

Soft Computing

Paper code : ETCS-456



Maharaja Agrasen Institute Of Technology

PSP Area, Sector-22, Rohini, New Delhi -110086

Submitted To :

Dr. Sandeep Tayal
(Assistant Professor)
(Department of Computer Science)

Submitted By :

Pavitra Walia
(Enrollment Number : 42414802718)
(Branch : Computer science)
(Semester : 8th)
(Batch : 8C789)

Index

Exp. no	Experiment Name	Date of performance	Date of checking	Marks	Signature
1	Implementation of Fuzzy Operations				
2	Implementation of fuzzy relations (Max-Min Composition)				
3	Implementation of fuzzy controller (Washing Machine)				
4	To implement Mc-Culloch pitts Model using XOR				
5	Implementation of Single layer Perceptron Learning Algorithm.				
6	Implementation of unsupervised learning algorithm – Hebbian Learning				
7	Implementation Genetic Application – Match Word Finding.				
8	Study of ANFIS Architecture.				

Experiment 1

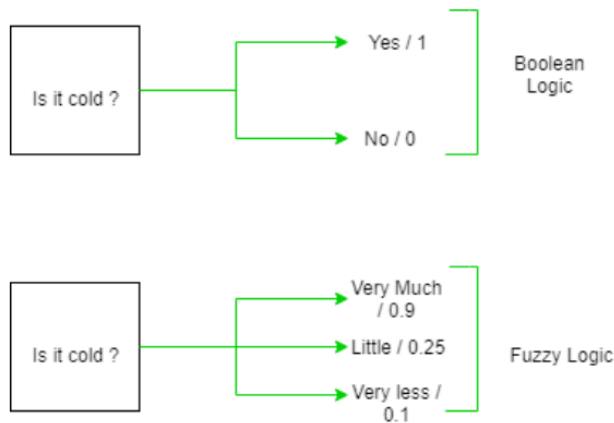
Aim : Implementation of Fuzzy Operations

Theory :

Fuzzy Logic:

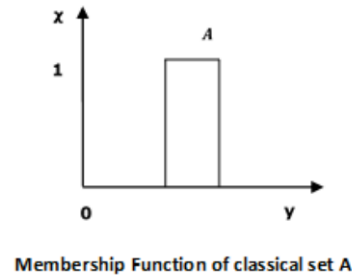
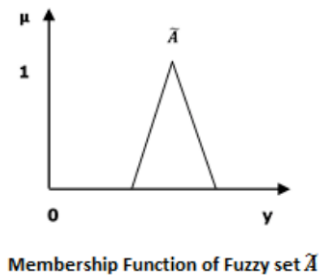
The term fuzzy refers to things which are not clear or are vague. In the real world many times we encounter a situation when we can't determine whether the state is true or false, their fuzzy logic provides a very valuable flexibility for reasoning. In this way, we can consider the inaccuracies and uncertainties of any situation.

In boolean system truth value, 1.0 represents absolute truth value and 0.0 represents absolute false value. But in the fuzzy system, there is no logic for absolute truth and absolute false value. But in fuzzy logic, there is intermediate value too present which is partially true and partially false.



Fuzzy Sets:

Fuzzy sets can be considered as an extension and gross oversimplification of classical sets. It can be best understood in the context of set membership. Basically it allows partial membership which means that it contain elements that have varying degrees of membership in the set. From this, we can understand the difference between classical set and fuzzy set. Classical set contains elements that satisfy precise properties of membership while fuzzy set contains elements that satisfy imprecise properties of membership.



Mathematical Concept:

A fuzzy set \tilde{A} in the universe of information U can be defined as a set of ordered pairs and it can be represented mathematically as:

$$\tilde{A} = \{ (y, \mu_{\tilde{A}}(y)) \mid y \in U \}$$

Here $\mu_{\tilde{A}}(y)$ = degree of membership of y in \tilde{A} , assumes values in the range from 0 to 1, i.e., $\mu_{\tilde{A}}(y) \in [0,1]$.

Operations on Fuzzy Sets:

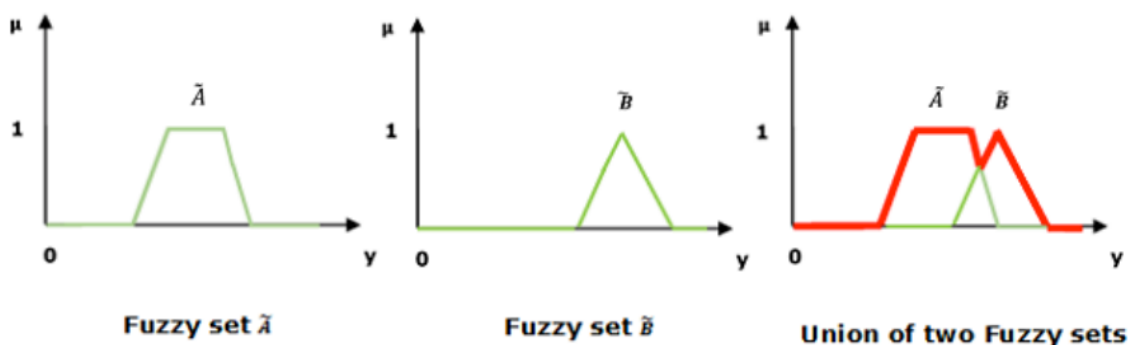
Having two fuzzy sets \tilde{A} and \tilde{B} , the universe of information U and an element \diamond of the universe, the following relations express the union, intersection and complement operation on fuzzy sets.

Union/Fuzzy ‘OR’

Let us consider the following representation to understand how the Union/Fuzzy ‘OR’ relation works –

$$\mu_{\tilde{A} \cup \tilde{B}}(y) = \mu_{\tilde{A}} \vee \mu_{\tilde{B}} \quad \forall y \in U$$

Here \vee represents the ‘max’ operation.

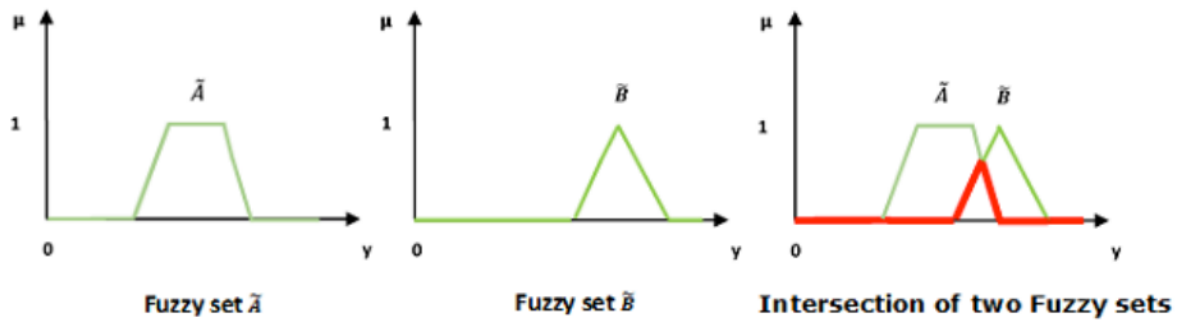


Intersection/Fuzzy 'AND'

Let us consider the following representation to understand how the Intersection/Fuzzy 'AND' relation works –

$$\mu_{\tilde{A} \cup \tilde{B}}(y) = \mu_{\tilde{A}} \wedge \mu_{\tilde{B}} \quad \forall y \in U$$

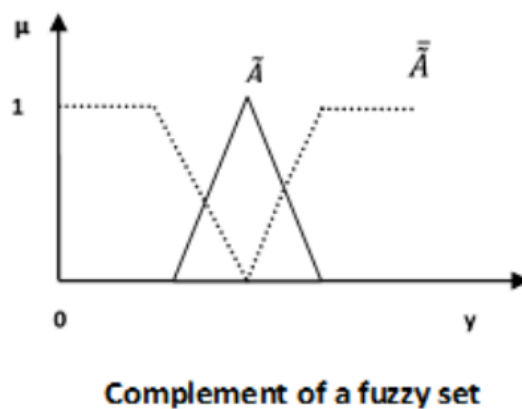
Here \wedge represents the 'min' operation.



Complement/Fuzzy 'NOT'

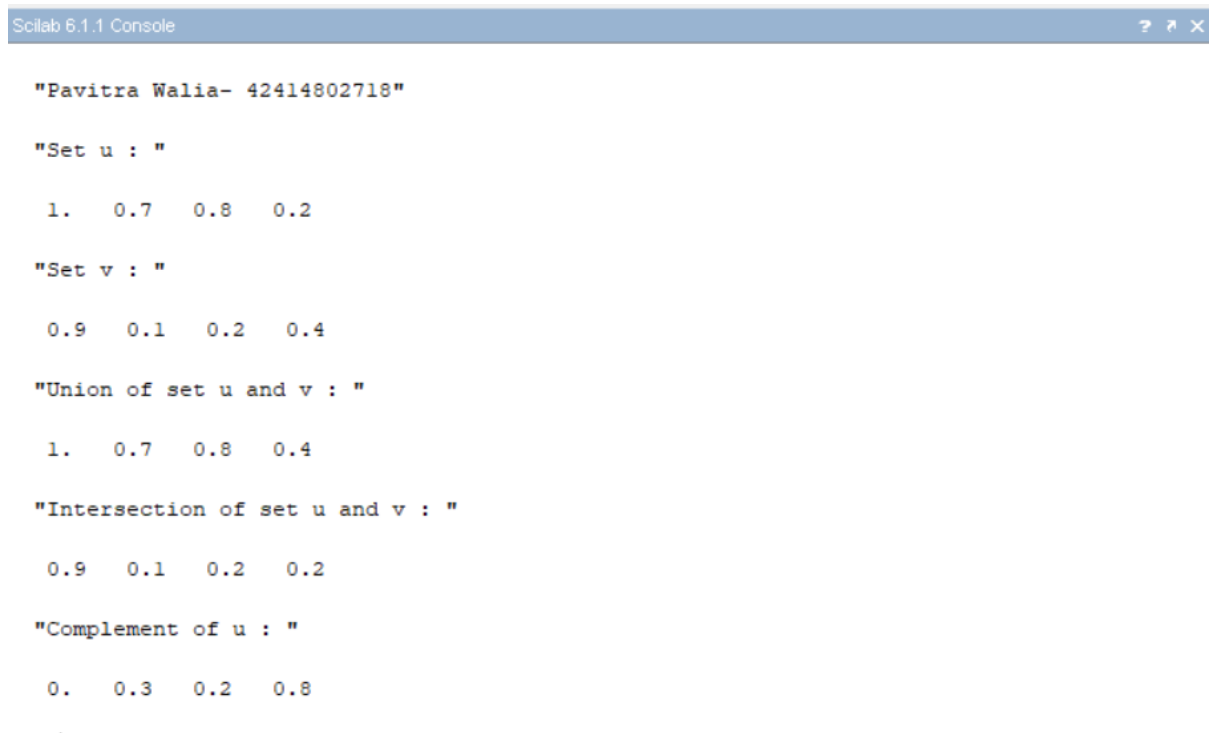
Let us consider the following representation to understand how the complement/Fuzzy 'NOT' relation works –

$$\mu_{\tilde{A}} = 1 - \mu_{\tilde{A}}(y) \quad \forall y \in U$$



Code :

```
clc;
disp('Pavitra Walia- 42414802718')
u=[1 0.7 0.8 0.2]
v=[0.9 0.1 0.2 0.4]
disp('Set u : ')
disp(u)
disp('Set v : ')
disp(v)
disp('Union of set u and v : ')
w=max(u,v)
disp(w)
disp('Intersection of set u and v : ')
p=min(u,v)
disp(p)
disp('Complement of u : ');
m=length(u);
q1=ones(m)-u
disp(q1)
```

Output :

Scilab 6.1.1 Console

```
"Pavitra Walia- 42414802718"

"Set u : "

1.    0.7    0.8    0.2

"Set v : "

0.9    0.1    0.2    0.4

"Union of set u and v : "

1.    0.7    0.8    0.4

"Intersection of set u and v : "

0.9    0.1    0.2    0.2

"Complement of u : "

0.    0.3    0.2    0.8

,
```

Viva Questions:

1. What are the properties of the Fuzzy set?

Commutativity

Associativity

Distributivity

Idempotency

Identity

Transitivity

2. What is De Morgan's Law in a Crisp Set?

For any two finite sets A and B;

- $(A \cup B)' = A' \cap B'$ (which is a De Morgan's law of union).
- $(A \cap B)' = A' \cup B'$ (which is a De Morgan's law of intersection).

3. What is the difference between the crisp set and fuzzy set?

FUZZY SET	CRISP SET
Prescribed by vague or ambiguous properties.	Defined by precise and certain characteristics.
Elements are allowed to be partially included in the set.	Element is either the member of a set or not
Used in fuzzy controllers	Digital design
Infinite-valued	bi-valued

4. List the different fuzzy set operations?

Union, intersection, complement, scalar product, vector product, Cartesian product and power.

5. What is fuzzy logic?

Fuzzy logic is a form of many-valued logic in which the truth values of variables may be any real number between 0 and 1 both inclusive. It is employed to handle the concept of partial truth, where the truth value may range between completely true and completely false

Experiment 2

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

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.

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,

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

Example: 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 $X \times Y \subseteq R$

Example: 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,

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

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

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

$$S = R \circ Q$$

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

Max min composition is then defined as,

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

Example: $R = \begin{bmatrix} 0.6 & 0.5 & 0.4 \\ 0.2 & 0.1 & 0.2 \end{bmatrix}$ and $Q = \begin{bmatrix} 0.2 & 0.6 \\ 0.1 & 0.3 \\ 0.7 & 0.5 \end{bmatrix}$ $S = \begin{bmatrix} 0.4 & 0.6 \\ 0.2 & 0.2 \end{bmatrix}$

Code :

```
clear;
clc;
disp('Pavitra Walia- 42414802718')
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 6.1.1 Console
Pavitra Walia- 42414802718
enter the first relation [1 0 1 0;0 0 0 1;0 0 0 0]

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

S=
0.  1.
0.  0.
0.  1.
0.  0.

the final max-product is "

t=
0.  1.
0.  0.
0.  0.

the final min-max output is "

h=
0.  1.
0.  0.
0.  0.

```

Viva Questions:

1. What is the main difference between probability and fuzzy logic?

Fuzzy Logic is all about the degree of truth. Probability theory has nothing to reason about things that aren't entirely true or false. In short, we can say that Fuzzy Logic captures the meaning of partial truth whereas Probability theory captures partial knowledge.

2. What are the types of fuzzy logic sets?

L-fuzzy sets

Neutrosophic fuzzy sets

Pythagorean fuzzy sets

3. Who is the founder of fuzzy logic?

Lotfi Zadeh

4. What is fuzzy arithmetic?

Fuzzy arithmetic or arithmetic of fuzzy numbers is generalization of interval arithmetic, where rather than considering intervals at one constant level only, several levels are considered in $[0, 1]$.

Experiment 3

Aim : Implementation of fuzzy controller (Washing Machine)

Theory :

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. Type of cloth 2. Type of dirt 3. Degree of dirt

Output: Wash time

Rules:

Type of Cloth	Type of Dirt	Degree of Dirt	Washing Time
Silk	Non greasy	Small	Very Short
Silk	Non greasy	Medium	Short
Silk	Non greasy	Large	Medium
Silk	Medium	Small	Medium
Silk	Medium	Medium	Long
Silk	Medium	Large	Long
Silk	Greasy	Small	Medium
Silk	Greasy	Medium	Long
Silk	Greasy	Large	Very Long
Woolen	Non greasy	Small	Short
Woolen	Non greasy	Medium	Medium
Woolen	Non greasy	Large	Long
Woolen	Medium	Small	Medium
Woolen	Medium	Medium	Medium
Woolen	Medium	Large	Long
Woolen	Greasy	Small	Long
Woolen	Greasy	Medium	Long
Woolen	Greasy	Large	Very Long
Cotton	Non greasy	Small	Short
Cotton	Non greasy	Medium	Medium
Cotton	Non greasy	Large	Long
Cotton	Medium	Small	Medium
Cotton	Medium	Medium	Long

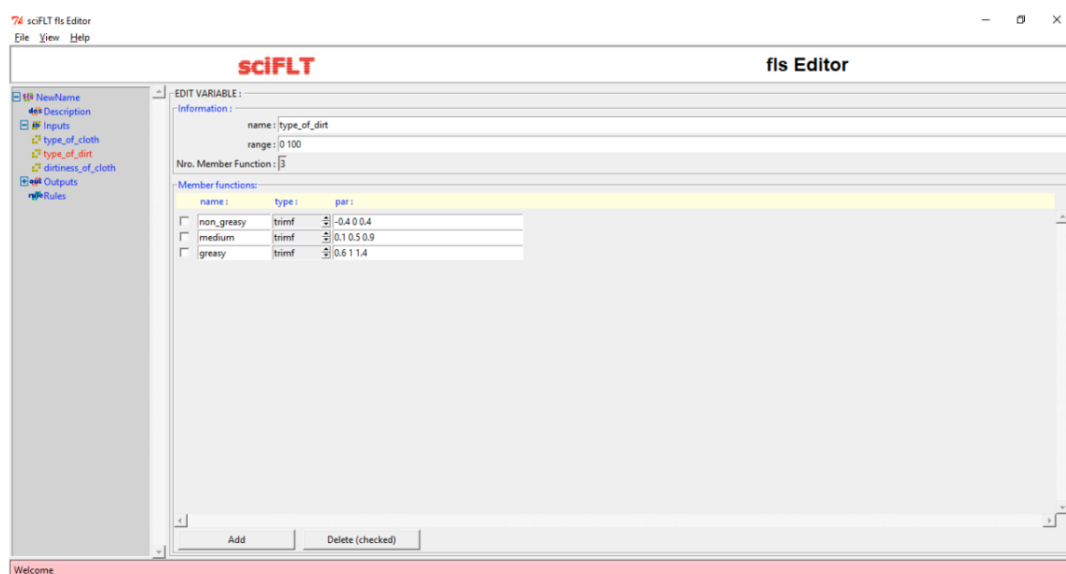
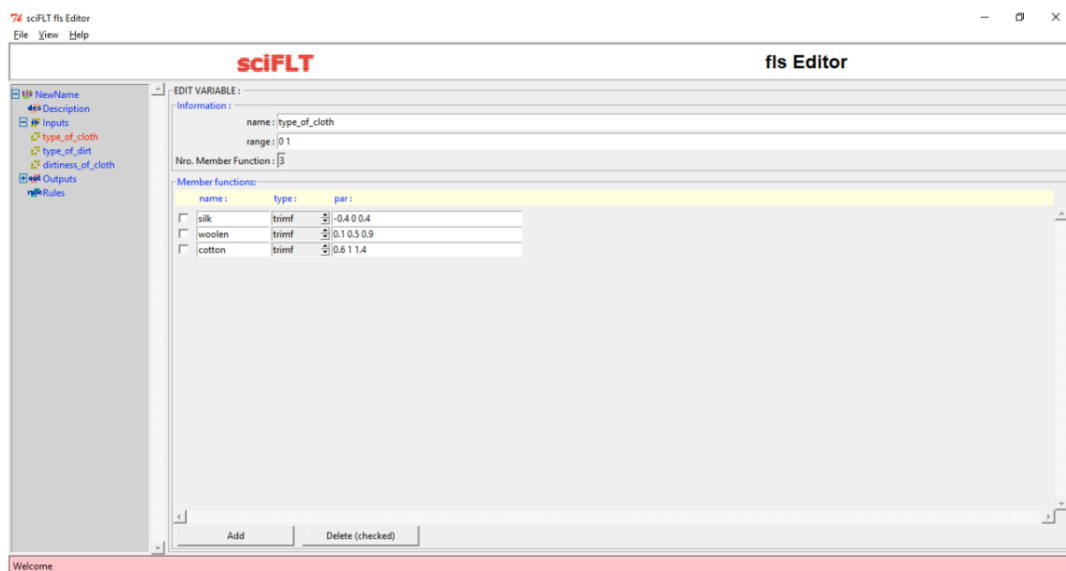
Execution:

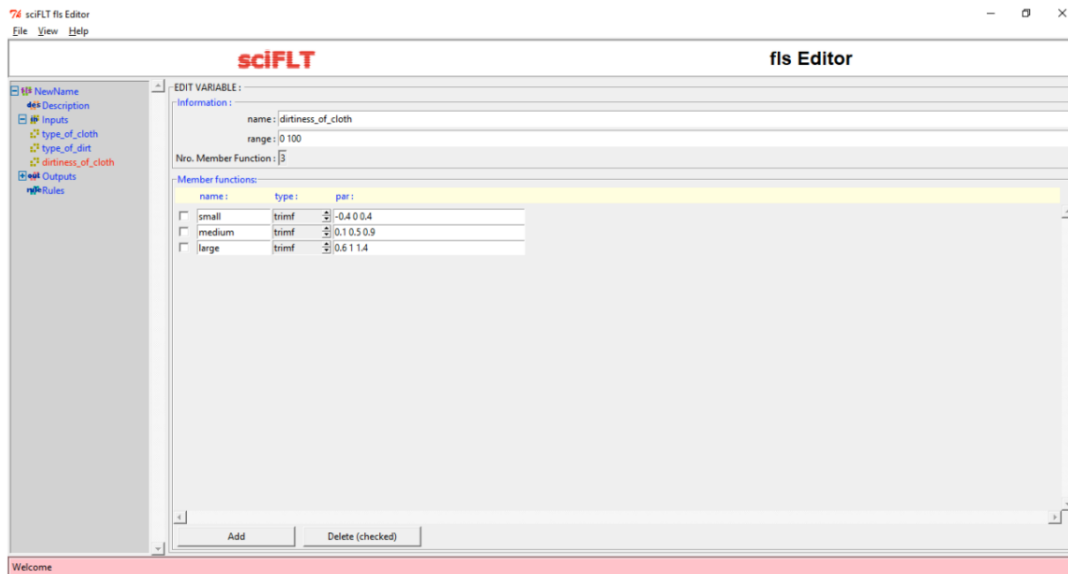
1. Run sciFLT Editor using sciFLTEditor() command.
2. Create New File.

The screenshot shows the 'sciFLT Editor' window. On the left is a sidebar with a tree view containing 'NewName', 'Description', 'Inputs', 'Outputs', and 'Rules'. The main area is titled 'Information:' and contains several text input fields: 'name: NewName', 'comment: NewComment', 'type: m', 'number of inputs: 3', 'number of outputs: 1', and 'number of rules: 27'. A 'Welcome' message is displayed at the bottom of the window.

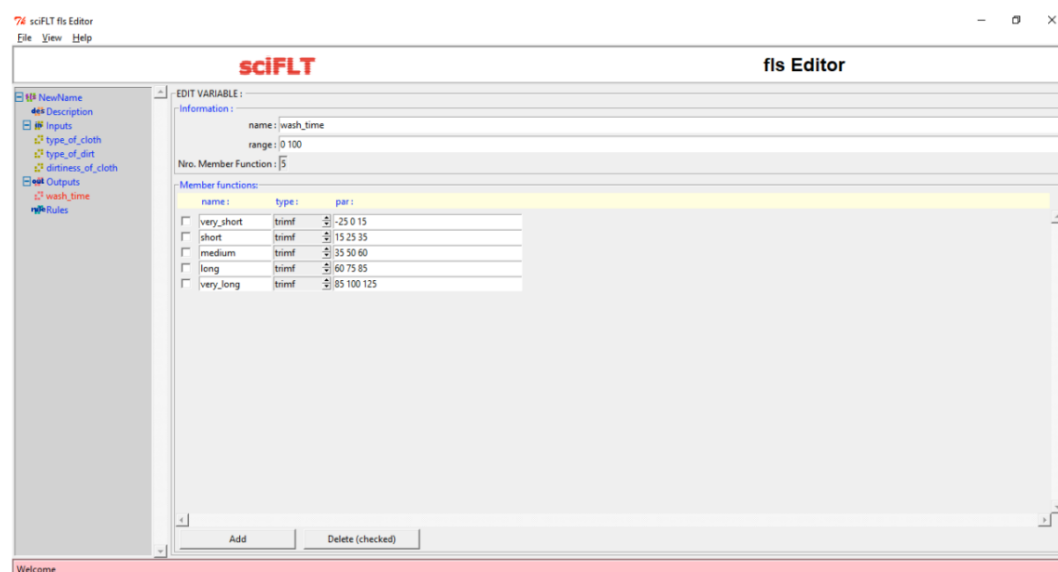
The screenshot shows the 'sciFLT Editor' window with the 'Description:' tab selected. The sidebar is the same as in the previous image. The main area contains various configuration options with radio buttons and text input fields. The 'Type' section has 'Takagi-Sugeno' and 'Mamdani' options, with 'Mamdani' selected. The 'S-Norm Class' section includes 'Dubois-Prade', 'Yager', 'Drastic sum', 'Einstein sum', 'Algebraic sum', and 'Maximum', with 'Maximum' selected. The 'T-Norm Class' section includes 'Dubois-Prade', 'Yager', 'Drastic product', 'Einstein product', 'Algebraic product', and 'Minimum', with 'Minimum' selected. The 'Complement' section has 'One', 'Yager', and 'Sugeno' options, with 'One' selected. The 'Implication Method' section has 'Minimum', 'Product', and 'Einstein Product' options, with 'Minimum' selected. The 'Aggregation Method' section has 'Maximum', 'Sum', 'Prob. OR', and 'Einstein Sum' options, with 'Maximum' selected. The 'Defuzzification Method' section has 'Centroide', 'Bisector', 'Mean of Maximum', 'Shortest of Maximum', 'Largest of Maximum', 'Weighted Average', and 'Weighted Sum' options, with 'Centroide' selected. A 'Parameter: 0' field is present in several sections. A 'Welcome' message is displayed at the bottom of the window.

3. Design the input variables.





4. Design Output Variable.



5. Define the rules.



6. Execute using loadfls and evalfls commands.

Output :

```
-->flsl=loadfls("washing.flsl")
flsl =

    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],flsl)
ans =

    58.507743

-->plotsurf(flsl)
```


74 Plot fis Surface

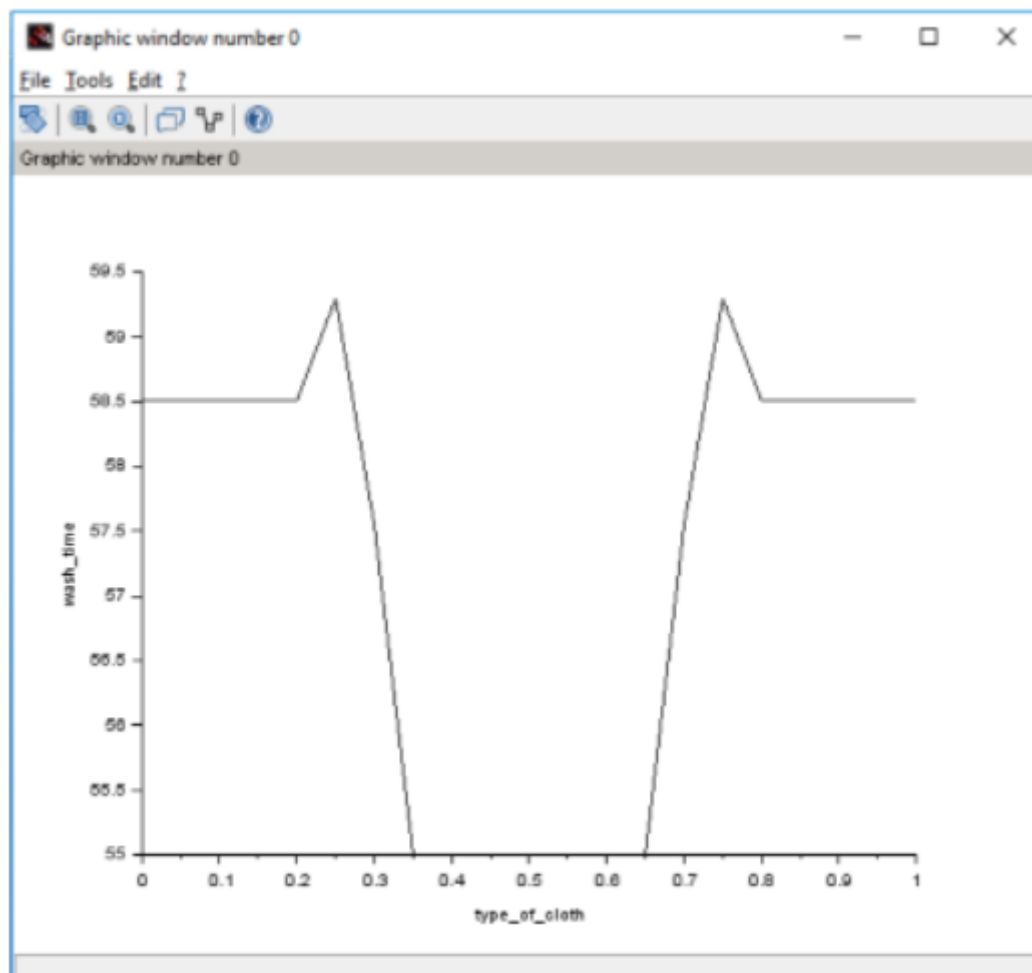
sciFLT **surfplot**

Plot the surface of : NewName

Var to plot	Nr. Points
X axe : (1) type_of_cloth	21
Y axe : (0) ----	21
Z axe : (1) wash_time	

Variable:	Value:
type_of_cloth	0.2
type_of_dirt	0.66
dirtiness_of_cloth	0.22

Plot ☐ In New window (1) Gray Color Cancel



Viva Questions:

1. What is the reason that logic function has rapidly become one of the most successful technologies for developing sophisticated control systems?

There are mainly two reasons:

- (i) Fuzzy logic applies the concept of 'certain degree' which is similar to the way human beings think. Instead of just being either true or false, fuzzy logic can be true partially and also false partially at the same time. This is similar to the human mind.
- (ii) Fuzzy logic can use exact points representing to what degree an event occurs and with fuzzy rules it generates precise outcomes.

2. What is the sequence of steps taken in designing a fuzzy logic machine?

Following is the sequence for the designing a fuzzy logic machine:

Fuzzification -> Rule Evaluation -> Defuzzification

When designing a fuzzy logic, we first have to define the fuzzy sets and make appropriate member functions. The rule evaluation comes in which matches the set to its corresponding rules.

3. What is the fuzzy inference system (FIS)?

A fuzzy inference system (FIS) is defined as a system that uses fuzzy membership functions to make a decision.

4. What is defuzzification and fuzzy controller?

Defuzzification is the process of producing a quantifiable result in Crisp logic, given fuzzy sets and corresponding membership degrees. It is the process that maps a fuzzy set to a crisp set

Experiment 4

Aim : To implement Mc-Culloch pitts Model using XOR

Theory :

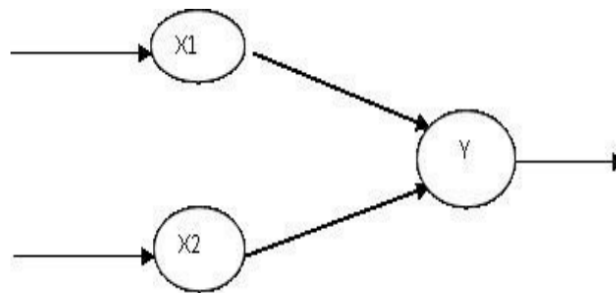
Neural network was inspired by the design and functioning of human brain and components.

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 examples. 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 problems. 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 :

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

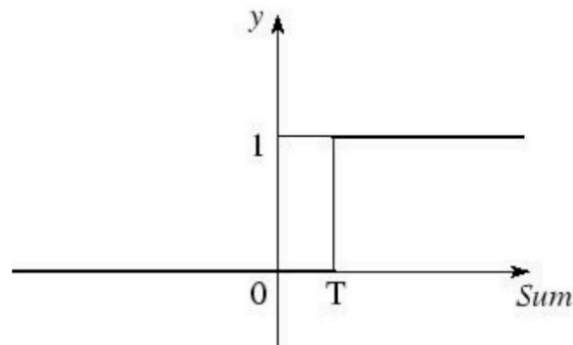
Output= function

The McCulloch-Pitts Model of Neuron:

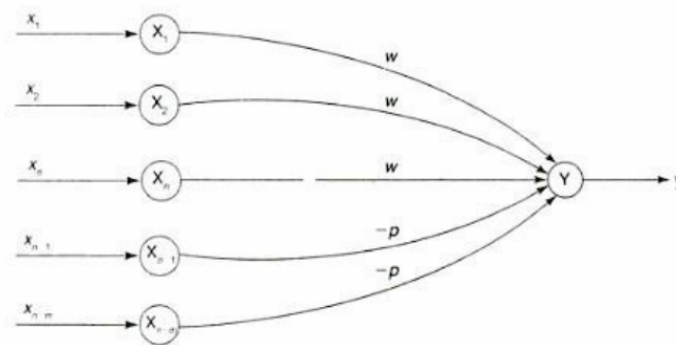
The early model of an artificial neuron is introduced by Warren McCulloch and Walter Pitts in 1943. The McCulloch-Pitts neural model is also known as linear threshold gate. It is a neuron of a set of inputs $I_1, I_2, I_3 \dots I_m$ and one output y . The linear threshold gate simply classifies the set 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:

$$\theta > n w - p$$

Output will fire if it receives $\lceil k \rceil$ or more excitatory inputs but no inhibitory inputs where $k w \geq \theta > (k-1) w$

- 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.

Code :

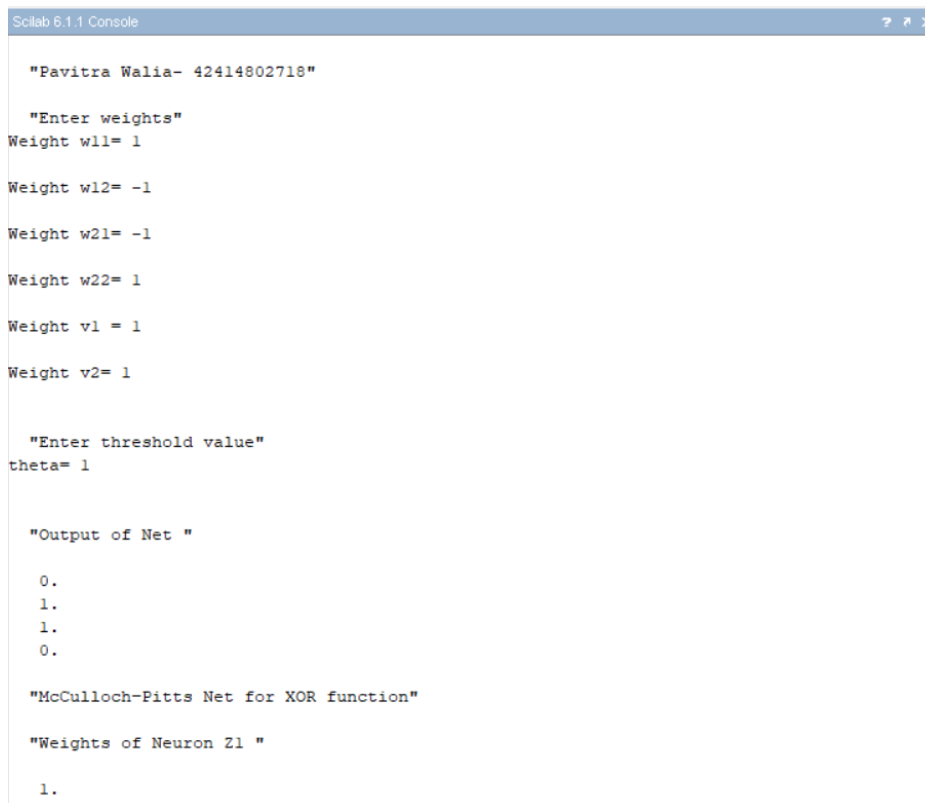
```
clear ;
clc ;
// Getting weights and thresholds value
disp('Pavitra Walia- 42414802718');
disp ('Enter weights') ;
w11 = input ( 'Weight w11=' ) ;
w12 = input ('Weight w12=' ) ;
w21 = input ('Weight w21=' ) ;
w22 = input ('Weight w22=' ) ;
v1 = input ('Weight v1 = ' ) ;
v2 = input ('Weight v2=' ) ;
disp ('Enter threshold value') ;
theta = input ('theta=' ) ;
x1 =[0 0 1 1];
x2 =[0 1 0 1];
z =[0;1;1;0];
con =1;
while con
zin1 = x1 * w11 + x2 * w21 ;
zin2 = x1 * w21 + x2 * w22 ;
for i =1:4
if zin1 ( i ) >= theta
y1 ( i ) =1;
else
y1 ( i ) =0;
end
if zin2 ( i ) >= theta
y2 ( i ) =1;
else
y2 ( i ) =0;
end
end
yin = y1 * v1 + y2 * v2 ;
for i =1:4
if yin ( i ) >= theta ;
y ( i ) =1;
```

```

else
y ( i )=0;
end
end
disp ('Output of Net ');
disp ( y );
if y == z
con =0;
else
disp ('Net is not learning to enter another set of weights and threshold values' );
w11 = input ( 'Weight w11=' );
w12 = input ( 'Weight w12=' );
w21 = input ( 'Weight w21=' );
w22 = input ( 'Weight w22=' );
v1 = input ( 'Weight v1=' );
v2 = input ( 'Weight v2=' );
theta = input ( 'theta=' );
end
end
disp ('McCulloch–Pitts Net for XOR function');
disp ( 'Weights of Neuron Z1 ' );
disp ( w11 );

```

Output :



```

Scilab 6.1.1 Console

"Pavitra Walia- 42414802718"

"Enter weights"
Weight w11= 1
Weight w12= -1
Weight w21= -1
Weight w22= 1
Weight v1 = 1
Weight v2= 1

"Enter threshold value"
theta= 1

"Output of Net "

0.
1.
1.
0.

"McCulloch-Pitts Net for XOR function"

"Weights of Neuron Z1 "

1.

```

Viva Questions:

1. What are Neural Networks? What are the types of neural networks?

A neural network is a series of algorithms that endeavors to recognize underlying relationships in a set of data through a process that mimics the way the human brain operates.

Types:

Feedforward Neural Network – Artificial Neuron.

Radial Basis Function Neural Network.

Multilayer Perceptron.

Convolutional Neural Network.

2. How are Artificial Neural Networks different from Normal Computers?

Difference between traditional computers and artificial neural networks is the way in which they function. While computers function logically with a set of rules and calculations, artificial neural networks can function via images, pictures, and concepts

3. What is a simple Artificial Neuron?

An artificial neuron is a mathematical function conceived as a model of biological neurons, a neural network. Usually each input is separately weighted, and the sum is passed through a nonlinear function known as an activation function or transfer function.

4. What is meant by training in artificial neural networks?

Once a network has been structured for a particular application, that network is trained. To start this process the initial weights are chosen randomly. Then, the training, or learning, begins. There are two approaches to training - supervised and unsupervised

Experiment 5

Aim : Implementation of Single layer Perceptron Learning Algorithm

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 problems. 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 problem. 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.

In late 1950s, Frank Rosenblatt introduced a network composed of the units that were enhanced version of McCulloch-Pitts Threshold Logic Unit (TLU) model. Rosenblatt's model of neuron, a perceptron, was the result of merger between two concepts from the 1940s, McCulloch-Pitts model of an artificial neuron and Hebbian learning rule of adjusting weights. In addition to the variable weight values, the perceptron model added an extra input that represents bias. Thus, the modified equation is now as follows:

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

where b represents the bias value.

Algorithm:

Perceptron Learning Algorithm

The perceptron learning rule was originally developed by Frank Rosenblatt in the late 1950s. Training patterns are presented to the network's inputs; the output is computed. Then the connection weights w_j are modified by an amount that is proportional to the product of the difference between the actual output, y , and the desired output, d , and the input pattern, x .

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

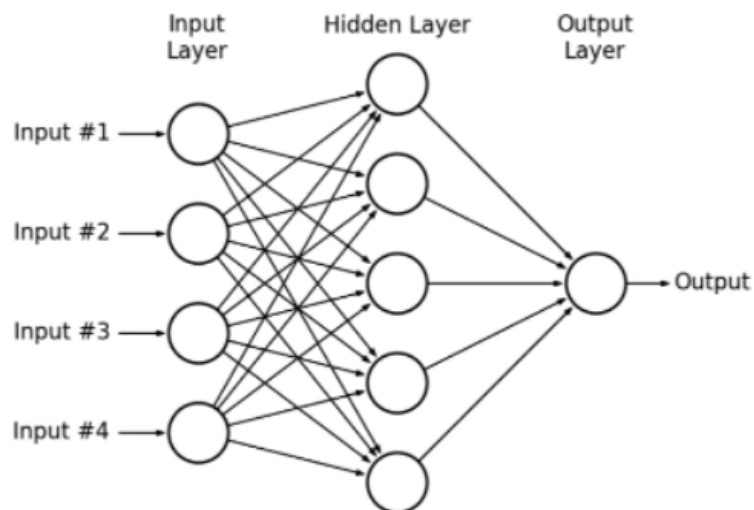
- d is the desired output,
- t is the iteration number, and
- eta is the gain or step size, where $0.0 < \eta < 1.0$

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.

Learning only occurs when an error is made; otherwise the weights are left unchanged.

Multilayer Perceptron:



Code :

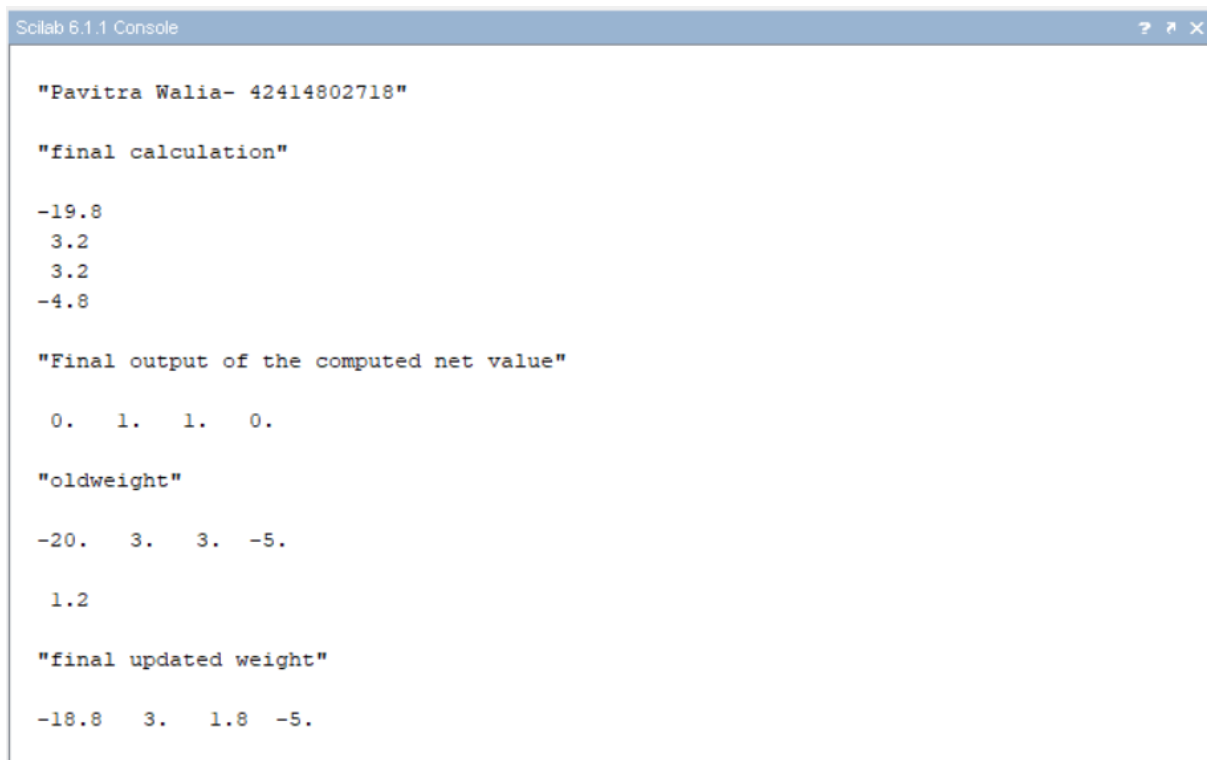
```
clear;
clc;
disp('Pavitra Walia- 42414802718')
x=[0 0 1 1; 0 1 0 1];
//input variable pass
d=[1 1 0 0];
//target output
w=[-20 3 3 -5];
//initialize weight for per input
z=[0 0 0 0];
//vector to store the calculated value of the sigma input*weight + bias
bias=0.2;
//iniatlize of bias to store the 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);
//updation to minimize the error
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
//final weight
disp('final updated weight');
disp(w);

```

Output :



```

Scilab 6.1.1 Console

"Pavitra Walia- 42414802718"

"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.

```

Viva Questions:

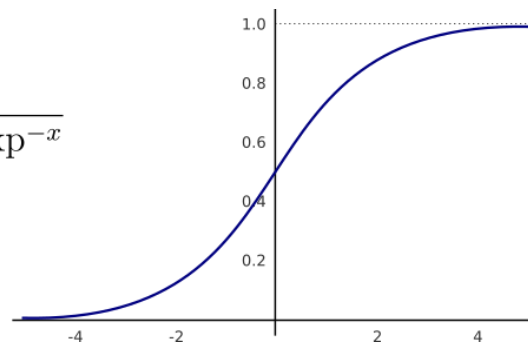
1. What is a feed forward network?

Deep feedforward networks, also often called feedforward neural networks, or multilayer perceptrons (MLPs), are the quintessential deep learning models. The goal of a feedforward network is to approximate some function f^* . These models are called feedforward because information flows through the function being evaluated from x , through the intermediate computations used to define f , and finally to the output y .

2. Write the logistic sigmoid function?

The logistic sigmoid has the following form.

$$\sigma(x) = \frac{1}{1 + \exp^{-x}}$$



3. Why use Artificial Neural Networks? What are its advantages?

ANNs have some key advantages that make them most suitable for certain problems and situations:

- ANNs have the ability to learn and model non-linear and complex relationships, which is really important because in real-life, many of the relationships between inputs and outputs are non-linear as well as complex.
- ANNs can generalize — After learning from the initial inputs and their relationships, it can infer unseen relationships on unseen data as well, thus making the model generalize and predict on unseen data.
- Unlike many other prediction techniques, ANN does not impose any restrictions on the input variables.

4. List some commercial practical applications of Artificial Neural Networks.

ANNs, due to some of its wonderful properties have many applications:

- Image Processing and Character recognition.
- Forecasting

5. What are the disadvantages of Artificial Neural Networks?

- Hardware dependence
- Unexplained behavior of the network
- Determination of proper network structure

Experiment 6

Aim : Implementation of unsupervised learning algorithm – Hebbian Learning

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 Rule, also known as Hebb Learning Rule, was proposed by Donald O Hebb. It is one of the first and also easiest learning rules in the neural network. It is used for pattern classification. It is a single layer neural network, i.e. it has one input layer and one output layer. The input layer can have many units, say n . The output layer only has one unit. Hebbian rule works by updating the weights between neurons in the neural network for each training sample.

Hebbian Learning Rule Algorithm :

- Set all weights to zero, $w_i = 0$ for $i=1$ to n , and bias to zero.
- For each input vector, $S(\text{input vector}) : t(\text{target output pair})$, repeat steps 3-5.
- Set activations for input units with the input vector $X_i = S_i$ for $i = 1$ to n .
- Set the corresponding output value to the output neuron, i.e. $y = t$.
- Update weight and bias by applying Hebb rule for all $i = 1$ to n :

$$w_i (\text{new}) = w_i (\text{old}) + x_i y$$

$$b (\text{new}) = b (\text{old}) + y$$

Code :

```
clc;
clear;
disp('Pavitra Walia- 42414802718')
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]);
plot(x(1,2), x(2,2), 'or', 'MarkerSize', 20,'MarkerFaceColor',[1 0 0]);
plot(x(1,3), x(2,3), 'or', 'MarkerSize', 20,'MarkerFaceColor',[1 0 0]);
plot(x(1,4), x(2,4), 'or', 'MarkerSize', 20,'MarkerFaceColor',[1 0 0]);

m=-(w(1)/w(2));
c=-b/w(2);
x1=linspace(-2,2,100);
x2=m*x1+c;
plot(x2,x1,'r');
axis([-2 2 -2 2]);
```

Output :



```
Scilab 6.1.1 Console

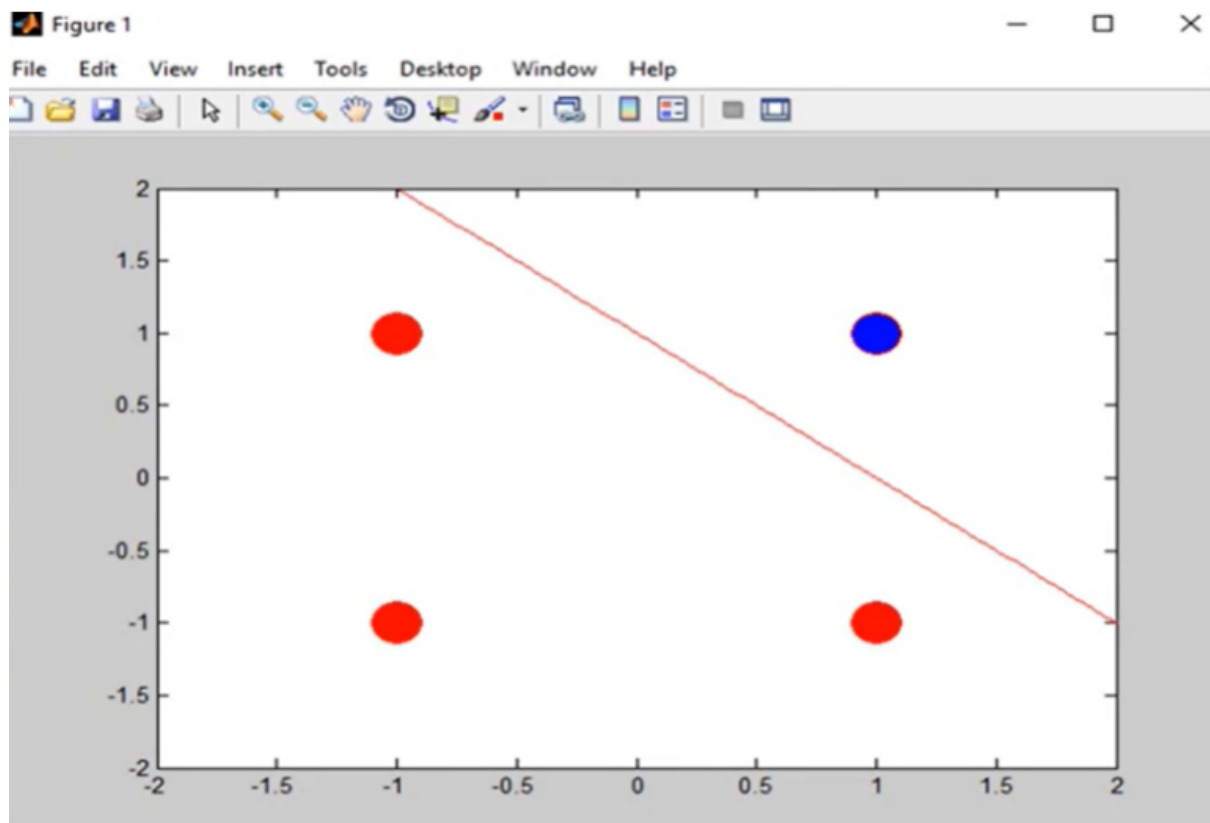
"Pavitra Walia- 42414802718"

"Final Weight Matrix:"

 2.   2.

"Final bias Values"

-2.
```



Viva Questions:

1. What is unsupervised and supervised training?

- In Supervised learning, you train the machine using data which is well "labeled."
- Unsupervised learning is a machine learning technique, where you do not need to supervise the model.

2. How do Artificial Neurons learn?

The smallest and most important unit of the artificial neural network is the neuron. As in biological neural systems, these neurons are connected with each other and together they have great processing power. Every neuron has input connections and output connections. These connections simulate the behavior of the synapses in the brain. The same way that synapses in the brain transfer the signal from one neuron to another, connections pass information between artificial neurons. These connections have weights, meaning that the value that is sent to every connection is multiplied by this factor.

3. What is the difference between neural network and fuzzy logic?

Fuzzy Logic

It is an extension of conventional set theory (also called crisp set) In conventional set logic true is 1(in general) and false is 0. But in fuzzy terms there is no concept of exact true or exact false; they are given membership.

Neural Network

It is nothing but trying to simulate our brain in the electronic domain so that it can learn like we do.

4. What is Unsupervised Hebbian learning algorithm

It is a linear feedforward neural network model for unsupervised learning with applications primarily in principal components analysis.

Experiment 7

Aim : Implementation Genetic Application – Match Word Finding.

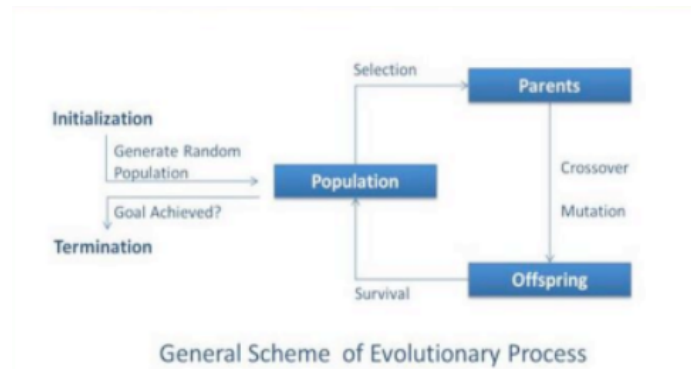
Theory :

Genetic algorithm:

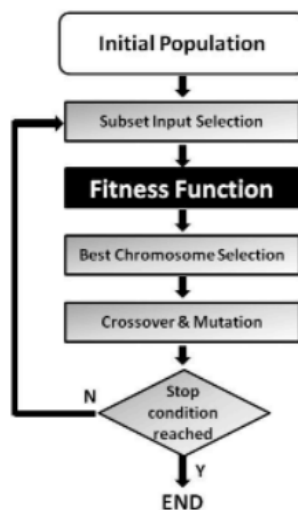
- Genetic algorithm is a search technique used in computing to find true or approximate solutions to approximate solutions to optimization & search problems.
- Genetic algorithms are inspired by Darwin's theory about evolution. Solution to a problem solved by genetic algorithms is evolved.
- Algorithm 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 example 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 problem)
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



Flowchart:



Problem 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.

Code :

```
# Python3 program to create target string, starting from
# random string using Genetic Algorithm

import random

# Number of individuals in each generation
POPULATION_SIZE = 100

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

# Target string to be generated
TARGET = "Soft Computing"

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
    print("Pavitra Walia- 42414802718")

    #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 :

```
Pavitra Walia- 42414802718
Generation: 1   String: v1fET;?1%D(s%6   Fitness: 13
Generation: 2   String: v1fET;?1%D(s%6   Fitness: 13
Generation: 3   String: S%f-_=G&]5C,Y/   Fitness: 12
Generation: 4   String: S%f-_=G&]5C,Y/   Fitness: 12
Generation: 5   String: AoXMDRoo,D(in6   Fitness: 10
Generation: 6   String: AoXMDRoo,D(in6   Fitness: 10
Generation: 7   String: AofgaRoocD(in6   Fitness: 9
Generation: 8   String: AofgaRoocD(in6   Fitness: 9
Generation: 9   String: FoftDRM
Wucin/   Fitness: 8
Generation: 10  String: kofE Co[ou(iB/   Fitness: 7
Generation: 11  String: kofE Co[ou(iB/   Fitness: 7
Generation: 12  String: kofE Co[ou(iB/   Fitness: 7
Generation: 13  String: Wofp CoX-ucin]   Fitness: 6
Generation: 14  String: Woft Co
-ucinp   Fitness: 5
Generation: 15  String: Woft Co
-ucinp   Fitness: 5
Generation: 16  String: Woft Co
-ucinp   Fitness: 5
Generation: 17  String: Woft Co
-ucinp   Fitness: 5
Generation: 18  String: =oft Comoutin8   Fitness: 3
Generation: 19  String: =oft Comoutin8   Fitness: 3
Generation: 20  String: =oft Comoutin8   Fitness: 3
Generation: 21  String: =oft Comoutin8   Fitness: 3
Generation: 22  String: =oft Comoutin8   Fitness: 3
Generation: 23  String: =oft Comoutin8   Fitness: 3
Generation: 24  String: =oft Comoutin8   Fitness: 3
Generation: 25  String: =oft Comoutin8   Fitness: 3
Generation: 26  String: =oft Computin/   Fitness: 2
Generation: 27  String: So2t Computing   Fitness: 1
Generation: 28  String: So2t Computing   Fitness: 1
Generation: 29  String: So2t Computing   Fitness: 1
Generation: 30  String: So2t Computing   Fitness: 1
Generation: 31  String: So2t Computing   Fitness: 1
```

Viva Questions:

1. Name some of the existing search methods.

- Calculus based Search
- Enumerative Search
- Random Search

2. What are the operators involved in a simple genetic algorithm?

There are three main types of operators which must work in conjunction with one another in order for the algorithm to be successful –

- Mutation
- Crossover
- Selection

3. What is reproduction?

Genetic algorithm reproduction methods for distribution system loss reduction and load balancing problems. Selected fittest parents create a new child.

4. What is crossover?

Crossover is a genetic operator used to combine the genetic information of two parents to generate new offspring

Experiment 8

Aim :Study of ANFIS Architecture.

Theory :

ANFIS:

An adaptive neuro-fuzzy inference system or adaptive network-based fuzzy inference system (ANFIS) is a kind of artificial neural network that is based on Takagi–Sugeno fuzzy inference system. Since it integrates both neural networks and fuzzy logic principles, it has potential to capture the benefits of both in a single framework. Its inference system corresponds to a set of fuzzy IF–THEN rules that have learning capability to approximate nonlinear functions. Hence, ANFIS is considered to be a universal estimator. For using the ANFIS in a more efficient and optimal way, one can use the best parameters obtained by genetic algorithm.

ANFIS Architecture:

1. Representing Takagi-Sugeno Fuzzy Model

For simplicity, we assume that the fuzzy inference system under consideration has two inputs x and y and one output z . For a first-order Takagi-Sugeno fuzzy model, a common rule set with two fuzzy if-then rules is the following:

Rule 1: If x is A_1 and y is B_1 , then $f_1 = p_1x + q_1y + r_1$;

Rule 2: If x is A_2 and y is B_2 , then $f_2 = p_2x + q_2y + r_2$;

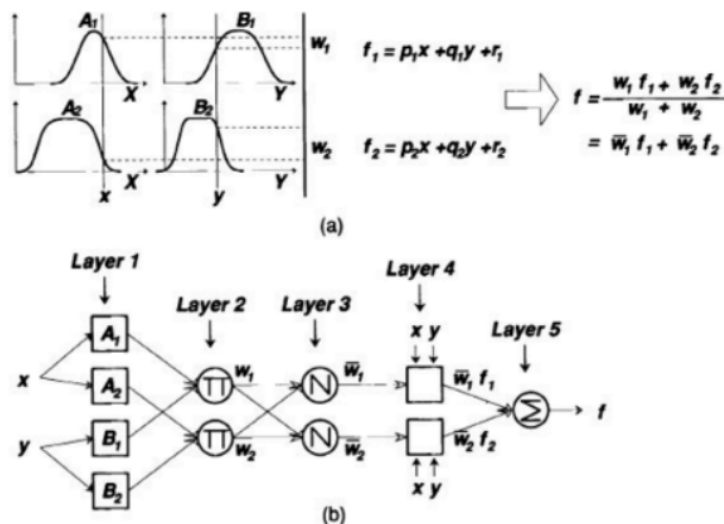


Figure 1: (a) A two inputs first order Takagi-Sugeno fuzzy model with two rules; (b) The equivalent ANFIS architecture.

Figure 1(a) illustrates the reasoning mechanism for this Takagi-Sugeno model; the corresponding equivalent ANFIS architecture is as shown in Figure 1(b), where nodes of the

same layer have similar functions, as described next. (Here we denote the output of the i th node in layer l as $O_{l,i}$)

Layer 1 Every node i in this layer is an adaptive node with a node function

$O_{1,i} = \mu_{A_i}(x)$, for $i=1,2$, or $O_{1,i} = \mu_{B_{i-2}}(y)$, for $i=3,4$, $O_{1,i} = \mu_{A_i}(x)$, for $i=1,2$, or $O_{1,i} = \mu_{B_{i-2}}(y)$, for $i=3,4$, where x (or y) is the input to node i and A_i (or B_{i-2}) is a linguistic label (such as "small" or "large") associated with this node. In other words, $O_{1,i}$ is the membership grade of a fuzzy set A ($=A_1, A_2, B_1$ or B_2) and it specifies the degree to which the given input x (or y) satisfies the quantifier A .

$$\mu_A(x) = \frac{1}{1 + |x - c_{ai}|^{2b}}, \mu_A(x) = \frac{1}{1 + |x - c_{ai}|^{2b}},$$

where $\{a_i, b_i, c_i\}$ is the parameter set. As the values of these parameters change, the bell-shaped function varies accordingly, thus exhibiting various forms of membership function for fuzzy set A . Parameters in this layer are referred to as premise parameters.

Layer 2 Every node in this layer is a fixed node labeled andfis , whose output is the product of all the incoming signals:

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

Each node output represents the firing strength of a rule. In general, any other T-norm operators that perform fuzzy AND can be used as the node function in this layer.

Layer 3 Every node in this layer is a fixed node labeled N . The i th node calculates the ratio of the i th rule's firing strength to the sum of all rules' firing strengths:

$$O_{3,i} = \bar{w}_i = \frac{w_i}{w_1 + w_2}, i=1,2. O_{3,i} = \bar{w}_i = \frac{w_i}{w_1 + w_2}, i=1,2.$$

For convenience, outputs of this layer are called normalized firing strengths.

Layer 4 Every node i in this layer is an adaptive node with a node function:

$$O_{4,i} = w_i \bar{f}_i = \bar{w}_i (p_i x + q_i y + r_i), O_{4,i} = w_i \bar{f}_i = \bar{w}_i (p_i x + q_i y + r_i),$$

where andfis is a normalized firing strength from layer 3 and $\{p_i, q_i, r_i\}$ is the parameter set of this node. Parameters in this layer are referred to as consequent parameters.

Layer 5 The single node in this layer is a fixed node labeled andfis , which computes the overall output as the summation of all incoming singals:

$$\text{overall output} = O_{5,i} = \sum_i w_i \bar{f}_i = \sum_i w_i \bar{f}_i \sum_i w_i \text{overall output} = O_{5,i} = \sum_i w_i \bar{f}_i = \sum_i w_i \bar{f}_i \sum_i w_i$$

Thus we have constructed an adaptive network that is functionally equivalent to a Sugeno fuzzy model.

2. Representing Tsukamoto Fuzzy Models:

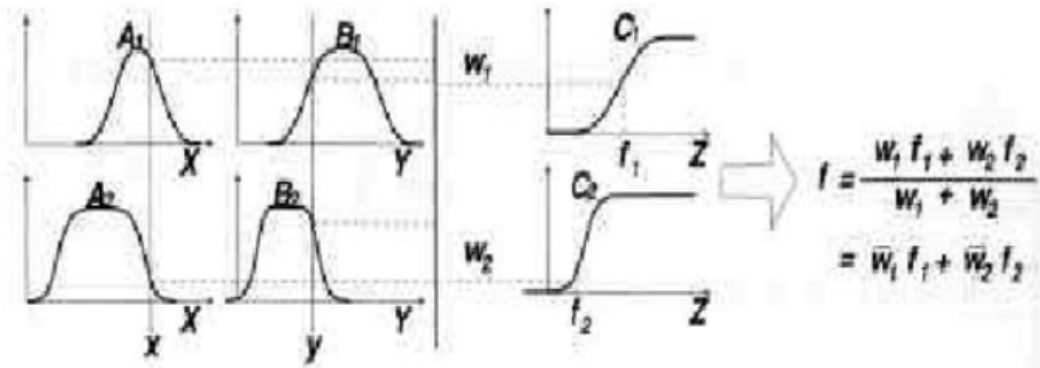


Figure 2(a)

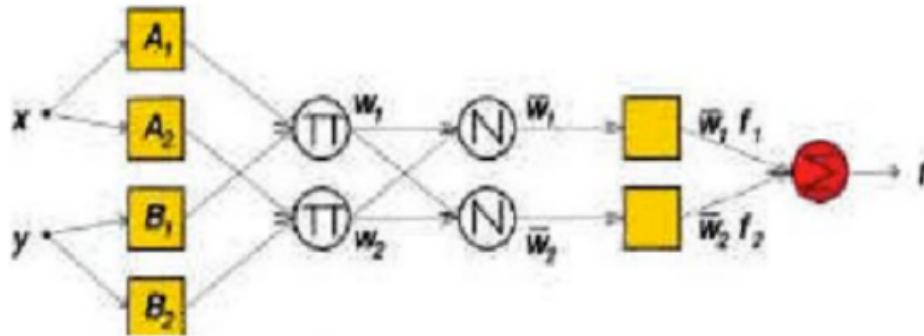


Figure 2: (b)

Figure 2: (a) A two-rule Tsukamoto fuzzy model; (b) The equivalent ANFIS architecture

The extension from TS ANFIS to Tsukamoto ANFIS is straightforward, as show in Figure 2, where the output of each rule (f_i , $i=1, 2$) is induced jointly by a consequent membership function and a firing strength.

3. Representing Mamdani Fuzzy Model

For the Mamdani fuzzy inference system with max-min composition, a corresponding ANFIS can be constructed if discrete approximations are used to replace the integrals in the centroid defuzzification scheme introduced in here. However, the resulting ANFIS is much more complicated than either TS ANFIS or Tsukamoto ANFIS. The extra complexity in structure and computation of Mamdani ANFIS with max-min composition does not necessarily imply better learning capability or approximation power. If we adopt sum-product composition and centroid defuzzification for a Mamdani fuzzy model, a corresponding ANFIS can be constructed easily based on Theorem directly without using any approximation at all.

Viva Questions:

1. What are hybrid systems?

A hybrid system is a dynamical system that exhibits both continuous and discrete dynamic behavior – a system that can both flow (described by a differential equation) and jump (described by a state machine or automaton)

2. What are fuzzy inference systems?

Fuzzy Inference Systems take inputs and process them based on the pre-specified rules to produce the outputs. Both the inputs and outputs are real valued, whereas the internal processing is based on fuzzy rules and fuzzy arithmetic.

3. How do neuro fuzzy inference systems work?

A neuro-fuzzy system is based on a fuzzy system which is trained by a learning algorithm derived from neural network theory. The learning procedure of a neuro-fuzzy system takes the semantical properties of the underlying fuzzy system into account. This results in constraints on the possible modifications applicable to the system parameters.

4. What is ANFIS architecture?

An adaptive neuro-fuzzy inference system or adaptive network-based fuzzy inference system (ANFIS) is a kind of artificial neural network that is based on Takagi–Sugeno fuzzy inference system. Since it integrates both neural networks and fuzzy logic principles, it has potential to capture the benefits of both in a single framework. Its inference system corresponds to a set of fuzzy IF–THEN rules that have learning capability to approximate nonlinear functions. Hence, ANFIS is considered to be a universal estimator