illustrations. It mimic human brain. It is configured for special application such as pattern recognition and data classification through a learning process. ANN is 85-90% accurate.

#### Basic Operation of a Neural Network:



X1 and X2 – input neurons.

Y- output neuron

Weighted interconnection links- W1 and W2.

**Net input calculation is :**

$$Y_{in} = x_1w_1 + x_2w_2$$

**Output is:**

$$y = f(Y_{in})$$

Output= function

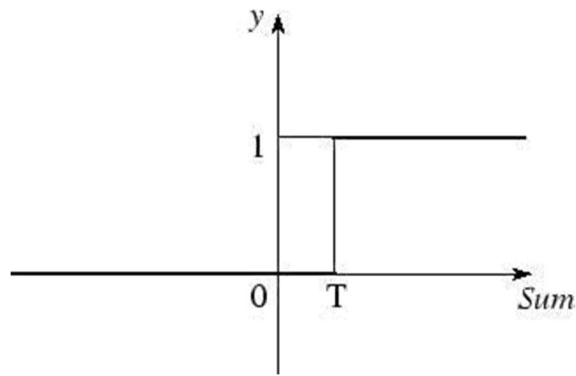
**The McCulloch-Pitts Model of Neuron:**

The early model of an artificial neuron is introduced by Warren McCulloch and Walter Pitts in 1943. The McCulloch-Pitts neural model is also known as linear threshold gate. It is a neuron of a set of inputs  $I_1, I_2, I_3 \dots I_m$  and one output  $y$ . The linear threshold gate simply classifies the set of inputs into two different classes. Thus the output  $y$  is binary. Such a function can be described mathematically using these equations:

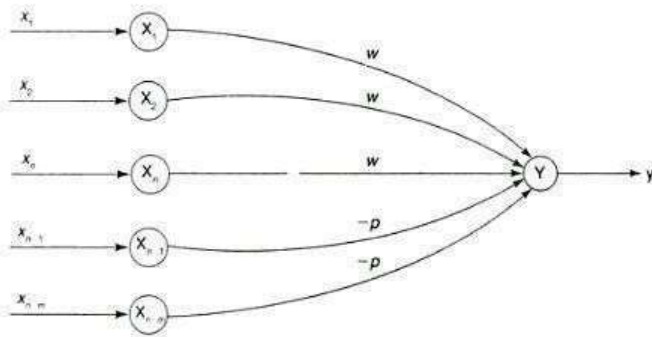
$$Sum = \sum_{i=1}^N I_i W_i,$$

$$y = f(Sum).$$

---



$W_1, W_2 \dots W_m$  are weight values normalized in the range of either  $(0,1)$  or  $(-1,1)$  and associated with each input line, Sum is the weighted sum, and  $T$  is a threshold constant. The function  $f$  is a linear step function at threshold  $T$  as shown in figure



A simple M-P neuron is shown in the figure.

It is excitatory with weight ( $w > 0$ ) / inhibitory with weight  $-p$  ( $p < 0$ ).

In the Fig., inputs from  $x_1$  to  $x_n$  possess excitatory weighted connection and  $x_{n+1}$  to  $x_{n+m}$  has inhibitory weighted interconnections.

Since the firing of neuron is based on threshold, activation function is defined as

$$f(y_{in}) = \begin{cases} 1 & \text{if } y_{in} \geq \theta \\ 0 & \text{if } y_{in} < \theta \end{cases}$$

For inhibition to be absolute, the threshold with the activation function should satisfy the following condition:

$$\theta > nw - p$$

Output will fire if it receives  $k$  or more excitatory inputs but no inhibitory inputs where

$$kw \geq \theta > (k-1)w$$

- The M-P neuron has no particular training algorithm.
- An analysis is performed to determine the weights and the threshold.
- It is used as a building block where any function or phenomenon is modelled based on a logic function.

**Illustration Statement:** Implement XOR function using MP model

Truth table for XOR function is:

X1	X2	Y
0	0	0
0	1	1
1	0	1
1	1	0

Activation function  $Y_{in}$  is as follows:

$$Y_{in} = x_1 w_1 + x_2 w_2$$

As we know,

$$x_1 \text{ XOR } x_2 = (x_1 \text{ AND NOT } x_2) \text{ OR } (x_2 \text{ AND NOT } x_1)$$

Let  $Z_1 = (x_1 \text{ AND NOT } x_2)$  and  $Z_2 = (x_2 \text{ AND NOT } x_1)$

X1	X2	Z1
0	0	0
0	1	0
1	0	1
1	1	0

For  $Z_1$ ,

$$W_{11} = 1 \text{ and } W_{12} = -1$$

$$\Theta = 1$$

X1	X2	Z2
0	0	0
0	1	1
1	0	0
1	1	0

For Z2,

$$W_{11}=-1 \text{ and } W_{12}=1$$

$$\Theta=1$$

$$Y=Z1+Z2$$

Z1	Z2	Y
0	0	0
0	1	1
1	0	1
1	1	0

For Y,

$$W_{11}=1 \text{ and } W_{12}=1$$

$$\Theta=1$$

## Source Code :

```
clear;
clc;
//Getting weights and threshold value
disp('Enter 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);
```

## OUTPUT/ Screenshots

```
Scilab 5.5.2 Console

Enter 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 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.

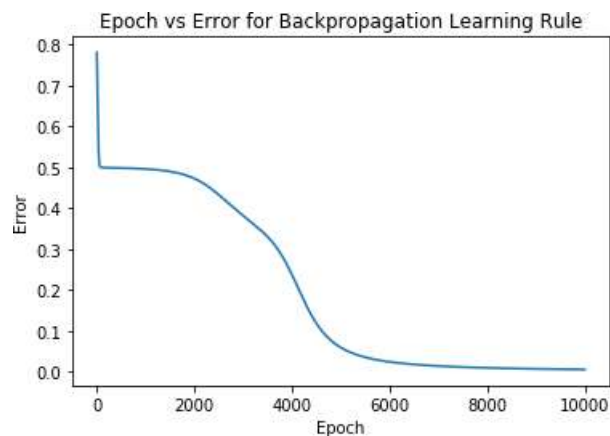
threshold value

1.

-->
```

## Result/Conclusion:

Mc-Culloch pits Model is implemented for XOR function by using the thresholding activation function. Activation of M-P neurons is binary (i.e) at any time step the neuron mayfire or may not fire. Threshold plays major role here.



## EXPERIMENT-5

**Aim: Implementation of Single layer Perceptron Learning Algorithm.**

### Objectives:

- To become familiar with neural networks learning algorithms from available Illustrations.
- Provide knowledge of learning algorithm in neural networks.

### Theory:

Neural networks are a branch of —Artificial Intelligence". Artificial Neural Network is a system loosely modelled based on the human brain. Neural networks are a powerful technique to solve many real world Illustrations. They have the ability to learn from experience in order to improve their performance and to adapt themselves to changes in the environment. In addition to that they are able to deal with incomplete information or noisy data and can be very effective especially in situations where it is not possible to define the rules or steps that lead to the solution of a Illustration. In a nutshell a Neural network can be considered as a black box that is able to predict an output pattern when it recognizes a given input pattern. Once trained, the neural network is able to recognize similarities when presented with a new input pattern, resulting in a predicted output pattern.

### Algorithm:

The algorithm is as follows:

1. Initialize the weights and threshold to small random numbers.
2. Present a vector  $x$  to the neuron inputs and calculate the output.
3. Update the weights according to:

$$w_j(t+1) = w_j(t) + \eta (d - y)x$$

where  $d$  is the desired output,

- $t$  is the iteration number, and
- $\eta$  is the gain or step size, where  $0.0 < \eta < 1.0$

4. Repeat steps 2 and 3 until:
  1. the iteration error is less than a user-specified error threshold or
  2. a predetermined number of iterations have been completed.

### Source Code:

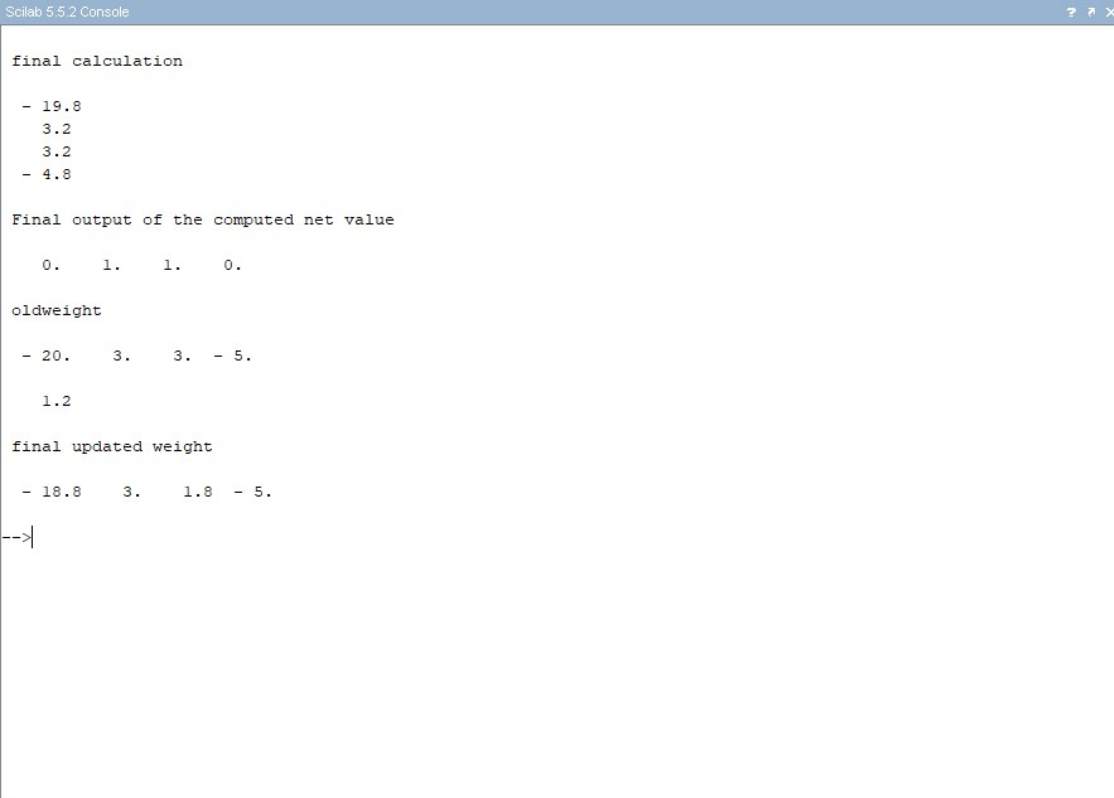
```
clear;
clc;
x=[0 0 1 1; 0 1 0 1];
//input variable pass
d=[1 1 0 0];
//target output
w=[-20 3 3 -5];
//initialize weight for per input
z=[0 0 0 0];
//vector to store the calculated value of the sigma input*weight + bias
bias=0.2;
//initiatlize of bias to store the value
//calculate the values total value
for j= 1:2
sigma=0;
for i=1 : 4
```

```

sigma=bias + x(j,i)*mtlb_t(w);
end
end
disp('final calculation');
disp(sigma);
//set the theta value for step function
theta=0.3;
for i=1:4
if sigma(i)> theta
z(i)=1;
elseif sigma(i)<=theta
z(i)=0;
end
end
disp('Final output of the computed net value');
disp(z);
disp('oldweight');
disp(w);
eta=1.2; // learning rate ;
for j= 1:4
lr=0;
for i=1 : 2
lr= x(i,j)*eta;
end end
disp(lr);
for i=1:4
if z(i)==1 & d(i)==0
w(i)=w(i)-lr;
elseif z(i)==0 & d(i)==1
w(i)=w(i)+lr;
end
end
disp('final updated weight');
disp(w);

```

## Output/ Screenshots:



```
Scilab 5.5.2 Console

final calculation

- 19.8
  3.2
  3.2
- 4.8

Final output of the computed net value

  0.   1.   1.   0.

oldweight

- 20.   3.   3.  - 5.

  1.2

final updated weight

- 18.8   3.   1.8  - 5.

-->|
```

## Result/Conclusion:

```
result_test=perceptron_test(d_test_X,d_test_y.shape,weights)
```

```
# Calculate score
```

```
score(result_test,d_test_y)
```

```
Score=46.15384615384615
```

Single layer perceptron learning algorithm is implemented for AND function. It is used for train the iterations of neural network. Neural network mimics the human brain and perceptron learning algorithm trains the neural network according to the input given.

## EXPERIMENT-6

**Aim: Implementation of unsupervised learning algorithm – Hebbian Learning**

### Objectives:

- To become familiar with neural networks learning algorithms from available Illustrations.
- To give design methodologies for artificial neural networks.
- Provide knowledge of un-supervised learning in neural networks.

### Theory:

#### Unsupervised Learning Algorithm:

These types of model are not provided with the correct results during the training. It can be used to cluster the input data in classes on the basis of their statistical properties only. The labelling can be carried out even if the labels are only available for a small number of objects represented of the desired classes. All similar input patterns are grouped together as clusters. If matching pattern is not found, a new cluster is formed. In contrast to supervised learning, unsupervised or self-organized learning does not require an external teacher. During the training session, the neural network receives a number of different patterns & learns how to classify input data into appropriate categories. Unsupervised learning tends to follow the neuro-biological organization of brain. It aims to learn rapidly & can be used in real-time.

Hebbian Learning:

Hebbian Learning is inspired by the biological neural weight adjustment mechanism. It describes the method to convert a neuron an inability to learn and enables it to develop cognition with response to external stimuli. These concepts are still the basis for neural learning today.

### Algorithm:

The algorithm is as follows:

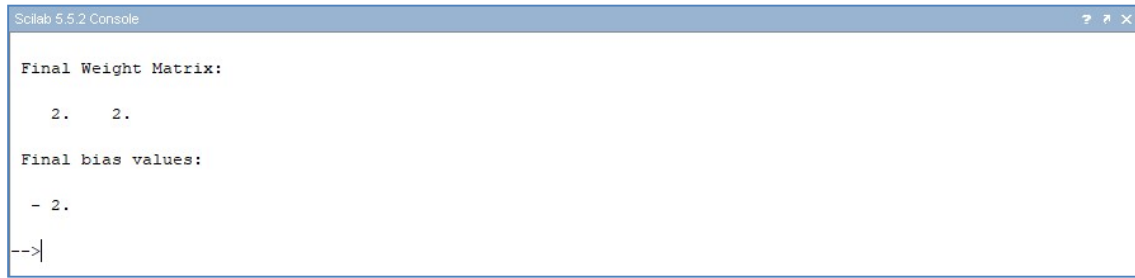
1. Set initial synaptic weights and thresholds to small random values, say in an interval  $[0,1]$ .
2. Compute the neuron output at iteration  $p$   
where  $n$  is number of neuron inputs, &  $\theta_j$  is the threshold value of neuron  $j$ .
3. Update the weights in the network

## Source Code:

```
clc;
clear;
x=[1 1 -1 -1;1 -1 1 -1];
t=[1 -1 -1 -1];
w=[0 0];
b=0;
for i=1:4
for j=1:2
w(j)=w(j)+t(i)*x(j,i);
end
b=b+t(i);
end
disp("Final Weight Matrix: ");
disp(w);
disp("Final bias values: ");
disp(b);
plot(x(1,1),x(2,1),'or','MarkerSize',20,'MarkerFaceColor',[0 0 1]);
set(gca(),"auto_clear","off");
plot(x(1,2),x(2,2),'or','MarkerSize',20,'MarkerFaceColor',[1 0 0]);
set(gca(),"auto_clear","off");
plot(x(1,3),x(2,3),'or','MarkerSize',20,'MarkerFaceColor',[1 0 0]);
set(gca(),"auto_clear","off");
plot(x(1,4),x(2,4),'or','MarkerSize',20,'MarkerFaceColor',[1 0 0]);
set(gca(),"auto_clear","off");
m=-(w(1)/w(2));
c=-b/w(2);
x1=linspace(-2,2,100);
x2=m*x1+c;
plot(x2,x1,'r');
a=gca() ;//get the current axes
a.box="on";
a.data_bounds=[-2,-2;2,2]; //define the bounds
```



## OUTPUT/ SCREENSHOTS:



```
Scilab 5.5.2 Console
Final Weight Matrix:
    2.    2.

Final bias values:
    - 2.

-->|
```

### Result/Conclusion:

Unsupervised Hebbian learning algorithm is implemented which does not require supervisor. It updates the weights the accordingly if error comes and train the network.

## EXPERIMENT-7

**Aim: Implementation Genetic Application – Match Word Finding.**

### Objectives:

- To familiarize with Mathematical foundations for Genetic algorithm, operator.
- To study the Applications of Genetic Algorithms.

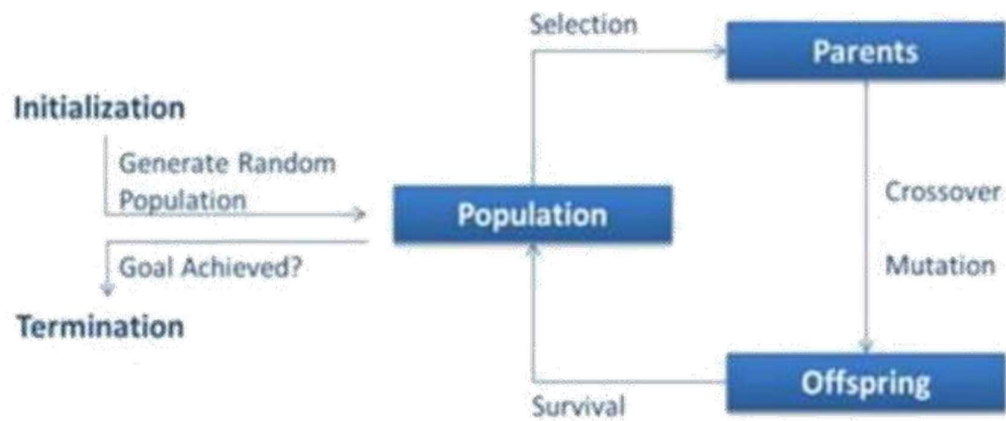
### Theory:

#### Genetic algorithm:

- Genetic algorithm is a search technique used in computing to find true or approximate solutions to optimization & search Illustrations.
- Genetic algorithms are inspired by Darwin's theory about evolution. Solution to an Illustration solved by genetic algorithms is evolved.
- Algorithm is started with a set of solutions (represented by chromosomes) called population. Solutions from one population are taken and used to form a new population. This is motivated by a hope, that the new population will be better than the old one. Solutions which are selected to form new solutions (offspring) are selected according to their fitness - the more suitable they are the more chances they have to reproduce.
- This is repeated until some condition (for Illustration number of populations or improvement of the best solution) is satisfied.

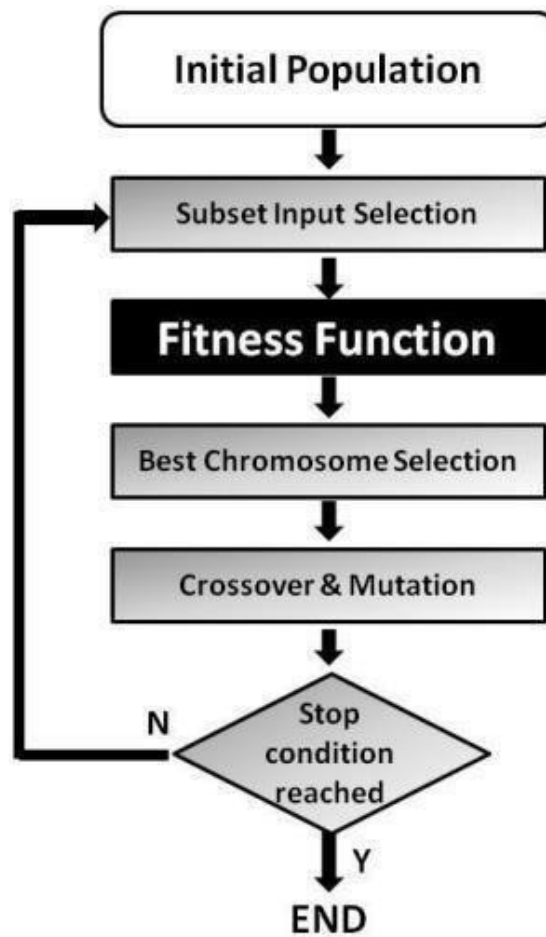
### Outline of the Basic Genetic Algorithm

1. [Start] Generate random population of  $n$  chromosomes (suitable solutions for the Illustration)
2. [Fitness] Evaluate the fitness  $f(x)$  of each chromosome  $x$  in the population
3. [New population] Create a new population by repeating following steps until the new population is complete
  1. [Selection] Select two parent chromosomes from a population according to their fitness (the better fitness, the bigger chance to be selected)
  2. [Crossover] With a crossover probability cross over the parents to form a new offspring (children). If no crossover was performed, offspring is an exact copy of parents.
  3. [Mutation] With a mutation probability mutate new offspring at each locus (position in chromosome).
  4. [Accepting] Place new offspring in a new population
  5. [Replace] Use new generated population for a further run of algorithm
  6. [Test] If the end condition is satisfied, stop, and return the best solution in current population
  7. [Loop] Go to step 2



General Scheme of Evolutionary Process

## Flowchart



**Illustration Statement:**

Match Word Finding Here we try to guess a word from the given population of word.

**Algorithm:**

Match Word Finding Algorithm:

**Step 1:** Select the word to be guessed

This value is taken through user input.

**Step 2:** Initialize the population

User inputs the population.

**Step 3:** Evaluate the population

Fitness is assigned based on number of correct letters in correct place.

**Step 4:** Select breeding population

Selection is done on the basis of fitness.

**Step 5:** Create new population

Population is created by using uniform crossover between breeding populations.

**Step 6:** Check for stopping condition

Here maximum fitness value in population is checked. If it is 60%.

**Step 7:** If stopping condition is not true, goto Step 3;

Else return the offspring with highest fitness value.

### Source Code:

```
import random

# Number of individuals in each generation
POPULATION_SIZE = 100

# Valid genes
GENES = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ
QRSTUVWXY 1234567890, .-;:_!\"#%&/'()=?@$%{}[]"

# Target string to be generated
TARGET = "Anant kedia"

class Individual(object):
    """
    Class representing individual in population
    """
    def __init__(self, chromosome):
        self.chromosome = chromosome
        self.fitness = self.cal_fitness()

    @classmethod
    def mutated_genes(self):
        """
        create random genes for mutation
        """
        global GENES
        gene = random.choice(GENES)
        return gene

    @classmethod
    def create_gnome(self):
        """
        create chromosome or string of genes
        """
        global TARGET
        gnome_len = len(TARGET)
        return [self.mutated_genes() for _ in range(gnome_len)]
```

```

def mate(self, par2):
    """
    Perform mating and produce new offspring
    """

    # chromosome for offspring
    child_chromosome = []
    for gp1, gp2 in zip(self.chromosome, par2.chromosome):

        # random probability
        prob = random.random()

        # if prob is less than 0.45, insert gene
        # from parent 1
        if prob < 0.45:
            child_chromosome.append(gp1)

        # if prob is between 0.45 and 0.90, insert
        # gene from parent 2
        elif prob < 0.90:
            child_chromosome.append(gp2)

        # otherwise insert random gene(mutate),
        # for maintaining diversity
        else:
            child_chromosome.append(self.mutated_genes())

    # create new Individual(offspring) using
    # generated chromosome for offspring
    return Individual(child_chromosome)

def cal_fitness(self):
    """
    Calculate fitness score, it is the number of
    characters in string which differ from target
    string.
    """

    global TARGET

```

```

    fitness = 0
    for gs, gt in zip(self.chromosome, TARGET):
        if gs != gt: fitness+= 1
    return fitness

# Driver code
def main():
    global POPULATION_SIZE

    #current generation
    generation = 1

    found = False
    population = []

    # create initial population
    for _ in range(POPULATION_SIZE):
        gnome = Individual.create_gnome()
        population.append(Individual(gnome))

    while not found:

        # sort the population in increasing order of fitness score
        population = sorted(population, key = lambda x:x.fitness)

        # if the individual having lowest fitness score ie.
        # 0 then we know that we have reached to the target
        # and break the loop
        if population[0].fitness <= 0:
            found = True
            break

        # Otherwise generate new offsprings for new generation
        new_generation = []

        # Perform Elitism, that mean 10% of fittest population
        # goes to the next generation
        s = int((10*POPULATION_SIZE)/100)
        new_generation.extend(population[:s])

```

```

# From 50% of fittest population, Individuals
# will mate to produce offspring
s = int((90*POPULATION_SIZE)/100)
for _ in range(s):
    parent1 = random.choice(population[:50])
    parent2 = random.choice(population[:50])
    child = parent1.mate(parent2)
    new_generation.append(child)

population = new_generation

print("Generation: {} \tString: {} \tFitness: {}".\
      format(generation,
            "".join(population[0].chromosome),
            population[0].fitness))

generation += 1

print("Generation: {} \tString: {} \tFitness: {}".\
      format(generation,
            "".join(population[0].chromosome),
            population[0].fitness))

if __name__ == '__main__':
    main()

```



## Output/Screenshots

Generation: 1	String: t0{"-?=jH[k8=B4]0e@}	Fitness: 18
Generation: 2	String: t0{"-?=jH[k8=B4]0e@}	Fitness: 18
Generation: 3	String: .#lRWf9k_Ifslw #0\$k_	Fitness: 17
Generation: 4	String: .-lRq?9mHqk3Wo]3rek_	Fitness: 16
Generation: 5	String: .-lRq?9mHqk3Wo]3rek_	Fitness: 16
Generation: 6	String: A#ldW) #lIkslw cVek)	Fitness: 14
Generation: 7	String: A#ldW) #lIkslw cVek)	Fitness: 14
Generation: 8	String: (, o x _x%Rs=, 6Peek3	Fitness: 13
	.	
	.	
	.	
Generation: 29	String: I lope Geeks#o, Geeks	Fitness: 3
Generation: 30	String: I loMe GeeksfoBGeeks	Fitness: 2
Generation: 31	String: I love Geeksfo0Geeks	Fitness: 1
Generation: 32	String: I love Geeksfo0Geeks	Fitness: 1
Generation: 33	String: I love Geeksfo0Geeks	Fitness: 1
Generation: 34	String: I love GeeksforGeeks	Fitness: 0

## Result/Conclusion:

The match word finding algorithm is implemented using the genetic algorithms which include all the genetic algorithm operators. Genetic algorithm includes the selection, crossover, mutation operators along with fitness function.

## EXPERIMENT-8

**Aim:** Study of ANFIS Architecture.

### Objectives:

- Study of hybrid systems.
- Prepare the students for developing intelligent system.

### Theory:

The adaptive network-based fuzzy inference systems (ANFIS) is used to solve Illustrations related to parameter identification. This parameter identification is done through a hybrid learning rule combining the back-propagation gradient descent and a least-squares method.

ANFIS is basically a graphical network representation of Sugeno-type fuzzy systems endowed with the neural learning capabilities. The network is comprised of nodes with specific functions collected in layers. ANFIS is able to construct a network realization of IF / THEN rules.

Consider a Sugeno type of fuzzy system having the rule base

1. If x is A1 and y is B1, then  $f_1 = c_{11}x + c_{12}y + c_{10}$
2. If x is A2 and y is B2, then  $f_2 = c_{21}x + c_{22}y + c_{20}$

Let the membership functions of fuzzy sets  $A_i, B_i, i=1,2$  be  $\mu_{A_i}, \mu_{B_i}$ .

In evaluating the rules, choose product for T-norm (logical and).

1. Evaluating the rule premises results in

$$w_i = \mu_{A_i}(x) \mu_{B_i}(y), \quad i=1,2.$$

2. Evaluating the implication and the rule consequences gives

$$f(x, y) = \frac{w_1(x, y) f_1(x, y) + w_2(x, y) f_2(x, y)}{w_1(x, y) + w_2(x, y)}.$$

Or leaving the arguments out

$$f = \frac{w_1 f_1 + w_2 f_2}{w_1 + w_2}$$

This can be separated to phases by first defining

$$\bar{w}_i = \frac{w_i}{w_1 + w_2}$$

Then f can be written as

$$f = \bar{w}_1 f_1 + \bar{w}_2 f_2$$

All computations can be presented in a diagram form. ANFIS normally has 5 layers of neurons of which neurons in the same layer are of the same function family.

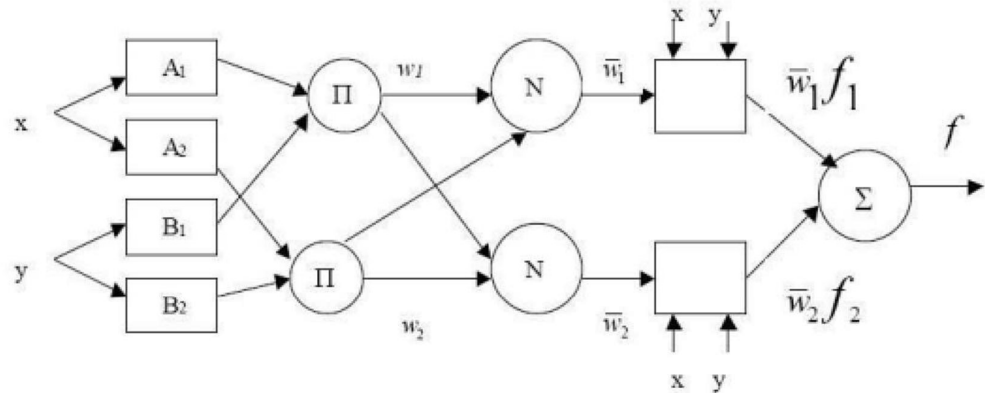


Figure: Structure of the ANFIS network.

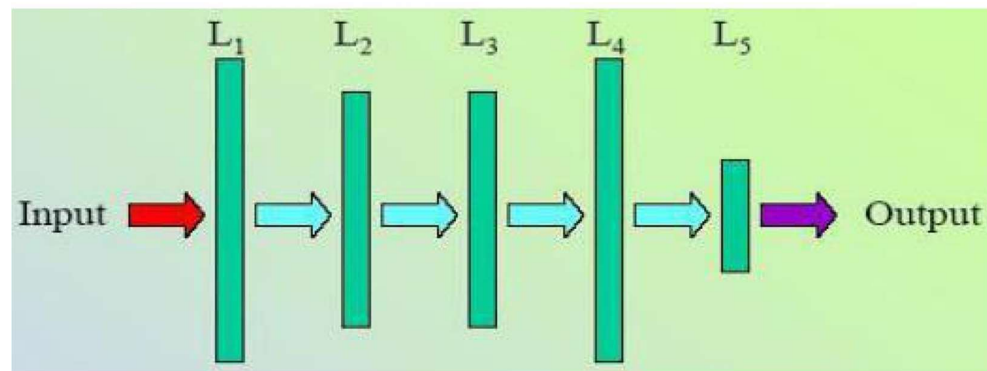


Figure : ANFIS Architecture

**Layer 1 (L1):** Each node generates the membership grades of a linguistic label.

An Illustration of a membership function is the generalised bell function: where  $\{a, b, c\}$  is the parameter set. As the values of the parameters change, the shape of the bell-shaped function varies. Parameters in that layer are called premise parameters.

**Layer 2 (L2):** Each node calculates the firing strength of each rule using the min or prod operator. In general, any other fuzzy AND operation can be used.

**Layer 3 (L3):** The nodes calculate the ratios of the rule firing strength to the sum of all the rules firing strength. The result is a normalised firing strength.

**Layer 4 (L4):** The nodes compute a parameter function on the layer 3 output. Parameters in this layer are called consequent parameters.

**Layer 5 (L5):** Normally a single node that aggregates the overall output as the summation of all incoming signals.

### Algorithm:

When the premise parameters are fixed, the overall output is a linear combination of the consequent parameters. In symbols, the output  $f$  can be written as which is linear in the consequent parameters  $c_{ij}$  ( $i = 1, 2, j = 0, 1, 2$ ). A hybrid algorithm adjusts the consequent

parameters  $c_{ij}$  in a forward pass and the premise parameters  $\{a_i, b_i, c_i\}$  in a backward pass (Jang et al., 1997). In the forward pass the network inputs propagate forward until layer 4, where the consequent parameters are identified by the least-squares method. In the backward pass, the error signals propagate backwards and the premise parameters are updated by gradient descent.

Because the update rules for the premise and consequent parameters are decoupled in the hybrid learning rule, a computational speedup may be possible by using variants of the gradient method or other optimisation techniques on the premise parameters.

### **Result/Conclusion:**

This study experiments describe the architecture of neuro fuzzy systems. Fuzzy rule based system includes the model like sugenor type fuzzy which is having neural learning capabilities.

## EXPERIMENT-9

**Aim:** Study of Derivative-free Optimization.

**Objectives:** From this experiment, the student will be able to study hybrid systems.

- Aware of the use of neuro fuzzy inference systems in the design of intelligent or humanistic systems.
- To become knowledgeable about neuro fuzzy inference systems.
- An ability to apply knowledge of computing and use of current computing techniques appropriate to the discipline.

### Theory:

Optimization Illustrations defined by functions for which derivatives are unavailable or available at a prohibitive cost are appearing more and more frequently in computational science and engineering. Increasing complexity in mathematical modelling, higher sophistication of scientific computing, and abundance of legacy codes are some of the reasons why derivative-free optimization is currently an area of great demand. Difficulties and challenges arise from multiple sources: expensive function evaluation, black-box/legacy codes, noise and uncertainty, unknown a priori function domains, and hidden constraints.

- **Derivative free Optimization**

Derivative free optimization is a subject of mathematical optimization. It may refer to Illustrations for which derivative information is unavailable, unreliable or impractical to obtain (derivative-free optimization Illustrations), or methods that do not use derivatives (derivative-free optimization methods)

- **Derivative free Optimization algorithm**

---

- Genetic algorithms (GA) ○
- Simulated Annealing (SA)

## **Genetic algorithms (GA)**

In the field of artificial intelligence, a genetic algorithm (GA) is a search heuristic that mimics the process of natural selection. This heuristic (also sometimes called a metaheuristic) is routinely used to generate useful solutions to optimization and search problems. [1] Genetic algorithms belong to the larger class of evolutionary algorithms (EA), which generate solutions to optimization problems using techniques inspired by natural evolution, such as inheritance, mutation, selection and crossover.

## **Optimization Illustrations**

In a genetic algorithm, a population of candidate solutions (called individuals, creatures, or phenotypes) to an optimization problem is evolved toward better solutions. Each candidate solution has a set of properties (its chromosomes or genotype) which can be mutated and altered; traditionally, solutions are represented in binary as strings of 0s and 1s, but other encodings are also possible. [2]

The evolution usually starts from a population of randomly generated individuals, and is an iterative process, with the population in each iteration called a generation. In each generation, the fitness of every individual in the population is evaluated; the fitness is usually the value of the objective function in the optimization problem being solved. The more fit individuals are stochastically selected from the current population, and each individual's genome is modified (recombined and possibly randomly mutated) to form a new generation. The new generation of candidate solutions is then used in the next iteration of the algorithm. Commonly, the algorithm terminates when either a maximum number of generations has been produced, or a satisfactory fitness level has been reached for the population.

## **A typical genetic algorithm requires:**

- a genetic representation of the solution domain,
- a fitness function to evaluate the solution domain.

A standard representation of each candidate solution is as an array of bits. [2] Arrays of other types and structures can be used in essentially the same way. The main property that makes these genetic representations convenient is that their parts are easily aligned

---

due to their fixed size, which facilitates simple crossover operations. Variable length representations may also be used, but crossover implementation is more complex in this case. Tree-like representations are explored in genetic programming and graph-form representations are explored in evolutionary programming; a mix of both linear chromosomes and trees is explored in gene expression programming.

Once the genetic representation and the fitness function are defined, a GA proceeds to initialize a population of solutions and then to improve it through repetitive application of the mutation, crossover, inversion and selection operators.

### **Initialization**

The population size depends on the nature of the Illustration, but typically contains several hundreds or thousands of possible solutions. Often, the initial population is generated randomly, allowing the entire range of possible solutions (the search space). Occasionally, the solutions may be "seeded" in areas where optimal solutions are likely to be found.

### **Selection**

During each successive generation, a proportion of the existing population is selected to breed a new generation. Individual solutions are selected through a fitness-based process, where fitter solutions (as measured by a fitness function) are typically more likely to be selected. Certain selection methods rate the fitness of each solution and preferentially select the best solutions. Other methods rate only a random sample of the population, as the former process may be very time-consuming.

The fitness function is defined over the genetic representation and measures the quality of the represented solution. The fitness function is always Illustration dependent. For instance, in the knapsack Illustration one wants to maximize the total value of objects that can be put in a knapsack of some fixed capacity. A representation of a solution might be an array of bits, where each bit represents a different object, and the value of the bit (0 or

1) represents whether or not the object is in the knapsack. Not every such representation is valid, as the size of objects may exceed the capacity of the knapsack. The fitness of the solution is the sum of values of all objects in the knapsack if the representation is valid, or 0 otherwise.

In some Illustrations, it is hard or even impossible to define the fitness expression; in these cases, a simulation may be used to determine the fitness function value of a phenotype (e.g. computational fluid dynamics is used to determine the air resistance of a vehicle

whose shape is encoded as the phenotype), or even interactive genetic algorithms are used.

### **Simulated annealing (SA)**

is a probabilistic technique for approximating the global optimum of a given function. Specifically, it is a metaheuristic to approximate global optimization in a large search space. It is often used when the search space is discrete (e.g., all tours that visit a given set of cities). For Illustrations where finding the precise global optimum is less important than finding an acceptable local optimum in a fixed amount of time, simulated annealing may be preferable to alternatives such as brute-force search or gradient descent.

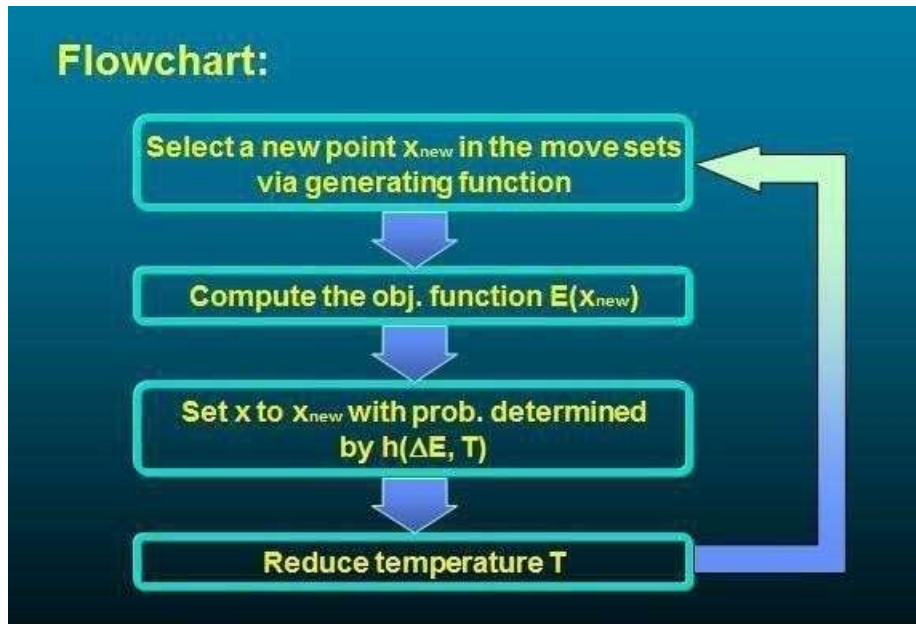
Simulated annealing interprets slow cooling as a slow decrease in the probability of accepting worse solutions as it explores the solution space. Accepting worse solutions is a fundamental property of metaheuristics because it allows for a more extensive search for the optimal solution.

#### **Terminology:**

1. Objective function  $E(x)$ : function to be optimized
2. Move set: set of next points to explore
3. Generating function: to select next point
4. Acceptance function  $h(DE, T)$ : to determine if the selected point should be accept or not. Usually  $h(DE, T) = 1/(1+\exp(DE/(cT)))$ .
5. Annealing (cooling) schedule: schedule for reducing the temperature  $T$



## Flowchart



## Result/Conclusion:

This study experiments describe the various techniques used for derivative free optimization. It also describes how to use optimization techniques in soft computing domain.

## EXPERIMENT-10

**Aim:** Study of research paper on Soft Computing and give a review report consists of abstract, introduction, state of the art, methodology, results, conclusion, reference.

**Objectives:** From this experiment, the student will be able to,

- Understand the research trends in soft computing.
- Create awareness among the students towards the recent trends in soft computing.

The learner will be able to,

- Understand the importance of research in soft computing.
- To become knowledgeable for developing soft computing applications.
- An ability to match the industry requirements in the domains of Programming with the required management skills to analyze the local and global impact of computing on individuals, organizations, and society.

### Theory:

Students can find the research papers based on the artificial neural network, hybrid systems, genetic algorithm, fuzzy system, fuzzy logic, fuzzy inference system etc.

Students need to search recent papers on any of the above mentioned topics, study it and prepare presentation on the same

### Result/Conclusion:

Through this experiment, we have understood the recent advancements and applications of various subdomains of soft computing.

### References:

1. [www.ieeeexplore.com](http://www.ieeeexplore.com)
  2. [www.Scencedirect.com](http://www.Scencedirect.com)
  3. Any open access journal
-