**SOFT COMPUTING LAB**

**(ETCS – 456)**

**Faculty name:** Ms. SUDHA **Student name:** Ayush Pandey

**Roll No.:** 45014802718

**Semester:** 8th Semester

**Group:** 8-C-9



Maharaja Agrasen Institute of Technology

PSP Area, Sector – 22, Rohini, New Delhi – 110085



**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 endeavor 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 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.

**Entrepreneurship**

The Institute strives to develop potential Engineers and Managers by enhancing their skills and research capabilities so that they become successful entrepreneurs and responsible citizens.



**MAHARAJA AGRASEN INSTITUTE OF TECHNOLOGY**

**COMPUTER SCIENCE AND ENGINEERING DEPARTMENT**

**VISION**

To produce “Critical Thinkers of Innovative Technology”.

**MISSION**

To foster an open, multidisciplinary and highly collaborative research environment for producing world-class engineers capable of providing innovative solutions to real life problems and fulfil societal needs.

**PRACTICAL RECORD**

**PAPER CODE : ETCS-456**

**Name of the student : Ayush Pandey**

**University Roll No. : 45014802718**

**Branch : CSE**

**Group : 8C-9**

**PRACTICAL DETAILS**

1. Experiments according to SC lab syllabus prescribed by GGSIPU

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ExpNo.** | **Experiment Name** | **Date Of Performance** | **Total Marks** | **Signature with Date** |
| 1. | Implementation of Fuzzy Operations. |  |  |  |
| 2. | Implementation of Fuzzy Relations (Maxmin 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. |  |  |  |
| 7. | Implementation of Simple Genetic Application. |  |  |  |
| 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 contains 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 Ã in the universe of information U can be defined as a set of ordered pairs and it can be represented mathematically as: Ã = {( y , μÃ ( y ) ) | y ε U}

Here μÃ (y) = degree of membership of y in Ã, assumes values in the range from 0 to 1, i.e., μÃ (y) ε [0,1].

**Operations on Fuzzy Sets:** Having two fuzzy sets A˜ and B˜, the universe of information U and an element 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 − μA˜∪B˜ (y) = μA˜ ∨ μB˜ ∀ y ε U

Here ∨ represents the ‘max’ operation.



**Intersection/Fuzzy ‘AND’**

Let us consider the following representation to understand how the Intersection/Fuzzy ‘AND’ relation works μA˜∪B˜ (y) = μA˜ ∧ μB˜ ∀ y ε U

Here ∧ represents the ‘min’ operation.



**Complement/Fuzzy ‘NOT’**

Let us consider the following representation to understand how the Complement/Fuzzy ‘NOT’ relation works− μA˜ = 1- μA˜ (y) ∀ y ε U



**CODE:**

clear;

clc;

x = (0:0.1:10)'; // The universe of discourse is [0,10]; the points are defined with a step of 0.1

u1 = gaussmf(x,[1,4]); // first membership function, gaussian type

u2 = trimf(x,[3 6.5 9]); // second membership function, triangular type

u\_union = max(u1,u2); // compute the membership degrees for the union using the "MAX" operator

u\_intersect = min(u1,u2);

u1\_fnot = 1-u1;

set(gca(),"auto\_clear","off");

subplot(4,1,1); // breaks the figure in four windows

plot(x,u1,'r');

set(gca(),"auto\_clear","off");

plot (x,u2,'m');

set(gca(),"auto\_clear","off");

legend('A','B');

title('Fuzzy sets A and B');

subplot(4,1,2); // the current plot appears in the second window

plot(x, u\_union,'color','b','linewidth',2);

title('Union');

subplot(4,1,3); // the current plot appears in the third window

plot(x, u\_intersect,'color','b','linewidth',2);

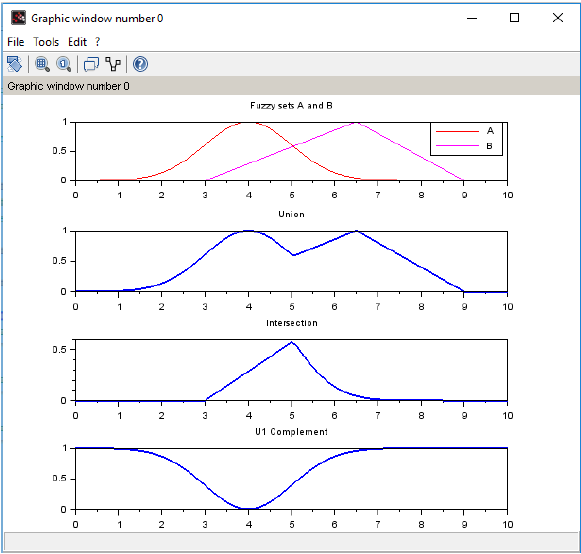
title('Intersection');

subplot(4,1,4); // the current plot appears in the fourth window

plot(x, u1\_fnot,'color','b','linewidth',2);

title('U1 Complement');

**OUTPUT:**



**VIVA QUESTIONS**

**Q. What are the properties of Fuzzy set?**

* Commutativity

Associativity

Distributivity

Idempotency

Identity

Transitivity

**Q. What is De Morgan’s Law in Crisp set?**

* For any two finite sets A and B:

(A U B)' = A' ∩ B' (which is a De Morgan's law of union).

(A ∩ B)' = A' U B' (which is a De Morgan's law of intersection).

**Q. 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 |

**Q. List the different fuzzy set operations?**

* Union, Intersection, Complement, Scalar product, Vector product, Cartesian product and Power.

**Q. 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.

**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,



The relation on this Cartesian product will be,



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



**Fuzzy relation** Let be universal sets then,



Is a fuzzy relation in

**Example:** Let X = {1, 2, 3} and Y = {1,2}

If,  then



**Max-Min Composition** Let X, Y and Z be universal sets and let R and Q be relations that relate them as,



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



Max min composition is then defined as,



Example:



and then,

**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 ")

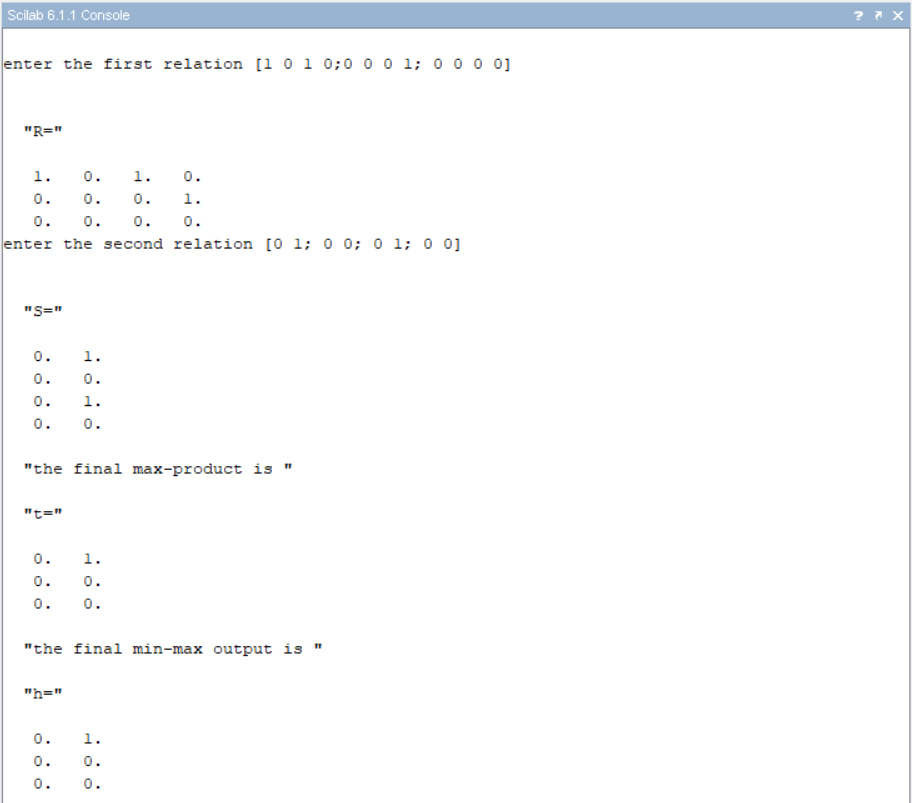
disp("h=",h);

else

disp("cannot find min-max");

end

**OUTPUT:**



**VIVA QUESTIONS**

**Q. What is the main difference between the 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.

**Q. What are formal parameters in functions?**

* L-fuzzy sets

Neutrosophic fuzzy sets

Pythagorean fuzzy sets

**Q. Who is the founder of fuzzy logic?**

* Lofti Zadeh

**Q. 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

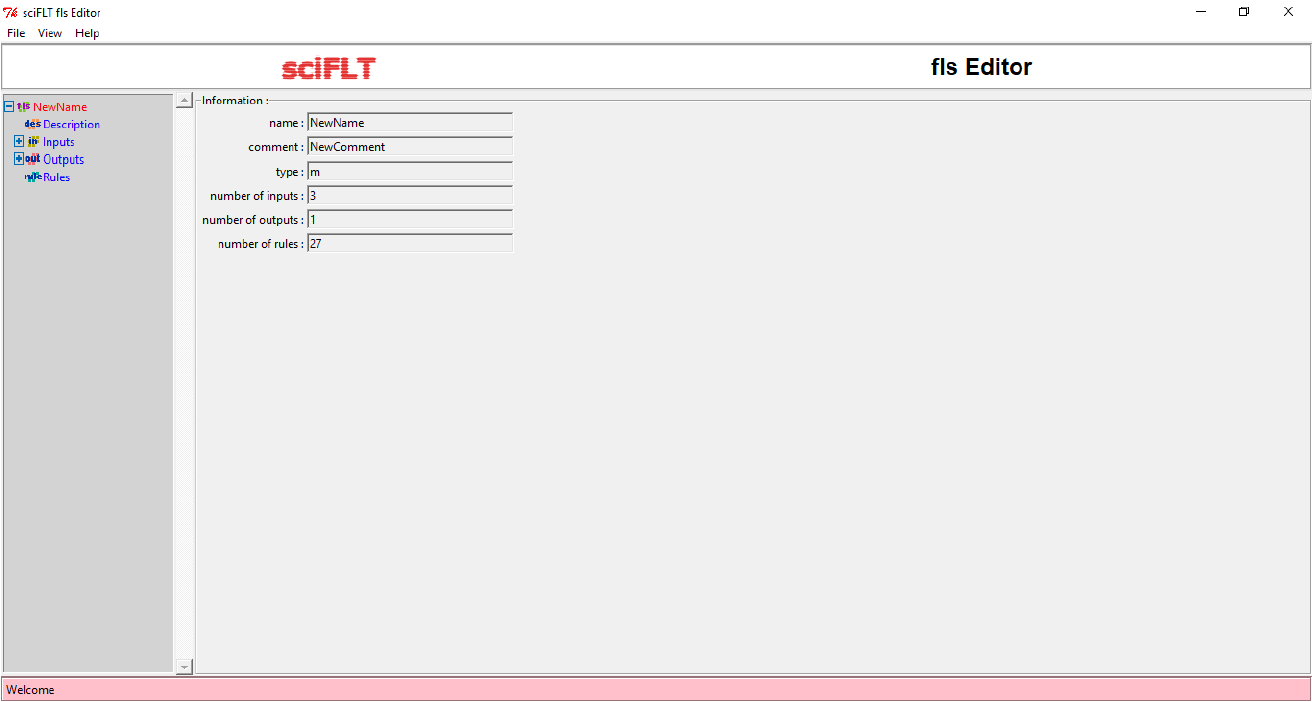
Output: Wash time

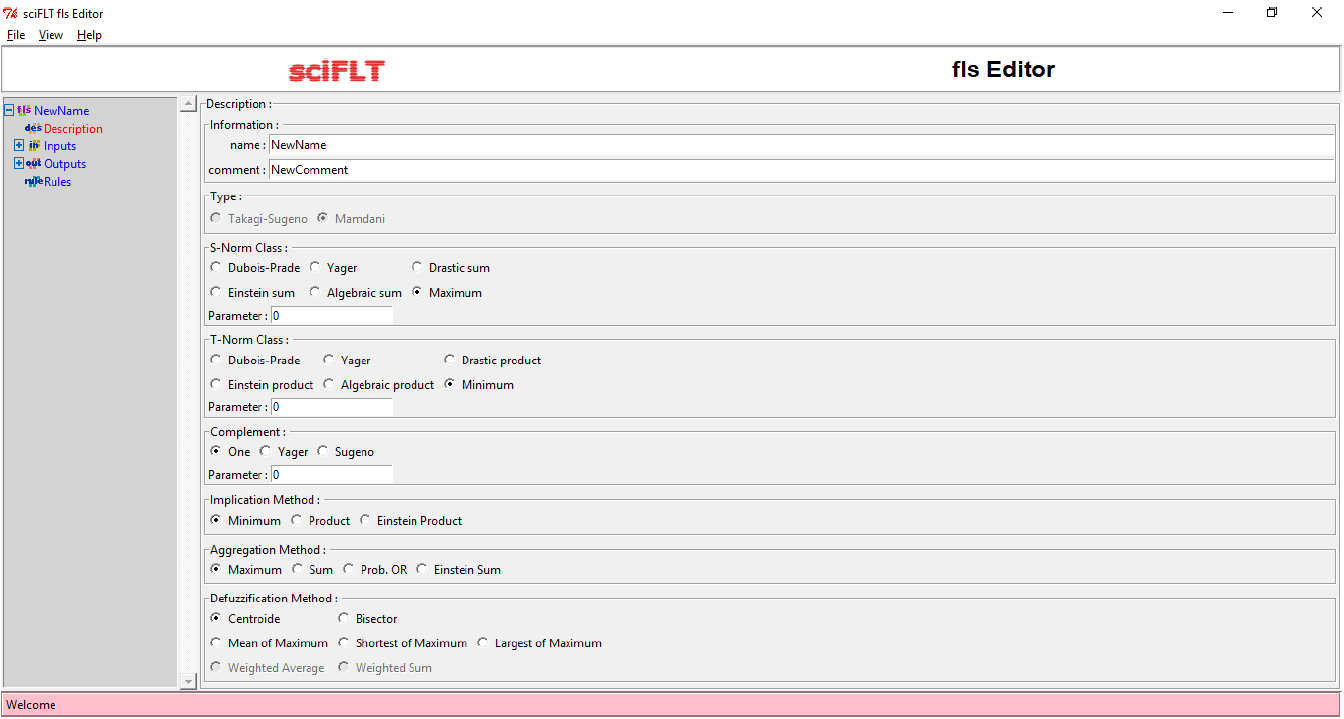
Rules:

|  |  |  |  |
| --- | --- | --- | --- |
| **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 |
| Woollen | Non greasy | Small | Short |
| Woollen | Non greasy | Medium | Medium |
| Woollen | Non greasy | Large | Long |
| Woollen | Medium | Small | Medium |
| Woollen | Medium | Medium | Medium |
| Woollen | Medium | Large | Long |
| Woollen | Greasy | Small | Long |
| Woollen | Greasy | Medium | Long |
| Woollen | 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 |
| Cotton | Medium | Large | Very Long |
| Cotton | Greasy | Small | Long |
| Cotton | Greasy | Medium | Long |
| Cotton | Greasy | Large | Very Long |

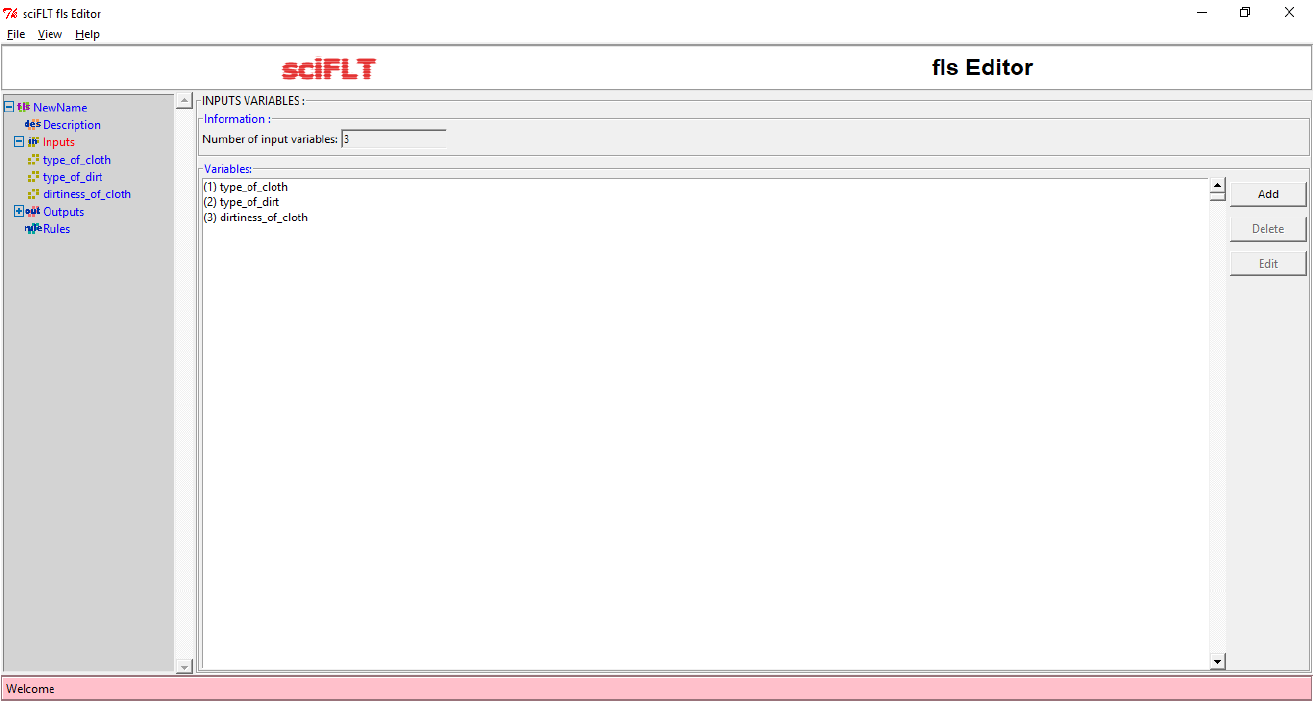
**Execution:**

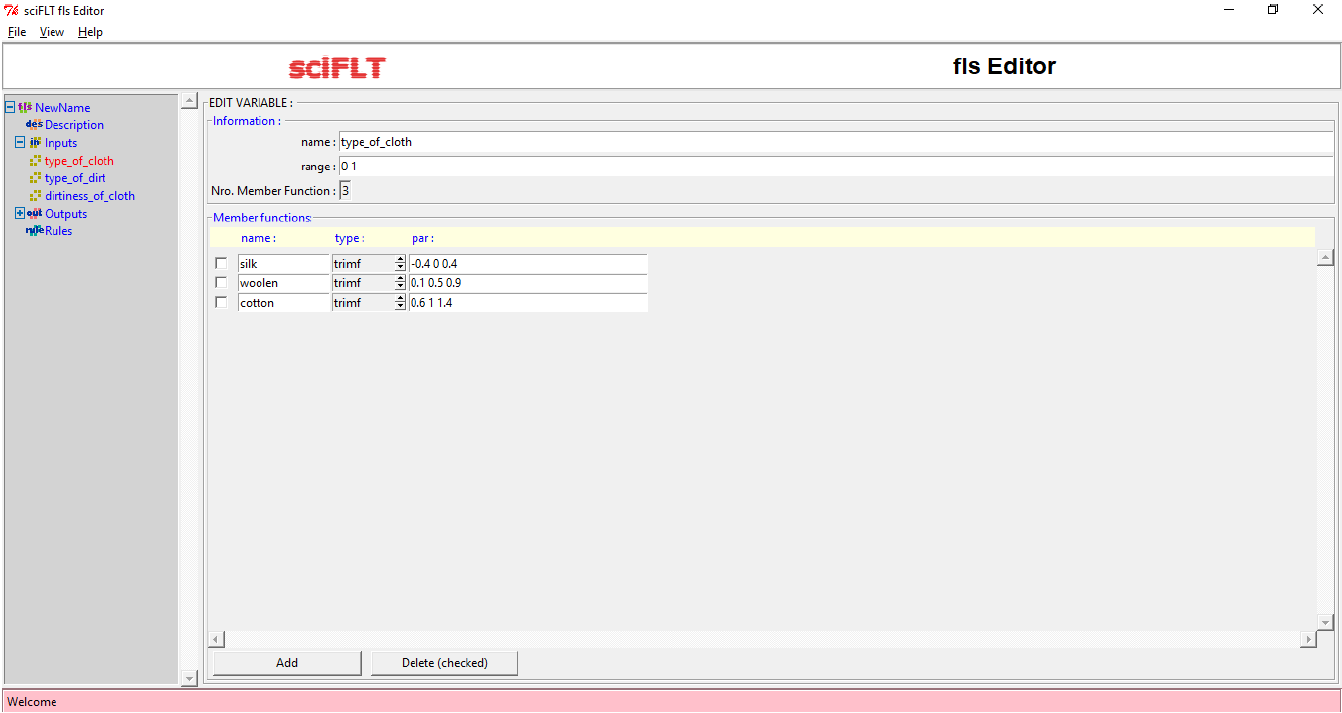
1. Run sciFLT Editor using sciFLTEditor() command.
2. Create New File.

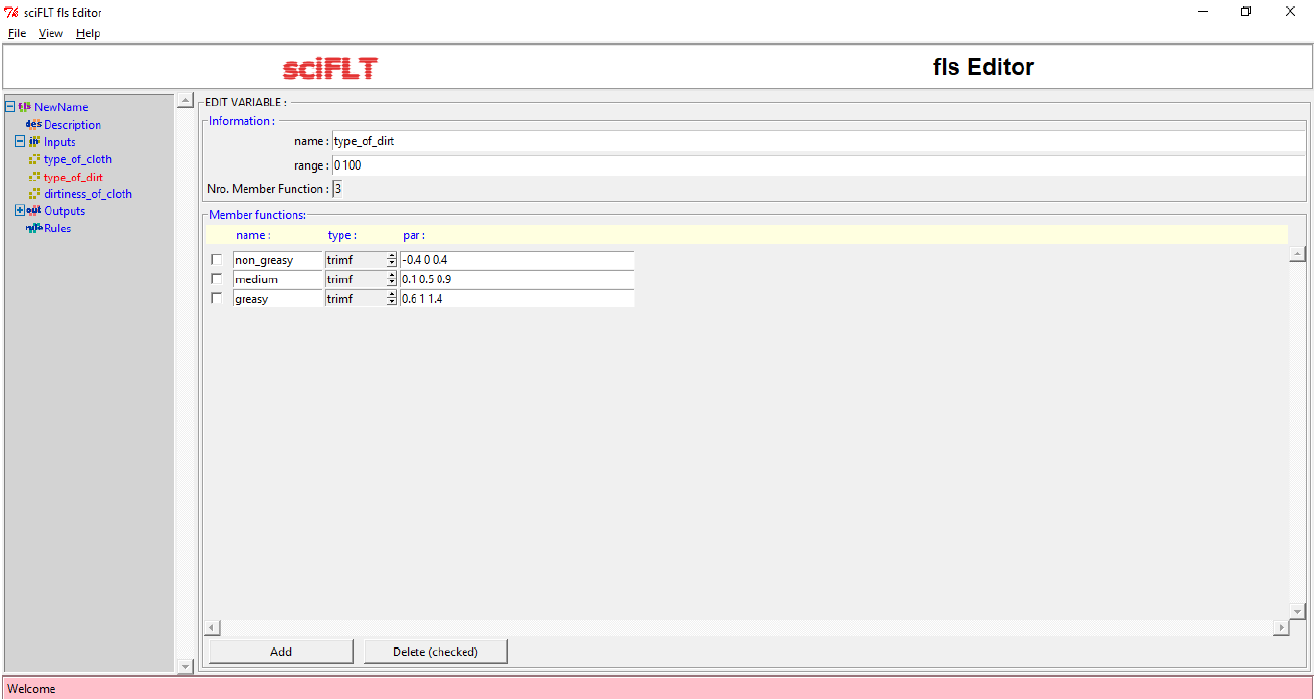


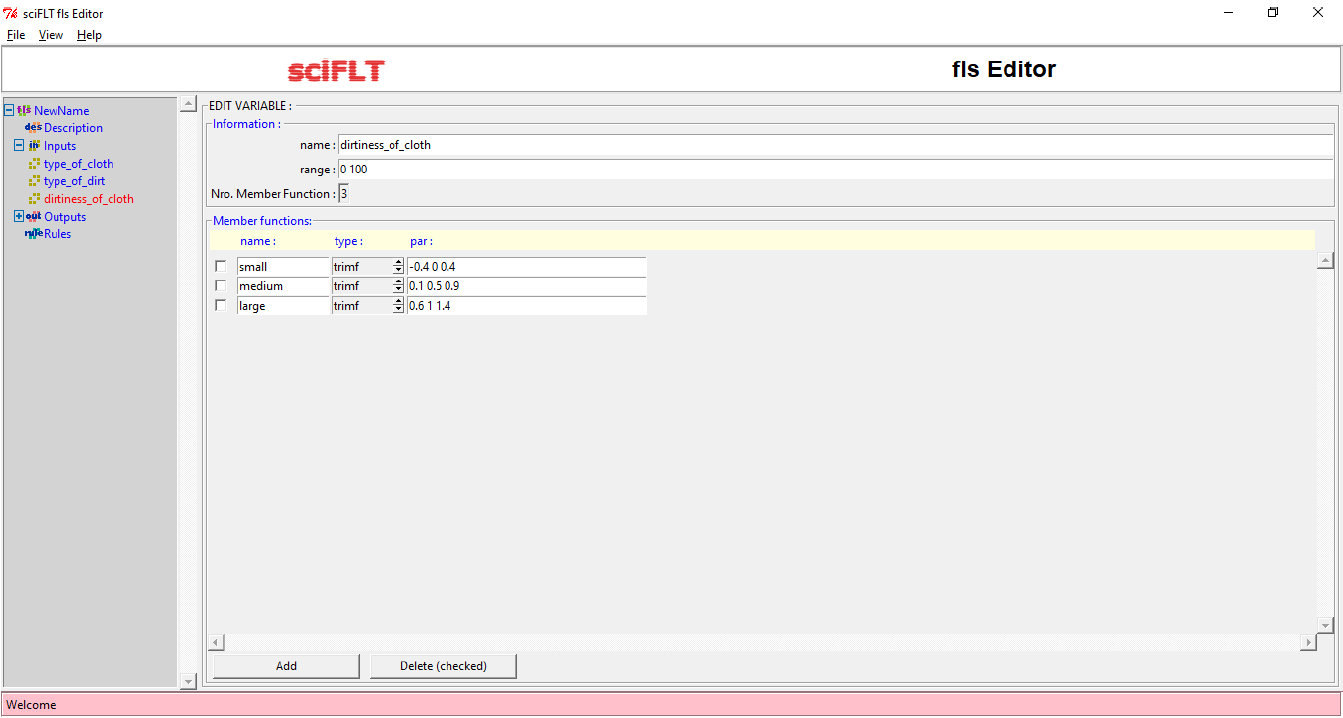


1. Design the input variables.

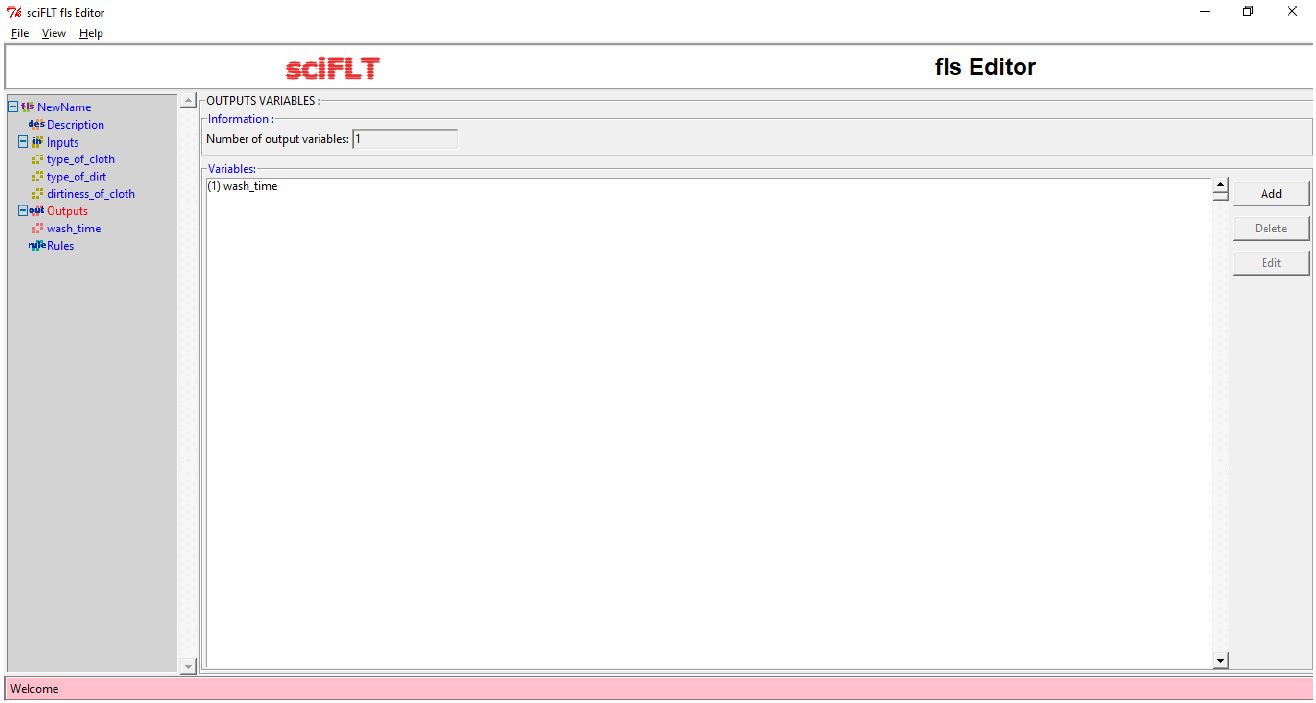


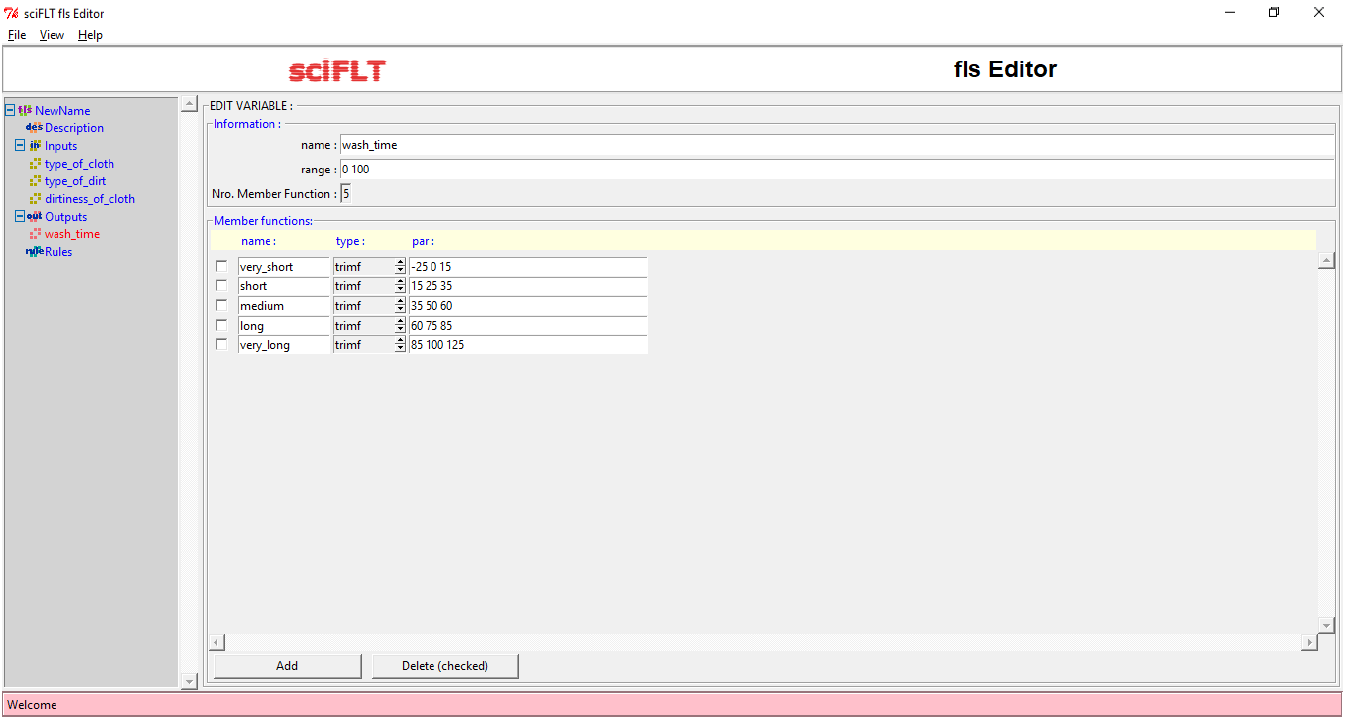






1. Design output variable.



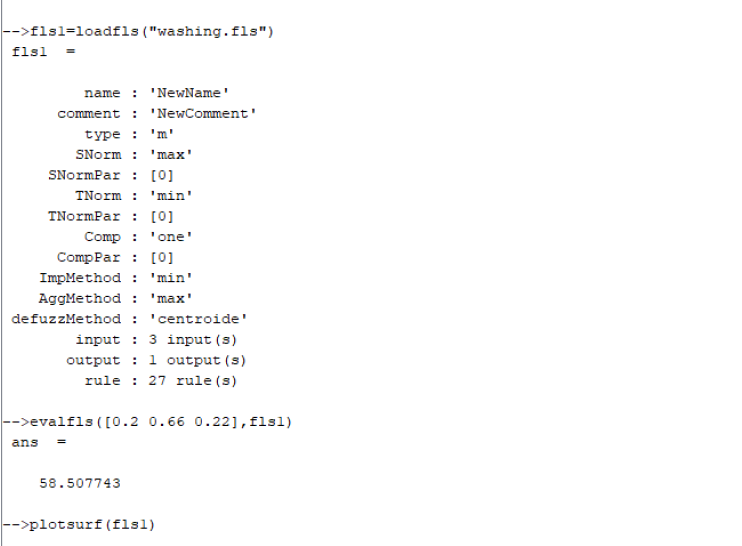
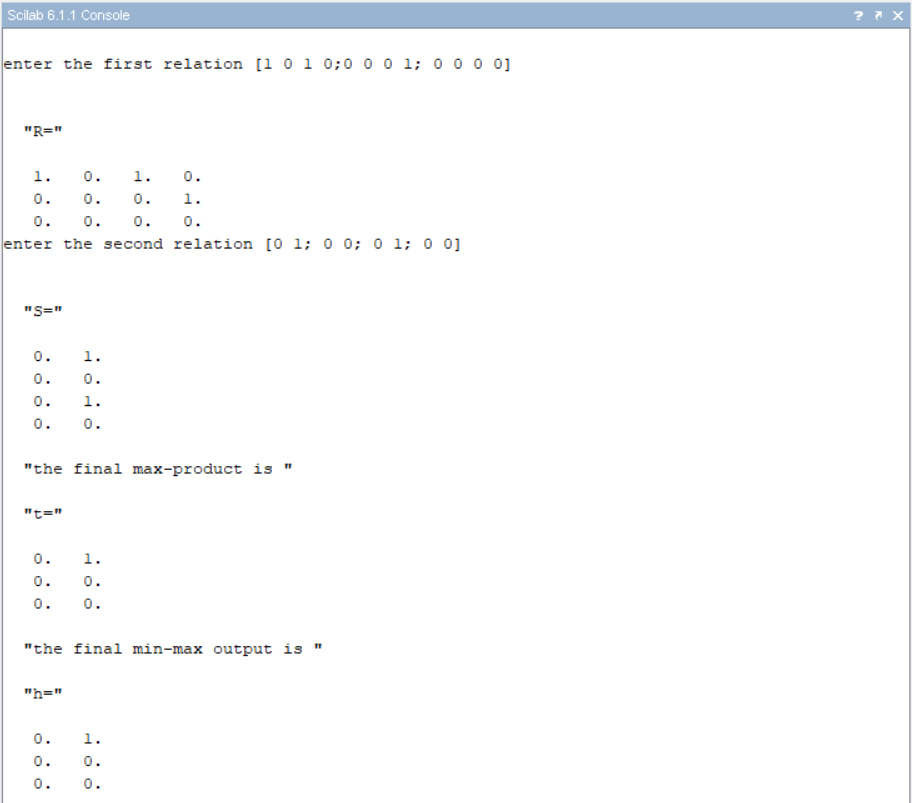


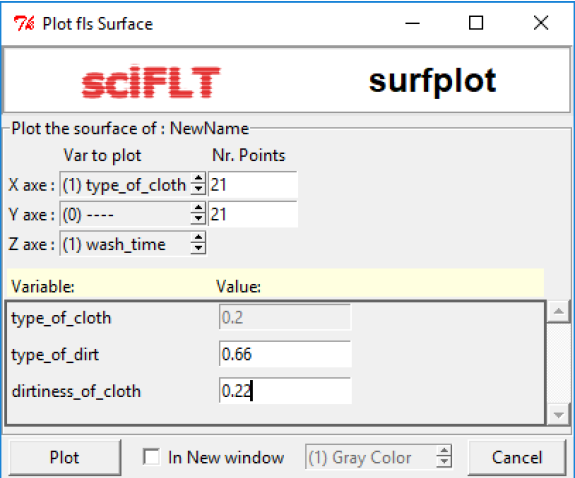
1. Define the rules.

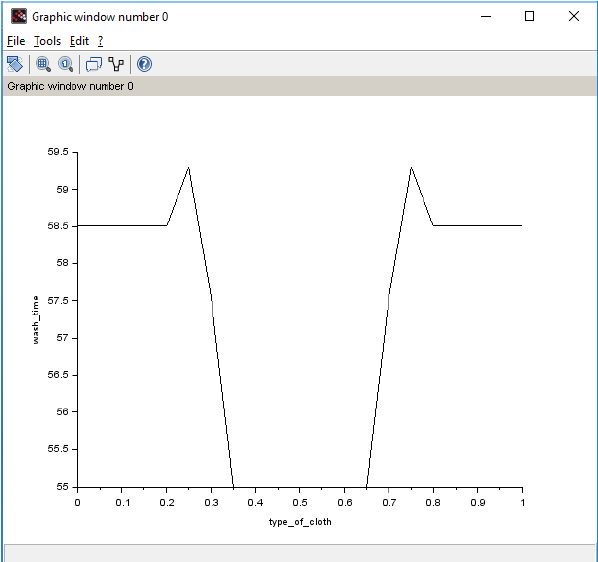


1. Execute using loadfls and evalfls commands.

**OUTPUT:**







**VIVA QUESTIONS**

**Q. 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:

Fuzzy logic applies the concept of ‘certain degree’ which is similar to the way human beings think. Instead of just 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.

Fuzzy logic can use exact points representing to what degree an event occurs and with fuzzy rules it generates precise outcomes.

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

* Following is the sequence for 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 function. The rule evaluation comes in which matches the sets to its corresponding rules.

**Q. What is fuzzy inference system (FIS)?**

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

**Q. What is defuzzification and fuzzy controller?**

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

**Experiment-4**

**Aim:** Implementation of simple neural network (McCulloh pits Model).

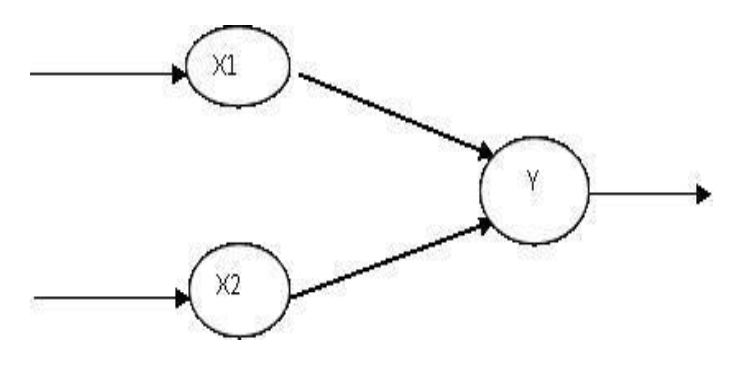
**Theory:**

Neural network was inspired by the design and functioning of human brainand 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

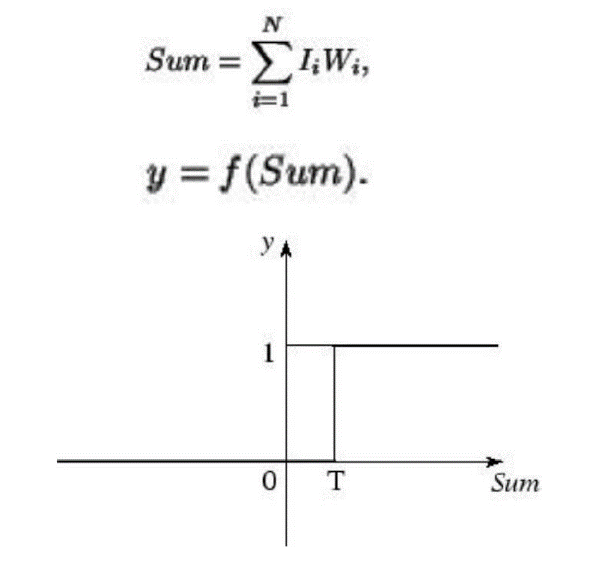
W1 and W2 – Weighted interconnection links

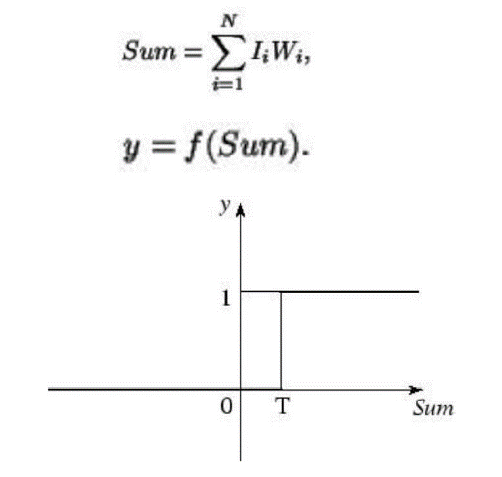
Net input calculation is: Yin = x1w1+x2w2

Output is: y = f(Yin)

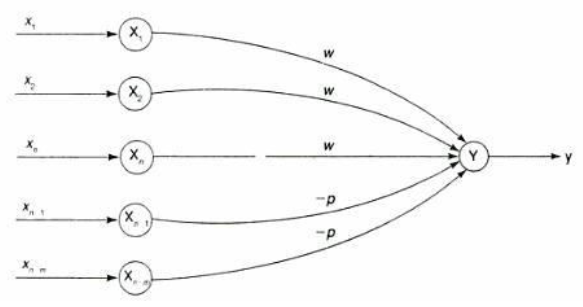
Output = function

**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 I1, I2, I3…Im 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:



W1, W2…Wm 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.

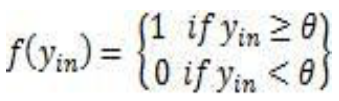
****

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 x1 to xn possess excitatory weighted connection and Xn+1 to xn+m has inhibitory weighted interconnections.

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

****

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

θ > nw – p

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

kw ≥ θ > (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.

**Problem Statement:** Generate AND-NOT function using McCulloch-Pitts neural net by a SCILAB program.

**CODE:**

clear;

clc;

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');

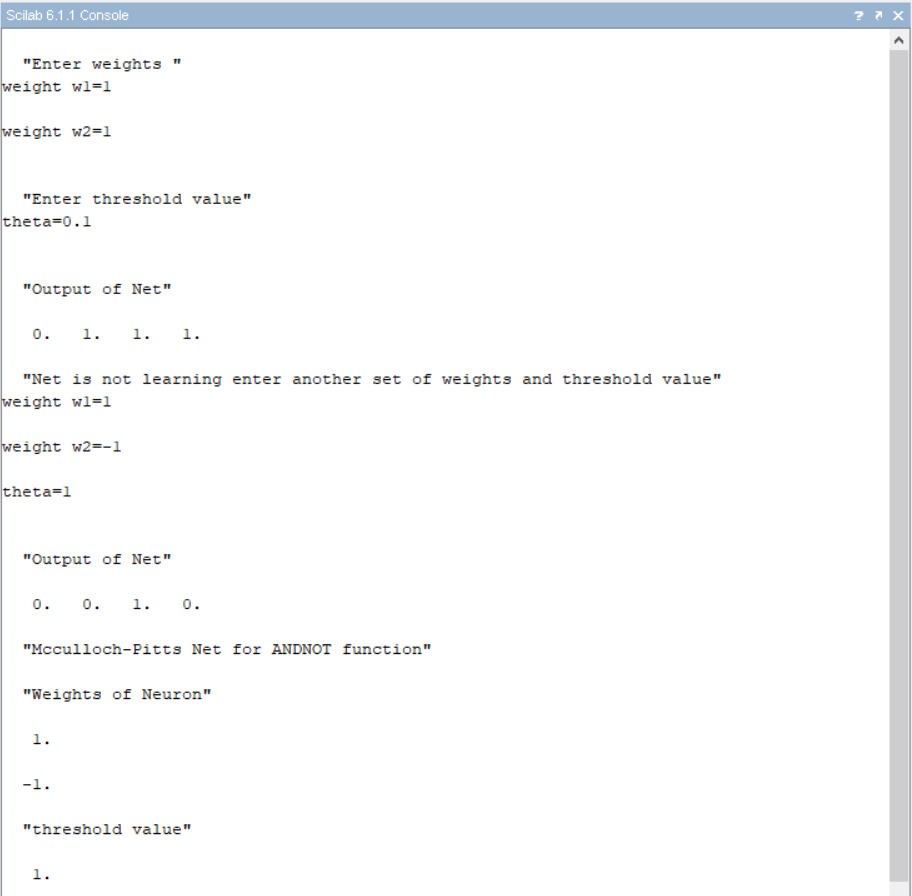
disp(w1);

disp(w2);

disp('threshold value');

disp(theta);

**OUTPUT:**



**VIVA QUESTIONS**

**Q. What are Neural Networks? What are the types of neural networks?**

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

Types:

Feed-forward Neural Network – Artificial Neuron

Multilayer Perceptron

Back-propagation Neural Network

Recurrent Neural Network

Convolution Neural Network

**Q. 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 functions logically with a set of rules and calculations artificial neural networks can function via images, pictures, and concepts.

**Q. What us a simple Artificial Neuron?**

* An artificial neuron is a mathematical function conceived as a model of biological neurons, a neural network. The artificial neuron receives one or more inputs and sums them to produce an output. Usually each input is separately weighted, and the sum is passed through a non-linear function known as activation function or transfer function.

**Q. What is meant by training of 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.

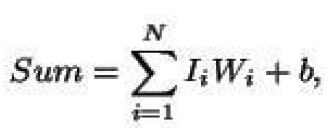
**Experiment-5**

**Aim:** Implementation of 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:



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 wj 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:



Where

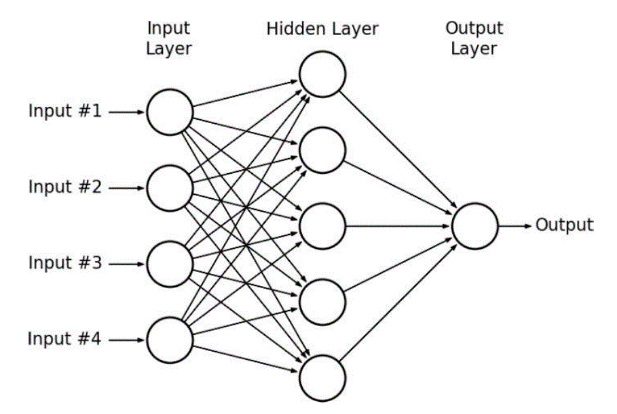
* d is the desired output,
* t is the iteration number, and
* eta is the gain or step size, where 0.0 < n < 1.0

1. Repeat steps 2 and 3 until:

* the iteration error is less than a user-specified error threshold or
* 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;

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; *//initialize of bias to store the 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);

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;

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;

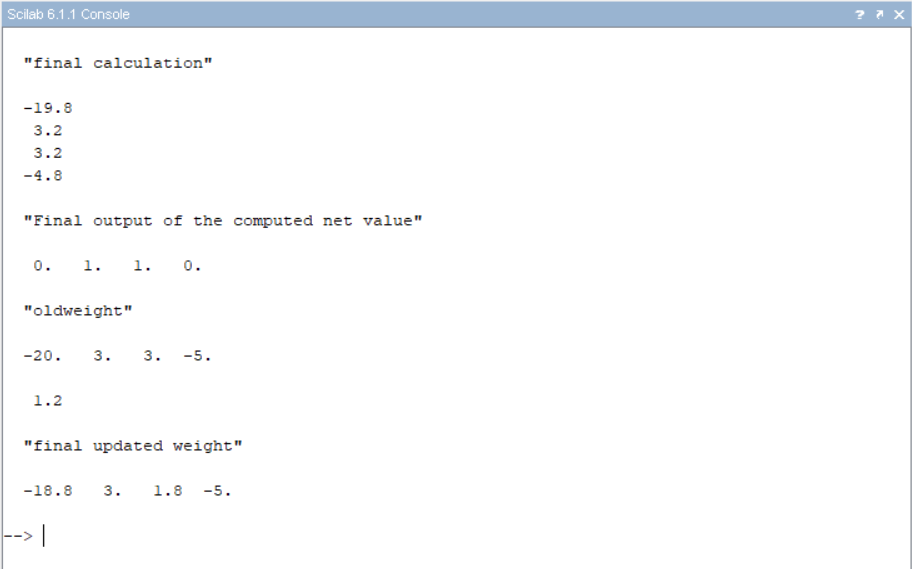
end

end

disp('final updated weight');

disp(w);

**OUTPUT:**



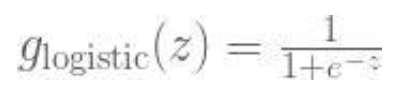
**VIVA QUESTIONS**

**Q. What is feed forward network?**

* Deep feedforward networks, also often called feedforward neural networks, or multiplayer 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.

**Q. Write the logistic sigmoid function?**

* The logistic sigmoid has the following form:



**Q. 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.

**Q. List some commercial practical applications on Artificial Neural Networks.**

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

Image Processing and Character recognition

Forecasting

**Q. What are the disadvantages of Artificial Neural Networks?**

* Hardware dependence

Unexplained behaviour 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 models 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 patters 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:** In 1949, Donald Hebb proposed one of the key ideas in biological learning, commonly known as Hebb‘s Law. Hebb‘s Law states that if neuron i is near enough is excite enough to excite neuron j & repeatedly participates in its activation, the synaptic connection between these two neurons is strengthened & neuron j becomes more sensitive to stimuli from neuron i.

Hebb‘s Law can be represented in the form of two rules:

1. If two neurons on either side of a connection are activated synchronously, then the weight of that connection is increased.
2. If two neurons on either side of a connection are activated asynchronously, then the weight of that connection is decreased.

Hebb‘s law provide basis for learning without a teacher. Learning here is a local phenomenon occurring without feedback from the environment.

* Using Hebb‘s Law we can express the adjustment applied to weight wij at iteration p in the following form:
* As a special case, we can represent Hebb‘s Law as follows:

where α is the learning rate parameter.

* Hebbian learning implies that weights can only increase. To resolve this problem, we might impose a limit on the growth of synaptic weights. It can be done by introducing non-linear forgetting factor into Hebb‘s Law:

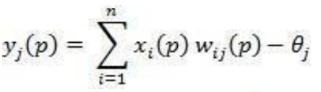


where φ is the forgetting factor.

**Hebbian learning algorithm**

**Step 1:** Initialization

Set initial synaptic weights and thresholds to small random values, say in an interval [0, 1].

**Step 2:** Activation

Compute the neuron output at iteration p

Where n is number of neuron inputs, &  is the threshold value of neuron j.

**Step 3:** Learning

Update the weights in the network

Where  is the weight correction at iteration p.

**Step 4:** Iteration

Increase iteration p by one, go back to step 2.

**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');

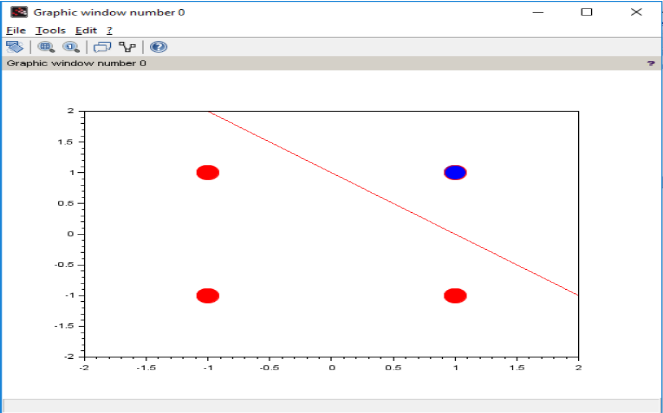
a=gca() ;//get the current axes

a.box="on";

a.data\_bounds=[-2,-2;2,2]; //define the bounds

**OUTPUT:**





**VIVA QUESTIONS**

**Q. What is unsupervised and supervised training?**

* In Supervised learning, you train the machine using data which is well “labelled”.
* Unsupervised learning is a machine learning technique, where you do not need to supervise the model.

**Q. How Artificial Neurons learns?**

* 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 the great processing power. Every neuron has input connections and output connections. These connections simulate the behaviour oof the synapses in the brain. The same way the synapses in the brain transfer the signal from one neuron to another, connections pass information between artificial neurons. These connections have weights, meaning that every value connection is multiplied by this factor.

**Q. 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 and false is 0. But in fuzzy 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 electronic domain so that it can learn like we do.

**Q. 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 of Simple Genetic Application (Match Word Finding).

**Theory:**

**Genetic algorithm:**

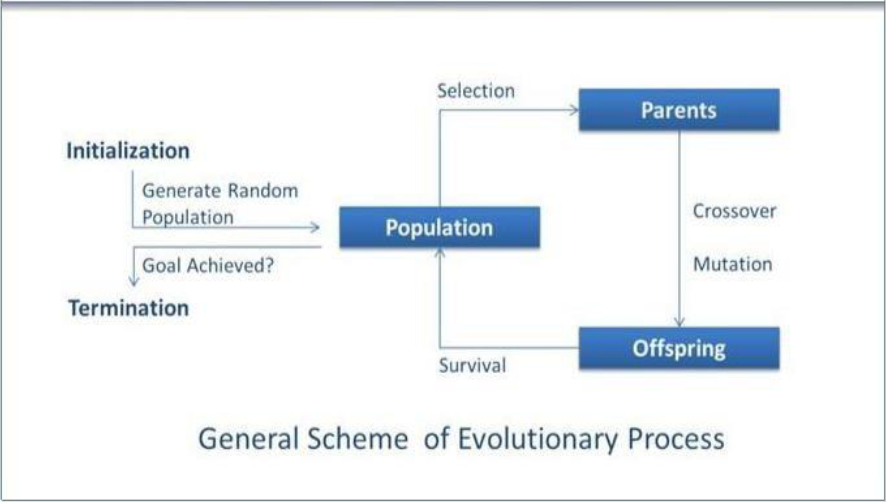
Genetic algorithm is a search technique used in computing to find true or approximate solutions to approximate solutions to optimization & search problems. Genetic algorithms are inspired by Darwin's theory about evolution. Solution to a problem solved by genetic algorithms is evolved.

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.

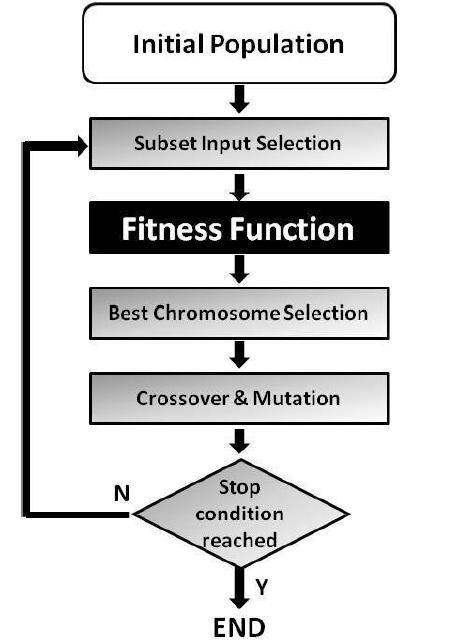
**Outline of the Basic Genetic Algorithm:**

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

* **[Selection]** Select two parent chromosomes from a population according to their fitness (the better fitness, the bigger chance to be selected)
* **[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.
* **[Mutation]** With a mutation probability mutate new offspring at each locus (position in chromosome).
* [**Accepting**] Place new offspring in a new population
* [**Replace**] Use new generated population for a further run of algorithm
* **[Test]** If the end condition is satisfied, stop, and return the best solution in current population
* **[Loop]** Go to step 2



**Flowchart:**



**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:**

import random

# Number of individuals in each generation

POPULATION\_SIZE = 100

# Valid genes

GENES = '''abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOP

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

# Target string to be generated

TARGET = "My name is Prateek"

class Individual(object):

def \_\_init\_\_(self, chromosome):

self.chromosome = chromosome

self.fitness = self.cal\_fitness()

@classmethod

def mutated\_genes(self):

global GENES

gene = random.choice(GENES)

return gene

@classmethod

def create\_gnome(self):

global TARGET

gnome\_len = len(TARGET)

return [self.mutated\_genes() for \_ in range(gnome\_len)]

def mate(self, par2):

# chromosome for offspring

child\_chromosome = []

for gp1, gp2 in zip(self.chromosome, par2.chromosome):

prob = random.random()

if prob < 0.45:

child\_chromosome.append(gp1)

elif prob < 0.90:

child\_chromosome.append(gp2)

else:

child\_chromosome.append(self.mutated\_genes())

return Individual(child\_chromosome)

def cal\_fitness(self):

global TARGET

fitness = 0

for gs, gt in zip(self.chromosome, TARGET):

if gs != gt: fitness+= 1

return fitness

def main():

global POPULATION\_SIZE

generation = 1

found = False

population = []

# create initial population

for \_ in range(POPULATION\_SIZE):

gnome = Individual.create\_gnome()

population.append(Individual(gnome))

while not found:

population = sorted(population, key = lambda x:x.fitness)

# if the individual having lowest fitness score i.e.

# 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))

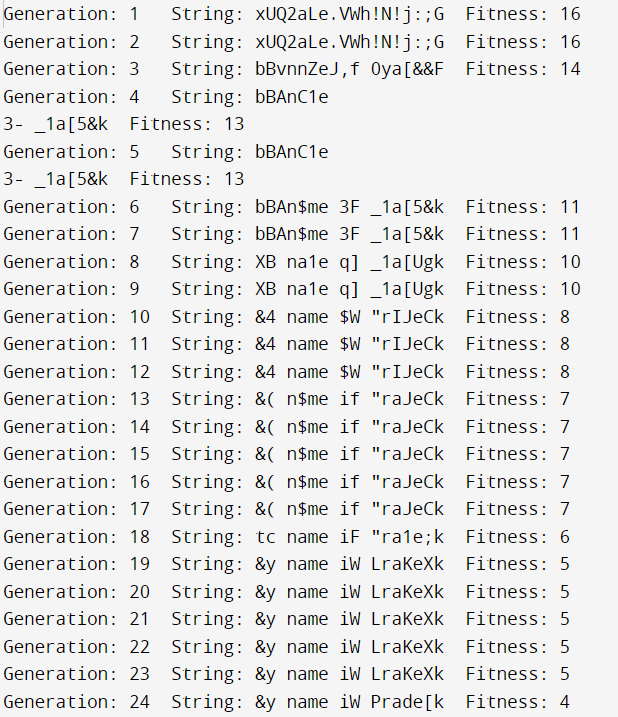
generation += 1

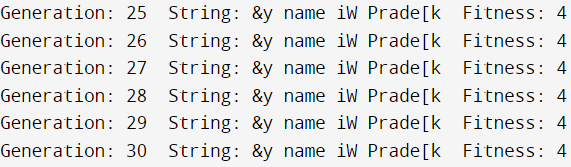
print("Generation: {}\tString: {}\tFitness: {}”.format( generation, "".join( population[0].chromosome), population[0].fitness))

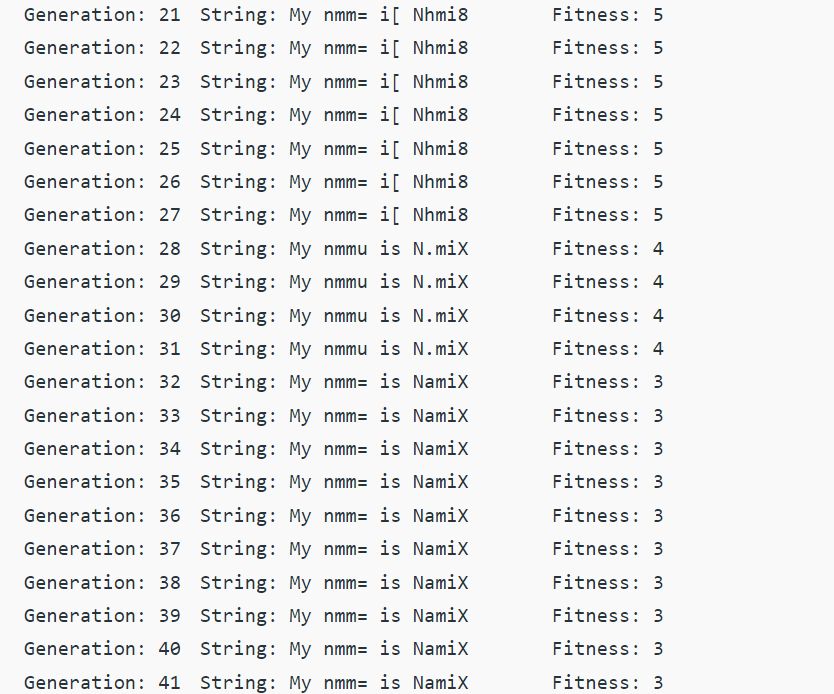
if \_\_name\_\_ == '\_\_main\_\_':

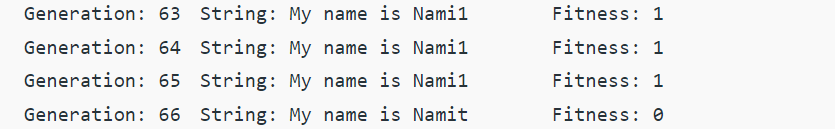
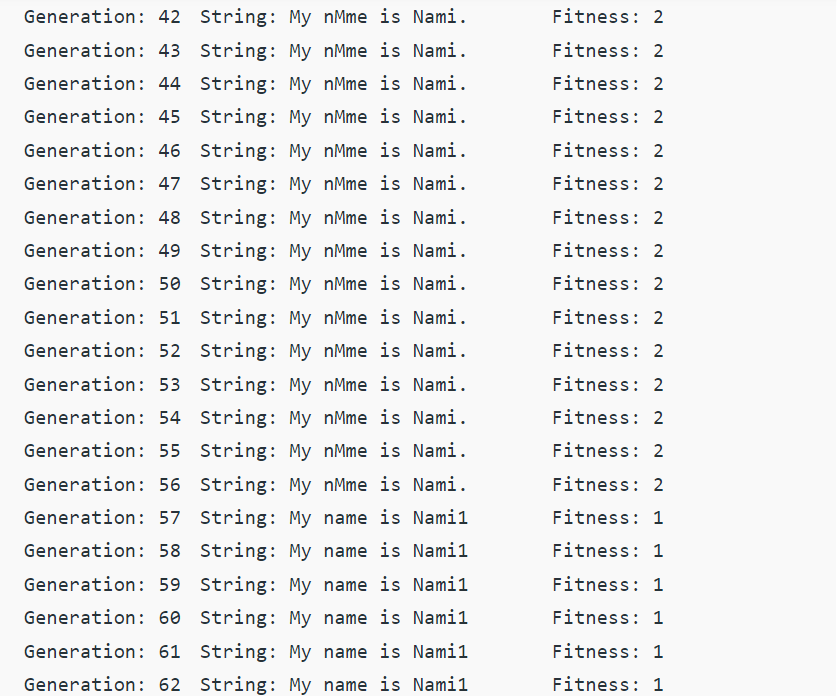
main()

**OUTPUT:**









**VIVA QUESTIONS**

**Q. Name some of the existing search methods.**

* Calculus based Search

Enumerative Search

Random Search

**Q. 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

**Q. What is reproduction?**

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

**Q. What is crossover?**

* Crossover is a genetic operator used to combine genetic information of two parents to create a new offspring.

**Experiment-8**

**Aim:** Study of AFNIS 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.

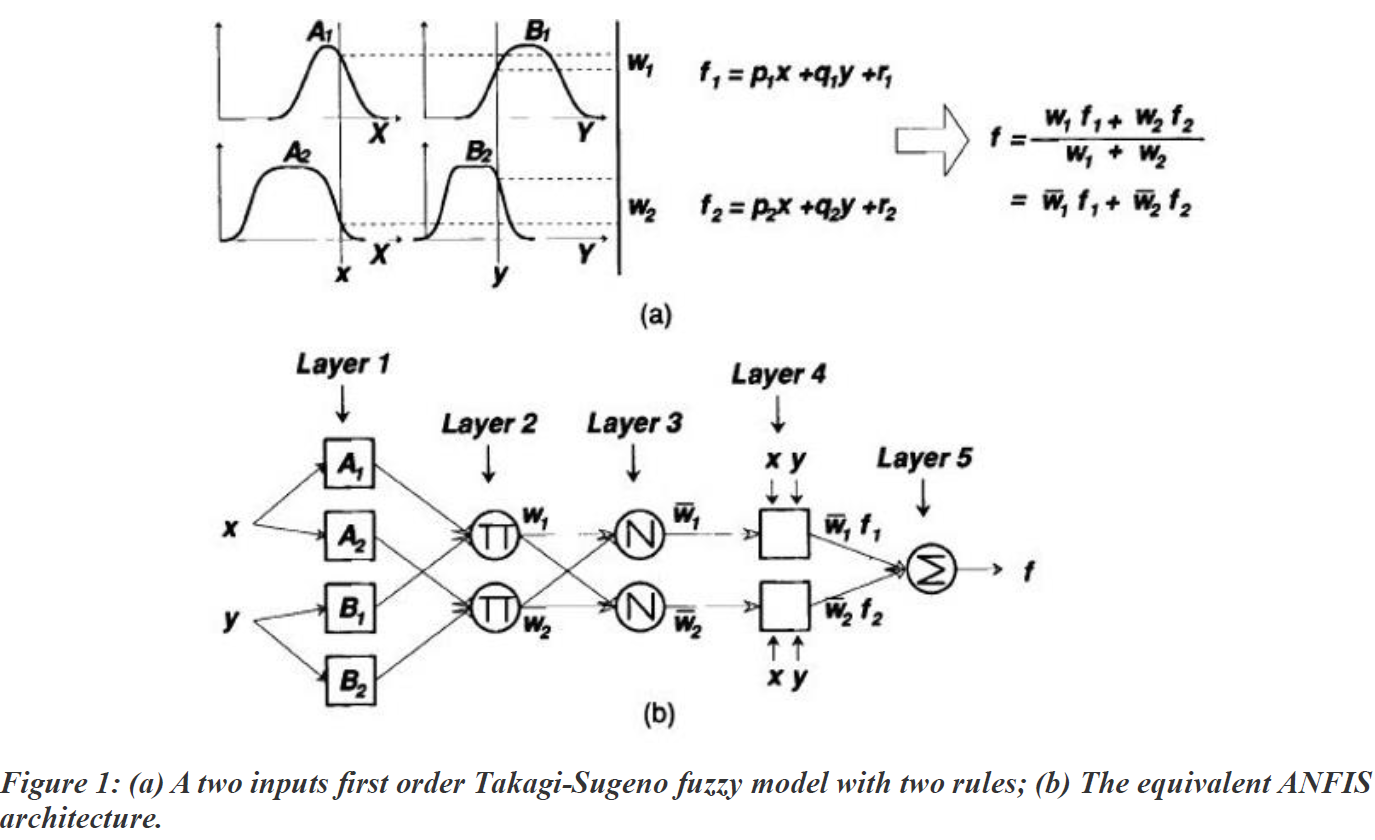
**ANFIS Architecture:**

**Representing Takagi-Sugeno Fuzzy Model**

For simplicity, we assume that the fuzzy inference sytem 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 A1 and y is B1, then f1=p1x+q1y+r1;

Rule 2: If x is A2 and y is B2, then f2=p2x+q2y+r2;



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

O1, i=μAi(x), for i=1,2, or O1, i=μBi−2(y), for i=3, 4, O1, i=μAi(x), for i=1, 2, or O1, i=μBi−2(y), for i=3, 4, where x (or y) is the input to node i and Ai(or Bi-2) is a linguistic label (such as "small" or "large") associated with this node. In other words, O1, i is the membership grade of a fuzzy set A (=A1, A2, B1 or B2) and it specifies the degree to which the given input x (or y) satifies the quantifier A.

μA(x)=11+|x−ciai|2b,μA(x)=11+|x−ciai|2b,

Where {ai, bi, ci} 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 labeledanfis, whose output is the product of all the incoming signals:

O2, i = wi = μAi(x) μBi(y), i= 1, 2. O2, I = wi = μAi(x) μBi(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 ith node calculates the ratio of the ith rule's firing strength to the sum of all rules' firing strenghts:

O3, i=w¯I = wiw1+w2′I = 1, 2. O3, I = w¯I = wiw1+w2′I = 1, 2.

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

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

O4, i= wifi¯= w¯i (pix + qiy + ri), O4, i= wifi¯= w¯i (pix + qiy + ri),

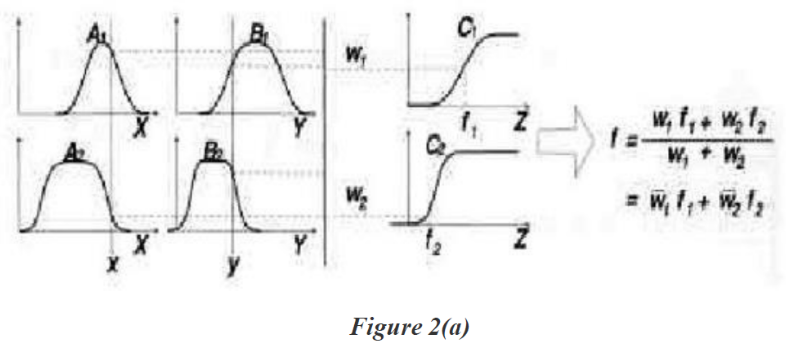
Where anfis is a normalized firing strength from layer 3 and {pi, qi, ri} is the parameter set of this node. Parameters in this layer are referred to as consequent parameters.

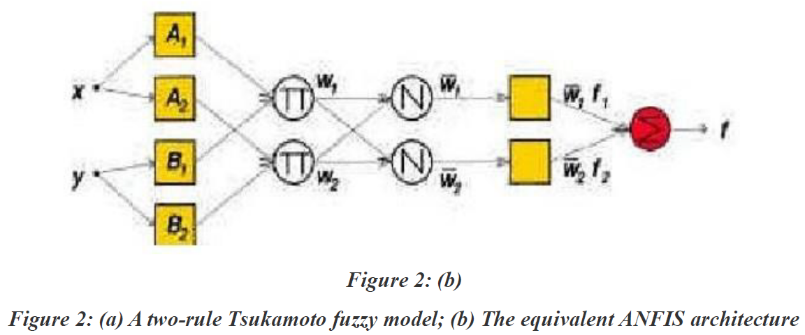
**Layer 5:** The single node in this layer is a fixed node labeledanfis, which computes the overall output as the summation of all incoming singals:

Overall output=O5, i=Σiw¯ifi=ΣiwifiΣiwioverall output=O5, i=Σiw¯ifi=ΣiwifiΣiwi

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

**Representing Tsukamoto Fuzzy Models:**





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

**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**

**Q. What are hybrid systems?**

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

**Q. 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.

**Q. How neuro fuzzy inference systems work?**

* A neuro-fuzzy system is based on a fuzzy system which is trained by 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.

**Q. What is ANFIS architecture?**

* An adaptive neuro-fuzzy inference system or adaptive network-based fuzzy inference system (AFNIS) 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 non-linear functions. Hence, ANFIS is considered to be a universal estimator.