

QUESTION BANK (DESCRIPTIVE)

Subject with Code: Soft Computing(20CS0531)

Course & Branch: B.Tech – CSE,CSM

Regulation: R20

Year &Sem: III-B.Tech & I – Sem

UNIT -I

INTRODUCTION TO SOFT COMPUTING

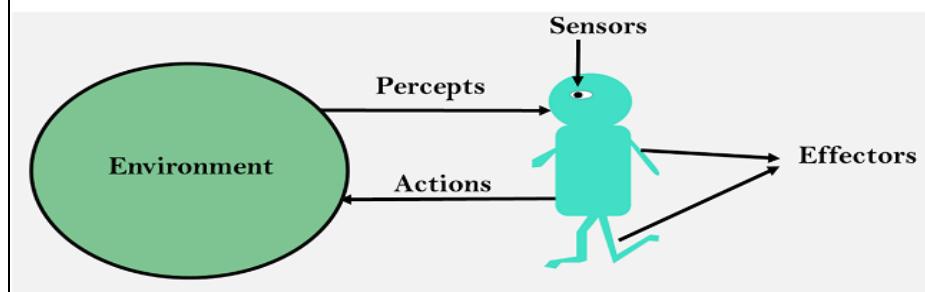
1.	<p>a Illustrate the basic components of Artificial Intelligence and its applications.</p> <p>The following are the components of Artificial Intelligence</p> <p>a. Learning Similar to humans, computer programs also learn in different manners. Talking of AI, learning by this platform is further segregated into a varied number of forms. One of the essential components of ai, learning for AI includes the trial-and-error method. The learning component of AI includes memorizing individual items like different solutions to problems, vocabulary, foreign languages, etc., also known as rote learning.</p> <p>b. Reasoning The art of reasoning was something that was only limited to humans until five decades ago. The ability to differentiate makes Reasoning one of the essential components of artificial intelligence. To reason is to allow the platform to draw inferences that fit with the provided situation.</p> <p>c. Problem-solving In its general form, the AI's problem-solving ability comprises data, where the solution needs to find x. The different methods of 'Problem-solving' count for essential artificial intelligence components that divide the queries into special and general purposes.</p> <p>d. Perception In using the 'perception' component of Artificial Intelligence, the element scans any given environment by using different sense-organs, either artificial or real.</p> <p>e. Language-understanding In simpler terms, language can be defined as a set of different system signs that justify their means using convention. Occurring as one of the widely used artificial intelligence components, language understanding uses distinctive types of language over different forms of natural meaning, exemplified overstatements.</p>	[L3][CO1]	[8M]
	<p style="text-align: center;">COMPONENTS OF ARTIFICIAL INTELLIGENCE</p> <p>Intelligent agents work through three main components: sensors, actuators, and effectors. Getting an overview of these components can improve our understanding of how</p>		

intelligent agents work.

Agent: This agent has some level of autonomy that allows it to perform specific, predictable, and repetitive tasks for users or applications.

- **Sensors:** These are devices that detect any changes in the environment. This information is sent to other devices. In artificial intelligence, the environment of the system is observed by intelligent agents through sensors.
- **Actuators:** These are components through which energy is converted into motion. They perform the role of controlling and moving a system. Examples include rails, motors, and gears.
- **Effectors:** The environment is affected by effectors. Examples include legs, fingers, wheels, display screen, and arms.

The following diagram shows how these components are positioned in the AI system.



Applications of AI

AI has been dominant in various fields such as –

- **Gaming** – AI plays crucial role in strategic games such as chess, poker, tic-tac-toe, etc.,
- **Natural Language Processing** – It is possible to interact with the computer that understands natural language spoken by humans.
- **Expert Systems** – There are some applications which integrate machine, software, and special information to impart reasoning and advising
- **Vision Systems** – These systems understand, interpret, and comprehend visual input on the computer.
- **Speech Recognition** – It can handle different accents, slang words, noise in the background, change in human's voice due to cold, etc.
- **Handwriting Recognition** – The handwriting recognition software reads the text written on paper by a pen or on screen by a stylus.
- **Intelligent Robots** – Robots are able to perform the tasks given by a human.

b Compare soft computing and hard computing

[L5][CO1]

[4M]

Difference between Soft Computing and Hard Computing:

S.NO	Soft Computing	Hard Computing
1.	Soft Computing is liberal of inexactness, uncertainty, partial truth and approximation.	Hard computing needs an exactly state analytic model.
2.	Soft Computing relies on formal logic and probabilistic reasoning.	Hard computing relies on binary logic and crisp system.
3.	Soft computing has the features of approximation and dispositionality.	Hard computing has the features of exactitude(precision) and categoricity.
4.	Soft computing is stochastic in nature.	Hard computing is deterministic in nature.

5.	Soft computing works on ambiguous and noisy data.	Hard computing works on exact data.
6.	Soft computing can perform parallel computations.	Hard computing performs sequential computations.
7.	Soft computing produces approximate results.	Hard computing produces precise results.
8.	Soft computing will emerge its own programs.	Hard computing requires programs to be written.
9.	Soft computing incorporates randomness .	Hard computing is settled.
10.	Soft computing will use multivalued logic.	Hard computing uses two-valued logic.

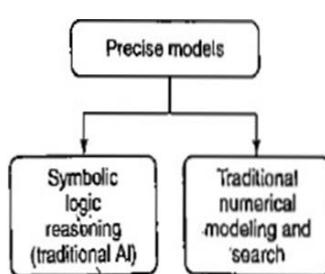


Figure 1.1: Hard Computing

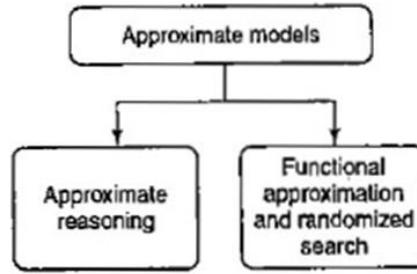
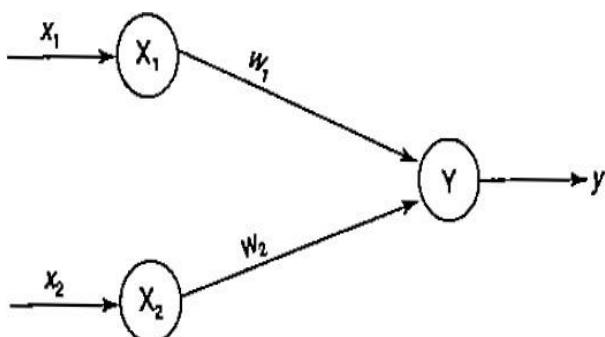


Figure 1.2: Soft Computing

2.	a	Explain the working principle of Artificial Neuron.	[L2][CO1]	[8M]
----	---	---	-----------	------

Artificial Neuron

An artificial neural network (ANN) is an efficient information processing system which resembles the characteristics of biological neural network. ANNs contain large number of highly interconnected processing elements called nodes or neurons or units. Each neuron is connected with other by connection link and each connection link is associated with weights which contain information about the input signal. This information is used by neuron net to solve a particular problem. ANNs have ability to learn, recall and



generalize training pattern or data similar to that of human brain. The ANN process sing elements called neurons or artificial neurons.

Each neuron has an internal state of its own, called activation or activity level of neuron which is the function of the inputs the neuron receives. The activation signal of a neuron is transmitted to other neurons. A neuron can send only one signal at a time which can be transmitted to several neurons.

Consider the figure 1.5, here X_1 and X_2 are input neurons, Y is the output neuron W_1 and W_2 are the weights net input is calculated as

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

where x_1 and x_2 are the activation of the input neurons X_1 and X_2 , i.e., is the output of the

inputs signals. The output y of the output neuron Y can be obtained by applying activations over the net input.

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

Output = Function (net input calculated)

The function to be applied over the net input is called activation function. The net input calculation is similar to the calculation of output of a pure linear straight line equation $y=mx$

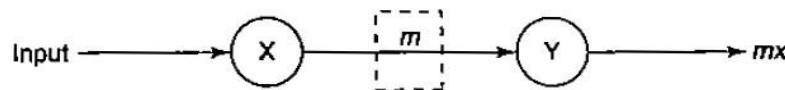


Figure 1.6: Neural net of pure linear equation

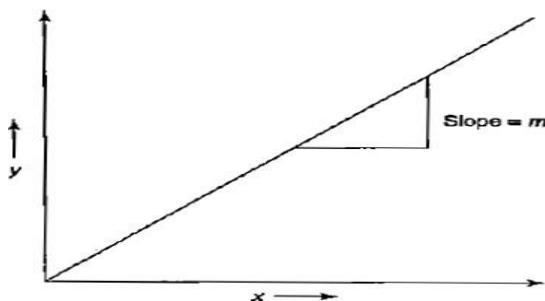


Figure 1.7: Graph for $y = mx$

The weight involved in the ANN is equivalent to the slope of the straight line.

b Differentiate Biological Neuron and Artificial Neuron.

[L4][CO1]

[4M]

Term	Brain	Computer
Speed	Execution time is few milliseconds	Execution time is few nano seconds
Processing	Perform massive parallel operations simultaneously	Perform several parallel operations simultaneously. It is faster than the biological neuron
Size and complexity	Number of Neuron is 10^{11} and number of interconnections is 10^{15} . So complexity of brain is higher than computer	It depends on the chosen application and network designer.
Storage capacity	<ul style="list-style-type: none"> Information is stored in interconnections or in synapse strength. New information is stored without destroying old one. Sometimes fails to recollect Information Draw biological neuron diagram	<ul style="list-style-type: none"> Stored in continuous memory location. Overloading may destroy older locations. Can be easily retrieved Draw the diagram of artificial neuron

Summarize the following terms:

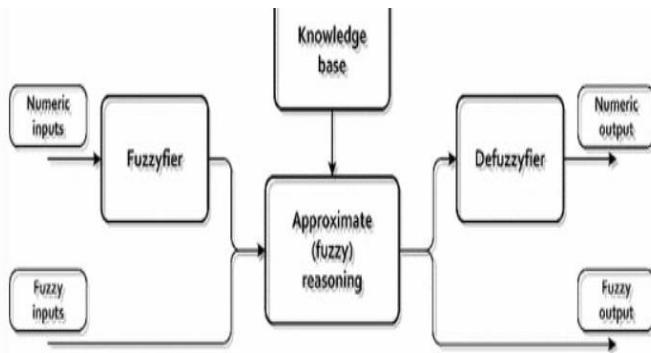
- i) Fuzzy Systems ii) Genetic Algorithm

[L5][CO1]

[12M]

i) Fuzzy Systems:

Fuzzy systems are structures based on fuzzy techniques oriented towards information processing, where the usage of classical sets theory and binary logic is impossible or difficult. In the literature, terms such as fuzzy system, fuzzy model, system based on fuzzy rules, fuzzy controller, or fuzzy associative memory are used interchangeably depending on the application type. Their main characteristic involves symbolic knowledge representation in a form of fuzzy conditional (if-then) rules.



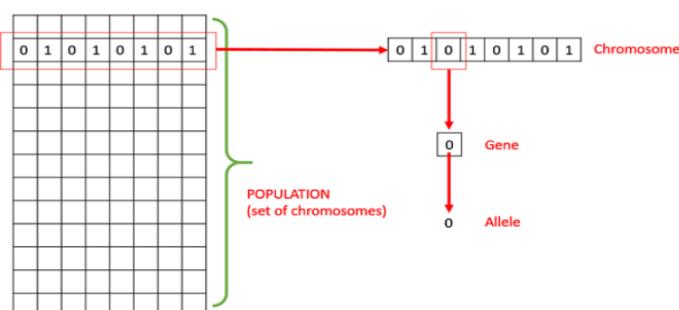
The typical structure of a fuzzy system

The typical structure of a fuzzy system consists of four functional blocks: the fuzzifier, the fuzzy inference engine, the knowledge base, and the defuzzifier. Both linguistic values (defined by fuzzy sets) and crisp (numerical) data can be used as inputs for a fuzzy system. If crisp data are applied, then the inference process is preceded by fuzzification, which assigns the appropriate fuzzy set to the nonfuzzy input. The values of input variables are mapped into linguistic values of the output variable by means of the appropriate method of approximate reasoning (inference engine) using expert knowledge, which is represented as a collection of fuzzy conditional rules (knowledge base). In addition to the linguistic values, the numerical data may be required as the fuzzy system output. In such cases defuzzification methods are used, which assign the representative crisp data to the resultant output fuzzy set.

- ii) **Genetic algorithm:** Genetic Algorithm (GA) is a search-based optimization technique based on the principles of **Genetics and Natural Selection**. GAs are a subset of a much larger branch of computation known as **Evolutionary Computation**.

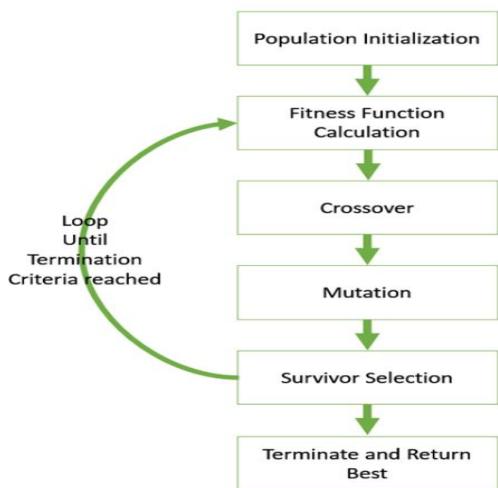
Basic Terminology

- **Population** – It is a subset of all the possible (encoded) solutions to the given problem. The population for a GA is analogous to the population for human beings except that instead of human beings, we have Candidate Solutions representing human beings.
- **Chromosomes** – A chromosome is one such solution to the given problem.
- **Gene** – A gene is one element position of a chromosome.
- **Allele** – It is the value a gene takes for a particular chromosome.



- **Genotype** – Genotype is the population in the computation space. In the computation space, the solutions are represented in a way which can be easily understood and manipulated using a computing system.
- **Phenotype** – Phenotype is the population in the actual real world solution space in which solutions are represented in a way they are represented in real world situations.

- Decoding and Encoding** – For simple problems, the **phenotype** and **genotype** spaces are the same.



Basic structure of GA

We start with an initial population (which may be generated at random or seeded by other heuristics), select parents from this population for mating. Apply crossover and mutation operators on the parents to generate new off-springs. And finally these off-springs replace the existing individuals in the population and the process repeats. In this way genetic algorithms actually try to mimic the human evolution to some extent.

4.

- a Distinguish between Supervised Learning and Unsupervised Learning.

[L4][CO1]

[6M]

Major differences between Supervised and Unsupervised Learning

Supervised Learning	Unsupervised Learning
Supervised Learning can be used for 2 different types of problems i.e. regression and classification	Unsupervised Learning can be used for 2 different types of problems i.e. clustering and association.
Input Data is provided to the model along with the output in the Supervised Learning.	Only input data is provided in Unsupervised Learning.
Output is predicted by the Supervised Learning.	Hidden patterns in the data can be found using the unsupervised learning model.
Labeled data is used to train supervised learning algorithms.	Unlabeled data is used to train unsupervised learning algorithms.
Accurate results are produced using a supervised learning model.	The accuracy of results produced are less in unsupervised learning models.
Training the model to predict output when a new data is provided is the objective of Supervised Learning.	Finding useful insights, hidden patterns from the unknown dataset is the objective of the unsupervised learning.
Supervised Learning includes various algorithms such as Bayesian Logic, Decision Tree, Logistic Regression, Linear Regression, Multi-class Classification, Support Vector Machine etc.	Unsupervised Learning includes various algorithms like KNN, Apriori Algorithm, and Clustering.
To assess whether right output is being predicted, direct feedback is accepted by the Supervised Learning Model.	No feedback will be taken by the unsupervised learning model.
In Supervised Learning, for right prediction of output, the model has to be trained for each data, hence Supervised Learning does	Unsupervised Learning has more resemblance to Artificial Intelligence as it learns learning

	<p>not have close resemblance to Artificial Intelligence.</p> <p>Number of classes are known in Supervised Learning.</p>	<p>new things with more experience.</p> <p>Number of classes are not known in Unsupervised Learning</p>		
	<p>In scenarios where one is aware of output and input data, supervised learning can be used.</p>	<p>In the scenarios where one is not aware of output data, but is only aware of the input data then Unsupervised Learning could be used.</p>		
	<p>Computational Complexity is very complex in Supervised Learning compared to Unsupervised Learning</p>	<p>There is less computational complexity in Unsupervised Learning when compared to Supervised Learning.</p>		
	<p>Supervised Learning will use off-line analysis</p>	<p>Unsupervised Learning uses Real time analysis of data.</p>		
	<p>Some of the applications of Supervised Learning are Spam detection, handwriting detection, pattern recognition, speech recognition etc.</p>	<p>Some of the applications of Unsupervised Learning are detecting fraudulent transactions, data preprocessing etc.</p>		
b	Describe the different activation functions in Neural Networks.			[L2][CO1] [6M]
	<p>Activation Functions:</p> <p>To make work more efficient and for exact output, some force or activation is given. Like that, activation function is applied over the net input to calculate the output of an ANN. Information processing of processing element has two major parts: input and output. An integration function (f) is associated with input of processing element. Several activation functions are there.</p> <p>1. Identity function: It is a linear function which is defined as</p> $f(x) = x \text{ for all } x$ <p>The output is same as the input.</p> <p>2. Binary step function: This function can be defined as</p> $f(x) = \begin{cases} 1 & \text{if } x \geq \theta \\ 0 & \text{if } x < \theta \end{cases}$ <p>Where, θ represents thresh hold value. It is used in single layer nets to convert the net input to an output that is binary (0 or 1).</p> <p>3. Bipolar step function: This function can be defined as</p> $f(x) = \begin{cases} 1 & \text{if } x \geq \theta \\ -1 & \text{if } x < \theta \end{cases}$ <p>Where, θ represents threshold value. It is used in single layer nets to convert the net input to an output that is bipolar (+1 or -1).</p>			

4. Sigmoid function: It is used in Back propagation nets.

Two types:

a) **Binary sigmoid function:** It is also termed as logistic sigmoid function or unipolar sigmoid function. It is defined as

$$f(x) = \frac{1}{1 + e^{-\lambda x}}$$

where, λ represents steepness parameter. The derivative of this function is

$$f'(x) = \lambda f(x)[1 - f(x)]$$

The range of sigmoid function is 0 to 1.

Bipolar sigmoid function: This function is defined as

$$f(x) = \frac{2}{1 + e^{-\lambda x}} - 1 = \frac{1 - e^{-\lambda x}}{1 + e^{-\lambda x}}$$

Where λ represents steepness parameter and the sigmoid range is between -1 and +1. The derivative of this function can be

$$f'(x) = \frac{\lambda}{2}[1 + f(x)][1 - f(x)]$$

It is closely related to hyperbolic tangent function, which is written as

$$h(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

$$h(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}}$$

The derivative of the hyperbolic tangent function is

$$h'(x) = [1 + h(x)][1 - h(x)]$$

Ramp function: The ramp function is defined as

$$f(x) = \begin{cases} 1 & \text{if } x > 1 \\ x & \text{if } 0 \leq x \leq 1 \\ 0 & \text{if } x < 0 \end{cases}$$

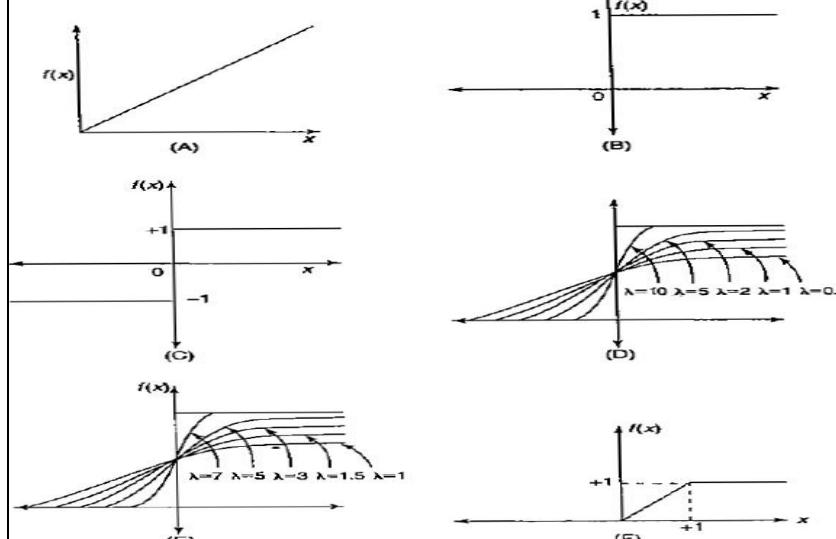


Figure (1): Depiction of activation functions: (A) identity function; (B) binary step function; (C) bipolar stepfunction; (D) binary sigmoidal function; (E) bipolar sigmoidal function; (F) ramp function

5.

Infer the classifications of Artificial Neural Networks.

[L2][CO1] [12M]

Classifications of ANNs(Connections)

The arrangement of neurons to form layers and the connection pattern formed within and between layers is called the network architecture. There exist five basic types of connection architecture. They are

1. Single layer feed forward network
2. Multilayer feed-forward network
3. Single node with its own feedback
4. Single-layer recurrent network
5. Multilayer recurrent network

Feed forward network: If no neuron in the output layer is an input to a node in the same layer / proceeding layer.

Feedback network: If outputs are directed back as input to the processing elements in the same layer/proceeding layer.

Lateral feedback: If the output is directed back to the input of the same layer.

Recurrent networks: Are networks with feedback networks with closed loop.

1.Single layer feed forward network

Layer is formed by taking processing elements and combining it with other processing elements. Input and output are linked with each other Inputs are connected to the processing nodes with various weights, resulting in series of outputs one per node.

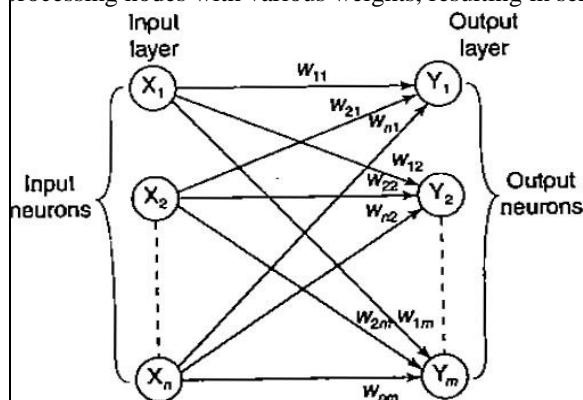


Figure 1.8: Single-layer feed-forward network

When a layer of processing nodes is formed the inputs can be connected to these nodes with various weights, resulting in a series of outputs, one per node. This is called single layer feedforward network.

2.Multilayer feed-forward network:

This network is formed by the interconnection of several layers. Input layer receives input and buffers input signal. Output layer generated output. Layer between input and output is called hidden layer. Hidden layer is internal to the network. There are Zero to several hidden layers in a network. More the hidden layer more is the complexity of network, but efficient output is produced.

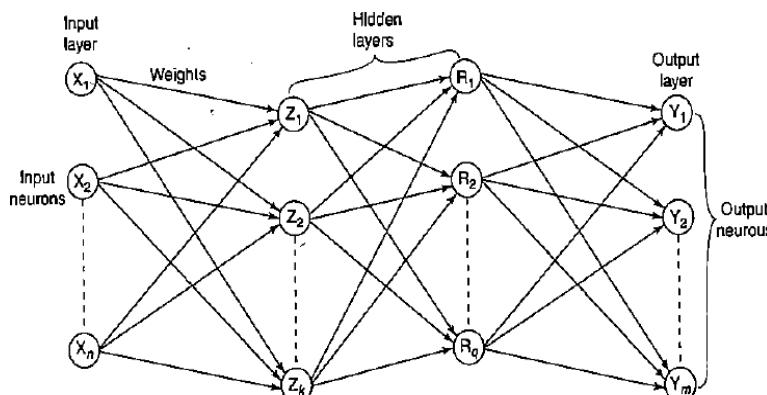


Figure 1.9: Multilayer feed-forward network

3.Single node with its own feedback:

It is a simple recurrent neural network having a single neuron with feedback to itself.

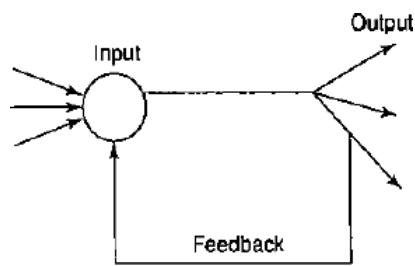


Figure 1.10: Single node with own feedback

4.Single layer recurrent network:

A single layer network with feedback from output can be directed to processing element itself or to other processing element/both.

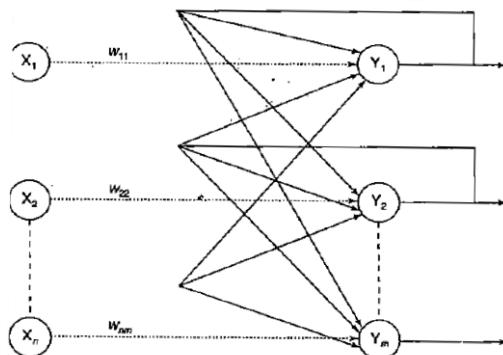


Figure 1.11: Single-layer recurrent network

5.Multilayer recurrent network:

Processing element output can be directed back to the nodes in the preceding layer, forming a multilayer recurrent network.

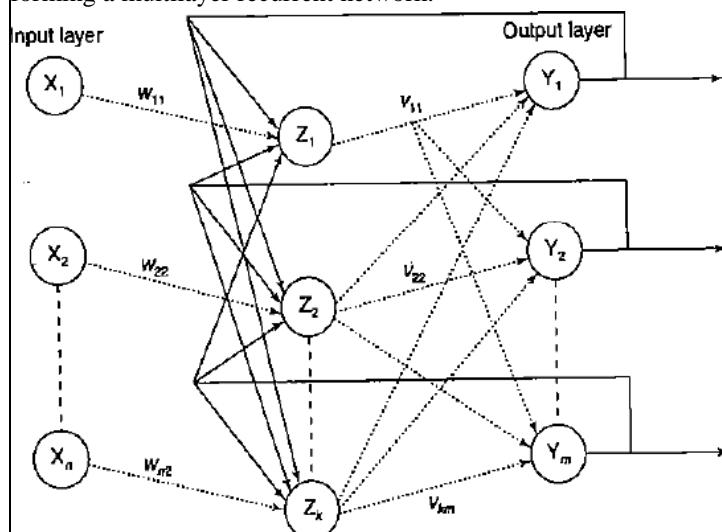


Figure 1.12: Multilayer recurrent network

Maxnet –competitive interconnections having fixed weights.

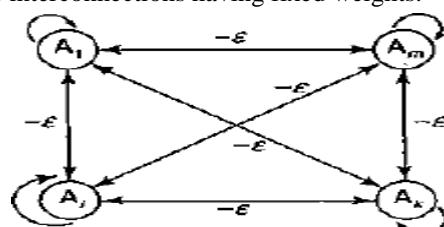


Figure 1.13: Competitive nets

On-center-off-surround/lateral inhibition structure – each processing neuron receives

two different classes of inputs- “excitatory” input from nearby processing elements & “inhibitory” elements from more distantly located processing elements. This type of interconnection is shown below

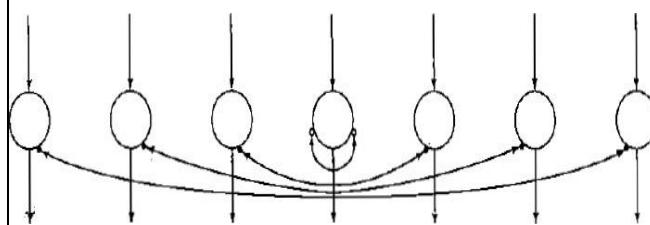


Figure 1.14: Lateral inhibition structure

<p>6.</p> <p>a Analyze Learning Techniques in ANN</p>	<p>[L4][CO1] [6M]</p> <p><i>Hebbian Learning Rule</i></p> <p>Developed by Donald Hebb in 1949, Hebbian learning is an unsupervised learning rule that works by adjusting the weights between two neurons in proportion to the product of their activation.</p> <p>According to this rule, the weight between two neurons is decreased if they are working in opposite directions and vice-versa. However, if there is no correlation between the signals, then the weight remains the same. As for the sign a weight carries, it is positive when both the nodes are either positive or negative. However, in case of a one node being either positive or negative, the weight carries a negative sign.</p> <p>Formula</p> $\Delta w = \alpha x_i y$ <p>where,</p> <ul style="list-style-type: none"> • Δw is the change in weight • α is the learning rate • x_i is the input vector and, • y is the output <p><i>Perceptron Learning Rule</i></p> <p>Developed by Rosenblatt, the perceptron learning rule is an error correction rule, used in a single layer feed forward network. Like Hebbian learning, this also, is a supervised learning rule.</p> <p>This rule works by finding the difference between the actual and desired outputs and adjusts the weights according to that. Naturally, the rule requires the input of a set of vectors, along with the weights so that it can produce an output.</p> <p>Formula</p> $w = w + \eta(y - \hat{y})x$ <ul style="list-style-type: none"> • w is the weight • η is the learning rate • x is the input vector
---	---

- y is the actual label of the input vector
- \hat{y} is the predicted label of the input vector

Correlation Learning Rule

With a similar principle as the Hebbian rule, the correction rule also increases or decreases the weights based on the phases of the two neurons.

If the neurons are in the opposite phase to each other, the weight should be towards the negative side and if they are in the same phase to each other, the weight should be towards the positive side. The only thing that makes this rule different from the Hebbian learning rule is that it is supervised in nature.

Formula

$$\Delta w = \alpha x_i t_j$$

- Δw is the change in weight
- α is the learning rate
- x_i is the input vector and,
- t_j is the target value

Competitive Learning Rule

The competitive learning rule is unsupervised in nature and as the name says, is based on the principle of competition amongst the nodes. That is why it is also known as the ‘Winner takes it all’ rule.

In this rule, all the output nodes represent the input pattern and the best one, meaning, the one having the most number of outputs is the winner. This winner node is then assigned the value 1 and the others that lose, remain at 0. Naturally, only one neuron remains active at once.

Formula

$$\Delta w_{ij} = \eta * (x_i - w_{ij})$$

where,

- Δw_{ij} is the changes in weight between the i th input neuron and j th output neuron
- η is the learning rate
- x_i is the input vector
- w_{ij} is the weight between the i th input neuron and j th output neuron

Delta Learning Rule

Developed by Bernard Widrow and Marcian Hoff, Delta learning rule is a supervised learning rule with a continuous activation function.

The main aim of this rule is to minimize error in the training patterns and thus, it is also known as the least mean square method. The principle used here is that of an infinite

gradient descent and the changes in weight is equal to the product of the error and the input.

Formula

$$\Delta w_{ij} = \eta * (d_j - y_i) * f'(h_j) * x_i$$

where,

- Δw_{ij} is the changes in weight between the i th input neuron and j th output neuron
- η is the learning rate
- d_j is the target output
- y_i is the actual output
- $f'(h_j)$ is the derivative of the activation function
- x_i is the i th input

Out Star Learning Rule

Developed by Grossberg, Out Star learning is a supervised learning rule that works with a pair of nodes arranged in a network layer.

According to this rule, the weights linked to a given node are equal to target output along that particular link. Also, there is a single output layer of neurons, where each neuron is a cluster of input vectors. This algorithm is mostly used for patterns and clustering related tasks.

Formula

$$\Delta w_{ij} = \eta * (x_i - w_j) * f(w_j)$$

where,

- Δw_{ij} is the change in weight
- η is the learning rate
- x_i is the input pattern
- w_j is the weight vector
- $f(w_j)$ is the activation function

b Describe the role of Evolutionary Programming in Soft Computing.

[L2][CO1]

[6M]

Evolutionary programming (EP) is an approach to simulated evolution that iteratively generates increasingly appropriate solutions in the light of a stationary or non-stationary environment and desired fitness function. Standard EP makes use of the same four components that are standard to all evolutionary algorithms (EAs):

- **initialization**,
- **variation**
- **evaluation**
- **selection**.

This approach to population-based search has its roots in the early work of Lawrence J. Fogel,

Evolutionary programming is **one of the four major evolutionary algorithm paradigms**. It is similar to genetic programming, but the structure of the program to be

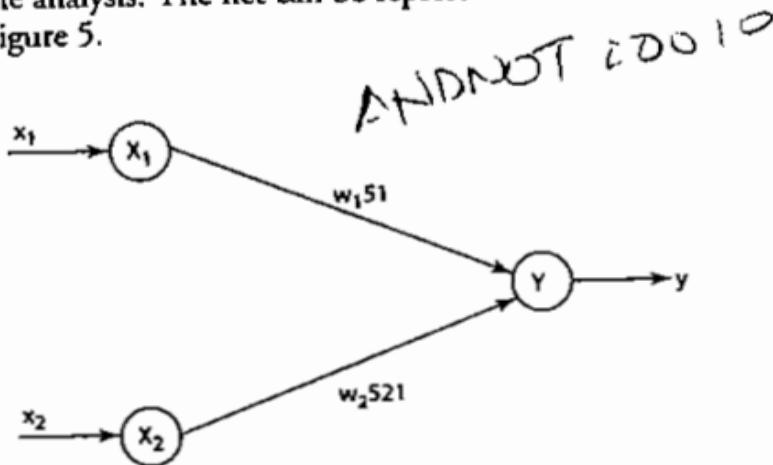
	<p>optimized is fixed, while its numerical parameters are allowed to evolve.</p> <ul style="list-style-type: none"> • There is no fixed structure for representation. • There is only mutation operation, and cross-over is not used in this method. • Each child is determined by its parent in a way of mutation. • So, we can conclude that there are three steps: <ul style="list-style-type: none"> - Initialize population and calculate fitness values for initial population - Mutate the parents and generate new population - Calculate fitness values of new generation and continue from the second step. <p>Some of the Applications of EP</p> <ul style="list-style-type: none"> • used to optimize the control gains for the kinematic control system • robotic control used EP coupled with fuzzy logic for the control of wheeled robots • area of mobile network strategic planning, applied EP to discover optimal cell size determination. 		
7.	<p>a Explain McCulloch and Pitts Neuron Model. [L2][CO1] [6M]</p> <p>McCulloch and Pitts Neuron:</p> <p>It is discovered in 1943 and usually called as M-P neuron. M-P neurons are connected by directed weighted paths. Activation of M-P neurons is binary (i.e.) at any time step the neuron may fire or may not fire. Weights associated with communication links may be excitatory (wgts are positive)/inhibitory (wgts are negative). Threshold plays major role here. There is a fixed threshold for each neuron and if the net input to the neuron is greater than the threshold then the neuron fires. They are widely used in logic functions. A simple M-P neuron is shown in the figure. It is excitatory with weight w ($w>0$) / 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.</p> <p>The given function gives: and $x_2 = 0$. The weights x_{n+1} to the analysis. The net can Figure 5.</p> <p>Figure 5 Neural net (weights fixed after analysis).</p> <p>Case 1: Assume that both weights w_1 and w_2 are excitatory, i.e., $w_1 = w_2 = 1$</p> <p>Then for the four inputs calculate the net input using $y_{in} = x_1 w_1 + x_2 w_2$</p> <p>For inputs</p> <p>(1, 1), $y_{in} = 1 \times 1 + 1 \times 1 = 2$ (1, 0), $y_{in} = 1 \times 1 + 0 \times 1 = 1$ (0, 1), $y_{in} = 0 \times 1 + 1 \times 1 = 1$ (0, 0), $y_{in} = 0 \times 1 + 0 \times 1 = 0$</p> <p>From the calculated net inputs, it is not possible to fire the neuron for input (1, 0) only. Hence, these weights are not suitable.</p> <p>Assume one weight as excitatory and the other as inhibitory, i.e., $w_1 = 1, w_2 = -1$</p> <p>b Implement in M-P Neuron Model. [L3][CO1] [6M]</p>	[L2][CO1] [6M]	

Table 2

x_1	x_2	y
0	0	0
0	1	0
1	0	1
1	1	0

★

The given function gives an output only when $x_1 = 1$ and $x_2 = 0$. The weights have to be decided only after the analysis. The net can be represented as shown in Figure 5.

**Figure 5** Neural net (weights fixed after analysis).

Case 1: Assume that both weights w_1 and w_2 are excitatory, i.e.,

$$w_1 = w_2 = 1$$

Then for the four inputs calculate the net input using

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

For inputs

$$(1, 1), \quad y_{in} = 1 \times 1 + 1 \times 1 = 2$$

$$(1, 0), \quad y_{in} = 1 \times 1 + 0 \times 1 = 1$$

$$(0, 1), \quad y_{in} = 0 \times 1 + 1 \times 1 = 1$$

$$(0, 0), \quad y_{in} = 0 \times 1 + 0 \times 1 = 0$$

From the calculated net inputs, it is not possible to fire the neuron for input $(1, 0)$ only. Hence, these weights are not suitable.

Assume one weight as excitatory and the other as inhibitory, i.e.,

Now calculate the net input. For the inputs

$$(1, 1), \quad y_{in} = 1 \times 1 + 1 \times -1 = 0$$

$$(1, 0), \quad y_{in} = 1 \times 1 + 0 \times -1 = 1$$

$$(0, 1), \quad y_{in} = 0 \times 1 + 1 \times -1 = -1$$

$$(0, 0), \quad y_{in} = 0 \times 1 + 0 \times -1 = 0$$

From the calculated net inputs, now it is possible to fire the neuron for input $(1, 0)$ only by fixing a threshold of 1, i.e., $\theta \geq 1$ for Y unit. Thus,

$$w_1 = 1; \quad w_2 = -1; \quad \theta \geq 1$$

Note: The value of θ is calculated using the following:

$$\theta \geq nw - p$$

$$\theta \geq 2 \times 1 - 1 \quad [for "p" \text{ inhibitory only}]$$

$$\theta \geq 1$$

Thus, the output of neuron Y can be written as

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

8.

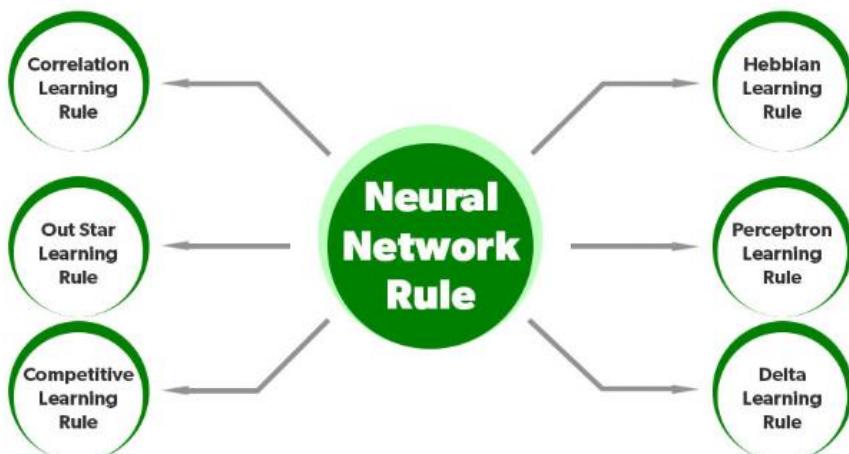
a

List out the different Learning rules and terminology in ANN.

[L2][CO1]

[4M]

Learning Rules:



Note: Give the explanation for each

ANN Terminologies:

Weights

Threshold value

Learning Rate

Bias

Target value

Error

Note: Give the explanation for each

b

Describe Hebbian Network in Artificial Neural Networks

[L2][CO1]

[8M]

Hebb network

Donald Hebb stated in 1949 that “In brain, the learning is performed by the change in the synaptic gap”. When an axon of cell A is near enough to excite cell B, and repeatedly or permanently takes place in firing it, some growth process or metabolic change takes place in one or both the cells such that A’s efficiency, as one of the cells firing B, is increased. According to Hebb rule, the weight vector is found to increase proportionately to the product of the input and the learning signal. In Hebb learning, two interconnected neurons are ‘on’ simultaneously. The weight update in Hebb rule is given by

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

Hebbs network is suited more for bipolar data. If binary data is used, the weight updatation formula cannot distinguish two conditions namely:

1. A training pair in which an input unit is “on” and the target value is “off”.
2. A training pair in which both the input unit and the target value is “off”.

Training algorithm

The training algorithm is used for the calculation and adjustment of weights. The flowchart for the training algorithm of Hebb network is given below

Step 0: First initialize the weights. Basically in this network they may be set to zero, i.e. $w_i = 0$, for $i = 1$ to n where “ n ” may be the total number of input neurons.

Step 1: Steps 2-4 have to be performed for each input training vector and target output pair, $s: t$.

Step 2: Input units activations are set. Generally, the activation function of input layer is identity function: $x_i = s_i$ for $i = 1$ to n

Step 3: Output units activations are set: $y = t$.

Step 4: Weight adjustments and bias adjustments are performed:

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

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

In step 4, the weight updatation formula can be written in vector form as

$$w(\text{new}) = w(\text{old}) + xy$$

Hence, Change in weight is expressed as $\Delta w = xy$

As a result,

$$w(\text{new}) = w(\text{old}) + \Delta w$$

Hebb rule is used for pattern association, pattern categorization, pattern classification and over a range of other areas.

Flowchart of Training algorithm

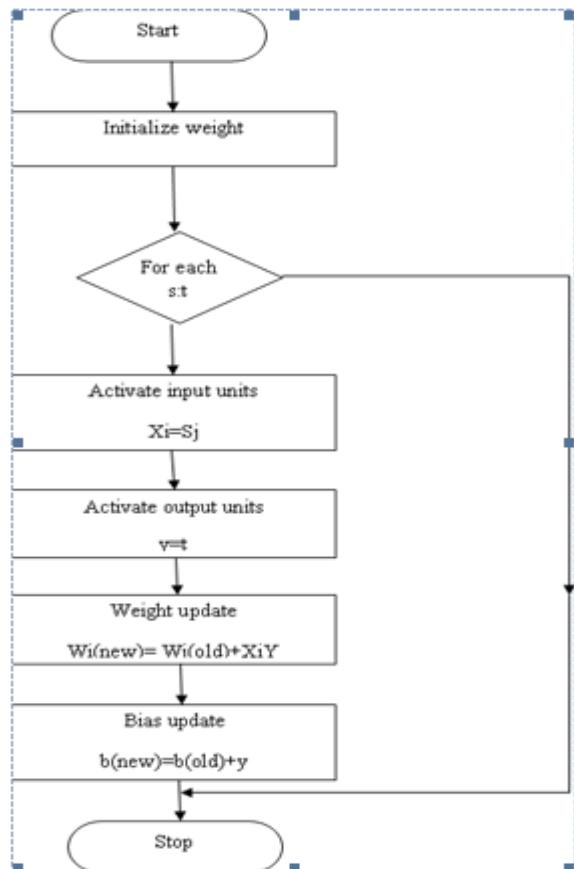


Figure 1.20: Flowchart of Hebb training algorithm

9.	<p>Illustrate the Perceptron Network with neat diagram.</p> <p>Perceptron networks Perceptron networks come under single-layer feed-forward networks and are also called simple perceptrons. Various types of perceptrons were designed by Rosenblatt (1962) and Minsky-Papert (1969, 1988).</p> <p>The key points to be noted in a perceptron network are:</p> <ol style="list-style-type: none"> 1. The perceptron network consists of three units, namely, sensory unit (input unit), associator unit (hidden unit), and response unit (output unit). 2. The sensory units are connected to associator units with fixed weights having values 1, 0 or -1, which are assigned at random. 3. The binary activation function is used in sensory unit and associator unit. 4. The response unit has an activation of 1, 0 or -1. The binary step with fixed threshold θ is used as activation for associator. The output signals that are sent from the associator unit to the response unit are only binary. 5. The output of the perceptron network is given by $y = f(y_{in})$ <p>where $f(y_{in})$ is activation function and is defined as</p> $f(y_{in}) = \begin{cases} 1 & \text{if } y_{in} > \theta \\ 0 & \text{if } -\theta \leq y_{in} \leq \theta \\ -1 & \text{if } y_{in} < -\theta \end{cases}$ <ol style="list-style-type: none"> 6. The perceptron learning rule is used in the weight updation between the associator unit and the response unit. For each training input, the net will calculate the response and it will determine whether or not an error has occurred. 7. The error calculation is based on the comparison of the values of targets with those of the calculated outputs. 	[L3][CO1]	[12M]
----	---	-----------	-------

8. The weights on the connections from the units that send the nonzero signal will get adjusted suitably.
9. The weights will be adjusted on the basis of the learning rule an error has occurred for a particular training patterns .i.e.,,

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

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

If no error occurs, there is no weight updation and hence the training process may be stopped. In the above equations, the target value "t" is +1 or -1 and α is the learning rate

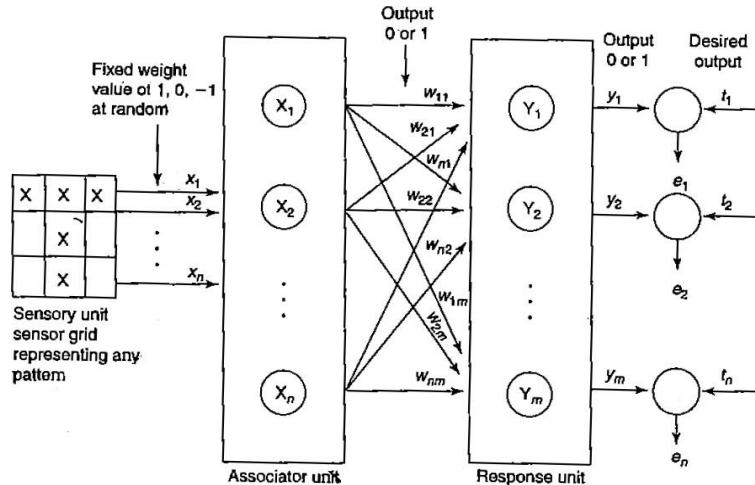


Figure 2.1: Original perceptron network

Architecture

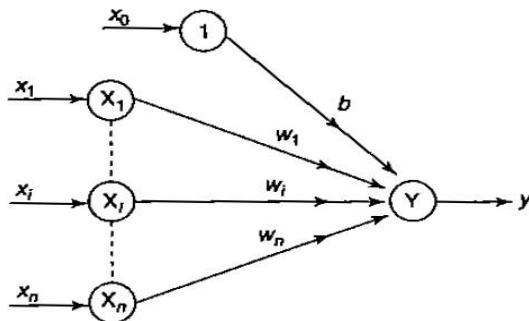


Figure 2.2: Single classification perceptron network

Here only the weights between the associator unit and the output unit can be adjusted, and the weights between the sensory and associator units are fixed.

Draw the flowchart.

- 10.** Explain briefly Adaline and Madaline Networks.

[L2][CO1]

[12M]

ADALINE

Adaline is a single unit neuron, which receives input from several units and also from one unit called bias. The basic Adaline model consists of trainable weights. Inputs are either of the two values (+ 1 or -1) and the weights have signs (positive or negative). Initially, random weights are assigned. The net input calculated is applied to a quantizer transfer function (possibly activation function) that restores the output to + 1 or -1.

Training Algorithm

Step 0: Weights and bias are set to some random values but not zero.

Set the learning rate parameter α .

Step 1: Perform Steps 2-6 when stopping condition is false.

Step 2: Perform Steps 3-5 for each bipolar training pair s:t.

Step 3: Set activations for input units $i = 1$ to n .

$$x_i = s_i$$

Step 4: Calculate the net input to the output unit.

n

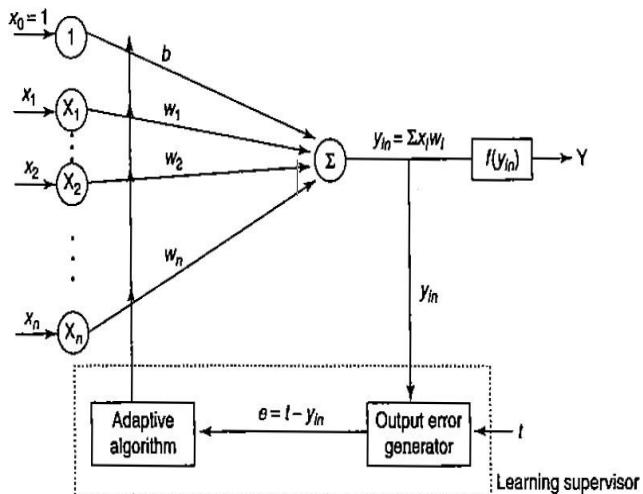
$$y_{in} = b + \sum_{i=1}^n x_i w_i$$

Step 5: Update the weights and bias for $i = 1$ to n

$$w_i(\text{new}) = w_i(\text{old}) + \alpha(t - y_{in}) x_i$$

$$b(\text{new}) = b(\text{old}) + \alpha(t - y_{in})$$

Step 6: If the highest weight change that occurred during training is smaller than a specified tolerance then stop the training process, else continue. This is the rest for stopping condition of a network.



Testing Algorithm

Step 0: Initialize the weights. (The weights are obtained from the training algorithm.)

Step 1: Perform Steps 2-4 for each bipolar input vector x .

Step 2: Set the activations of the input units to x .

Step 3: Calculate the net input to the output unit:

$$y_{in} = b + \sum x_i w_i$$

Step 4: Apply the activation function over the net input calculated:

$$y = \begin{cases} 1 & \text{if } y_{in} \geq 0 \\ -1 & \text{if } y_{in} < 0 \end{cases}$$

in

Multiple Adaptive Linear Neuron (Madaline)

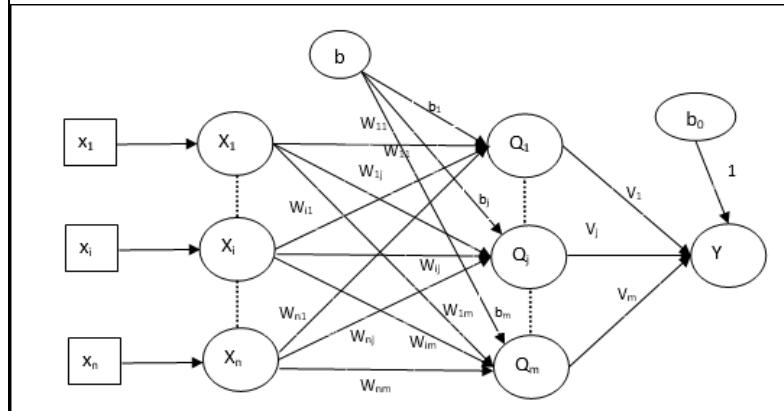
Madaline which stands for Multiple Adaptive Linear Neuron, is a network which consists of many Adalines in parallel. It will have a single output unit. Some important points about Madaline are as follows –

- It is just like a multilayer perceptron, where Adaline will act as a hidden unit between the input and the Madaline layer.
- The weights and the bias between the input and Adaline layers, as in we see in the Adaline architecture, are adjustable.
- The Adaline and Madaline layers have fixed weights and bias of 1.

- Training can be done with the help of Delta rule.

Architecture

The architecture of Madaline consists of “**n**” neurons of the input layer, “**m**” neurons of the Adaline layer, and 1 neuron of the Madaline layer. The Adaline layer can be considered as the hidden layer as it is between the input layer and the output layer, i.e. the Madaline layer.



Training Algorithm

By now we know that only the weights and bias between the input and the Adaline layer are to be adjusted, and the weights and bias between the Adaline and the Madaline layer are fixed.

Step 1 – Initialize the following to start the training –

- Weights
- Bias
- Learning rate α

for easy calculation and simplicity, weights and bias must be set equal to 0 and the learning rate must be set equal to 1.

Step 2 – Continue step 3-8 when the stopping condition is not true.

Step 3 – Continue step 4-7 for every bipolar training pair s:t.

Step 4 – Activate each input unit as follows –

$$x_i = s_i \quad (i = 1 \text{ to } n)$$

Step 5 – Obtain the net input at each hidden layer, i.e. the Adaline layer with the following relation –

$$Q_{inj} = b_j + \sum_i^n x_i w_{ij} \quad j = 1 \text{ to } m$$

Here ‘**b**’ is bias and ‘**n**’ is the total number of input neurons.

Step 6 – Apply the following activation function to obtain the final output at the Adaline and the Madaline layer –

$$f(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ -1 & \text{if } x < 0 \end{cases}$$

Output at the hidden (Adaline) unit

$$Q_j = f(Q_{inj})$$

Final output of the network

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

$$\text{i.e. } y_{inj} = b_0 + \sum_{j=1}^m Q_j v_j$$

Step 7 – Calculate the error and adjust the weights as follows –

Case 1 – if $y \neq t$ and $t = 1$ then,

$$w_{ij}(\text{new}) = w_{ij}(\text{old}) + \alpha(1 - Q_{inj})x_i$$

$$b_j(\text{new}) = b_j(\text{old}) + \alpha(1 - Q_{inj})$$

In this case, the weights would be updated on Q_j where the net input is close to 0 because $t = 1$.

Case 2 – if $y \neq t$ and $t = -1$ then,

$$w_{ik}(\text{new}) = w_{ik}(\text{old}) + \alpha(-1 - Q_{ink})x_i$$

$$b_k(\text{new}) = b_k(\text{old}) + \alpha(-1 - Q_{ink})$$

In this case, the weights would be updated on Q_k where the net input is positive because $t = -1$.

Here 'y' is the actual output and 't' is the desired/target output.

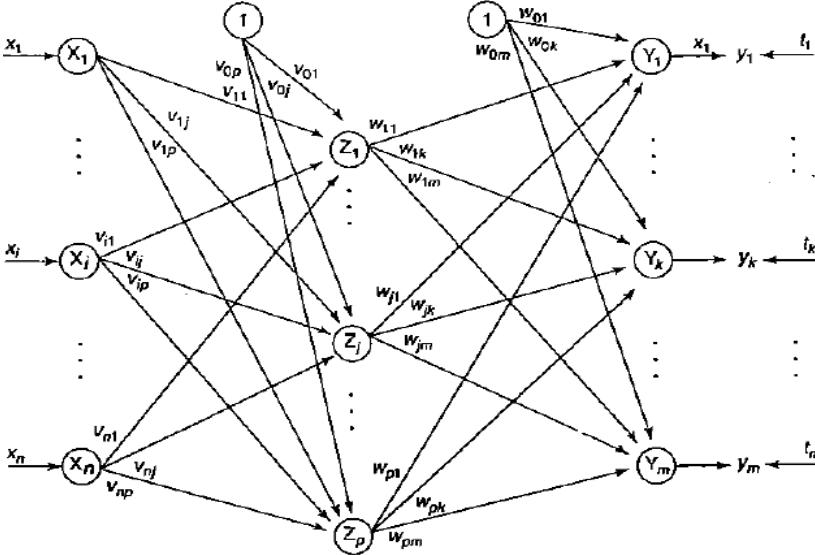
Case 3 – if $y = t$ then

There would be no change in weights.

Step 8 – Test for the stopping condition, which will happen when there is no change in weight or the highest weight change occurred during training is smaller than the specified tolerance.

The architecture of Madaline consists of “n” neurons of the input layer, “m” neurons of the Adaline layer, and 1 neuron of the Madaline layer. The Adaline layer can be considered as the hidden layer as it is between the input layer and the output layer, i.e. the Madaline layer.

UNIT -II
ARTIFICIAL NEURAL NETWORKS

1	Analyze the Back propagation of Neural Network with neat diagram.	[L4][CO2]	[12M]
	<p>Back propagation Network</p> <p>The back propagation learning algorithm is one of the most important developments in neural networks (Bryson and Ho, 1969; Werbos, 1974; Lecun, 1985; Parker, 1985; Umelhart, 1986). This learning algorithm is applied to multilayer feed-forward networks consisting of processing elements with continuous differentiable activation functions. The networks associated with back-propagation learning algorithm are also called back-propagation networks. (BPNS).</p> <p>For a given set of training input-output pair, this algorithm provides a procedure for changing the weights in a BPN to classify the given input patterns correctly. The</p>  <p>basic concept for this weight update algorithm is simply the gradient descent method. This is a method where error is propagated back to the hidden unit. Back propagation network is a training algorithm.</p> <p>A back-propagation neural network is a multilayer, feed-forward neural network consisting of an input layer, a hidden layer and an output layer. The neurons present in the hidden and output layers have biases, which are the connections from the units whose activation is always 1. The bias terms also act as weights. During the back propagation phase of learning, signals are sent in the reverse direction. The inputs sent to the BPN and the output obtained from the net could be either binary (0, 1) or bipolar (-1, +1). The activation function could be any function which increases monotonically and is also differentiable. The commonly used activation functions are binary, sigmoidal and bipolar sigmoidal activation functions. These functions are used in the BPN because of the following characteristics: (i) Continuity (ii) Differentiability (iii) Non-decreasing monotonic.</p> <p>Training Algorithm</p>		

Step 0: Initialize weights and learning rate (take some small random values).
 Step 1: Perform Steps 2-9 when stopping condition is false.
 Step 2: Perform Steps 3-8 for each training pair.

Feed forward Phase 1

Step 3: Each input unit receives input signal x_i and sends it to the hidden unit ($i = 1$ to n).

Step 4: Each hidden unit z_j ($j = 1$ to p) sums its weighted input signals to calculate net input:

$$z_{inj} = v_{0j} + \sum_{i=1}^n x_i v_{ij}$$

Calculate output of the hidden unit by applying its activation functions over z_{inj}

(binary or bipolar sigmoidal activation function):

$$z_j = f(z_{inj})$$

and send the output signal from the hidden unit to the input of output layer units.

Step 5: For each output unit y_k ($k = 1$ to m), calculate the net input:

$$y_{ink} = w_{0k} + \sum_{j=1}^p z_j w_{jk}$$

and apply the activation function to compute output signal

$$y_k = f(y_{ink})$$

Back-propagation of error (Phase II)

Step 6: Each output unit y_k ($k = 1$ to m) receives a target pattern corresponding to the input training pattern and computes the error correction term:

$$\delta_k = (t_k - y_k) f'(y_{ink})$$

The derivative $f'(y_{ink})$ can be calculated as in activation function section. On the basis of the calculated error correction term, update the change in weights and bias:

$$\Delta w_{jk} = \alpha \delta_k z_j; \quad \Delta w_{0k} = \alpha \delta_k$$

Also, send δ_k to the hidden layer backwards.

Step 7: Each hidden unit ($z_j = 1$ to p) sums its delta inputs from the output units:

$$\delta_{inj} = \sum_{k=1}^m \delta_k w_{jk}$$

The term δ_{inj} gets multiplied with the derivative of $f(z_{inj})$ to calculate the error term:

$$\delta_j = \delta_{inj} f'(z_{inj})$$

The derivative $f'(z_{inj})$ can be calculated as activation function section depending on whether binary or bipolar sigmoidal function is used. On the basis of the calculated δ_j , update the change in weights and bias:

$$\Delta v_{ij} = \alpha \delta_j x_i; \quad \Delta v_{0j} = \alpha \delta_j$$

Weight and bias updation (Phase III):

Step 8: Each output unit (y_k , $k = 1$ to m) updates the bias and weights:

$$w_{jk}(new) = w_{jk}(old) + \Delta w_{jk}$$

$$w_{0k}(new) = w_{0k}(old) + \Delta w_{0k}$$

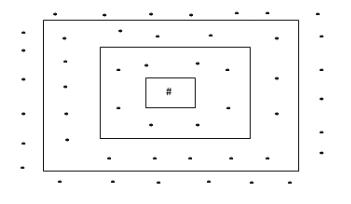
Each hidden unit (z_j , $j = 1$ to p) updates its bias and weights:

$$v_{ij}(new) = v_{ij}(old) + \Delta v_{ij}$$

$$v_{0j}(new) = v_{0j}(old) + \Delta v_{0j}$$

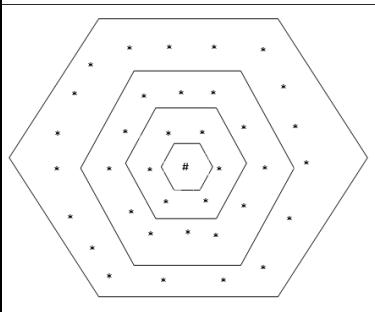
Step 9: Check for the stopping condition. The stopping condition may be certain number of epochs reached or when the actual output equals the target output.

2	Discuss Self –Organizing Map algorithm and its features .	[L2][CO2]	[12M]
	<p>Kohonen Self-Organizing Feature Maps</p> <p>It is one of the Dimensionality Reduction Technique. Suppose we have some pattern of arbitrary dimensions, however, we need them in one dimension or two dimensions. Then the process of feature mapping would be very useful to convert the wide pattern space into a typical feature space. to convert the arbitrary dimensions into 1-D or 2-D, it must also have the ability to preserve the neighbor topology like grid topology, rectangular grid topology.</p> <p>Rectangular Grid Topology</p> <p>This topology has 24 nodes in the distance-2 grid, 16 nodes in the distance-1 grid, and 8 nodes in the distance-0 grid, which means the difference between each rectangular grid is 8 nodes. The winning unit is indicated by #.</p>		



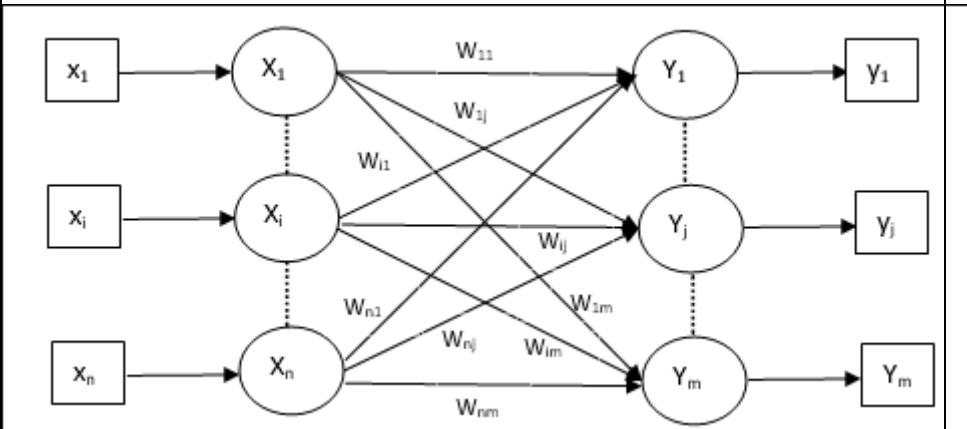
Hexagonal Grid Topology

This topology has 18 nodes in the distance-2 grid, 12 nodes in the distance-1 grid, and 6 nodes in the distance-0 grid, which means the difference between each rectangular grid is 6 nodes. The winning unit is indicated by #.



Architecture

The architecture of KSOM is similar to that of the competitive network. With the help of neighborhood schemes, discussed earlier, the training can take place over the extended region of the network.



Algorithm for training

Step 1 – Initialize the weights, the learning rate α and the neighborhood topological scheme.

Step 2 – Continue step 3-9, when the stopping condition is not true.

Step 3 – Continue step 4-6 for every input vector \mathbf{x} .

Step 4 – Calculate Square of Euclidean Distance for $j = 1 \text{ to } m$

$$D(j) = \sum_{i=1}^n \sum_{j=1}^m (x_i - w_{ij})^2$$

Step 5 – Obtain the winning unit J where $D(j)$ is minimum.

Step 6 – Calculate the new weight of the winning unit by the following relation –

$$w_{ij}(\text{new}) = w_{ij}(\text{old}) + \alpha[x_i - w_{ij}(\text{old})]$$

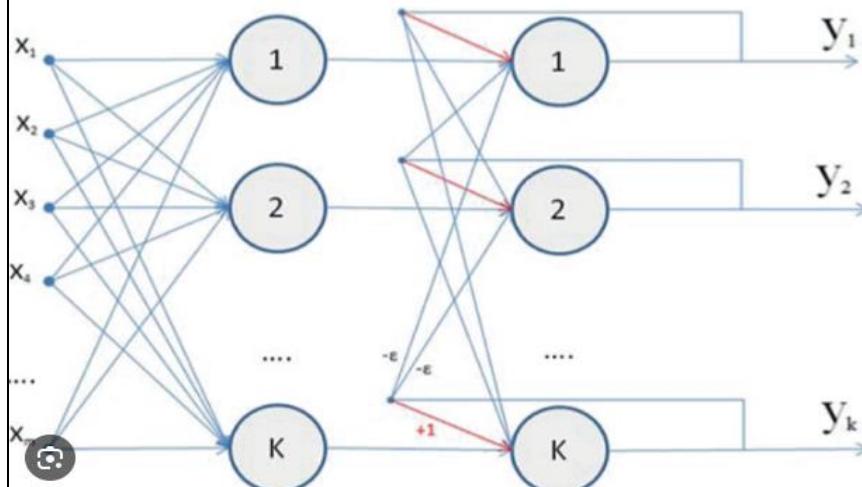
Step 7 – Update the learning rate α by the following relation –

$$\alpha(t+1) = 0.5\alpha t$$

Step 8 – Reduce the radius of topological scheme.

Step 9 – Check for the stopping condition for the network.

3		Analyze Fixed weight networks advanced neural network.	[L4][CO2]	[12M]
		Ans: 4(a) Hamming Networks and (5) Hopfield Network Explanation		
4	a	Explain Hamming neural network with neat diagram.	[L2][CO2]	[8M]
		<p>Hamming Network</p> <p>In most of the neural networks using unsupervised learning, it is essential to compute the distance and perform comparisons. This kind of network is Hamming network, where for every given input vectors, it would be clustered into different groups. Following are some important features of Hamming Networks –</p> <ul style="list-style-type: none"> • Lippmann started working on Hamming networks in 1987. • It is a single layer network. • The inputs can be either binary {0, 1} or bipolar {-1, 1}. • The weights of the net are calculated by the exemplar vectors. • It is a fixed weight network which means the weights would remain the same even during training. 		



Give Explanation of the above diagram.

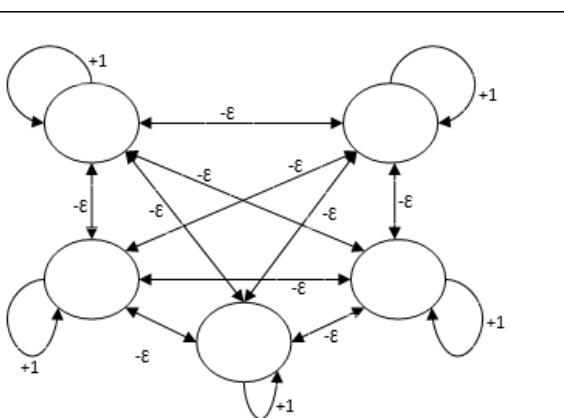
- b Explain Max network with architecture.

[L2][CO2] [4M]

Max Net

This is also a fixed weight network, which serves as a subnet for selecting the node having the highest input. All the nodes are fully interconnected and there exists symmetrical weights in all these weighted interconnections.

Architecture



It uses the mechanism which is an iterative process and each node receives inhibitory inputs from all other nodes through connections. The single node whose value is maximum would be active or winner and the activations of all other nodes would be inactive. Max Net uses identity activation function with

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{if } x \leq 0 \end{cases}$$

The task of this net is accomplished by the self-excitation weight of +1 and mutual inhibition magnitude, which is set like $[0 < \epsilon < \frac{1}{m}]$ where "m" is the total number of the nodes.

5

- Describe architectural functions and its characteristics of Hopfield Neural Network with neat sketch.

[L2][CO2] [12M]

Hopfield Networks

Hopfield neural network was invented by Dr. John J. Hopfield in 1982. It consists of a single layer which contains one or more fully connected recurrent neurons. The Hopfield network is commonly used for auto-association and optimization tasks.

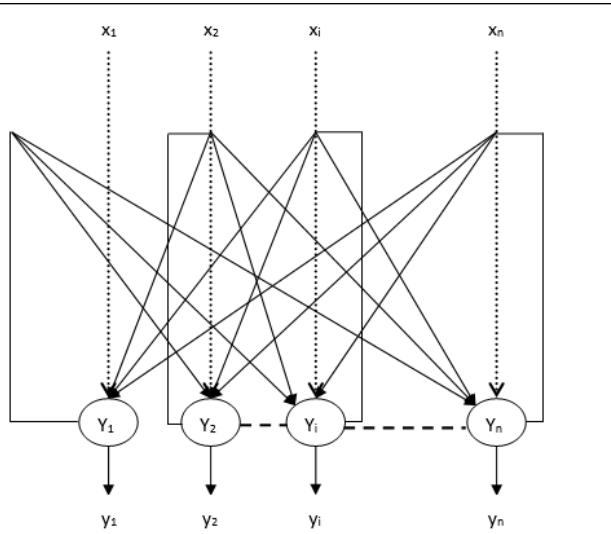
Discrete Hopfield Network

A Hopfield network which operates in a discrete line fashion or in other words, it can be said the input and output patterns are discrete vector, which can be either binary (0,1) or bipolar (+1, -1) in nature. The network has symmetrical weights with no self-connections i.e., $w_{ij} = w_{ji}$ and $w_{ii} = 0$.

Architecture

Following are some important points to keep in mind about discrete Hopfield network –

- This model consists of neurons with one inverting and one non-inverting output.
- The output of each neuron should be the input of other neurons but not the input of self.
- Weight/connection strength is represented by w_{ij} .
- Connections can be excitatory as well as inhibitory. It would be excitatory, if the output of the neuron is same as the input, otherwise inhibitory.
- Weights should be symmetrical, i.e. $w_{ij} = w_{ji}$



The output from \mathbf{Y}_1 going to \mathbf{Y}_2 , \mathbf{Y}_i and \mathbf{Y}_n have the weights w_{12} , w_{1i} and w_{1n} respectively. Similarly, other arcs have the weights on them.

Training Algorithm

During training of discrete Hopfield network, weights will be updated. As we know that we can have the binary input vectors as well as bipolar input vectors. Hence, in both the cases, weight updates can be done with the following relation

Case 1 – Binary input patterns

For a set of binary patterns $s(p)$, $p = 1$ to P

Here, $s(p) = s_1(p), s_2(p), \dots, s_i(p), \dots, s_n(p)$

Weight Matrix is given by

$$w_{ij} = \sum_{p=1}^P [2s_i(p) - 1][2s_j(p) - 1] \quad \text{for } i \neq j$$

Case 2 – Bipolar input patterns

For a set of binary patterns $s(p)$, $p = 1$ to P

Here, $s(p) = s_1(p), s_2(p), \dots, s_i(p), \dots, s_n(p)$

Weight Matrix is given by

$$w_{ij} = \sum_{p=1}^P [s_i(p)][s_j(p)] \quad \text{for } i \neq j$$

Testing Algorithm

Step 1 – Initialize the weights, which are obtained from training algorithm by using Hebbian principle.

Step 2 – Perform steps 3-9, if the activations of the network is not consolidated.

Step 3 – For each input vector \mathbf{X} , perform steps 4-8.

Step 4 – Make initial activation of the network equal to the external input vector \mathbf{X} as follows –

$$y_i = x_i \quad \text{for } i = 1 \text{ to } n$$

Step 5 – For each unit Y_i , perform steps 6-9.

Step 6 – Calculate the net input of the network as follows –

$$y_{ini} = x_i + \sum_j y_j w_{ji}$$

Step 7 – Apply the activation as follows over the net input to calculate the output –

$$y_i = \begin{cases} 1 & \text{if } y_{ini} > \theta_i \\ y_i & \text{if } y_{ini} = \theta_i \\ 0 & \text{if } y_{ini} < \theta_i \end{cases}$$

Here θ_i is the threshold.

Step 8 – Broadcast this output y_i to all other units.

Step 9 – Test the network for conjunction.

Energy Function Evaluation

An energy function is defined as a function that is bonded and non-increasing function of the state of the system.

Energy function E_f , also called **Lyapunov function** determines the stability of discrete Hopfield network, and is characterized as follows -

$$E_f = -\frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n y_i y_j w_{ij} - \sum_{i=1}^n x_i y_i + \sum_{i=1}^n \theta_i y_i$$

Condition – In a stable network, whenever the state of node changes, the above energy function will decrease.

Suppose when node i has changed state from $y_i^{(k)}$ to $y_i^{(k+1)}$ then the Energy change ΔE_f is given by the following relation

6	a Discuss Bidirectional Associate Memory with neat architecture	[L2][CO2]	[7M]
Bidirectional Associative Memory (BAM) <p>Bidirectional associative memory (BAM), first proposed by Bart Kosko in the year 1988. The BAM network performs forward and backward associative searches for stored stimulus responses. The BAM is a recurrent hetero associative pattern-marching nerwork that encodes binary or bipolar patterns using Hebbian learning rule</p> <p>Architecture</p> <p>The architecture of BAM network consists of two layers of neurons which are connected by directed weighted pare interconnections. The network dynamics involve two layers of interaction. The BAM network iterates by sending the signals back and forth between the two layers until all the neurons reach equilibrium. The weights associated with the network are bidirectional. Thus, BAM can respond to the inputs in either layer.</p> <p>Figure shows a BAM network consisting of n units in X layer and m units in Y layer. The layers can be connected in both directions(bidirectional) with the result the weight matrix sent from the X layer to the Y layer is W and the weight matrix for signals sent from the Y layer to the X layer is W^T. Thus, the Weight matrix is calculated in both directions.</p> <p>Determination of Weights</p> <p>Let the input vectors be denoted by $s(p)$ and target vectors by $t(p)$. $p = 1, \dots, P$. Then</p>			

the weight matrix to store a set of input and target vectors, where

$$\begin{aligned}s(p) &= (s_1(p), \dots, s_i(p), \dots, s_n(p)) \\ t(p) &= (t_1(p), \dots, t_j(p), \dots, t_m(p))\end{aligned}$$

can be determined by Hebb rule training algorithm. In case of input vectors being binary, the weight matrix $W = \{w_{ij}\}$ is given by

$$w_{ij} = \sum_{p=1}^P [2s_i(p) - 1][2t_j(p) - 1]$$

When the input vectors are bipolar, the weight matrix $W = \{w_{ij}\}$ can be defined as

$$w_{ij} = \sum_{p=1}^P [s_i(p)][t_j(p)]$$

The activation function is based on whether the input target vector pairs used are binary or bipolar

The activation function for the Y-layer

1. With binary input vectors is

$$y_j = \begin{cases} 1 & \text{if } y_{inj} > 0 \\ y_j & \text{if } y_{inj} = 0 \\ 0 & \text{if } y_{inj} < 0 \end{cases}$$

1. With binary input vectors is

$$x_i = \begin{cases} 1 & \text{if } x_{ini} > 0 \\ x_i & \text{if } x_{ini} = 0 \\ 0 & \text{if } x_{ini} < 0 \end{cases}$$

2. With bipolar input vectors is

$$y_j = \begin{cases} 1 & \text{if } y_{inj} > \theta_j \\ y_j & \text{if } y_{inj} = \theta_j \\ -1 & \text{if } y_{inj} < \theta_j \end{cases}$$

2. With bipolar input vectors is

$$x_i = \begin{cases} 1 & \text{if } x_{ini} > \theta_i \\ x_i & \text{if } x_{ini} = \theta_i \\ -1 & \text{if } x_{ini} < \theta_i \end{cases}$$

b

Analyze Auto Associative memory and Hetero Associative memory.

[L4][CO2]

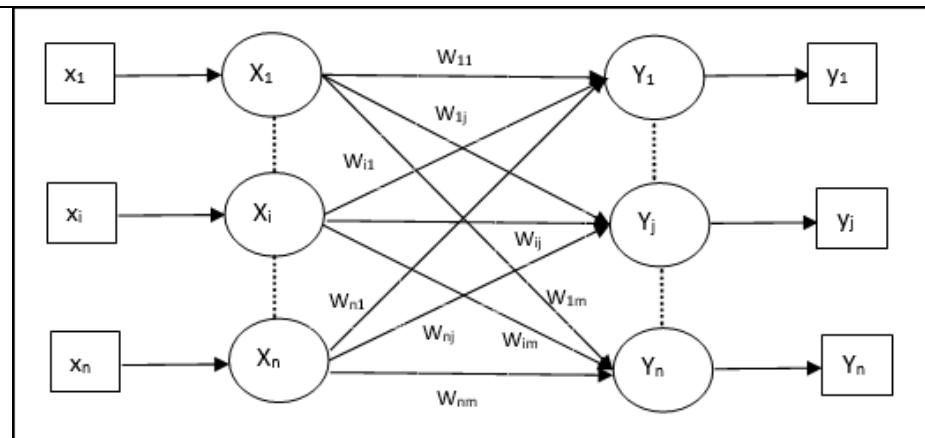
[5M]

Auto Associative Memory

This is a single layer neural network in which the input training vector and the output target vectors are the same. The weights are determined so that the network stores a set of patterns.

Architecture

As shown in the following figure, the architecture of Auto Associative memory network has ' n ' number of input training vectors and similar ' n ' number of output target vectors.



Testing Algorithm

Step 1 – Set the weights obtained during training for Hebb's rule.

Step 2 – Perform steps 3-5 for each input vector.

Step 3 – Set the activation of the input units equal to that of the input vector.

Step 4 – Calculate the net input to each output unit $j = 1 \text{ to } n$

$$y_{inj} = \sum_{i=1}^n x_i w_{ij}$$

Step 5 – Apply the following activation function to calculate the output

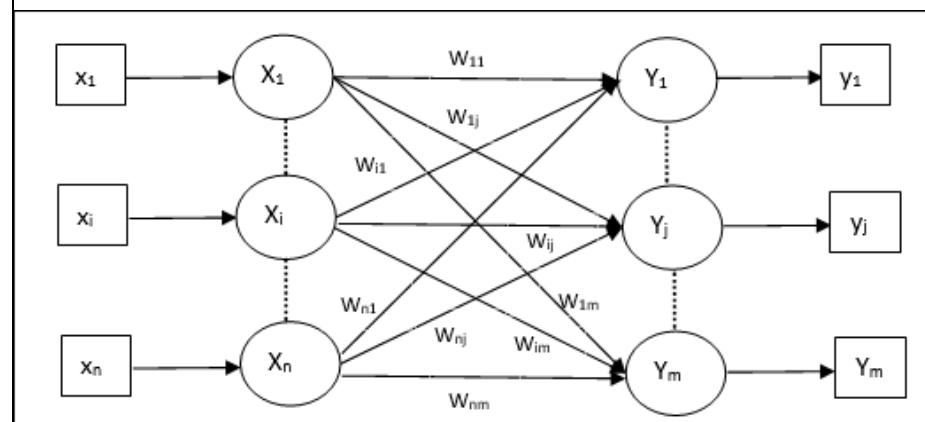
$$y_j = f(y_{inj}) = \begin{cases} +1 & \text{if } y_{inj} > 0 \\ -1 & \text{if } y_{inj} \leq 0 \end{cases}$$

Hetero Associative memory

Similar to Auto Associative Memory network, this is also a single layer neural network. However, in this network the input training vector and the output target vectors are not the same. The weights are determined so that the network stores a set of patterns. Hetero associative network is static in nature, hence, there would be no non-linear and delay operations.

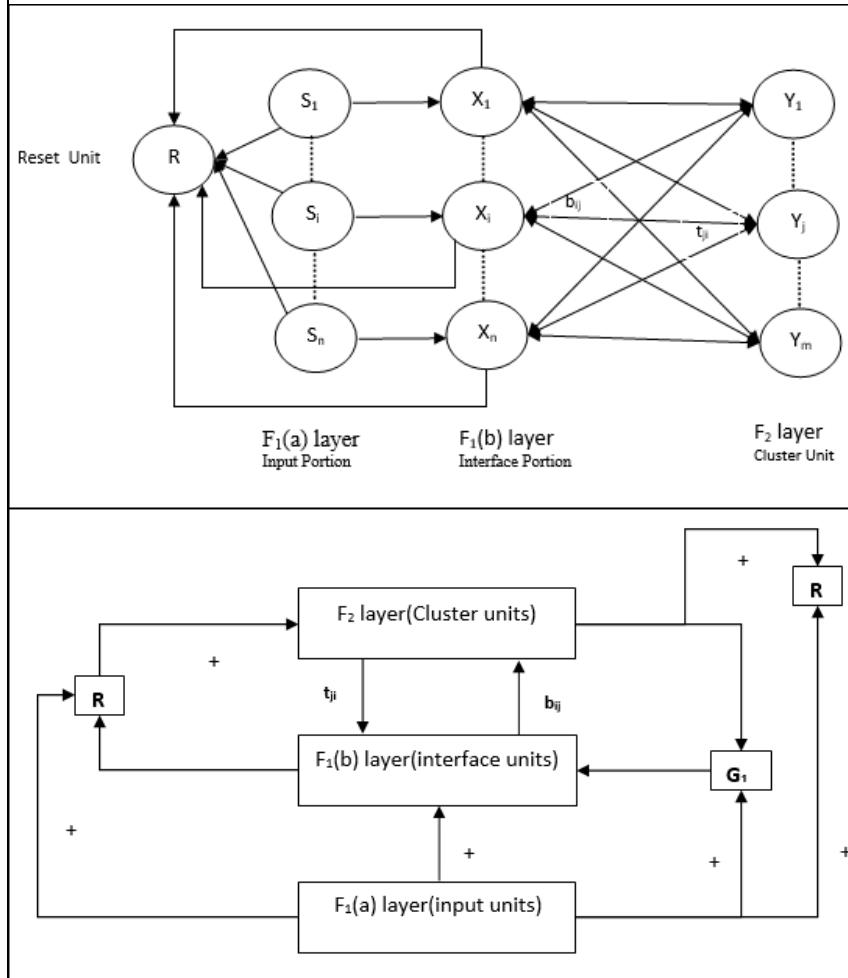
Architecture

As shown in the following figure, the architecture of Hetero Associative Memory network has ' n ' number of input training vectors and ' m ' number of output target vectors.



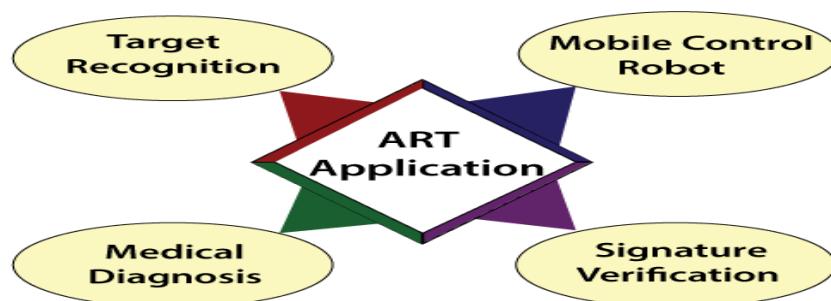
	<p>Testing Algorithm</p> <p>Step 1 – Set the weights obtained during training for Hebb's rule.</p> <p>Step 2 – Perform steps 3-5 for each input vector.</p> <p>Step 3 – Set the activation of the input units equal to that of the input vector.</p> <p>Step 4 – Calculate the net input to each output unit $j = 1 \text{ to } m$;</p> $y_{inj} = \sum_{i=1}^n x_i w_{ij}$ <p>Step 5 – Apply the following activation function to calculate the output</p> $y_j = f(y_{inj}) = \begin{cases} +1 & \text{if } y_{inj} > 0 \\ 0 & \text{if } y_{inj} = 0 \\ -1 & \text{if } y_{inj} < 0 \end{cases}$		
7	<p>a Generalize the Adaptive Resonance Theory Neural Network</p> <p>Adaptive Resonance Theory</p> <p>This network was developed by Stephen Grossberg and Gail Carpenter in 1987. It is based on competition and uses unsupervised learning model. Adaptive Resonance Theory (ART) networks, as the name suggests, is always open to new learning (adaptive) without losing the old patterns (resonance). Basically, ART network is a vector classifier which accepts an input vector and classifies it into one of the categories depending upon which of the stored pattern it resembles the most.</p> <p>Architecture of ART1</p> <p>It consists of the following two units –</p> <p>Computational Unit – It is made up of the following –</p> <ul style="list-style-type: none"> • Input unit (F_1 layer) – It further has the following two portions – <ul style="list-style-type: none"> $F_1(a)$ layer (Input portion) – In ART1, there would be no processing in this portion rather than having the input vectors only. It is connected to $F_1(b)$ layer (interface portion). $F_1(b)$ layer (Interface portion) – This portion combines the signal from the input portion with that of F_2 layer. $F_1(b)$ layer is connected to F_2 layer through bottom up weights b_{ij} and F_2 layer is connected to $F_1(b)$ layer through top down weights t_{ji}. • Cluster Unit (F_2 layer) – This is a competitive layer. The unit having the largest net input is selected to learn the input pattern. The activation of all other cluster unit are set to 0. • Reset Mechanism – The work of this mechanism is based upon the similarity between the top-down weight and the input vector. Now, if the degree of this similarity is less than the vigilance parameter, then the cluster is not allowed to learn the pattern and a rest would happen. 	[L6][CO2]	[8M]

Supplement Unit – Actually the issue with Reset mechanism is that the layer F_2 must have to be inhibited under certain conditions and must also be available when some learning happens. That is why two supplemental units namely, G_1 and G_2 is added along with reset unit, R . They are called **gain control units**. These units receive and send signals to the other units present in the network. ‘+’ indicates an excitatory signal, while ‘-’ indicates an inhibitory signal.



- b Identify some applications of ART Model

[L2][CO2] [4M]



Target recognition:

Fuzzy ARTMAP neural network can be used for automatic classification of targets depend on their radar range profiles. Tests on synthetic data show the fuzzy ARTMAP can result in substantial savings in memory requirements when related to k nearest neighbor(kNN) classifiers. The utilization of multi wave length profiles mainly improves the performance of both kinds of classifiers.

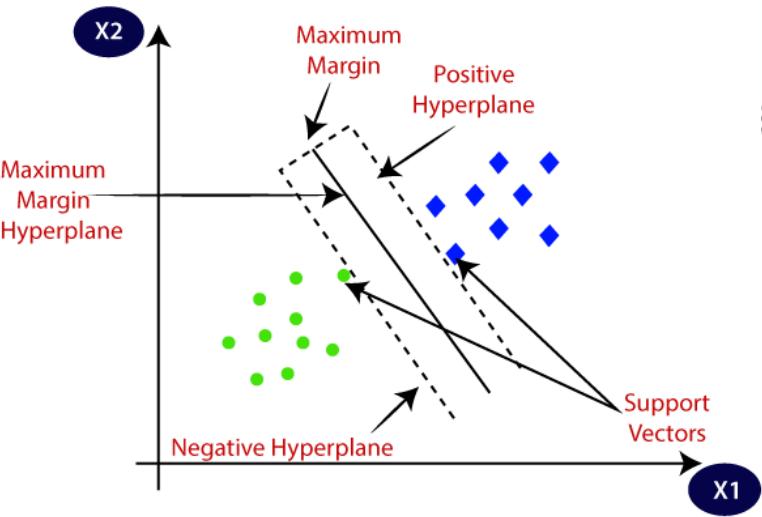
Medical diagnosis:

Medical databases present huge numbers of challenges found in general information management settings where speed, use, efficiency, and accuracy are the prime concerns. A direct objective of improved computer-assisted medicine is to help to deliver intensive care in situations that may be less than ideal. Working with these issues has stimulated several ART architecture developments, including ARTMAP-IC.

Signature verification:

Automatic signature verification is a well known and active area of research with various applications such as bank check confirmation, ATM access, etc. the training of the network is finished using ART1 that uses global features as input vector and the verification and recognition phase uses a two-step process. In the initial step, the input vector is coordinated with the stored reference vector, which was used as a training set, and in the second step, cluster formation takes place.

Mobile control robot: Nowadays, we perceive a wide range of robotic devices. It is still a field of research in their program part, called artificial intelligence. The human brain is an interesting subject as a model for such an intelligent system. Inspired by the structure of the human brain, an artificial neural emerges. Similar to the brain, the artificial neural network contains numerous simple computational units, neurons that are interconnected mutually to allow the transfer of the signal from the neurons to neurons. Artificial neural networks are used to solve different issues with good outcomes compared to other decision algorithms.

8	<p>a Illustrate the Support Vector Machine</p>	[L3][CO2]	[8M]
	<p>SVM:</p> <p>Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning. It is inherently binary classifier.</p>  <p>The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane</p> <p>Types of SVM</p> <p>SVM can be of two types:</p>		

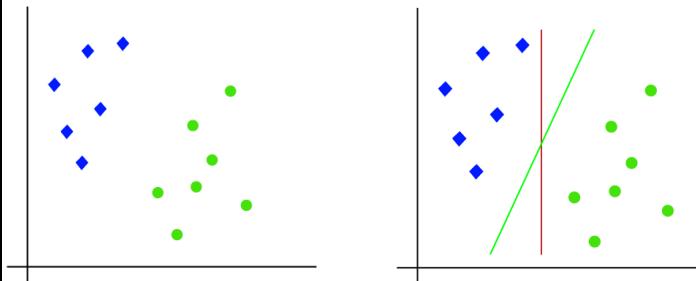
- **Linear SVM:** Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.
- **Non-linear SVM:** Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier.

Support Vectors:

The data points or vectors that are the closest to the hyperplane and which affect the position of the hyperplane are termed as Support Vector. Since these vectors support the hyperplane, hence called a Support vector.

Linear SVM:

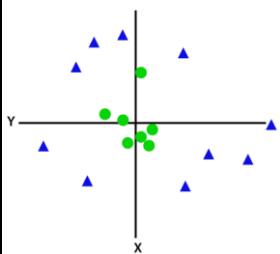
The working of the SVM algorithm can be understood by using an example. Suppose we have a dataset that has two tags (green and blue), and the dataset has two features x_1 and x_2 . We want a classifier that can classify the pair (x_1, x_2) of coordinates in either green or blue. Consider the below image:



So as it is 2-d space so by just using a straight line, we can easily separate these two classes. But there can be multiple lines that can separate these classes.

Non-Linear SVM:

If data is linearly arranged, then we can separate it by using a straight line, but for non-linear data, we cannot draw a single straight line. Consider the below image:



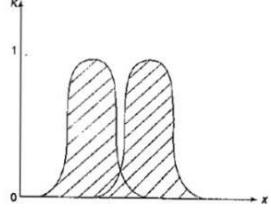
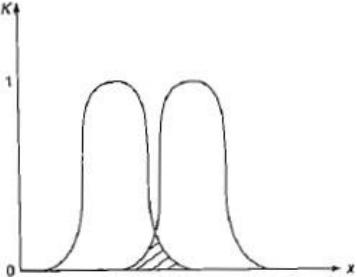
So to separate these data points, we need to add one more dimension. For linear data, we have used two dimensions x and y, so for non-linear data, we will add a third dimension z. It can be calculated as:

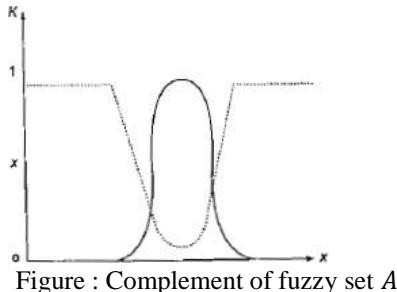
$$z = x^2 + y^2$$

By adding the third dimension, the sample space will become as below image:

	b	List out the Applications of SVM.	[L1][CO2]	[4M]
		<p>Applications of SVM</p> <ul style="list-style-type: none"> • Face detection • image classification • Medical Diagnosis • text categorization, etc. <p>Note: Give Explanations for each.</p>		
9		Explain the types of Hopfield Network with neat architectures.	L2][CO2]	[12M]
		For Answer: Refer Que No:5		
10		Summarize the following i)Feed forward phase in BPNN ii) Back propagation Error iii) Updation of weight and bias in BPNN	[L2][CO2]	[12M]
		It is same as question 1 answer		

UNIT -III
FUZZY SYSTEMS

1	<p>Explain the various types of operations and properties on Fuzzy Sets with examples.</p>	[L2][CO3]	[12M]
	<p><i>Operations on fuzzy sets</i></p> <p>1. Union: The union of fuzzy sets A and B, denoted by $\tilde{A} \cup \tilde{B}$ is defined as</p> $\mu_{\tilde{A} \cup \tilde{B}}(x) = \max_{\sim} [\mu_{\tilde{A}}(x), \mu_{\tilde{B}}(x)] = \mu_{\tilde{A}}(x) \vee \mu_{\tilde{B}}(x) \text{ for all } x \in U$ <p>where \vee indicates max operator. The Venn diagram for union operation of fuzzy sets A and B is shown below figure.</p>  <p>Figure: Union of fuzzy sets \tilde{A} and \tilde{B}</p> <p>2. Intersection: The intersection of fuzzy sets A and B, denoted by $\tilde{A} \cap \tilde{B}$, is defined as</p> $\mu_{\tilde{A} \cap \tilde{B}}(x) = \min_{\sim} [\mu_{\tilde{A}}(x), \mu_{\tilde{B}}(x)] = \mu_{\tilde{A}}(x) \wedge \mu_{\tilde{B}}(x) \text{ for all } x \in U$ <p>where \wedge indicates min operator. The Venn diagram for intersection operation of fuzzy sets A and B is shown below figure.</p> <p>Figure : Intersection of fuzzy sets A and B</p>  <p>Complement: When $\mu_{\tilde{A}}(x) \in [0,1]$, the complement of A, denoted as \tilde{A}' is defined by,</p> $\mu_{\tilde{A}'}(x) = 1 - \mu_{\tilde{A}}(x) \text{ for all } x \in U$ <p>The Venn diagram for complement operation of fuzzy set A is shown below figure.</p>		



More operations on Fuzzy sets

Algebraic sum

The algebraic sum ($A + B$) of fuzzy sets, fuzzy sets *A and B* is defined as

$$\mu_{A+B}(x) = \mu_A(x) + \mu_B(x) - \mu_A(x) \cdot \mu_B(x)$$

Algebraic product

The algebraic product ($A \cdot B$) of fuzzy sets, fuzzy sets *A and B* is defined as

$$\sim \quad \sim \quad \sim$$

$$\mu_{A \cdot B}(x) = \mu_A(x) \cdot \mu_B(x)$$

Bounded sum

The bounded sum ($\underset{\sim}{A} \oplus \underset{\sim}{B}$) of fuzzy sets, fuzzy sets *A and B* is defined as

$$\mu_{A \oplus B}(x) = \min \{1, \mu_A(x) + \mu_B(x)\}$$

Bounded difference

The bounded difference ($\underset{\sim}{A} \ominus \underset{\sim}{B}$) of fuzzy sets, fuzzy sets *A and B* is defined as

$$\mu_{A \ominus B}(x) = \max \{0, \mu_A(x) - \mu_B(x)\}$$

2

a Explain the various components of a Fuzzy Logic System with a neat block diagram.

[L2][CO3]

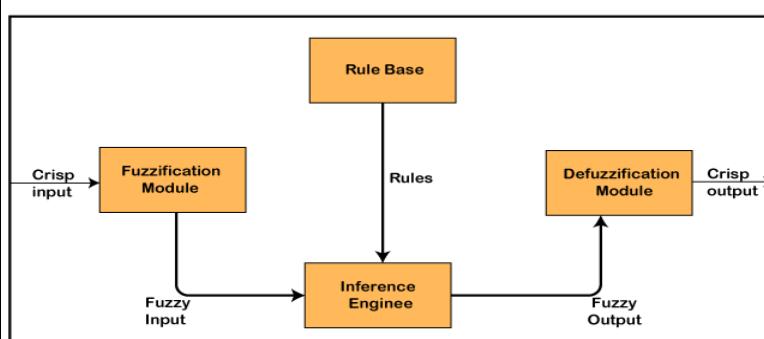
[8M]

Architecture of a Fuzzy Logic System

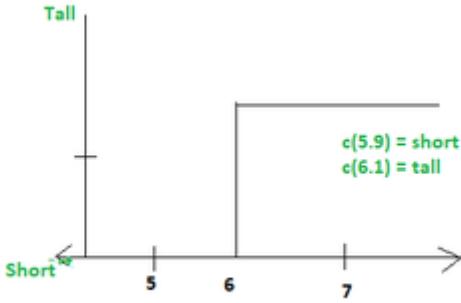
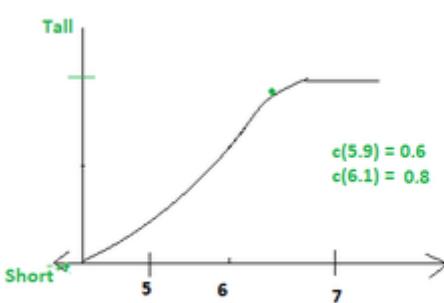
In the architecture of the **Fuzzy Logic** system, each component plays an important role. The architecture consists of the different four components which are given below.

1. Rule Base
2. Fuzzification
3. Inference Engine
4. Defuzzification

Following diagram shows the architecture or process of a Fuzzy Logic system:



	<p>1. Rule Base</p> <p>Rule Base is a component used for storing the set of rules and the If-Then conditions given by the experts are used for controlling the decision-making systems. There are so many updates that come in the Fuzzy theory recently, which offers effective methods for designing and tuning of fuzzy controllers. These updates or developments decreases the number of fuzzy set of rules.</p> <p>2. Fuzzification</p> <p>Fuzzification is a module or component for transforming the system inputs, i.e., it converts the crisp number into fuzzy steps. The crisp numbers are those inputs which are measured by the sensors and then fuzzification passed them into the control systems for further processing. This component divides the input signals into following five states in any Fuzzy Logic system:</p> <ul style="list-style-type: none"> ○ Large Positive (LP) ○ Medium Positive (MP) ○ Small (S) ○ Medium Negative (MN) ○ Large negative (LN) <p>3. Inference Engine</p> <p>This component is a main component in any Fuzzy Logic system (FLS), because all the information is processed in the Inference Engine. It allows users to find the matching degree between the current fuzzy input and the rules. After the matching degree, this system determines which rule is to be added according to the given input field. When all rules are fired, then they are combined for developing the control actions.</p> <p>4. Defuzzification</p> <p>Defuzzification is a module or component, which takes the fuzzy set inputs generated by the Inference Engine, and then transforms them into a crisp value. It is the last step in the process of a fuzzy logic system. The crisp value is a type of value which is acceptable by the user. Various techniques are present to do this, but the user has to select the best one for reducing the errors.</p>																	
b	Differentiate the fuzzy sets and classical sets	[L4][CO3]	[4M]															
	<table border="1"> <thead> <tr> <th>S.No</th><th>Crisp Set</th><th>Fuzzy Set</th></tr> </thead> <tbody> <tr> <td>1</td><td>Crisp set defines the value is either 0 or 1.</td><td>Fuzzy set defines the value between 0 and 1 including both 0 and 1.</td></tr> <tr> <td>2</td><td>It is also called a classical set.</td><td>It specifies the degree to which something is true.</td></tr> <tr> <td>3</td><td>It shows full membership</td><td>It shows partial membership.</td></tr> <tr> <td>4</td><td>Eg1. She is 18 years old.</td><td>Eg1. She is about 18 years old.</td></tr> </tbody> </table>	S.No	Crisp Set	Fuzzy Set	1	Crisp set defines the value is either 0 or 1.	Fuzzy set defines the value between 0 and 1 including both 0 and 1.	2	It is also called a classical set.	It specifies the degree to which something is true.	3	It shows full membership	It shows partial membership.	4	Eg1. She is 18 years old.	Eg1. She is about 18 years old.		
S.No	Crisp Set	Fuzzy Set																
1	Crisp set defines the value is either 0 or 1.	Fuzzy set defines the value between 0 and 1 including both 0 and 1.																
2	It is also called a classical set.	It specifies the degree to which something is true.																
3	It shows full membership	It shows partial membership.																
4	Eg1. She is 18 years old.	Eg1. She is about 18 years old.																

	Eg2. Rahul is 1.6m tall	Eg2. Rahul is about 1.6m tall.		
5	Crisp set application used for digital design.	Fuzzy set used in the fuzzy controller.		
6	It is bi-valued function logic.	It is infinite valued function logic		
7	Full membership means totally true/false, yes/no, 0/1.	Partial membership means true to false, yes to no, 0 to 1.		
 <p style="text-align: center;"><i>Crisp Set</i></p>				
 <p style="text-align: center;"><i>Fuzzy Set</i></p>				
3	a	Discuss the various operations and properties on Classical Sets with simple examples.	[L2][CO3]	[6M]
	<p>Operations on Classical sets</p> <p>Union :</p> <p>The union between two sets gives all those elements in the universe that belong to either set A or set B or both sets A and B. The union operation can be termed as a logical OR operation. The union of two sets A and B is given as</p> $A \cup B = \{x x \in A \text{ or } x \in B\}$ <p>The union of sets A and B is illustrated by the Venn diagram shown below</p>			

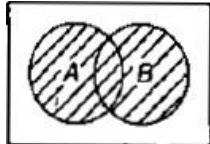


Figure 3.3: Union of two sets

Intersection

The intersection between two sets represents all those elements in the universe that simultaneously belong to both the sets. The intersection operation can be termed as a logical AND operation. The intersection of sets A and B is given by

$$A \cap B = \{x | x \in A \text{ and } x \in B\}$$

The intersection of sets A and B is illustrated by the Venn diagram shown below

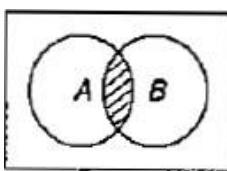


Figure 3.4: Intersection of two sets

Complement

The complement of set A is defined as the collection of all elements in universe X that do not reside in set A, i.e., the entities that do not belong to A. It is denoted by \bar{A} and is defined as

$$\bar{A} = \{x | x \notin A, x \in X\}$$

where X is the universal set and A is a given set formed from universe X. The complement operation of set A is show below

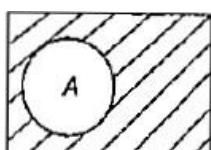


Figure 3.5: Complement of set A

Difference (Subtraction)

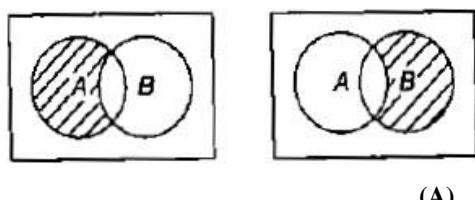
The difference of set A with respect to set B is the collection of all elements in the universe that belong to A but do not belong to B, i.e., the difference set consists of all elements that belong to A but do not belong to B. It is denoted by $A|B$ or $A - B$ and is given by

$$A|B \text{ or } (A - B) = \{x | x \in A \text{ and } x \notin B\} = A - (A \cap B)$$

The vice versa of it also can be performed

$$B|A \text{ or } (B - A) = B - (B \cap A) = \{x | x \in B \text{ and } x \notin A\}$$

The above operations are shown below

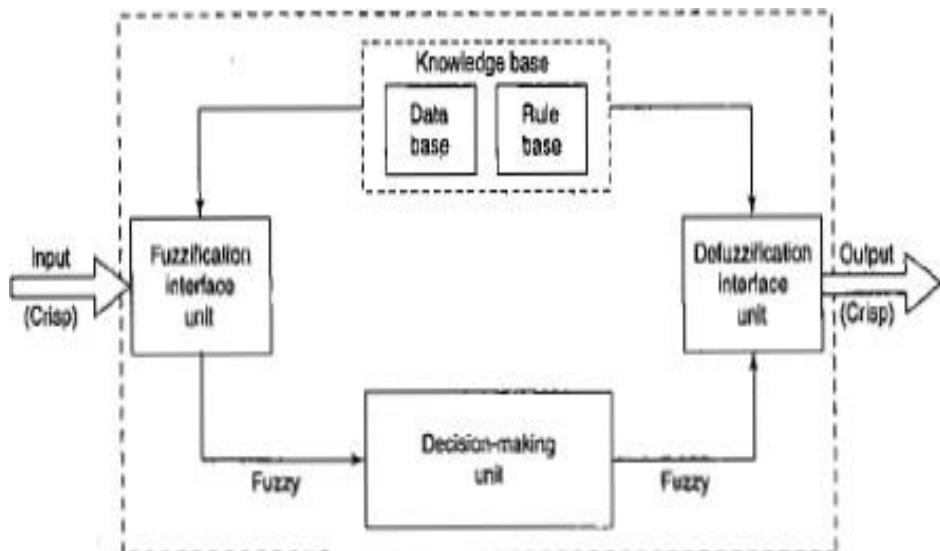
Figure 3.6: (A) Difference $A|B$ or $(A-B)$; (B) Difference $B|A$ or $(B-A)$

b	List out the various operations and composition operations on classical relations.explain it.	[L1][CO3]	[6M]
	<p><i>Classical relations</i></p> <p>A classical binary relation represents the presence or absence of a connection or interaction between the elements of two sets.</p> <p><i>Cartesian Product of Relation:</i></p> <p>An ordered r-tuple is an ordered sequence of r-elements expressed in the form $(a_1, a_2, a_3, \dots, a_r)$. An unordered tuple is a collection of r-elements without any restrictions in order.</p> <p>For crisp sets A_1, A_2, \dots, A_r, the set of all r-tuples $(a_1, a_2, a_3, \dots, a_r)$, where $a_1 \in A_1, a_2 \in A_2 \dots, a_r \in A_r$ is called the Cartesian product of $A_1, A_2 \dots, A_r$ and is denoted by $A_1 \times A_2 \times \dots \times A_r$.</p> <p><i>Cardinality of Classical Relation</i></p> <p>Consider n elements of the universe X being related to m elements of universe Y. When the cardinality of X = n_X and the cardinality of Y = n_Y, then the cardinality of relation R between the two universes is</p> $n_{X \times Y} = n_X \times n_Y$ <p>The cardinality of the power set P(X × Y) describing the relation is given by</p> $n_{P(X \times Y)} = 2^{(n_X n_Y)}$ <p><i>Operations on classical relations</i></p> <p>Let R and S be two separate relations on the Cartesian universe $X \times Y$. The null relation and the complete relation are defined by the relation matrices \emptyset_R and E_R. An example of a 3 × 3 form of the \emptyset_R and E_R matrices is given below:</p> $\emptyset_R = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad \text{and} \quad E_R = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$ <ol style="list-style-type: none"> 1. Union $R \cup S \rightarrow \chi_{R \cup S}(x, y) : \chi_{R \cup S}(x, y) = \max[\chi_R(x, y), \chi_S(x, y)]$ <ol style="list-style-type: none"> 2. Intersection $R \cap S \rightarrow \chi_{R \cap S}(x, y) : \chi_{R \cap S}(x, y) = \min[\chi_R(x, y), \chi_S(x, y)]$ <ol style="list-style-type: none"> 3. Complement $R \rightarrow \chi_{\bar{R}}(x, y) : \chi_{\bar{R}}(x, y) = 1 - \chi_R(x, y)$ <ol style="list-style-type: none"> 4. Containment $R \subset S \rightarrow \chi_R(x, y) : \chi_R(x, y) \leq \chi_S(x, y)$ <ol style="list-style-type: none"> 5. Identity $\emptyset \rightarrow \emptyset_R \quad \text{and} \quad X \rightarrow E_R$ <p><i>Composition of Classical Relations</i></p> <p>Let R be a relation that maps elements from universe X to universe Y and S be a relation that maps elements from universe Y to universe Z</p> $R \subseteq X \times Y \text{ and } S \subseteq Y \times Z$ <p>The composition operations are of two types:</p> <p><i>Max-min composition</i></p> <p>The max-min composition is defined by the function theoretic expression as</p> $T = R \circ S$ $\chi_T(x, z) = \vee_{y \in Y} [\chi_R(x, y) \wedge \chi_S(y, z)]$ <p><i>Max-product composition</i></p>		

		The max-product composition is defined by the function theoretic expression as $T = R \circ S$ $\chi_T(x, z) = \bigvee_{y \in Y} [\chi_R(x, y) \cdot \chi_S(y, z)]$		
4	a	Describe the various fuzzy composition relations with suitable examples. Fuzzy relations A fuzzy relation is a fuzzy set defined on the Cartesian product of classical sets $\{X_1, X_2, \dots, X_n\}$ where tuples (x_1, x_2, \dots, x_n) may have varying degrees of membership $\mu_R(x_1, x_2, \dots, x_n)$ within the relation. $R(X_1, X_2, \dots, X_n) = \bigcup_{X_1, X_2, \dots, X_n} \mu_R(x_1, x_2, \dots, x_n) (x_1, x_2, \dots, x_n), x_i \in X_i$ A fuzzy relation between two sets X and Y is called binary fuzzy relation and is denoted by $R(X, Y)$. Let $X = \{x_1, x_2, \dots, x_n\} \quad \text{and} \quad Y = \{y_1, y_2, \dots, y_m\}$ Fuzzy relation $R(X, Y)$ can be expressed by an $n \times m$ matrix as follows: $R(X, Y) = \begin{bmatrix} \mu_R(x_1, y_1) & \mu_R(x_1, y_2) & \dots & \mu_R(x_1, y_m) \\ \mu_R(x_2, y_1) & \mu_R(x_2, y_2) & \dots & \mu_R(x_2, y_m) \\ \vdots & \vdots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ \mu_R(x_n, y_1) & \mu_R(x_n, y_2) & \dots & \mu_R(x_n, y_m) \end{bmatrix}$ Operations on fuzzy relations 1. Union $\mu_{R \cup S}(x, y) = \max [\mu_R(x, y), \mu_S(x, y)]$ 2. Intersection $\mu_{R \cap S}(x, y) = \min [\mu_R(x, y), \mu_S(x, y)]$ 3. Complement $\mu_R^c(x, y) = 1 - \mu_R(x, y)$ 4. Containment $R \subset S \rightarrow \mu_R(x, y) \leq \mu_S(x, y)$ 5. Inverse The inverse of a fuzzy relation R on $X \times Y$ is denoted by R^{-1} . It is a relation on $Y \times X$ defined by $R^{-1}(y, x) = R(x, y)$ for all pairs $(y, x) \in Y \times X$. 6. Projection For a fuzzy relation $R(X, Y)$, let $[R \downarrow Y]$ denote the projection of R onto Y . Then $[R \downarrow Y]$ is a fuzzy relation in Y whose membership function is defined by There are two types of fuzzy composition techniques 1. Max- Min Composition 2. Min-Max Composition Fuzzy MAX-Min composition	[L2][CO3]	[8M]

	<p>The max-min composition of $R(X, Y)$ and $S(Y, Z)$, denoted by $R(X, Y) \circ S(Y, Z)$ is defined by $T(X, Z)$ as</p> $\mu_T(x, z) = \mu_{R \circ S}(x, z) = \max_{y \in Y} \{\min [\mu_R(x, y), \mu_S(y, z)]\}$ $= \vee_{y \in Y} [\mu_R(x, y) \wedge \mu_S(y, z)] \quad \forall x \in X, z \in Z$ <p>The min-max composition of $R(X, Y)$ and $S(Y, Z)$, denoted by $R(X, Y) \circ S(Y, Z)$ is defined by $T(X, Z)$ as</p> $\mu_T(x, z) = \mu_{R \circ S}(x, z) = \min_{y \in Y} \{\max [\mu_R(x, y), \mu_S(y, z)]\}$ $= \wedge_{y \in Y} [\mu_R(x, y) \vee \mu_S(y, z)] \quad \forall x \in X, z \in Z$ <p>Fuzzy max-product composition</p> <p>The max-product composition of $R(X, Y)$ and $S(Y, Z)$, denoted by $R(X, Y) \circ S(Y, Z)$ is defined by $T(X, Z)$ as</p> $\mu_T(x, z) = \mu_{R \circ S}(x, z) = \min_{y \in Y} [\mu_R(x, y) \cdot \mu_S(y, z)]$ $= \vee_{y \in Y} [\mu_R(x, y) \cdot \mu_S(y, z)]$ <p>The properties of fuzzy composition can be given as follows:</p> $\begin{aligned} R \circ S &\neq S \circ R \\ (R \circ S)^{-1} &= S^{-1} \circ R^{-1} \\ (R \circ S) \circ M &= R \circ (S \circ M) \end{aligned}$												
b	Differentiate classical relations and Fuzzy relations	[L4][CO3]	[4M]										
	<p>Classical relations Vs Fuzzy relations</p> <p>The difference between the classical and fuzzy relations can be expressed in terms of</p> <ol style="list-style-type: none"> 1. Cardinality 2. Operations 3. Properties <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center; padding: 2px;">Classical Relations</th> <th style="text-align: center; padding: 2px;">Fuzzy Relations</th> </tr> </thead> <tbody> <tr> <td colspan="2" style="text-align: center; padding: 2px;">Cardinality:</td> </tr> <tr> <td style="padding: 2px;">Consider n elements of universe X being related to the m elements of universe Y.</td> <td style="padding: 2px;">The cardinality of fuzzy sets on any universe is infinity; hence the cardinality of a fuzzy relation between two or more universe is also infinity.</td> </tr> <tr> <td style="padding: 2px;">When the cardinality of $X = {}^n X$ & the cardinality of $Y = {}^n Y$, then the cardinality of relation R between the two universe is</td> <td style="padding: 2px;">${}^n X * Y = {}^n X * {}^n Y$</td> </tr> <tr> <td style="padding: 2px;">The cardinality of the power set $P(X * Y)$ describing the relation is given by</td> <td style="padding: 2px;">${}^n P(X * Y) = 2^{({}^n X * {}^n Y)}$</td> </tr> </tbody> </table>	Classical Relations	Fuzzy Relations	Cardinality:		Consider n elements of universe X being related to the m elements of universe Y .	The cardinality of fuzzy sets on any universe is infinity; hence the cardinality of a fuzzy relation between two or more universe is also infinity.	When the cardinality of $X = {}^n X$ & the cardinality of $Y = {}^n Y$, then the cardinality of relation R between the two universe is	${}^n X * Y = {}^n X * {}^n Y$	The cardinality of the power set $P(X * Y)$ describing the relation is given by	${}^n P(X * Y) = 2^{({}^n X * {}^n Y)}$		
Classical Relations	Fuzzy Relations												
Cardinality:													
Consider n elements of universe X being related to the m elements of universe Y .	The cardinality of fuzzy sets on any universe is infinity; hence the cardinality of a fuzzy relation between two or more universe is also infinity.												
When the cardinality of $X = {}^n X$ & the cardinality of $Y = {}^n Y$, then the cardinality of relation R between the two universe is	${}^n X * Y = {}^n X * {}^n Y$												
The cardinality of the power set $P(X * Y)$ describing the relation is given by	${}^n P(X * Y) = 2^{({}^n X * {}^n Y)}$												

Operations	Classical Relations	Fuzzy Relations		
Union	$R \cup S \rightarrow \chi_R \cup \chi_S(x, y)$ = $\max [\chi_R(x, y), \chi_S(x, y)]$	$\mu_{R \cup S}(x, y)$ = $\max [\mu_R(x, y), \mu_S(x, y)]$		
Intersection	$R \cap S \rightarrow \chi_R \cap \chi_S(x, y) =$ $\min [\chi_R(x, y), \chi_S(x, y)]$	$\mu_{R \cap S}(x, y)$ = $\min [\mu_R(x, y), \mu_S(x, y)]$		
Complement	$R \rightarrow \chi_R(x, y); \chi_R(x, y) = 1 - \chi_R(x, y)$	$\mu_R(x, y) = 1 - \mu_R(x, y)$		
Containment	$R \subset S$ $\rightarrow \chi_R(x, y); \chi_R(x, y) \leq \chi_S(x, y)$	$R \subset S \Rightarrow \mu_R(x, y) \leq \mu_S(x, y)$		
Identity	$\phi \rightarrow \phi R \quad \& \quad X \rightarrow E_R$			
Inverse		The inverse of fuzzy relation R on X^*Y is denoted by R^{-1} . It is relation on Y^*X defined by $R^{-1}(y, x) = R(x, y)$ for all pairs $(y, x) \in Y^*X$		
Projection		For fuzzy relation $R(X, Y)$, let $[R \downarrow Y]$ denote the projection of R onto Y . $\overline{\mu_{[R \downarrow Y]}}(x, y) = \max_y \mu_R(x, y)$		
Properties	Classical Relations	Fuzzy Relations		
<ul style="list-style-type: none"> • Commutativity • Associativity • Distributivity • Involution • Idempotency • DeMorgan's Law • Excluded middle law 	<ul style="list-style-type: none"> • Commutativity • Associativity • Distributivity • Involution • Idempotency • DeMorgan's Law 			
5	Explain the Frame work of Fuzzy Inference Systems with neat sketch.			[L2][CO3] [12M]
	<p>Fuzzy Inference Systems</p> <p>Fuzzy rule based systems, fuzzy models, and fuzzy expert systems are generally known as systems. The key unit of a fuzzy logic system is FIS. The primary work of this system is decision making. FIS uses "IF ... THEN" rules along with connectors "OR" or "AND" for making necessary decision rules. The input to FIS may be fuzzy or crisp, but the output from FIS is always a fuzzy set.</p> <p>Construction and Working Principle of FIS</p>			



1. A Rule base that contains numerous fuzzy IF-THEN rules.
2. A database that defines the membership functions of fuzzy sets used in fuzzy rules.
3. Decision-making unit that performs operation on the rules.
4. Fuzzification interface unit that converts the crisp quantities into fuzzy quantities.

5. Defuzzification interface that converts the fuzzy quantities into crisp quantities. Initially, in the fuzzification unit, the crisp input is converted into a fuzzy input. Various fuzzification methods are employed for this. After this process, rule base is formed. Database and rule base are collectively called the knowledge base. Finally, defuzzification process is carried out to produce crisp output. Mainly, the fuzzy rules are formed in the rule base and suitable decisions are made in the decision-making unit

Methods of FIS:

There are two important types of FIS. They are

1. Mamdani FIS(1975);
2. Sugeno FIS(1985);

Advantages of Fuzzy Logic System

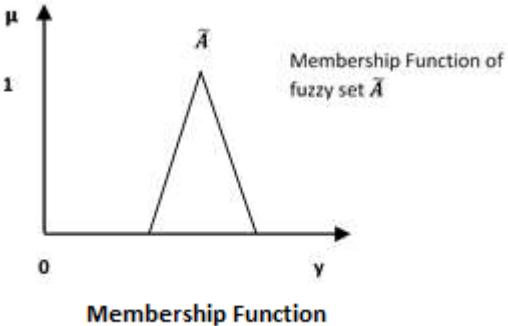
- This system can work with any type of inputs whether it is imprecise, distorted or noisy input information.
- The construction of Fuzzy Logic Systems is easy and understandable.
- Fuzzy logic comes with mathematical concepts of set theory and the reasoning of that is quite simple.
- It provides a very efficient solution to complex problems in all fields of life as it resembles human reasoning and decision-making.
- The algorithms can be described with little data, so little memory is required.

Disadvantages of Fuzzy Logic Systems

- Many researchers proposed different ways to solve a given problem through fuzzy logic which leads to ambiguity. There is no systematic approach to solve a given problem through fuzzy logic.
- Proof of its characteristics is difficult or impossible in most cases because every time we do not get a mathematical description of our approach.
- As fuzzy logic works on precise as well as imprecise data so most of the time accuracy is compromised.

Application

- It is used in the aerospace field for altitude control of spacecraft and satellites.
- It has been used in the automotive system for speed control, traffic control.
- It is used for decision-making support systems and personal evaluation in the large company business.

		<ul style="list-style-type: none"> It has application in the chemical industry for controlling the pH, drying, chemical distillation process. Fuzzy logic is used in Natural language processing and various intensive applications in Artificial Intelligence. Fuzzy logic is extensively used in modern control systems such as expert systems. Fuzzy Logic is used with Neural Networks as it mimics how a person would make decisions, only much faster. It is done by Aggregation of data and changing it into more meaningful data by forming partial truths as Fuzzy sets. 		
6	a	Demonstrate the membership functions in fuzzy logic.	[L3][CO4]	[6M]
		<p>Fuzzy logic is not logic that is fuzzy but logic that is used to describe fuzziness. This fuzziness is best characterized by its membership function. In other words, we can say that membership function represents the degree of truth in fuzzy logic.</p>  <p>Fuzzy membership functions</p> <p>Membership function defines the fuzziness in a fuzzy set irrespective of the elements in the set, which are discrete or continuous. A fuzzy set \tilde{A} in the universe of discourse X can be defined as a set of ordered pairs:</p> $\tilde{A} = \{(x, \mu_{\tilde{A}}(x)) \mid x \in X\}$ <p>where $\mu_{\tilde{A}}(\cdot)$ is called membership function of \tilde{A}. The membership function $\mu_{\tilde{A}}(\cdot)$ maps X to the membership space M, i.e., $\mu_{\tilde{A}}: X \rightarrow M$. The membership value ranges in the interval $[0, 1]$ i.e., the range of the membership function is a subset of the non-negative real numbers whose supremum is finite.</p> <p>Features of Membership Functions</p> <p>The three main basic features involved in characterizing membership function are the following.</p> <ol style="list-style-type: none"> Core <p>The core of a membership function for some fuzzy set \tilde{A} is defined as that region of universe that is characterized by complete membership in the set \tilde{A}. The core has elements x of the universe such that</p> $\mu_{\tilde{A}}(x) = 1$ <p>The core of a fuzzy set may be an empty set.</p>		

2. Support

The support of a membership function for a fuzzy set \tilde{A} is defined as that region of universe that is characterized by a non-zero membership in the set \tilde{A} .

$$\mu_{\tilde{A}}(x) > 0$$

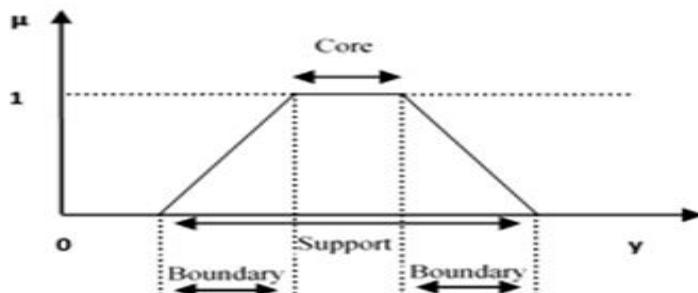
A fuzzy set whose support is a single element in X with $\mu_{\tilde{A}}(x) = 1$ is referred to as a fuzzy singleton.

3. Boundary

The support of a membership functions as the region of universe containing elements that have a non-zero but not complete membership. The boundary comprises those elements of x of the universe such that

$$0 < \mu_{\tilde{A}}(x) < 1$$

The boundary elements are those which possess partial membership in the fuzzy set \tilde{A} .



Features of Membership Function

Normal and Subnormal fuzzy sets

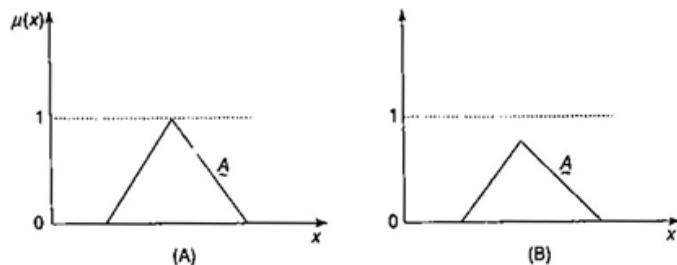


Figure 4.2: (A) Normal fuzzy set and (B) subnormal fuzzy set

Convex and Nonconvex fuzzy sets

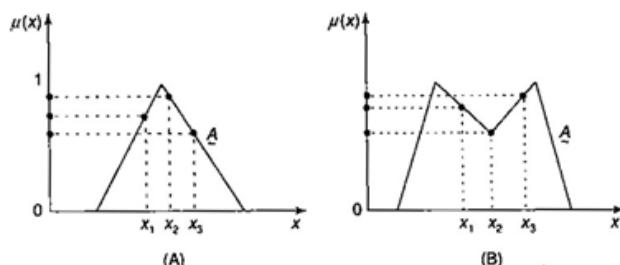


Figure 4.3: (A) Convex normal fuzzy set and (B) Nonconvex normal fuzzy set

Crossover point of fuzzy set

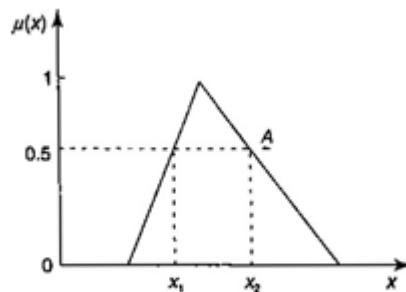


Figure 4.4: Crossover point of a fuzzy set

b Define Fuzzification and explain membership value assignment in fuzzy logic.

[L4][CO4]

[6M]

Fuzzification

It is defined as the process of transforming a crisp set to a fuzzy set or a fuzzy set to fuzzier set. Basically, this operation translates accurate crisp input values into linguistic variables.

Following are the two important methods of fuzzification –

Support Fuzzification(s-fuzzification) Method

In this method, the fuzzified set can be expressed with the help of the following relation –

$$A^s = \mu_1 Q(x_1) + \mu_2 Q(x_2) + \dots + \mu_n Q(x_n)$$

Here the fuzzy set $Q(x_i)Q(x_i)$ is called as kernel of fuzzification. This method is implemented by keeping μ_i constant and x_i being transformed to a fuzzy set $Q(x_i)Q(x_i)$.

Grade Fuzzification (g-fuzzification) Method

It is quite similar to the above method but the main difference is that it kept x_i constant and μ_i is expressed as a fuzzy set.

Membership value assignments

Intuition

Intuition method is based upon the common intelligence of human. It is the capacity of the human to develop membership functions on the basis of their own intelligence and understanding capacity.

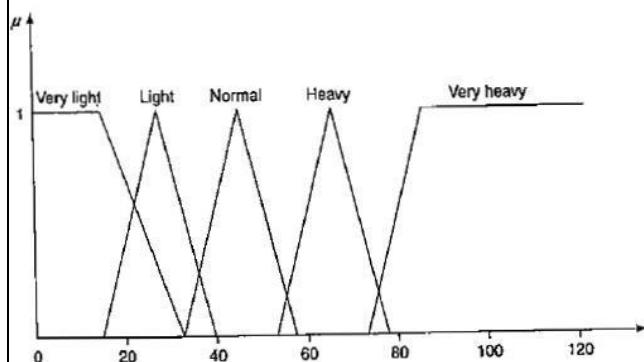


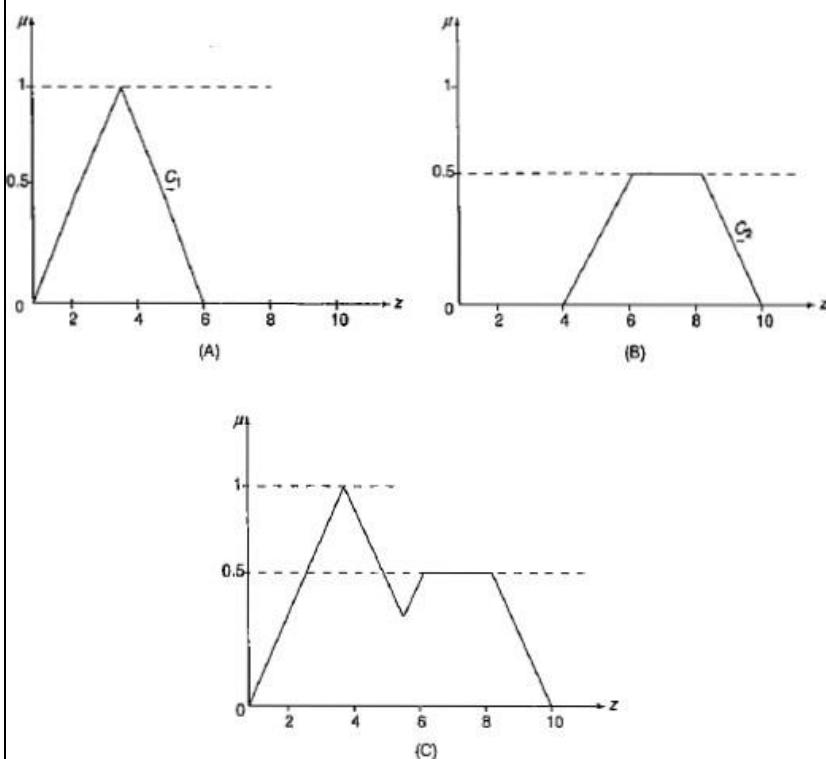
Figure : Membership function for the fuzzy variable “weight”

Inference

The inference method uses knowledge to perform deductive reasoning. Deduction achieves conclusion by means inference. There are various methods for performing deductive reasoning.

The membership functions may be defined by various shapes: triangular, trapezoidal, bell-shaped, Gaussian and so on. The inference method here is

	<p>discussed via triangular shape.</p> <p>Consider a triangle, where X, Y and Z are angles such that $X \geq Y \geq Z \geq 0$, and let U be the universe of triangles i.e.,</p> <p>There are various types of triangles available.</p> <p>$I =$ isosceles triangle (approximate) \sim</p> <p>$E =$ equilateral triangle (approximate) \sim</p> <p>$R =$ right-angle triangle (approximate) \sim</p> <p>$IR =$ isosceles and right-angle triangle (approximate) \sim</p> <p>$T =$ other triangles $\sim I$</p> <p>Rank ordering:</p> <p>The formation of government is based on the polling concept; to identify a best student, ranking may be performed; to buy a car, one can ask for several opinions and so on.</p> <p>Lambda -cuts for fuzzy sets:</p> <p>Consider a fuzzy set A. The set A_λ ($0 < \lambda < 1$), called the lambda (λ) -cut (or alpha [α] -cut) set, is a crisp set of the fuzzy set and is defined as follows:</p> $A_\lambda = \{(x, \mu_A(x) \geq \lambda)\} \quad \lambda \in [0, 1]$ <p>The set A_λ is called weak lambda-cut set if it consists of all the elements of a fuzzy set whose membership function have values greater than or equal to the specified value. The set A_λ is called strong lambda cut if it consist of all elements of a fuzzy set whose membership functions have values strictly greater than a specified value. A strong λ - cut set is given by</p> $A_\lambda = \{(x, \mu_A(x) > \lambda)\} \quad \lambda \in [0, 1]$ <p>The properties of λ cut are as follows</p> <ol style="list-style-type: none"> 1. $(A \cup B)_\lambda = A_\lambda \cup B_\lambda$ 2. $(A \cap B)_\lambda = A_\lambda \cap B_\lambda$ 3. $(A)_\lambda \neq \overline{A}$ except when $\lambda = 0.5$ 4. For any $\lambda \leq \beta$, where $0 \leq \beta \leq 1$, it is true that $A_\beta \subseteq A_\lambda$, where $A_0 = X$ 		
7	Analyze the different types of defuzzification methods with relevant mathematical expression and diagram.	[L4][CO4]	[12M]
	<p>Defuzzification methods</p> <p>Defuzzification is the process of conversion of a fuzzy quantity into a precise quantity. The output of a fuzzy set process may be union of two or more fuzzy membership functions defined on the universe of discourse of the output variable.</p>		



Defuzzification methods include the following:

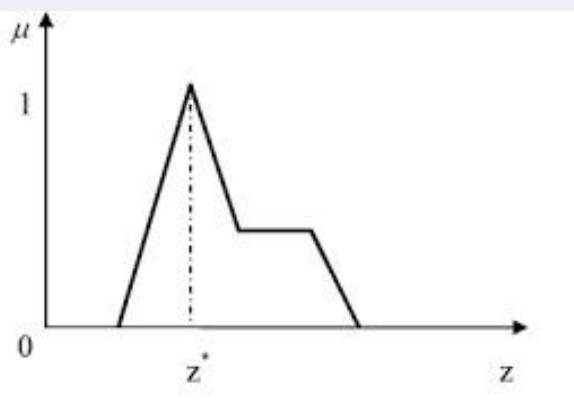
1. Max membership principle
2. Centroid method
3. Weighted average method
4. Mean-max membership
5. Center of sums
6. Center of largest area
7. First of maxima, last of maxima

Defuzzification Methods:

1. Max-Membership Principle

This method is also known as height method and is limited to peak output functions. This method is given by the algebraic expression:

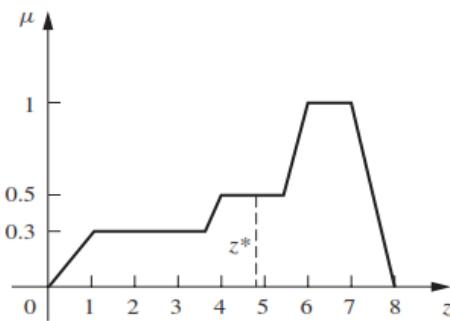
$$\mu(z^*) \geq \mu(z) \text{ for all } z \in Z.$$



2. Centroid Method

This method is also known as the centre of mass, centre of area or centre of gravity. It is the most commonly used defuzzification method. The defuzzified output z^* is given by:

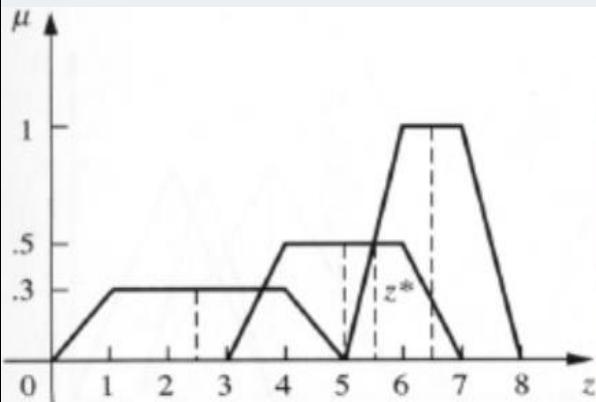
$$z^* = \int \mu(z) \cdot z dz / \int \mu(z) dz$$



3. Weighted Average Method

This method is valid for symmetrical output membership functions only. Each membership function is weighted by its maximum membership value. The output in the case is given by

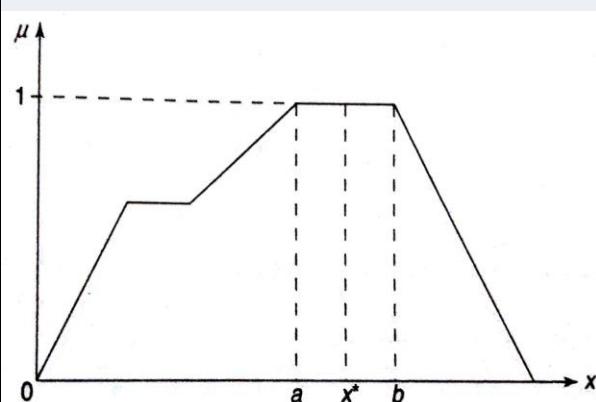
$$z^* = \frac{\sum \mu(z') \cdot z'}{\sum \mu(z')} ; \text{ where } z' \text{ is the maximum value of the membership function.}$$



4. Mean-Max Membership

This method is also known as the middle of the maxima. This is closely related to the max-membership method, except that the locations of the maximum membership can be nonunique. The output here is given by:

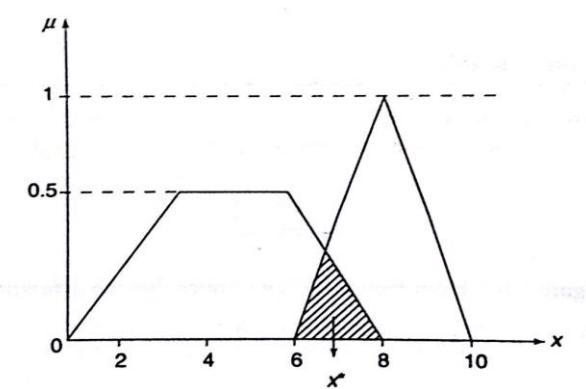
$$z^* = \frac{\sum z'}{n} ; \text{ where } z' \text{ is the maximum value of the membership function.}$$



5. Centre of Sums

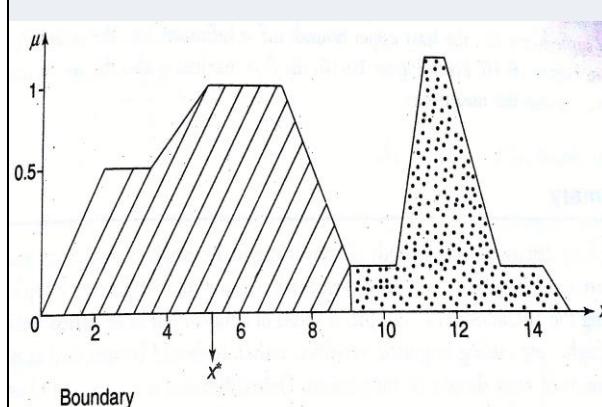
This method employs the algebraic sum of the individual fuzzy subsets instead of their union. The calculations here are very fast, but the main drawback is that the intersecting areas are added twice. The defuzzified value z* is given by

$$z^* = \frac{\int z^* \sum \mu(z) dz}{\int \sum \mu(z) dz}$$



6. Centre of Largest Area

This method can be adopted when the output of at least two convex fuzzy subsets which are not overlapping. The output, in this case, is biased towards a side of one membership function. When output fuzzy set has at least two convex regions, then the centre of gravity of the convex fuzzy subregion having the largest area is used to obtain the defuzzified value z^* . The value is given by

$$z^* = \frac{\int \mu c(z) \cdot z dz}{\int \sum \mu c(z) dz}$$


First of maxima, last of maxima

This method uses the overall output or union of all individual output fuzzy set \tilde{c}_i for determining the smallest value of the domain with maximized membership in \tilde{c}_j . The steps used for obtaining x^* are:

- Initially, the maximum height in the union is found:

$$hgt(\tilde{c}_i) = \sup_{x \in X} \mu_{\tilde{c}_i}(x)$$

where sup is the supremum that is the least upper bound.

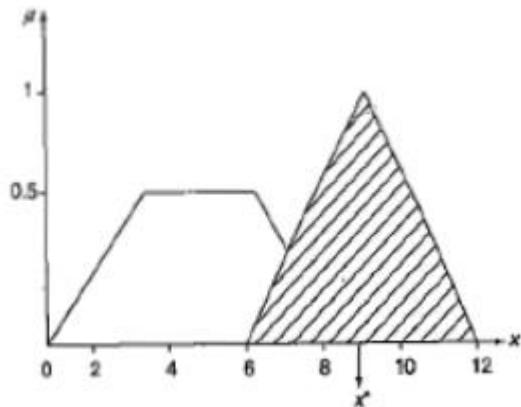
- Then the first of maxima is found:

$$x^* = \inf_{x \in X} [x \in X | \mu_{\tilde{c}_i}(x) = hgt(\tilde{c}_i)]$$

where inf is the infimum that is the greatest lower bound.

- After this the last maxima is found:

$$x^* = \sup_{x \in X} [x \in X | \mu_{\tilde{c}_i}(x) = hgt(\tilde{c}_i)]$$



8	<p>Summarize the following terms with suitable examples:</p> <p>i) Fuzzy Arithmetic ii) Fuzzy Measures</p>	[L5][CO4]	[12M]																					
	<p>Fuzzy Arithmetic:</p> <p>The imprecise data from the measuring instruments are generally expressed in the form of intervals, and suitable mathematical operations are performed over these intervals for obtain a reliable data of the measurements (which are also in the form of intervals). This type of computation is called <i>interval arithmetic</i> or <i>interval analysis</i>. Fuzzy arithmetic is a major concept in possibility theory.</p> <p>Fuzzy arithmetic is also a tool for dealing with fuzzy quantifiers in approximate reasoning. Fuzzy numbers are an extension of the concept of intervals. Intervals are considered at only one unique level. Fuzzy numbers consider them at several levels varying from 0 to 1.</p> <p>Interval Analysis of Uncertain Values</p> <p>Table 11-1 Set operations on intervals</p> <table border="1"> <thead> <tr> <th>Conditions</th> <th>Union, \cup</th> <th>Intersection, \cap</th> </tr> </thead> <tbody> <tr> <td>$a_1 > b_2$</td> <td>$[b_1, b_2] \cup [a_1, a_2]$</td> <td>$\emptyset$</td> </tr> <tr> <td>$b_1 > a_2$</td> <td>$[a_1, a_2] \cup [b_1, b_2]$</td> <td>$\emptyset$</td> </tr> <tr> <td>$a_1 > b_1, a_2 < b_2$</td> <td>$[b_1, b_2]$</td> <td>$[a_1, a_2]$</td> </tr> <tr> <td>$b_1 > a_1, b_2 < a_2$</td> <td>$[a_1, a_2]$</td> <td>$[b_1, b_2]$</td> </tr> <tr> <td>$a_1 < b_1 < a_2 < b_2$</td> <td>$[a_1, b_2]$</td> <td>$[b_1, a_2]$</td> </tr> <tr> <td>$b_1 < a_1 < b_2 < a_2$</td> <td>$[b_1, a_2]$</td> <td>$[a_1, b_2]$</td> </tr> </tbody> </table> <p>The mathematical operations performed on intervals are as follows:</p> <ol style="list-style-type: none"> 1. Addition (+): Let $A = [a_1, a_2]$ and $B = [b_1, b_2]$ be the two intervals defined. If $x \in [a_1, a_2]$ and $y \in [b_1, b_2]$, then $(x+y) \in [a_1 + b_1, a_2 + b_2]$ <p>This can be written as</p> $A + B = [a_1, a_2] + [b_1, b_2] = [a_1 + b_1, a_2 + b_2]$ <ol style="list-style-type: none"> 2. Subtraction (-): The subtraction for the two intervals of confidence is given by $A - B = [a_1, a_2] - [b_1, b_2] = [a_1 - b_2, a_2 - b_1]$ <p>That is, we subtract the larger value out of b_1 and b_2 from a_1 and the smaller value out of b_1 and b_2 from a_2.</p> <ol style="list-style-type: none"> 3. Multiplication (>): Let the two intervals of confidence be $A = [a_1, a_2]$ and $B = [b_1, b_2]$ defined on non-negative real line. The multiplication of these two intervals is given by $A \cdot B = [a_1, a_2] \cdot [b_1, b_2] = [a_1 \cdot b_1, a_2 \cdot b_2]$	Conditions	Union, \cup	Intersection, \cap	$a_1 > b_2$	$[b_1, b_2] \cup [a_1, a_2]$	\emptyset	$b_1 > a_2$	$[a_1, a_2] \cup [b_1, b_2]$	\emptyset	$a_1 > b_1, a_2 < b_2$	$[b_1, b_2]$	$[a_1, a_2]$	$b_1 > a_1, b_2 < a_2$	$[a_1, a_2]$	$[b_1, b_2]$	$a_1 < b_1 < a_2 < b_2$	$[a_1, b_2]$	$[b_1, a_2]$	$b_1 < a_1 < b_2 < a_2$	$[b_1, a_2]$	$[a_1, b_2]$		
Conditions	Union, \cup	Intersection, \cap																						
$a_1 > b_2$	$[b_1, b_2] \cup [a_1, a_2]$	\emptyset																						
$b_1 > a_2$	$[a_1, a_2] \cup [b_1, b_2]$	\emptyset																						
$a_1 > b_1, a_2 < b_2$	$[b_1, b_2]$	$[a_1, a_2]$																						
$b_1 > a_1, b_2 < a_2$	$[a_1, a_2]$	$[b_1, b_2]$																						
$a_1 < b_1 < a_2 < b_2$	$[a_1, b_2]$	$[b_1, a_2]$																						
$b_1 < a_1 < b_2 < a_2$	$[b_1, a_2]$	$[a_1, b_2]$																						

	<p>If we multiply an interval with a non-negative real number α, then we get</p> $\alpha \cdot \underline{A} = [\alpha, \alpha] \cdot [a_1, a_2] = [\alpha \cdot a_1, \alpha \cdot a_2]$ $\alpha \cdot \overline{B} = [\alpha, \alpha] \cdot [b_1, b_2] = [\alpha \cdot b_1, \alpha \cdot b_2]$ <p>4. Division (\div): The division of two intervals of confidence defined on a non-negative real line is given by</p> $\underline{A} \div \overline{B} = [a_1, a_2] \div [b_1, b_2] = \left[\frac{a_1}{b_2}, \frac{a_2}{b_1} \right]$ <p>If $b_1 = 0$ then the upper bound increases to $+\infty$. If $b_1 = b_2 = 0$, then interval of confidence is extended to $+\infty$.</p> <p>5. Image (\overline{A}): If $x \in [a_1, a_2]$ then its image $-x \in [-a_2, -a_1]$. Also if $\underline{A} = [a_1, a_2]$ then its image $\overline{A} = [-a_2, -a_1]$. Note that</p> $\underline{A} + \overline{A} = [a_1, a_2] + [-a_2, -a_1] = [a_1 - a_2, a_2 - a_1] \neq 0$ <p>That is, with image concept, the subtraction becomes addition of an image.</p> <p>6. Inverse (A^{-1}): If $x \in [a_1, a_2]$ is a subset of a positive real line, then its inverse is given by</p> $\left(\frac{1}{x} \right) \in \left[\frac{1}{a_2}, \frac{1}{a_1} \right]$ <p>Similarly, the inverse of \underline{A} is given by</p> $\underline{A}^{-1} = [a_1, a_2]^{-1} = \left[\frac{1}{a_2}, \frac{1}{a_1} \right]$ <p>That is, with inverse concept, division becomes multiplication of an inverse. For division by a non-negative number $\alpha > 0$, i.e. $(1/\alpha) \cdot \underline{A}$, we obtain</p> $\underline{A} \div \alpha = \underline{A} \cdot \left[\frac{1}{\alpha}, \frac{1}{\alpha} \right] = \left[\frac{a_1}{\alpha}, \frac{a_2}{\alpha} \right]$ <p>7. Max and min operations: Let two intervals of confidence be $\underline{A} = [a_1, a_2]$ and $\overline{B} = [b_1, b_2]$. Their max and min operations are defined by</p> $\text{Max: } \underline{A} \vee \overline{B} = [a_1, a_2] \vee [b_1, b_2] = [a_1 \vee b_1, a_2 \vee b_2]$ $\text{Min: } \underline{A} \wedge \overline{B} = [a_1, a_2] \wedge [b_1, b_2] = [a_1 \wedge b_1, a_2 \wedge b_2]$
--	--

Fuzzy Measures:

A fuzzy measure explains the imprecision or ambiguity assignment of an element a to two or more crisp sets. For representing uncertainty condition, known as ambiguity, we assign a value in the unit interval $[0, 1]$.

1. Belief and Plausibility Measures
2. Probability measures
3. Possibility, necessity measures

A fuzzy measure is defined by a function

$$g : P(X) \rightarrow [0, 1]$$

which assigns to each crisp subset of a universe of discourse X a number in the unit interval $[0, 1]$, where $P(X)$ is power set of X .

A fuzzy measure is also described as follows:

$$g : B \rightarrow [0, 1]$$

where $B \subseteq P(X)$ is a family of crisp subsets of X . Here B is a Borel field or a σ field. Also

Belief Measure:

The belief measure is a fuzzy measure that satisfies three axioms g1, g2 and g3 and an additional axiom of subadditivity. A belief measure is a function $\text{bel} : B \rightarrow [0, 1]$

satisfying axioms g1, g2 and g3 of fuzzy measures and subadditivity axiom. It is defined as follows:

$$\begin{aligned} \text{bel}(A_1 \cup A_2 \cup \dots \cup A_n) &\geq \sum_i \text{bel}(A_i) - \sum_{i < j} \text{bel}(A_i \cap A_j) \\ &\quad + \dots + (-1)^{n-1} \text{bel}(A_1 \cap A_2 \cap \dots \cap A_n) \end{aligned}$$

Possibility measures:

	<p>The possibility measure \prod and necessity measure N are functions</p> $\prod : \mathcal{B} \rightarrow [0, 1]$ $N : \mathcal{B} \rightarrow [0, 1]$ <p>such that both \prod and N satisfy the axioms g1, g2 and g3 of fuzzy measures and the following additional axiom (g7):</p> $\prod(A \cup B) = \max(\prod(A), \prod(B)) \quad \forall A, B \in \mathcal{B}$ $N(A \cap B) = \min(N(A), N(B)) \quad \forall A, B \in \mathcal{B}$ <p>As necessity and possibility measures are special subclasses of belief and plausibility measures, respectively, they are related to each other by</p> $\prod(A) = 1 - N(\bar{A})$ $N(A) = 1 - \prod(\bar{A}) \quad \forall A \in \sigma \text{ field}$ <p>Probability Measures</p> <p>A probability measure is a function</p> $P : \mathcal{B} \rightarrow [0, 1]$ <p>satisfying the three axioms g1, g2 and g3 of fuzzy measures and the additivity axiom (axiom g6) as follows:</p> $P(A \cup B) = P(A) + P(B) \text{ whenever } A \cap B = \emptyset, A, B \in \mathcal{B}$ $P : X \rightarrow [0, 1] \text{ such that } P(x) = m(\{x\})$ <p>This function $P(X)$ is called probability distribution function. Within probability measure, the total ignorance is expressed by the uniform probability distribution function</p> $P(x) = m(\{x\}) = \frac{1}{ X } \text{ for all } x \in X$		
9	Explain the Fuzzy rule base and approximate reasoning in Fuzzy logic.	[L2][CO4]	[12M]
	<p>Fuzzy Logic Rule Base</p> <p>It is a known fact that a human being is always comfortable making conversations in natural language. The representation of human knowledge can be done with the help of following natural language expression –</p> <p style="text-align: center;">IF antecedent THEN consequent</p> <p>The expression as stated above is referred to as the Fuzzy IF-THEN rule base.</p> <p>Canonical Form</p> <p>Following is the canonical form of Fuzzy Logic Rule Base –</p> <p>Rule 1 – If condition C1, then restriction R1</p> <p>Rule 2 – If condition C1, then restriction R2</p> <p>.</p> <p>.</p> <p>.</p> <p>Rule n – If condition C1, then restriction Rn</p> <p><i>Interpretations of Fuzzy IF-THEN Rules</i></p> <p>Fuzzy IF-THEN Rules can be interpreted in the following four forms –</p> <p>Assignment Statements</p> <p>These kinds of statements use “=” (equal to sign) for the purpose of assignment. They are of the following form –</p> $a = hello$ $climate = summer$ <p>Conditional Statements</p> <p>These kinds of statements use the “IF-THEN” rule base form for the purpose of condition. They are of the following form –</p> $IF temperature is high THEN Climate is hot$ $IF food is fresh THEN eat.$		

	<p>Unconditional Statements They are of the following form –</p> <p style="text-align: center;"><i>GOTO 10</i> <i>turn the Fan off</i></p> <p>Linguistic Variable We have studied that fuzzy logic uses linguistic variables which are the words or sentences in a natural language. For example, if we say temperature, it is a linguistic variable; the values of which are very hot or cold, slightly hot or cold, very warm, slightly warm, etc. The words very, slightly are the linguistic hedges.</p> <p>Following are the different modes of approximate reasoning –</p> <p>Categorical Reasoning In this mode of approximate reasoning, the antecedents, containing no fuzzy quantifiers and fuzzy probabilities, are assumed to be in canonical form.</p> <p>Qualitative Reasoning In this mode of approximate reasoning, the antecedents and consequents have fuzzy linguistic variables; the input-output relationship of a system is expressed as a collection of fuzzy IF-THEN rules. This reasoning is mainly used in control system analysis.</p> <p>Syllogistic Reasoning In this mode of approximation reasoning, antecedents with fuzzy quantifiers are related to inference rules. This is expressed as –</p> <p style="text-align: center;">x = $S_1 A$'s are B's y = $S_2 C$'s are D's ----- z = $S_3 E$'s are F's</p> <p>Here A,B,C,D,E,F are fuzzy predicates.</p> <ul style="list-style-type: none"> • S_1 and S_2 are given fuzzy quantifiers. • S_3 is the fuzzy quantifier which has to be decided. <p>Dispositional Reasoning In this mode of approximation reasoning, the antecedents are dispositions that may contain the fuzzy quantifier “usually”. The quantifier Usually links together the dispositional and syllogistic reasoning; hence it plays an important role. For example, the projection rule of inference in dispositional reasoning can be given as follows –</p> <p style="text-align: center;">usually((L,M) is R) \Rightarrow usually (L is [R ↓ L])</p> <p>Here $[R \downarrow L]$ is the projection of fuzzy relation R on L</p>			
10	a	Compare Mamdani FIS and Sugeno FIS.	[L5][CO4]	[6M]
		The main difference between Mamdani and Sugeno methods lies in the output membership functions. The Sugeno output membership functions are either linear or constant. The difference also lies in the consequents of their fuzzy rules as a result their aggregation and defuzzification procedures differ suitably. A large number of fuzzy rules must be employed in Sugeno method		

	<p>for approximating periodic or highly oscillatory functions. The configuration of Sugeno fuzzy systems can be reduced and it becomes smaller than that of Mamdani fuzzy systems if nontriangular or nontrapezoidal fuzzy input sets are used. Sugeno controllers have more adjustable parameters in the rule consequent and the number of parameters grows exponentially with the increase of the number of input variables. There exist several mathematical results for Sugeno fuzzy controllers than for Mamdani controllers. Formation of Mamdani FIS is easier than Sugeno FIS.</p> <ul style="list-style-type: none"> • The main advantage of Mamdani method are: <ol style="list-style-type: none"> 1. It has widespread acceptance. 2. It is well-suitable for human input. 3. It is intuitive. • The advantages of Sugeno method include: <ol style="list-style-type: none"> 1. It is computationally efficient. 2. It is compact and works well with linear technique, optimization technique and adaptive technique. 3. It is best suited for analysis. 4. It has a guaranteed continuity of the output surface. 		
b	Demonstrate the Fuzzy Decision Making briefly.	[L3][CO4]	[6M]
	<p>Fuzzy decision making:</p> <p>It is an activity which includes the steps to be taken for choosing a suitable alternative from those that are needed for realizing a certain goal.</p> <p>Steps for Decision Making</p> <p>Determining the Set of Alternatives – In this step, the alternatives from which the decision has to be taken must be determined.</p> <p>Evaluating Alternative – Here, the alternatives must be evaluated so that the decision can be taken about one of the alternatives.</p> <p>Comparison between Alternatives – In this step, a comparison between the evaluated alternatives is done.</p> <p>Types of Decision making:</p> <p>Individual Decision Making: In this type of decision making, only a single person is responsible for taking decisions. The decision making model in this kind can be characterized as –</p>		

- Set of possible actions
- Set of goals $G_i (i \in X_n)$;
- Set of Constraints $C_j (j \in X_m)$

The goals and constraints stated above are expressed in terms of fuzzy sets.

Now consider a set A. Then, the goal and constraints for this set are given by –

$$G_i(a) = \text{composition } [G_i(a)] = G_i^1(G_i(a)) \text{ with } G_i^1$$

$$C_j(a) = \text{composition } [C_j(a)] = C_j^1(C_j(a)) \text{ with } C_j^1 \quad \text{for } a \in A$$

The fuzzy decision in the above case is given by –

$$F_D = \min[i \in X_n^{in} fG_i(a), j \in X_m^{in} fC_j(a)]$$

Multi-person Decision Making

Decision making in this case includes several persons so that the expert knowledge from various persons is utilized to make decisions.

$$\text{Number of persons preferring } x_i \text{ to } x_j = N(x_i, x_j)$$

$$\text{Total number of decision makers} = n$$

$$\text{Then, } SC(x_i, x_j) = \frac{N(x_i, x_j)}{n}$$

Multi-objective Decision Making

Multi-objective decision making occurs when there are several objectives to be realized. There are following two issues in this type of decision making –

- To acquire proper information related to the satisfaction of the objectives by various alternatives.
- To weigh the relative importance of each objective.

Mathematically we can define a universe of n alternatives as –

$A = [a_1, a_2, \dots, a_i, \dots, a_n]$ And the set of "m" objectives as $O = [o_1, o_2, \dots, o_i, \dots, o_n]$

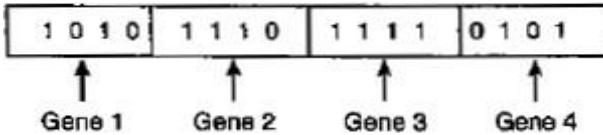
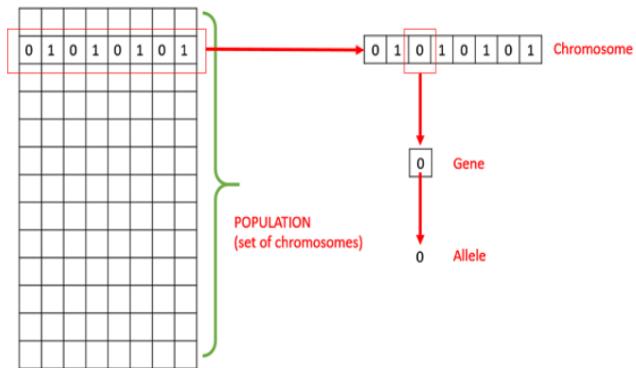
Multi-attribute Decision Making

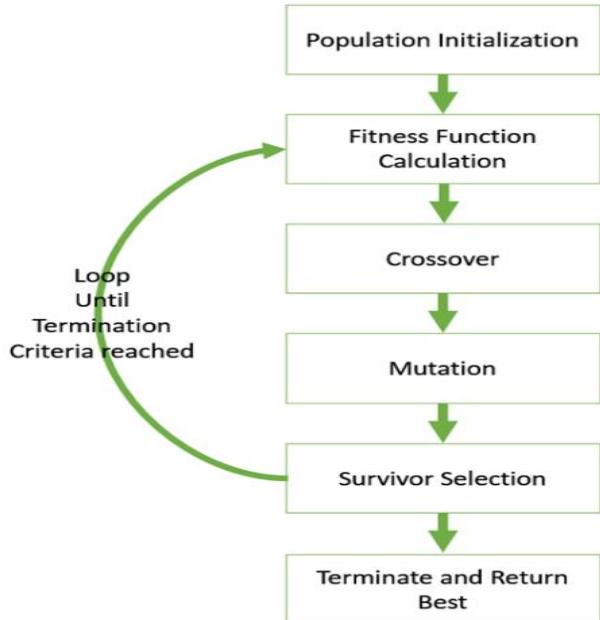
Multi-attribute decision making takes place when the evaluation of alternatives can be carried out based on several attributes of the object. The attributes can be numerical data, linguistic data and qualitative data.

Mathematically, the multi-attribute evaluation is carried out on the basis of linear equation as follows –

$$Y = A_1X_1 + A_2X_2 + \dots + A_iX_i + \dots + A_rX_r$$

UNIT -IV
GENETIC ALGORITHMS

1	<p>Define the Genetic Algorithm with basic terminologies and illustrate the working principle of Genetic Algorithm?</p> <p>Basic Terminologies in Genetic Algorithm</p> <p>Individuals: An individual is a single solution.</p> <p>Population – It is a subset of all the possible (encoded) solutions to the given problem.</p> <p>Chromosomes – A chromosome is one such solution to the given problem.</p> <p>Gene – A gene is one element position of a chromosome. Genes are the basic "instructions" for building a GA. A gene is a bit string of arbitrary lengths</p>  <p style="text-align: center;">Figure: Representation of a gene</p> <p>Allele – It is the value a gene takes for a particular chromosome.</p>  <p>Genotype – Genotype is the population in the computation space. In the computation space, the solutions are represented in a way which can be easily understood and manipulated using a computing system.</p> <p>Phenotype – Phenotype is the population in the actual real world solution space in which solutions are represented in a way they are represented in real world situations.</p> <p>Decoding and Encoding – For simple problems, the phenotype and genotype spaces are the same.</p> <p>Fitness Function – A fitness function simply defined is a function which takes the solution as input and produces the suitability of the solution as the output. In some cases, the fitness function and the objective function may be the same, while in others it might be different based on the problem.</p> <p>Genetic Operators – These alter the genetic composition of the offspring. These include crossover, mutation, selection, etc.</p>	[L3][CO5]	[12M]



Basic structure of GA

We start with an initial population (which may be generated at random or seeded by other heuristics), select parents from this population for mating. Apply crossover and mutation operators on the parents to generate new off-springs. And finally these off-springs replace the existing individuals in the population and the process repeats. In this way genetic algorithms actually try to mimic the human evolution to some extent.

Working Principle of Genetic Algorithm (General Genetic Algorithm)

Step 1: Create a random initial state

An initial population is created from a random selection of solutions (which are analogous to chromosomes). This is unlike the situation for symbolic AI systems, where the initial State in a problem is already given.

Step 2: Evaluate fitness

A value for fitness is assigned to each solution (chromosome) depending on how close it actually is w solving the problem (thus arriving to the answer of the desired problem). (These "solutions" are not to be confused with "answers" to the problem; think of them as possible characteristics that the system would employ in order to reach the answer.)

Step 3: Reproduce (and children mutate)

Those chromosomes with a higher fitness value are more likely to reproduce offspring (which can mutate after reproduction). The offspring is a product of the father and mother, whose composition consists of a combination of genes from the two (this process is known as "crossing over").

Step 4: Next generation

If the new generation contains a solution that produces an output that is close enough or equal to the desired then the problem has been solved. If this is not the case, then the new generation will go through the same process as their parents did. This will continue until a solution is reached.

2	Discuss about Simple genetic algorithm with neat sketch	[L2][CO5]	[12M]
	<p>Simple Genetic Algorithm (SGA) is one of the three types of strategies followed in Genetic algorithm.</p> <p>SGA starts with the creation of an initial population of size N. Then, we evaluate the goodness/fitness of each of the solutions/individuals. After that, the convergence criterion is checked, if it meets then we converge the algorithm otherwise go for the next steps –</p>		

1. Select Np individuals from the previous population.
2. Create the mating pool randomly.
3. Perform Crossover.
4. Perform Mutation in offspring solutions.
5. Perform inversion in offspring solutions.
6. Replace the old solutions of the last generation with the newly created solutions and go to step (2).

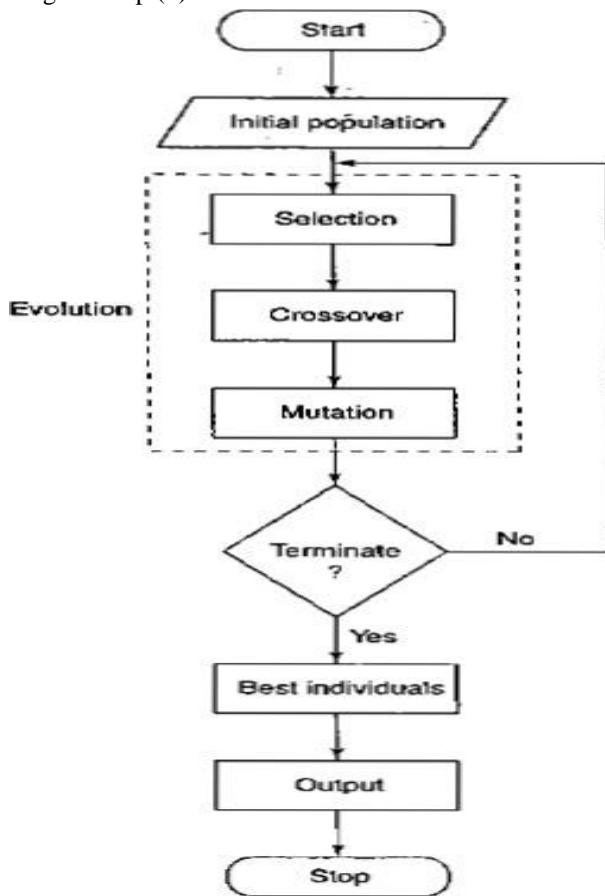


Figure: Flowchart for genetic algorithm

Important Parameters while solving problems in Simple Genetic Algorithm:

1. Initial Population N
An individual is a single solution while a population is a set of individuals at an instant of searching process.
2. Mating pool size Np
3. Convergence threshold.
4. Selection
Selection is the process of choosing two parents randomly from the population for crossing. The idea is to give preference to the individuals with good fitness scores and allow them to pass their genes to successive generations.
5. Crossover
Crossover is the process of taking two parent solutions and producing from them a child. After the selection (reproduction) process, the population is enriched with better individuals. Reproduction makes clones of good strings but does not create new ones. Crossover operator is applied to the mating pool with the hope that it creates a better offspring. Crossover is a recombination operator that proceeds in three steps:
 1. The reproduction operator selects at random a pair of two individual strings for the mating.
 2. A cross site is selected at random along the string length.
 3. Finally, the position values are swapped between the two strings following the cross site.
6. Mutation
Mutation to apply random changes to individual parents in order to form

	<p>children</p> <p>7. Inversion A string from the population is selected and the bits between two random sites are inverted</p> <p>Applications :</p> <ul style="list-style-type: none"> • Learning robot behavior using Simple Genetic Algorithm. • In the finance industry. • Neural networks. • Soft computing Applications like Fuzzy logic, Neuro computing. • Optimization problems. 		
3	<p>Explain the Various Operators in genetic algorithm?</p>	[L2][CO5]	[12M]
	<p>Once the initial generation is created, the algorithm evolves the generation using following operators –</p> <p>1) Selection Operator: Selection is the process of choosing two parents from the population for crossing. The idea is to give preference to the individuals with good fitness scores and allow them to pass their genes to successive generations</p> <p>Various Selection techniques</p> <ol style="list-style-type: none"> 1. Roulette wheel Selection 2. Random selection 3. Rank Selection 4. Tournament Selection 5. Steady State Selection 6. Boltzmann Selection <p>Give Explanation for each method</p> <p>2) Crossover Operator: This represents mating between individuals. Two individuals are selected using selection operator and crossover sites are chosen randomly. Then the genes at these crossover sites are exchanged thus creating a completely new individual (offspring).</p> <p>Various techniques of Crossover Operation</p> <ol style="list-style-type: none"> 1. Single Point Crossover 2. Two –Point Crossover 3. Multipoint Crossover 4. Uniform Crossover 5. Uniform crossover with mask 6. Linear Crossover <p>Give Explanation for each method</p> <p>For example –</p> <pre> graph LR P1[Parent 1: A B C D E F G H] --- C1(()) P2[Parent 2: F G H A D B E A] --- C2(()) C1 --- O[Offspring: F G H B C D E A] C2 --- O </pre> <p>3) Mutation Operator: The key idea is to insert random genes in offspring to</p>		

	<p>maintain the diversity in the population to avoid premature convergence. For example –</p> <p>Before Mutation</p>  <p>After Mutation</p>  <p>Types of Mutation operations</p> <ol style="list-style-type: none"> 1. Bitflip mutation 2. Swapping 3. Reversing 4. Scramble 5. Random Resetting 6. Inversion <p>Give the explanation for each method.</p> <p>Refer Q.No.4</p>												
4	<p>Summarize the following terms:</p> <p>i)Mutation operation ii)Selection operation.</p>	[L5][CO5]	[12M]										
	<p>a) <u>Mutation operation</u></p> <p>Mutation</p> <p>After crossover, the strings are subjected to mutation. Mutation plays the role of recovering the lost genetic materials as well as for randomly distributing generic information. It introduces new generic structures in the population by randomly modifying some of its building blocks.</p> <p>BIT Flipping</p> <p>Flipping of a bit involves changing 0 to 1 and 1 to 0 based on a mutation chromosome generated. A parent is considered and a mutation chromosome is randomly generated. For a 1 in mutation chromosome, the corresponding bit in parent chromosome is flipped (0 to 1 and 1 to 0) and child chromosome is produced.</p> <table border="1" data-bbox="436 1483 873 1640"> <tr> <td>Parent</td> <td>1 0 1 1 0 1 0 1</td> </tr> <tr> <td>Mutation chromosome</td> <td>1 0 0 0 1 0 0 1</td> </tr> <tr> <td>Child</td> <td>0 0 1 1 1 1 0 0</td> </tr> </table> <p>Figure : Mutation flipping</p> <p>Interchanging or Swapping</p> <p>Two random positions of the string are chosen and the bits corresponding to those positions are interchanged.</p> <table border="1" data-bbox="444 1797 889 1909"> <tr> <td>Parent</td> <td>1 0 1 1 0 1 0 1</td> </tr> <tr> <td>Child</td> <td>1 1 1 1 0 0 0 1</td> </tr> </table> <p>Figure: Interchanging</p> <p>Reversing</p> <p>A random position is chosen and the bits next to that position are reversed and child chromosome is produced.</p>	Parent	1 0 1 1 0 1 0 1	Mutation chromosome	1 0 0 0 1 0 0 1	Child	0 0 1 1 1 1 0 0	Parent	1 0 1 1 0 1 0 1	Child	1 1 1 1 0 0 0 1		
Parent	1 0 1 1 0 1 0 1												
Mutation chromosome	1 0 0 0 1 0 0 1												
Child	0 0 1 1 1 1 0 0												
Parent	1 0 1 1 0 1 0 1												
Child	1 1 1 1 0 0 0 1												

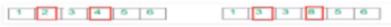
Parent	1 0 1 1 0 1 0 1
Child	1 0 1 1 0 1 1 0

Figure : Reversing

Random Resetting

- Random Resetting is an extension of the bit flip for the integer representation.
- In this, a random value from the set of permissible values is assigned to a randomly chosen gene.

Value ranges from [1, 8]



Scramble Mutation

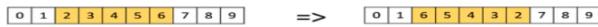
- Scramble mutation is also popular with permutation representations.
- In this, from the entire chromosome, a subset of genes is chosen and their values are scrambled or shuffled randomly.



=> 0 1 3 6 4 2 5 7 8 9

Inversion Mutation

Select a subset of genes like in scramble mutation, but instead of shuffling the subset, we merely invert the entire string in the subset.



=> 0 1 6 5 4 3 2 7 8 9

b) Selection operation

Selection is the process of choosing two parents randomly from the population for crossing. The purpose of selection is to emphasize fitter individuals in the population in hopes that their offspring have higher fitness. According to Darwin's theory of evolution the best ones survive to create new offspring.

Two types of selection:

- **Proportionate-based selection**

Proportionate-based selection picks out individuals based upon their fitness values relative to the fitness of the other individuals in the population.

- **Ordinal-based selection**

Ordinal-based selection schemes select individuals not upon their raw fitness, but upon their rank within the population. This requires that the selection pressure is independent of the fitness distribution of the population, and is solely based **upon the relative ordering (ranking) of the population**.

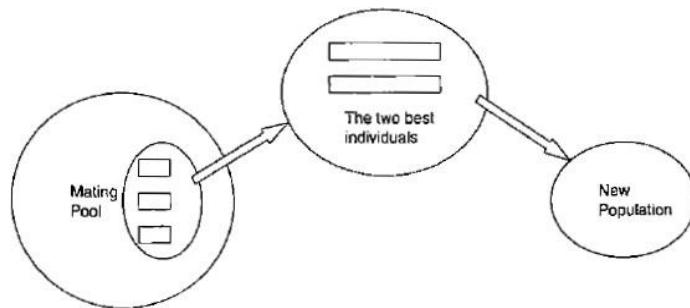


Figure: Selection

Roulette Wheel Selection

The Roulette process can also be explained as follows:

The expected value of an individual is individual's fitness divided by the actual fitness of the population. In a roulette wheel selection, the circular wheel is divided as described before. A fixed point is chosen on the wheel circumference as shown and the wheel is rotated. The region of the wheel which comes in front of the fixed point is chosen as the parent. For the second parent, the same process is repeated.

Each individual is assigned a slice of the Roulette wheel, the size of the slice being proportional to the individual's fitness. A target value is set, which is a random proportion of the sum of the fitness's in the population. A fit individual will contribute more to the target value, but if it does not exceed it, the next chromosome in line has a chance, and it may be weak. The wheel is spun N times, where N is the number of individuals in the population. On each spin, the individual under the wheel's marker is selected to be in the pool of parents for the next generation.

Roulette-Wheel selection



- The most common **fitness-proportionate selection** technique is called **Roulette Wheel Selection**.
- Probability for an individual being selected is considered to be proportional to its fitness.

The probability that i-th individual will be pointed is

$$P_i = \frac{f_i}{\sum_{i=1}^N f_i}$$

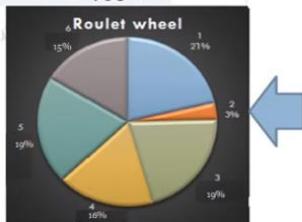
- f_i is the evaluation associated with the individual i in the population.
- N – Size of population
- These probabilities are represented on a wheel

Roulette-Wheel selection

Parents are selected according to their fitness values

The better chromosomes have more chances to be selected

Chromosome	Fitness	Probability	Chance (N*Pi)	Actual c
1	50	50/186 = .2688	.2688*6 = 1.61	2
2	6	6/186 = 0.0322	.0322*6=0.19	0
3	36	36/186 = 0.1935	0.1935*6 =1.16	1
4	30	30/186 = 0.1612	0.1612*6 =0.97	1
5	36	36/186 = 0.1935	0.1935*6 =1.16	1
6	28	28/186 = 0.1505	0.1505*6= 0.90	1
		186		6



Random Selection

This technique randomly selects a parent from the population. In terms of disruption of generic codes, random selection is a little more disruptive, on average, than Roulette wheel selection.

Rank Selection

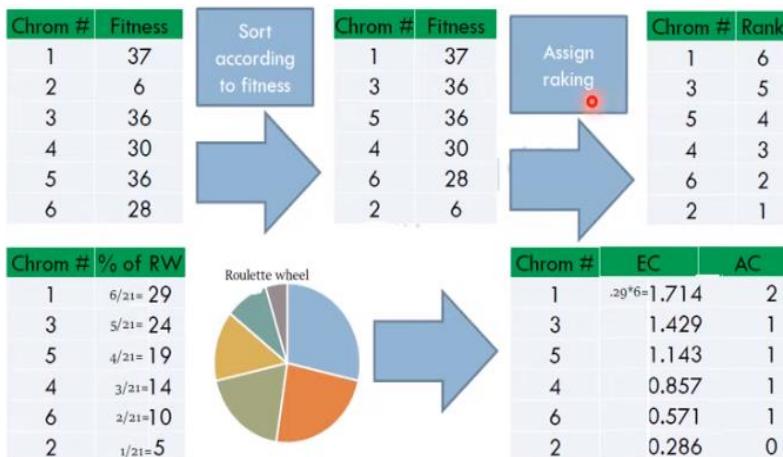
Rank selection

- Individuals are arranged in an ascending order of their fitness values.
- The individual, which has the lowest value of fitness is assigned rank 1, and other individuals are ranked accordingly.
- The individual, which has the best value of fitness is assigned rank N
- Apply the Roulette-Wheel selection based on their assigned ranks.

The % area to be occupied by a particular individual i , is given by

$$\frac{r_i}{\sum_{i=1}^N r_i} \times 100$$

where r_i indicates the rank of $i - th$ individual.



Tournament Selection

In K-Way tournament selection, we select K individuals from the population at random and select the best out of these to become a parent. The same process is repeated for selecting the next parent..

Tournament selection

- In tournament selection several tournaments are played among a few individuals.
- Select the tournament size n (say 2 or 3) at random.
- Pick n individuals from the population, at random and determine the best one in terms of their fitness values.
- The best individual is selected

Tournament selection

$$N = 8, N_U = 2, N_P = 8$$

Input :

Individual	1	2	3	4	5	6	7	8
Fitness	1.0	2.1	3.1	4.0	4.6	1.9	1.8	4.5

Output :

Trial	Individuals	Selected
1	2, 4	4
2	3, 8	8
3	1, 3	3
4	4, 5	5
5	1, 6	6
6	1, 2	2
7	4, 2	4
8	8, 3	8

Boltzmann Selection

In Boltzmann selection, a continuously varying temperature controls the rate of selection according to a preset schedule. The temperature starts high, which means that the selection pressure is low. The temperature is gradually lowered, which gradually increases the selection pressure, thereby allowing the GA to narrow in more closely to the best part of the search space while maintaining the appropriate degree of diversity.

Steady-State selection

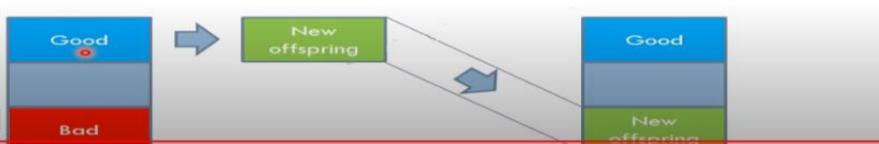
- This is not a particular method of selecting parents.
- Select good parents from population
- Create children, evaluate
- If new offspring is better than bad individual in the population, replace bad individual with this new offspring
- The rest of population survives to new generation.

Steady-State selection

In this method, a few good chromosomes are used for creating new offspring in every iteration.

Then some bad chromosomes are removed and the new offspring is placed in their places

The rest of population migrates to the next generation without going through the selection process.



5	a	Analyze Inversion and Deletion Operators in GA.	[L4][CO5]	[6M]
		Inversion A string from the population is selected and the bits between two random sites are inverted. Linear+end-inversion Linear+end-inversion performs linear inversion with a specified		

probability of 0.75. If linear inversion was not performed, the end inversion would be performed with equal probability of 0.125 at either the left or right end of the string. Under end inversion, the left or right end of the string was picked as one inversion-point and a second inversion-point was picked uniformly at random from the point no farther away than one half of the string length.

Continuous inversion

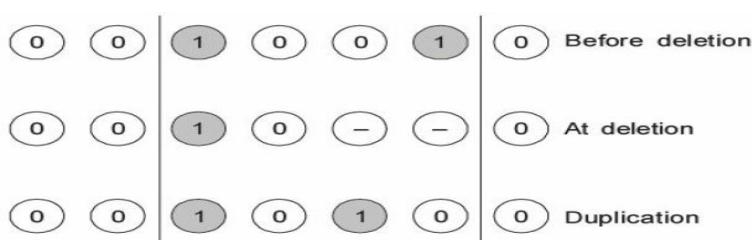
In continuous inversion, inversion was applied with specified inversion probability P_r to each new individual when it is created.

Mass inversion

No inversion takes place until a new population is created and thereafter, one-half of the population undergoes identical inversion (using the same two inverting points).

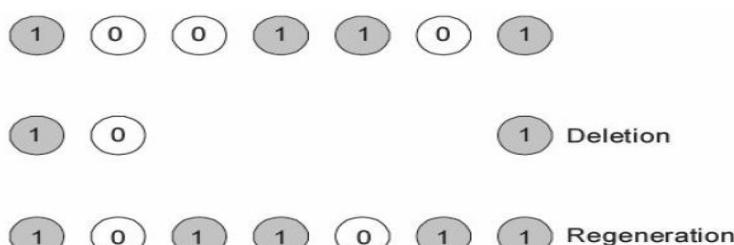
Deletion and Duplication

Any two or three bits at random in order are selected and the previous bits are duplicated



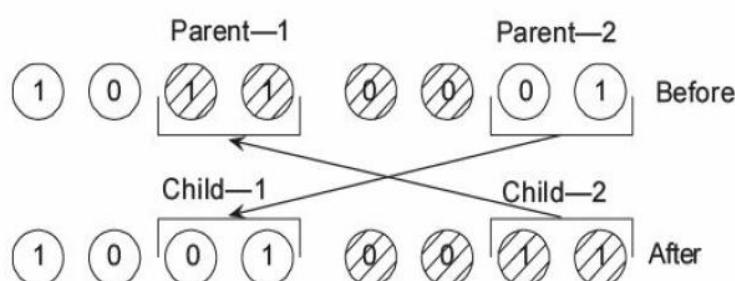
Deletion and Regeneration

Genes between two cross-sites are deleted and regenerated randomly.



Segregation

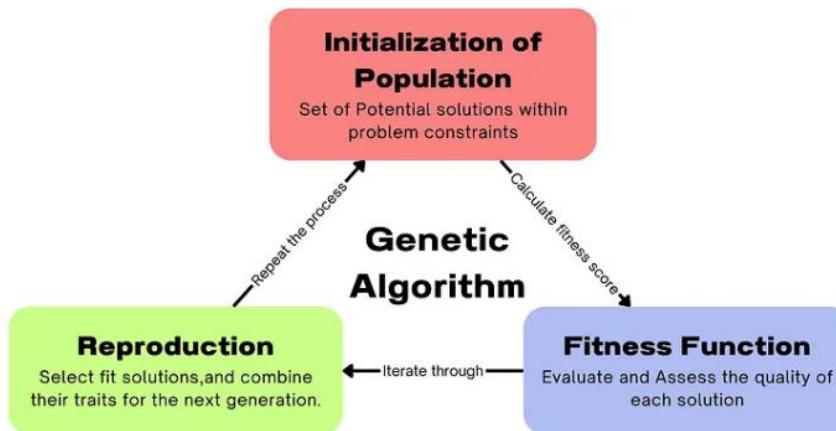
The bits of the parents are segregated and then crossed over to produce Offspring



	b	Describe the applications of genetic algorithm.	[L1][CO5]	[6M]
		<p>The Various applications of Genetic Algorithm are:</p> <p>Optimization – Genetic Algorithms are most commonly used in optimization problems wherein we have to maximize or minimize a given objective function value under a given set of constraints. The approach to solve Optimization problems has been highlighted throughout the tutorial.</p> <p>Economics – GAs are also used to characterize various economic models like the cobweb model, game theory equilibrium resolution, asset pricing, etc.</p> <p>Neural Networks – GAs are also used to train neural networks, particularly recurrent neural networks.</p> <p>Parallelization – GAs also have very good parallel capabilities, and prove to be very effective means in solving certain problems, and also provide a good area for research.</p> <p>Image Processing – GAs are used for various digital image processing (DIP) tasks as well like dense pixel matching.</p> <p>Vehicle routing problems – With multiple soft time windows, multiple depots and a heterogeneous fleet.</p> <p>Scheduling applications – GAs are used to solve various scheduling problems as well, particularly the time tabling problem.</p> <p>Machine Learning – as already discussed, genetics based machine learning (GBML) is a niche area in machine learning.</p> <p>Robot Trajectory Generation – GAs have been used to plan the path which a robot arm takes by moving from one point to another.</p> <p>Parametric Design of Aircraft – GAs have been used to design aircrafts by varying the parameters and evolving better solutions.</p> <p>DNA Analysis – GAs have been used to determine the structure of DNA using spectrometric data about the sample.</p> <p>Multimodal Optimization – GAs are obviously very good approaches for multimodal optimization in which we have to find multiple optimum solutions.</p> <p>Traveling salesman problem and its applications – GAs have been used to solve the TSP, which is a well-known combinatorial problem using novel crossover and packing strategies.</p>		
6	a	How Fitness function can be evaluated in Genetic Algorithm?	[L1][CO5]	[4M]
		<p>Fitness function</p> <p>The fitness function simply defined is a function which takes a candidate solution to the problem as input and produces as output how “fit” our how “good” the</p>		

solution is with respect to the problem in consideration.

Calculation of fitness value is done repeatedly in a GA and therefore it should be sufficiently fast. A slow computation of the fitness value can adversely affect a GA and make it exceptionally slow.



A fitness function should possess the following characteristics –

- The fitness function should be sufficiently fast to compute.
- It must quantitatively measure how fit a given solution is or how fit individuals can be produced from the given solution.

The following image shows the fitness calculation for a solution of the 0/1 Knapsack. It is a simple fitness function which just sums the profit values of the items being picked (which have a 1), scanning the elements from left to right till the knapsack is full.

0 1 2 3 4 5 6	Item Number
0 1 0 1 1 0 1	Chromosome
2 9 8 5 4 0 2	Profit Values
7 5 3 1 5 9 8	Weight Values
Knapsack capacity = 15 Total associated profit = 18 Last item not picked as it exceeds knapsack capacity	

b) Describe various Encoding Techniques of Genetic algorithm.

[L2][CO5]

[8M]

Encoding

Encoding is a process of representing individual genes. The process can be performed using bits, numbers, trees, arrays, lists or any other object.

Binary Encoding

Each chromosome encodes a binary (bit) string. Each bit in the string can represent some characteristics of the solution. Every bit string therefore is a solution but not necessarily the best solution. Another possibility is that the whole string can represent a number. The way bit strings can code differs from problem to problem.

Binary encoding gives many possible chromosomes with a smaller number of alleles. Binary coded strings with 1s and 0s are mostly used. The length of the string depends on the accuracy.

Chromosome 1	1 1 0 1 0 0 0 1 1 0 1 0
Chromosome 2	0 1 1 1 1 1 1 1 1 0 0

Figure: Binary encoding

Octal Encoding

This encoding uses string made up of octal numbers (0-7).

Chromosome 1	03467216
Chromosome 2	15723314

Figure: Octal encoding**Hexadecimal Encoding**

This encoding uses string made up of hexadecimal numbers (0-9, A-F).

Chromosome 1	9CE7
Chromosome 2	3DBA

Figure: Hexadecimal encoding**Permutation Encoding (Real Number Coding)**

Every chromosome is a string of numbers, represented in a sequence. In permutation encoding, every chromosome is a suing of integer/real values, which represents number in a sequence.

Chromosome A	1 5 3 2 6 4 7 9 8
Chromosome B	8 5 6 7 2 3 1 4 9

Figure : Permutation encoding**Value Encoding**

Every chromosome is a string of values and the values can be anything connected to the problem. In value encoding, every chromosome is a string of some values. Values can be anything connected to problem, form numbers, real numbers or characters to some complicated objects.

Chromosome A	1.2324 5.3243 0.4556 2.3293 2.4545
Chromosome B	ABDJEIFJDHDIERJFDLDFLFEGT
Chromosome C	(back), (back), (right), (forward), (left)

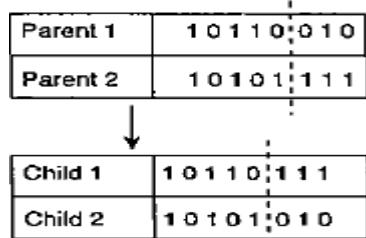
Figure : Value encoding**Tree Encoding**

This encoding is mainly used for evolving program expressions for genetic programming. Every chromosome is a tree of some objects such as functions and commands of a programming language

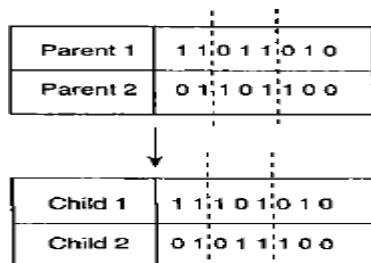
7	a	List out the various Bitwise Operators are in GA.	[L3][CO5]	[4M]
		<ol style="list-style-type: none"> 1. The one's complement operator, 2. The logical bit-wise operator, and 3. The shift operator. <p>Logical Bit-wise Operators</p> <p>There are three logical bit-wise operators, namely,</p> <ol style="list-style-type: none"> 1. Bit-wise AND, 2. Bitwise Exclusive-OR, and 3. Bit-wise OR. <p>Shift Operators</p> <ol style="list-style-type: none"> 1.Shift left operator (<<) 2. Shift right operator (>>) <p>Masking</p>		

	b	Illustrate the bitwise operators in GA with suitable examples.	[L3][CO5]	[8M]
		<p>Usually, binary coding is used more extensively in the coding mechanism to generate algorithm structure. In the present work, genetic algorithm program as written in “C” language with the help of built in operators in “C” (the bit-wise operators), we can directly manipulate the individual bits within a word of memory. These operators can be carried out easily and efficiently. These can operate on integers and characters but not on floats and doubles. Byron S. Gotfried (1990) categorizes bit-wise operators into three, namely,</p> <ol style="list-style-type: none"> 1. The one's complement operator, 2. The logical bit-wise operator, and 3. The shift operator. <p>One's Complement Operator</p> <p>The one's complement operator (\sim) is an unary operator that causes the bits of its operand to be inverted (i.e. reversed), so that 1 becomes zero and zero becomes 1. This operator always precedes its operand.</p> <p>Logical Bit-wise Operators</p> <p>There are three logical bit-wise operators, namely,</p> <ol style="list-style-type: none"> 1. Bit-wise AND, 2. Bitwise Exclusive-OR, and 3. Bit-wise OR. <p>Bit-wise AND (&) operator</p> <p>A bit-wise AND (&) expression returns 1 if both the bits have a value 1, otherwise it returns a value 0.</p> <p>Parent 1a = 1010 1010 → 10 10 Parent 2b = 1100 0011 → 12 3 Child a&b = 1000 0010 → 8 2</p> <p>Bit-wise exclusive-OR (\wedge) operator</p> <p>A bit-wise exclusive-OR (\wedge) expression returns a 1 if one of the bits have a value of 1 and the other has a value of 0 otherwise it returns a value 0.</p> <p>Parent 1a = 1010 1010 → 10 10 Parent 2b = 1100 0011 → 12 3 Child a&b = 0110 1001 → 6 9</p> <p>Bit-wise OR () operator</p> <p>A bit-wise OR () expression returns a 1 if one or more bits have a value of 1 otherwise it returns a value 0.</p> <p>Parent 1a = 1010 1010 → 10 10 Parent 2b = 1100 0011 → 12 3 Child a&b = 1110 1011 → 13 11</p> <p>Shift Operators</p> <p>Two bit-wise shift operators are, shift left ($<<$) and shift right ($>>$) operators. Each operator operates on a single variable but requires two operands. The first operand is an integer type operand that represents the bit pattern to be shifted and the second is an unsigned integer that indicates the number of displacements.</p> <p>Shift left operator ($<<$)</p>		

	<p>The shift left operator causes all the bits in the first operand to be shifted to the left by the number of positions indicated by the second operand. The leftmost bits (i.e. the overflow bits) in the original bit pattern is lost. The Right most bit positions that become vacant are to be filled with zeroes.</p> <p>a = 1010 0110 → 10 6</p> <p>a << 2 = 1001 1000 → 9 8</p> <p>Shift right operator (>>)</p> <p>The shift right operator causes all the bits in the first operand to be shifted to the right by the number of positions indicated by the second operand. The right most bits (i.e. the underflow bits) in the original bit pattern are lost. The left most bit positions that become vacant are then filled with zeroes.</p> <p>a = 1010 0110 → 10 6</p> <p>a >> 2 = 0010 1001 → 2 9</p> <p>Masking</p> <p>Masking is a process in which a given bit pattern is transformed into another bit pattern by means of logical bit-wise operation. The original bit pattern is one of the operands in the bit-wise operation. The second operand called mask, is a specially selected bit pattern that brings about the desired transformation. There are several different kinds of masking operations. For example, a portion of a given bit pattern can be copied to a new word, while the remainder of the new word is filled with 0. Thus, part of the original bit pattern will be “masked off” from the final result.</p> <p>BIT-WISE OPERATORS USED IN GA</p> <p>Logical bit-wise operators are used in different combinations. Each operator operates on two individuals and generates one resultant so as to keep the number of individuals in the population constant. Two different operators are used in GA process.</p> <p>Populations are selected randomly for mating and on each pair bit-wise AND and bit-wise OR operators are performed. Similarly, AND and exclusive-OR or OR and exclusive-OR operations can be performed to produce children or population for the next generation.</p>		
8	Analyze the various cross over operations performed in GA	[L2][CO5]	[12M]
	<p>Crossover (Recombination)</p> <p>Crossover is the process of taking two parent solutions and producing from them a child. After the selection (reproduction) process, the population is enriched with better individuals. Reproduction makes clones of good strings but does not create new ones. Crossover operator is applied to the mating pool with the hope that it creates a better offspring. Crossover is a recombination operator that proceeds in three steps:</p> <ol style="list-style-type: none"> 1. The reproduction operator selects at random a pair of two individual strings for the mating. 2. A cross site is selected are random along the string length. 3. Finally, the position values are swapped between the two strings following the cross site. <p>Single-Point Crossover</p> <p>The two mating chromosomes are cut once at corresponding points and the sections after the cuts exchanged. Here, a cross site or crossover point is selected randomly along the length of the mated strings and bits next to the cross sites are exchanged. If appropriate site is chosen, better children can be obtained by combining good parents, else it severely hampers string quality.</p>		

**Figure: Single – point crossover****Two-Point Crossover**

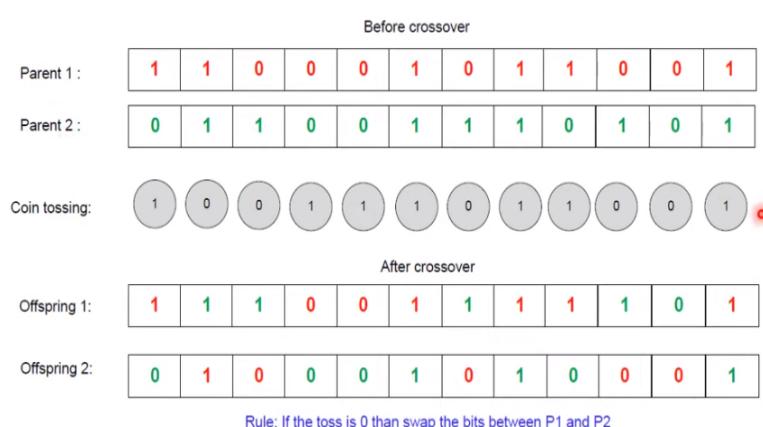
In two-point crossover, two crossover points are chosen and the contents between these points are exchanged between two mated parents.

**Figure: Two – point crossover**

Originally, GAs were using one point crossover which cuts two chromosomes in one point and splices the two halves to create new ones. But with this one-point crossover, the head and the tail of one chromosome cannot be passed together to the offspring. If both the head and the tail of a chromosome contain good genetic information, none of the offspring obtained directly with one-point crossover will share the two good features. Using a two- point crossover one can avoid this drawback, and so it is generally considered better than one-point crossover.

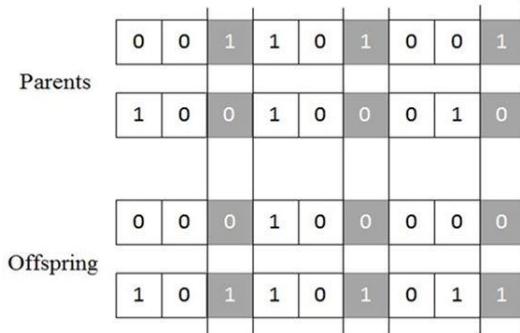
Uniform Crossover

Each gene in the offspring is created by copying the corresponding gene from one or the other parent chosen according to a random generated binary crossover mask of the same length as the chromosomes. Where there is a 1 in the crossover mask, the gene is copied from the first parent, and where there is a 0 in the mask the gene is copied from the second parent. A new crossover mask is randomly generated for each pair of parents. In the below figure, while producing child 1, when there is a 1 in the mask, the gene is copied from parent 1 else it is copied from parent 2. On producing child 2, when there is a 1 in the mask, the gene is copied from parent 2, and when there is a 0 in the mask, the gene is copied from the parent 1.

Uniform Crossover

Multipoint Crossover (N-Point Crossover)

There are two ways in this crossover. One is even number of cross sites and the other odd number of cross sites. In the case of even number of cross sites, the cross sites are selected randomly around a circle and information is exchanged. In the case of odd number of cross sites, a different cross point is always assumed at the string beginning.

**Three-Parent Crossover**

In this crossover technique, three parents are randomly chosen. Each bit of the first parent is compared with the bit of the second parent. If both are the same, the bit is taken for the offspring, otherwise the bit from the third parent is taken for the offspring.

Parent 1	1 1 0 1 0 0 0 1
Parent 2	0 1 1 0 1 0 0 1
Parent 3	0 1 1 0 1 1 0 0
Child	0 1 1 0 1 0 0 1

Figure : Three - parent crossover

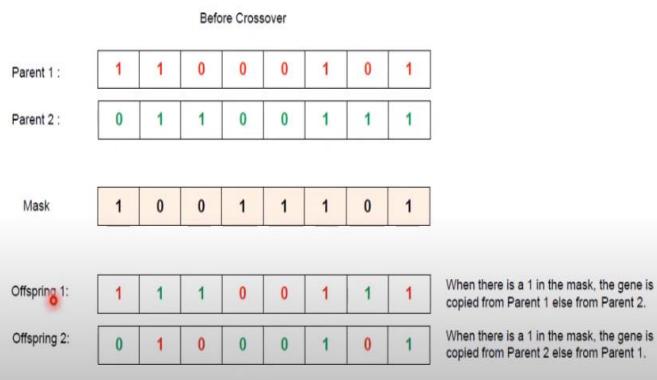
Uniform crossover with crossover mask

- Each gene is created in the offspring by copying the corresponding gene from one or the other parent chosen according to a random generated binary crossover mask of the same length as the chromosome.
- Where there is a 1 in the mask, the gene is copied from the first parent
- Where there is a 0 in the mask, the gene is copied from the second parent.

Uniform crossover with crossover mask

- Each gene is created in the offspring by copying the corresponding gene from one or the other parent chosen according to a random generated binary crossover mask of the same length as the chromosome.
- Where there is a 1 in the mask, the gene is copied from the first parent
- Where there is a 0 in the mask, the gene is copied from the second parent.

Uniform crossover with crossover mask



Shuffle Crossover

Shuffle crossover is related to uniform crossover. A single crossover position (as in crossover) is selected. But before the variables are exchanged, they are randomly shuffled in both parents. After recombination, the variables in the offspring are unshuffled. This removes positional bias as the variables are randomly reassigned each time crossover is performed.

Linear crossover in Real-coded GAs

This scheme uses some linear functions of the parent chromosomes to produce the new children.

•

$$\text{Child1} = \alpha.x + (1-\alpha).y$$

$$\text{Child2} = \alpha.x + (1-\alpha).y$$

$$\alpha = 0.5$$

x	0.1	0.1	0.2	0.2	0.3	0.3	0.4	0.4	0.5	0.5	0.15	0.2	0.2	0.2	0.3	0.25	0.35	0.3	0.2	0.35
---	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	------	-----	-----	-----	-----	------	------	-----	-----	------

$$\Rightarrow$$

y	0.2	0.3	0.2	0.2	0.3	0.2	0.3	0.2	0.3	0.2	0.15	0.2	0.2	0.2	0.3	0.25	0.35	0.3	0.2	0.35
---	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	------	-----	-----	-----	-----	------	------	-----	-----	------

Crossover Probability

Crossover probability is a parameter to describe how often crossover will be performed. If there is no crossover, offspring are exact copies of parents. If there is crossover, offspring are made from parts of both parents chromosome. If crossover probability is 100%, then all offspring are made by crossover. If it is 0%, whole new generation is made from exact copies of chromosomes from old population.

9	<p>a List out the different reproduction and inheritance operators used in GA.</p> <p>REPRODUCTION OPERATORS: also called Selection techniques.</p> <p>Various Selection techniques</p> <ol style="list-style-type: none"> 1. Roulette wheel Selection 2. Random selection 3. Rank Selection 4. Tournament Selection 5. Steady State Selection 6. Boltzmann Selection 	[L2][CO5]	[6M]
---	--	-----------	------

	<p>INHERITANCE OPERATORS</p> <p>As already seen, genetic algorithm makes use of the Darwinian survival of the fittest procedure. Genetic algorithms are search procedures based on mechanics of natural genetics and natural selection.</p> <p>Genetic algorithm derives power from the genetic operators listed here.</p> <p>Low-level operators, namely</p> <ol style="list-style-type: none"> 1. Inversion 2. Dominance 3. Deletion 4. Intrachromosomal duplication 5. Translocation 6. Segregation 7. Speciation 8. Migration 9. Sharing 10. Mating <p>A simple genetic algorithm largely uses three basic operators which are</p> <ol style="list-style-type: none"> 1. Reproduction 2. Cross over 3. Mutation 		
	<p>b Identify the Advantages and Disadvantages of Genetic Algorithm.</p> <p>Advantages of Genetic Algorithm</p> <ul style="list-style-type: none"> • It can find solutions to problems that are difficult or impossible to solve using traditional methods. • The parallel capabilities of genetic algorithms are best. • It helps in optimizing various problems such as discrete functions, multi-objective problems, and continuous functions. • It provides a solution for a problem that improves over time. • A genetic algorithm does not need derivative information. <p>Limitations of Genetic Algorithms</p> <ul style="list-style-type: none"> • Genetic algorithms are not efficient algorithms for solving simple problems. • It does not guarantee the quality of the final solution to a problem. • Repetitive calculation of fitness values may generate some computational challenges. • It is difficult to debug and can be computationally expensive. • It is sensitive to the initial conditions and can sometimes converge to local optima. 	[L2][CO5]	[6M]
10	<p>a Draw the flow chart for the working principle of Genetic Algorithm. Explain it</p> <p>For Answer: Refer Que no:2</p>	[L2][CO5]	[6M]
	<p>b Briefly explain Convergence of Genetic Algorithm.</p>	[L2][CO5]	[6M]
	<p>The termination condition of a Genetic Algorithm is important in determining when a GA run will end. It has been observed that initially, the GA progresses very fast with better solutions coming in every few iterations, but this tends to saturate in the later stages where the improvements are very small. We usually want a termination condition such that our solution is close to the optimal, at the end of the run. In a standard genetic algorithm, binary strings of 1s and 0s represent the chromosomes. Each chromosome is assigned a fitness value expressing its</p>		

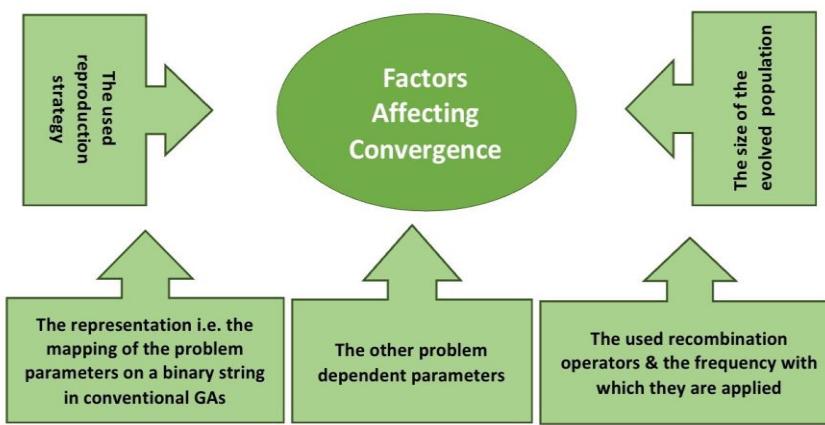
quality reflecting the given objective function. Such a population is evolved by means of reproduction and recombination operators in order to breed the optimal solution's chromosome. The evolution keeps running until some termination condition is fulfilled. The best chromosome encountered so far is then considered as the found solution.

- Usually, we keep one of the following termination conditions –
- When there has been no improvement in the population for X iterations.
- When we reach an absolute number of generations.
- When the objective function value has reached a certain pre-defined value.
- For example, in a genetic algorithm we keep a counter which keeps track of the generations for which there has been no improvement in the population.
- Initially, we set this counter to zero. Each time we don't generate off-springs which are better than the individuals in the population, we increment the counter.
- However, if the fitness any of the off-springs is better, then we reset the counter to zero. The algorithm terminates when the counter reaches a predetermined value.
- Like other parameters of a GA, the termination condition is also highly problem specific and the GA designer should try out various options to see what suits his particular problem the best.

Termination or Convergence Criteria

- Each iteration should be tested with some convergence test.
 - Commonly known convergence test or termination conditions are :
1. A solution is found that satisfies the objective criteria.
 2. Fixed number of generation is executed.
 3. The highest ranking solution fitness is reaching or has reached a solution such that successive iterations no longer produce better result.
 4. Allocated budget (such as computation time) has reached.
 5. Combination of the above.

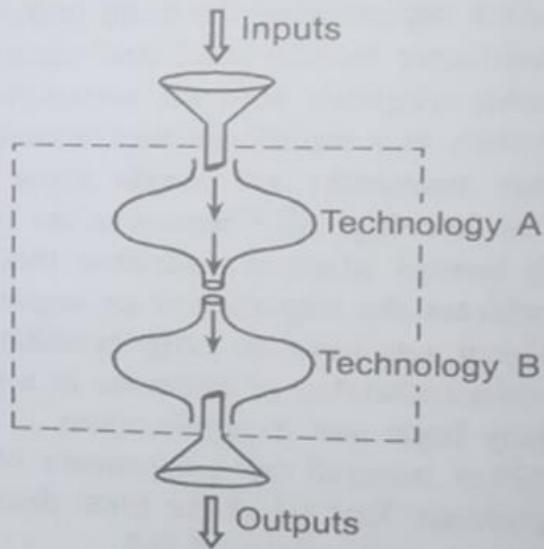
Factors affecting Convergence:



UNIT -V

HYBRID SYSTEMS

		HYBRID SYSTEMS		
1	a	List out the various types of hybrid systems.	[L1][CO6]	[4M]
		<p>There are 3 different types of Hybrid Systems</p> <p>1. Sequential Hybrids,</p> <p>2. Auxiliary Hybrids</p> <p>3. Embedded Hybrids.</p> <p>By Combining the Soft computing constituents, Types of Hybrid Systems:</p> <ul style="list-style-type: none"> • Neuro-Fuzzy Hybrid systems • Neuro Genetic Hybrid systems • Fuzzy Genetic Hybrid systems 		
	b	Explain the basic categories of Hybrid system with neat architectures.	[L3][CO6]	[8M]
		<p>Hybrid systems:</p> <p>Hybrid systems have been classified as (Refer Gray and Kilgour, 1997) A Hybrid system is an intelligent system that is framed by combining at least two intelligent technologies like Fuzzy Logic, Neural networks, Genetic algorithms, reinforcement learning, etc. The combination of different techniques in one computational model makes these systems possess an extended range of capabilities i.e. Hybrid systems are those for which more than one technology is employed to solve the problem These systems are capable of reasoning and learning in an uncertain and imprecise environment. These systems can provide human-like expertise like domain knowledge, adaptation in noisy environments, etc. There are 3 different types of Hybrid Systems</p> <p>1. Sequential Hybrids,</p> <p>2. Auxiliary Hybrids</p> <p>3. Embedded Hybrids.</p> <p>Sequential Hybrid Systems</p> <p>As the name indicates, sequential hybrid systems make use of technologies in a pipeline-like fashion. Thus, one technology's output becomes another's input and so on. Fig(1) illustrates the schema for a sequential hybrid.</p>		



Fig(1) Sequential hybrid systems

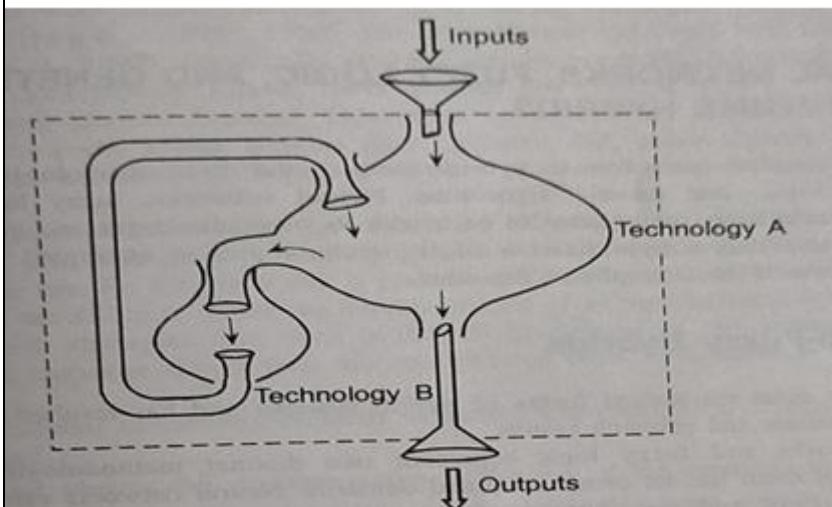
This is one of the weakest forms of hybridization since an integrated combination of the technologies is not present.

An example is a GA preprocessor which obtains the optimal parameters for different instances of a problem and hands over the 'preprocessed' data set to an NN for further processing.

Auxiliary Hybrid Systems

In this, one technology calls the other as a "subroutine" to process or manipulate information needed by it. The second technology processes the information provided by the first and hands over for further use. Figure 2 illustrates an auxiliary hybrid system. This type of hybridization though better than sequential hybrids, is considered to be of intermediary level only.

An example is a neuro-genetic system in which an NN employs a GA to optimize its structural parameters, i.e. parameters which defines its architecture.

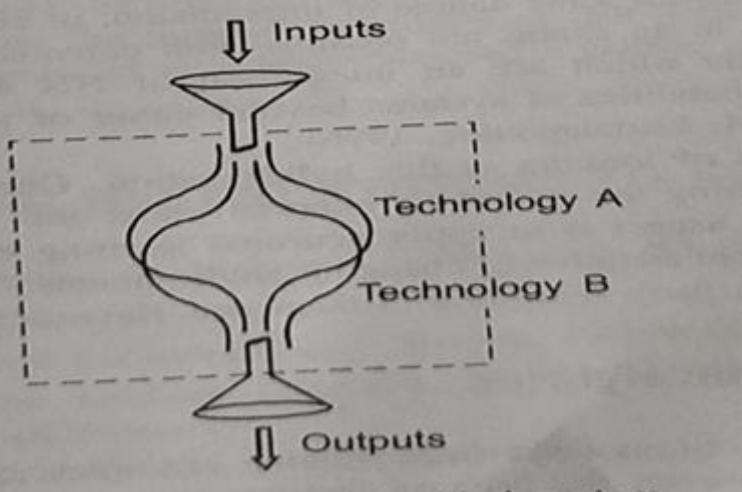


Fig(2) Auxillary Hybrid Systems

Embedded Hybrid Systems

In Embedded hybrid systems, the technologies participating are integrated in such a manner that they appear intertwined. The fusion is so complete that it would appear that no technology can be used without the others for solving the

problem. Figure 3 illustrates the schema for an embedded hybrid system. Here, the hybridization is absolute.



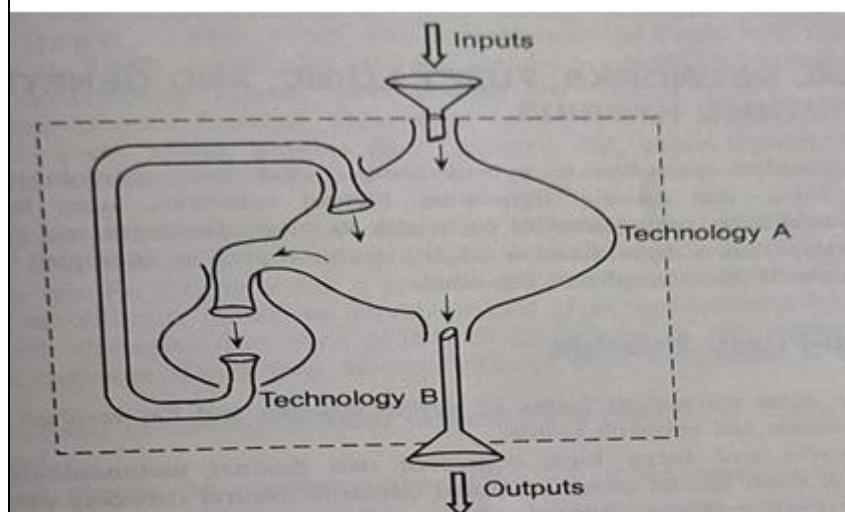
Fig(3) Embedded Hybrid Systems

2	a	List out the applications of hybrid system.	[L4][CO6]	[4M]
		<ul style="list-style-type: none"> • Mechanical Engineering • Electrical Engine • Artificial Intelligence • Economics • Automative • Manufacturing • Smart Grids 		
	b	Analyze the Auxiliary hybrid system with neat architecture.	[L2][CO6]	[8M]

Auxiliary Hybrid Systems

In this, one technology calls the other as a "subroutine" to process or manipulate information needed by it. The second technology processes the information provided by the first and hands over for further use. Figure 2 illustrates an auxiliary hybrid system. This type of hybridization though better than sequential hybrids, is considered to be of intermediary level only.

An example is a neuro-genetic system in which an NN employs a GA to optimize its structural parameters, i.e. parameters which defines its architecture.

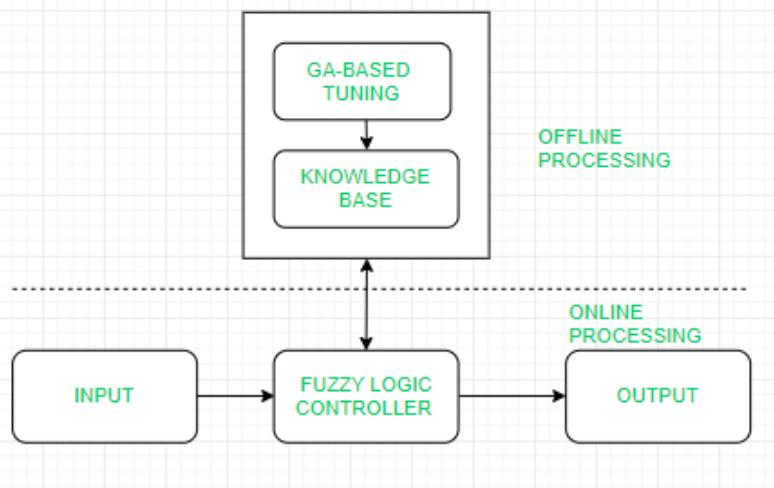


Fig(2) Auxillary Hybrid Systems

		Advantages and disadvantages of auxillary hybrid systems In the context of soft computing, auxiliary hybrid systems can refer to the integration of soft computing techniques with traditional or conventional computing methods to improve overall system performance. Here are some advantages and disadvantages of using auxiliary hybrid systems in soft computing: Advantages: 1. Enhanced Performance: Soft computing techniques, such as fuzzy logic, neural networks, and genetic algorithms, can complement traditional computing methods to enhance overall system performance. The combination of these approaches can provide more accurate and robust solutions. 2. Adaptability: Soft computing is known for its adaptability and ability to handle uncertainty and imprecision. Integrating soft computing techniques into auxiliary systems allows for better handling of real-world problems where precise modeling may be challenging. 3. Optimization: The hybridization of soft computing with traditional methods can lead to optimization of various processes. Genetic algorithms, for example, can be used to optimize parameters in a system, leading to improved efficiency. 4. Learning and Adaptation: Soft computing techniques often involve learning and adaptation mechanisms, which can be valuable in scenarios where the system needs to evolve or adjust to changing conditions. Neural networks, for instance, can learn from data and improve their performance over time. 5. Fault Tolerance: Soft computing techniques are often resilient to noisy or incomplete data. By integrating them into auxiliary systems, it is possible to improve fault tolerance and robustness, making the system more reliable in real-world applications. Disadvantages: 1. Complexity: The integration of soft computing with traditional computing can introduce additional complexity into the system. Understanding, implementing, and maintaining hybrid systems may require specialized knowledge and expertise. 2. Training and Tuning: Soft computing techniques often require training and tuning of parameters. Designing effective hybrid systems may involve the challenge of finding the right balance and configuration for optimal performance. 3. Computational Overhead: Some soft computing techniques, particularly neural networks and genetic algorithms, can be computationally intensive. This may lead to increased computational overhead and longer processing times in hybrid systems. 4. Interpretability: Soft computing techniques are sometimes criticized for their lack of interpretability. The integration of these techniques into auxiliary systems may make it challenging to explain the decision-making processes, especially in applications where transparency is crucial. 5. Data Dependency: Soft computing techniques often rely on large amounts of data for training and learning. In situations where sufficient data is not available, or the quality of the data is poor, the effectiveness of the soft computing component in the hybrid system may be compromised.	
3	a	Discuss in detail about Fuzzy-Genetic hybrid system.	[L4][CO6] [8M]

Fuzzy Genetic Hybrid systems:

A Fuzzy Genetic Hybrid System is developed to use fuzzy logic-based techniques for improving and modeling Genetic algorithms and vice-versa.



Three approaches that can be used to develop such a system are:

- Michigan Approach
- Pittsburgh Approach
- IRL Approach

Working Flow:

- Start with an initial population of solutions that represent the first generation.
- Feed each chromosome from the population into the Fuzzy logic controller and compute performance index.
- Create a new generation using evolution operators till some condition is met

Note: Explain the diagram in Proper manner.

b

Identify the advantages and disadvantages of Fuzzy-Genetic hybrid systems.

[L1][CO6]

[4M]

Advantages:

- GAs are used to develop the best set of rules to be used by a fuzzy inference engine
- GAs are used to optimize the choice of membership functions.
- A Fuzzy GA is a directed random search over all discrete fuzzy subsets.
- It can mimic the human decision-making process.

Disadvantages:

- Interpretation of results is difficult.
- Difficult to build membership values and rules.
- Takes lots of time to converge.

Applications:

- Mechanical Engineering
- Electrical Engine
- Artificial Intelligence
- Economics

4

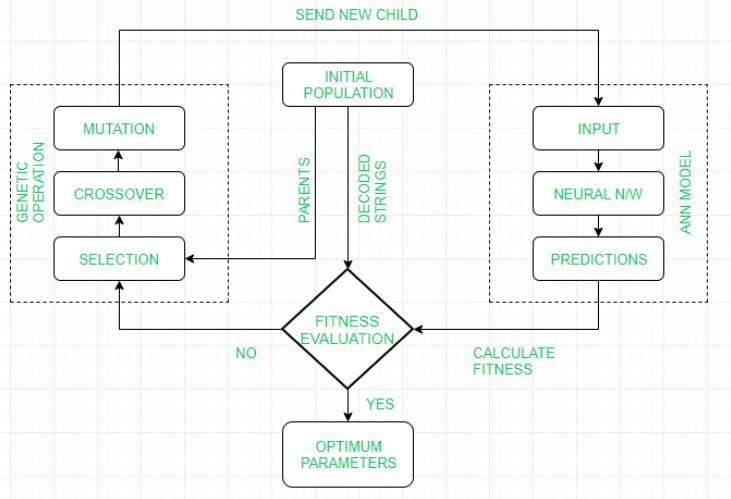
a

Illustrate the Neuro-Genetic Hybrid system with neat architecture.

[L3][CO6]

[8M]

A Neuro Genetic hybrid system is a system that combines **Neural networks**: which are capable to learn various tasks from examples, classify objects and establish relations between them, and a **Genetic algorithm**: which serves important search and optimization techniques. Genetic algorithms can be used to improve the performance of Neural Networks and they can be used to decide the connection weights of the inputs. These algorithms can also be used for topology selection and training networks.

**Working Flow:**

GA repeatedly modifies a population of individual solutions. GA uses three main types of rules at each step to create the next generation from the current population:

1. **Selection** to select the individuals, called parents, that contribute to the population at the next generation
2. **Crossover** to combine two parents to form children for the next generation
3. **Mutation** to apply random changes to individual parents in order to form children

GA then sends the new child generation to ANN model as a new input parameter. Finally, calculating the fitness by the developed ANN model is performed.

b

Explain the advantages and disadvantages of Neuro-genetic hybrid Systems

[L2][CO6]

[4M]

Advantages:

- GA is used for topology optimization i.e to select the number of hidden layers, number of hidden nodes, and interconnection pattern for ANN.
- In GAs, the learning of ANN is formulated as a weight optimization problem, usually using the inverse mean squared error as a fitness measure.
- Control parameters such as learning rate, momentum rate, tolerance level, etc are also optimized using GA.
- It can mimic the human decision-making process.

Disadvantages:

- Highly complex system.
- The accuracy of the system is dependent on the initial population.
- Maintenance costs are very high.

Applications:

- Face recognition
- DNA matching
- Animal and human research
- Behavioral system

5

a

Explain Neuro – Fuzzy hybrid system with neat diagram.

[L2][CO6]

[8M]

The Neuro-fuzzy system is based on fuzzy system which is trained on the basis of the working of neural network theory. The learning process operates only on the local information and causes only local changes in the underlying fuzzy system. A neuro-fuzzy system can be seen as a 3-layer feedforward neural network. The first layer represents input variables, the middle (hidden) layer represents fuzzy rules and the third layer represents output variables. Fuzzy sets are encoded as connection weights within the layers of the network, which provides functionality in processing and training the model.

		<pre> graph LR A[CRISP INPUT] --> B[FUZZIFICATION] B --> C[NEURAL NETWORK] C --> D[DEFUZZIFICATION] E[FUZZY RULE BASE] --> B E --> C D --> F[CRISP OUTPUT] </pre> <p>Working flow:</p> <ul style="list-style-type: none"> • In the input layer, each neuron transmits external crisp signals directly to the next layer. • Each fuzzification neuron receives a crisp input and determines the degree to which the input belongs to the input fuzzy set. • The fuzzy rule layer receives neurons that represent fuzzy sets. • An output neuron combines all inputs using fuzzy operation UNION. • Each defuzzification neuron represents the single output of the neuro-fuzzy system. 		
	b	Illustrate the advantages and disadvantages of Neuro-Fuzzy hybrid systems.	[L3][CO5]	[4M]
		<p>Advantages:</p> <ul style="list-style-type: none"> • It can handle numeric, linguistic, logic, etc kind of information. • It can manage imprecise, partial, vague, or imperfect information. • It can resolve conflicts by collaboration and aggregation. • It has self-learning, self-organizing and self-tuning capabilities. • It can mimic the human decision-making process. <p>Disadvantages:</p> <ul style="list-style-type: none"> • Hard to develop a model from a fuzzy system • Problems of finding suitable membership values for fuzzy systems • Neural networks cannot be used if training data is not available. <p>Applications:</p> <ul style="list-style-type: none"> • Student Modelling • Medical systems • Traffic control systems • Forecasting and predictions 		
6	a	Draw the Architecture of Fuzzy Back propagation. Explain it.	[L2][CO6]	[6M]
		<p>FUZZY BP ARCHITECTURE</p> <p>Fuzzy BP is a three layered feed forward architecture. The three layers are-input layer, hidden layer, and output layer. As in BPN, the functioning of fuzzy BP proceeds in two stages, namely</p> <ol style="list-style-type: none"> 1. Learning or Training, and 2. Inference. <p>Consider a configuration of l - m - n (l input neurons, m hidden neurons, and output neurons) for the fuzzy BP model. Figure illustrates the architecture of fuzzy BP.</p>		

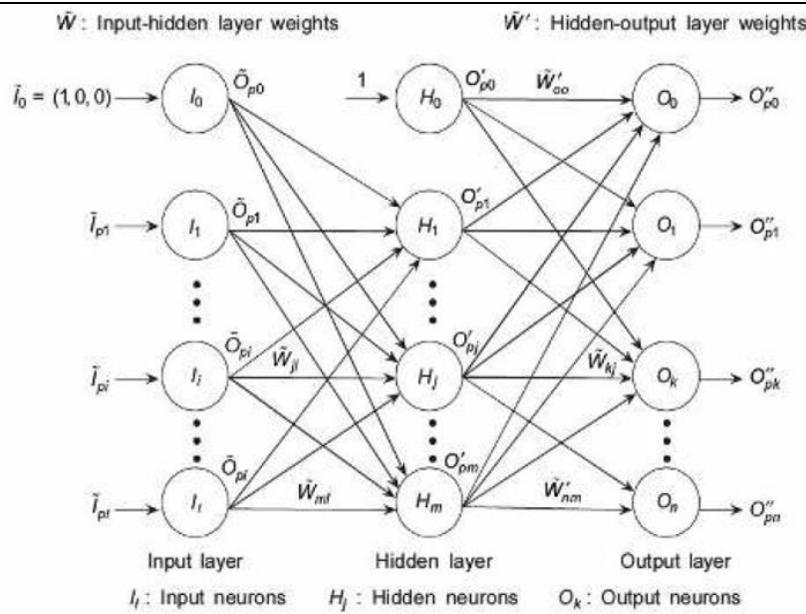


Fig. Architecture of Fuzzy BP

Let $\tilde{I}_p = (\tilde{I}_{p1}, \tilde{I}_{p2}, \dots, \tilde{I}_{pl})$ be the p th pattern among N input patterns that fuzzy BP needs to be trained, with $\tilde{I}_0 = (1, 0, 0)$ as the bias.

Here, \tilde{I}_{pi} indicates the i th input component of the input pattern p and is an LR-type triangular fuzzy number, i.e. $\tilde{I}_{pi} = (I_{pmi}, I_{pa\beta i}, I_{p\beta i})$. Let \tilde{O}_{pi} be the output value of the i th input neuron, O'_{pj} and O''_{pk} are the j th and k th crisp defuzzification outputs of the hidden and output layer neurons respectively. \tilde{W}_{ji} and \tilde{W}'_{kj} are the LR-type fuzzy connection weights between the i th input neuron and the j th hidden neuron, and the j th hidden neuron and k th output neuron respectively. In addition, CE and f are the sigmoidal and centroid functions as explained in Eqs. (12.9) and (12.10).

The computations carried out by each layer are:

The input neurons, hidden neurons and output neurons are represented as

$$\tilde{W}_{kj}$$

$$\tilde{O}_{pi} = \tilde{I}_{pi} = 1, 2, \dots, l; \quad \tilde{O}_{po} = (1, 0, 0)$$

$$O'_{pj} = f(NET_{pj}), \quad j = 1, 2, \dots, m; \quad O'_{po} = 1$$

$$NET_{pj} = CE \left(\sum_{i=0}^l \tilde{W}_{ji} \tilde{O}_{pi} \right)$$

$$O''_{pk} = f(NET'_{pk}), \quad k = 0, 1, 2, \dots, n-1$$

$$NET'_{pk} = CE \left(\sum_{j=0}^m \tilde{W}'_{kj} O'_{pj} \right)$$

b Describe LR Type Fuzzy numbers

[L2][CO6]

[6M]

LR-Type Fuzzy Numbers

The LR-type fuzzy numbers are special type of representations for fuzzy numbers, proposed by Dubois and Prade (1979). They introduced functions called L (and R) which map $R^+ \rightarrow [0, 1]$ and are decreasing shape functions

if
 $L(0) = 1,$
 $L(x) < 0, \forall x < 1,$
 $L(1) = 0 \text{ or } [L(x) > 0 \ \forall x, \text{ and } L(\infty) = 0]$

Definition

A fuzzy number is of LR-type if there exist reference functions L (for left), R (for right) and scalars, $\alpha > 0, \beta > 0$

Here, m , called the mean value of, is a real number and α and β are called the left and right spreads, respectively.

Here, M is the membership function of fuzzy number. An LR-type fuzzy number M can be expressed as $(m, \alpha, \beta)_{LR}$

If α and β are both zero, the LR-type function indicates a crisp value.

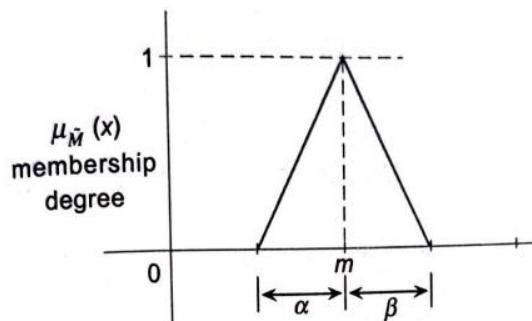


Fig. Symmetric triangular LR-type fuzzy number

In the case of trapezoidal fuzzy numbers, LR-type fuzzy number can be represented as

$$\mu_M(x) = \begin{cases} L\left(\frac{m_1 - x}{\alpha}\right) & \text{for } x \leq m_1, \alpha > 0 \\ R\left(\frac{x - m_2}{\beta}\right) & \text{for } x \geq m_2, \beta > 0 \\ 1 & \text{otherwise} \end{cases}$$

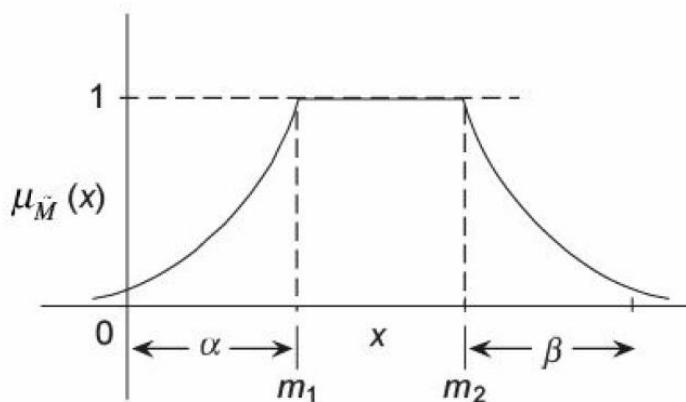
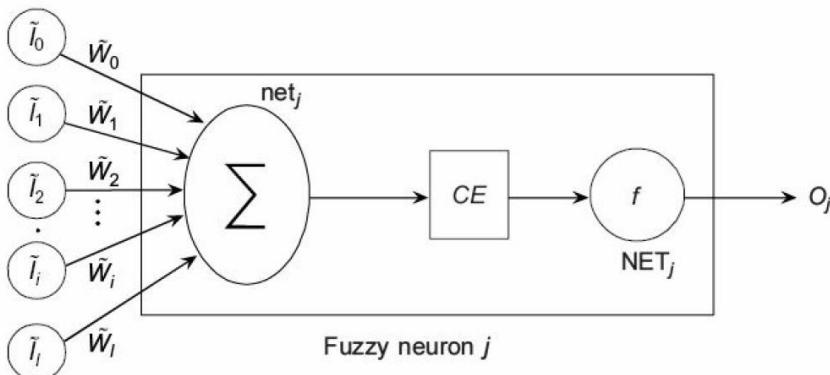


Fig. Trapezoidal LR-type fuzzy number.

The basic operations on LR type fuzzy numbers are:

Let M and N be two LR-type fuzzy numbers given by $M = (m, \alpha, \beta)$ and $N = (n, \gamma, \delta)$. The basic operations are

Addition

	$(m, \alpha, \beta)_{LR} \oplus (n, \gamma, \delta)_{LR} = (m+n, \alpha+\gamma, \beta+\delta)_{LR}$ $(m, \alpha, \beta)_{LR} \oplus (n, \gamma, \delta)_{RL} = (m-n, \alpha+\delta, \beta+\gamma)_{LR}$ Multiplication $(m, \alpha, \beta)_{LR} \otimes (n, \gamma, \delta)_{LR} = (mn, m\gamma + n\alpha, m\delta + n\beta)_{LR}$ for $m \geq 0, n \geq 0$ $(m, \alpha, \beta)_{RL} \otimes (n, \gamma, \delta)_{LR} = (mn, n\alpha - m\delta, n\beta - m\gamma)_{RL}$ for $n \geq 0, m < 0$ $(m, \alpha, \beta)_{LR} \otimes (n, \gamma, \delta)_{RL} = (mn, -n\beta - m\delta, -n\alpha - m\gamma)_{RL}$ for $m < 0, n < 0$ Scalar Multiplication $\lambda \bullet (m, \alpha, \beta)_{LR} = (\lambda m, \lambda \alpha, \lambda \beta)_{LR}, \forall \lambda \geq 0, \lambda \in R$ $\lambda \bullet (m, \alpha, \beta)_{RL} = (\lambda m, -\lambda \alpha, -\lambda \beta)_{RL}, \forall \lambda < 0, \lambda \in R$			
7	a	Explain about Fuzzy Neuron.	[L2][CO6]	[6M]
	i) FUZZY NEURON The fuzzy neuron is the basic element of the fuzzy BP model. Figure illustrates the architecture of the fuzzy neuron. Given the input vector (I) and weight vector W , the fuzzy neuron computes the crisp output given by $O = f(NET) = f\left(CE\left(\sum_{i=0}^l \tilde{W}_i \cdot \tilde{I}_i\right)\right)$ <p>where, $I_0 = (1, 0, 0)$ is the bias. Here, the fuzzy weighted summation</p> $\tilde{net} = \sum_{i=0}^l \tilde{W}_i \cdot \tilde{I}_i$  <p>Fig. fuzzy neuron architecture of fuzzy BP model</p> <p>The function CE is the centroid of the tS triangular fuzzy number and can be treated as a defuzzification operation which maps fuzzy weighted summation value to a crisp value. Thus, if $\tilde{net} = (net_m, net_a, net_\beta)$ is the fuzzy weighted summation then the function CE is given by</p> $CE(\tilde{net}) = net_m + \frac{1}{3}(net_\beta - net_a) = NET$ <p>The function f is the sigmoidal function which performs nonlinear mapping between the input and output. f is defined as</p>			

$$f(NET) = \frac{1}{1 + \exp(-NET)}$$

This is the final computation to obtain the crisp output value O. In the fuzzy neuron, both input vector i and weight vector W are represented by triangular LR-type fuzzy numbers. Thus, for $\tilde{I} = (\tilde{I}_0, \tilde{I}_1, \dots, \tilde{I}_l)$ the input component vector i ; is represented by the LR-type fuzzy number $(I_{mi}, I_{\alpha i}, I_{\beta i})$. Similarly, for $\tilde{W} = (\tilde{W}_0, \tilde{W}_1, \dots, \tilde{W}_l)$ the weight vector component W ; is represented as $\tilde{W}_i = (W_{mi}, W_{\alpha i}, W_{\beta i})$.

b Infer the Fuzzy Logic Controller with neat Architecture.

[L4][CO6] [6M]

FUZZY LOGIC CONTROLLER

Soh and Yang (1996). Yang and Soh (2000) have investigated fuzzy based structural optimization using GA. Using the approach of Soh and Yang as seen a fuzzy logic controller (FLC) a rule based system incorporating the flexibility of human decision making is used for fuzzy structural optimization.

The fuzzy functions are intended to represent a human expert's conception of the linguistic terms, thus giving an approximation of the confidence with which precise numeric value is described by a linguistic label.

Components of Fuzzy Logic Controller (FLC)

As shown in Fig., fuzzy logic controller process is divided into three stages.

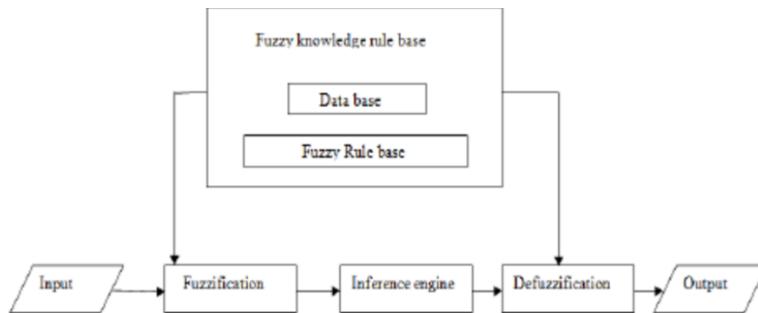


Fig: Frame work for fuzzy logic controller

- Fuzzification**-To calculate fuzzy input (i.e. to evaluate the input variables with respect to corresponding linguistic terms in the condition side).
- Fuzzy inference**-To calculate fuzzy output (i.e. to evaluate the activation strength of every rule base and combine their action sides).
- Defuzzification**-To calculate the actual output (i.e. to convert the fuzzy output into precise numerical value).

Fuzzy IF-THEN Rules

Fuzzy rules take the form IF (conditions) and THEN (actions), where conditions and actions are linguistic variables, respectively. An example of Fuzzy IF-THEN rule is given below.

- Increase interest rates slightly if unemployment is low and inflation is moderate.
- Increase interest rates sharply if unemployment is low and inflation is moderate but rising sharply.
- Decrease interest rates slightly if unemployment is low but increasing and inflation rate is low and stable.

The primary format of IF-THEN rules is given in Fig.

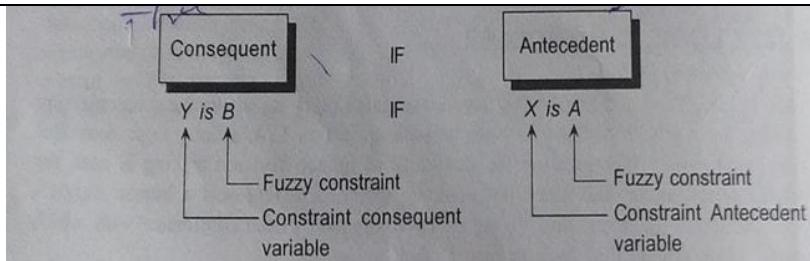


Fig formatt of IF THEN rules

Example

Volume is small if pressure high.

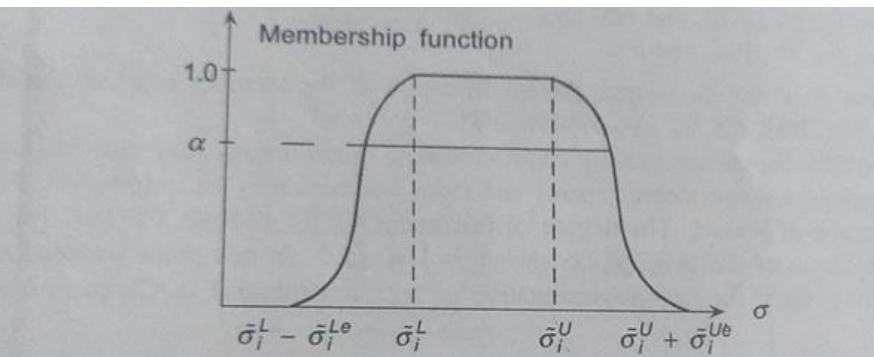


Fig. Fuzzy constraints for stress

Here, the symbol means fuzzy variable operator and α represents a series of linguistic variables that means "very very small", "small", "medium", "large", "very large", and "very very large" and so forth. According to Zadeh (1987), the following seven fuzzy variables are usually in the study of structural optimization as negative large (NL), negative medium (NM), negative small (NS), zero (ZE), positive small (PS), positive medium (PM), and positive large (PL).

They are defined by the membership functions as shown in Fig. For the convergence of implementation, seven fuzzy variables are assigned seven integer reference numbers, namely -3, -2, -1, 0, 1, 2, 3 respectively.

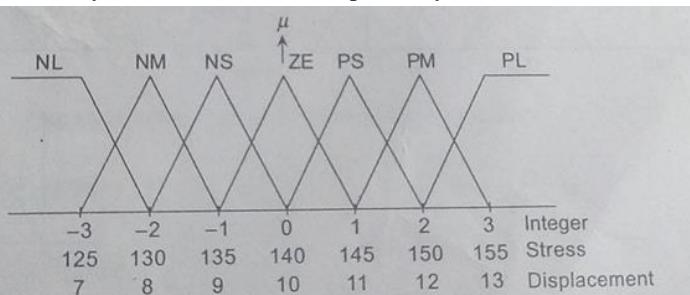


Fig. Memebership function for fuzzy variables

8

a Design a Fuzzy Logic Controller using Genetic Algorithm.

[L6][CO6]

[6M]

GA IN FUZZY LOGIC CONTROLLER DESIGN

- For optimal control problems, fuzzy logic techniques are primarily applied since quick control strategy is needed and imprecise and qualitative definition of action plans are available.
- While designing an optimal fuzzy controller, one has to look for two primary activities
 - Find optimal membership functions for control and action variable.
 - Find an optimal set of rules between control and action variable.

In the above two cases, GAs have been suitably used.

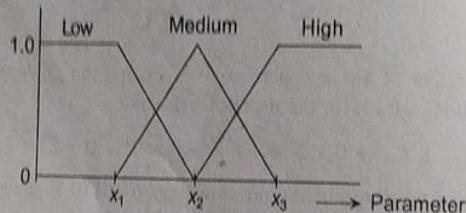


Fig.Fuzzy membership functions and typical variables used to optimize the parameters

The above Figure shows typical membership functions for a variable (control or action) having three choices **low, medium, and high**. Since the maximum membership function value of these choices is always one, the abscissa marked by X are usually chosen by the user.

These abscissa can be treated as variables in GA and an optimization problem can be posed to find these variables for minimizing or maximizing a control strategy such as time of variable operation, product quality, and others.

Consider an example given by Deb (1999) to illustrate how GA can be uniquely applied to the above problem.

Let us assume that there **are two control variables (temperature and humidity)** and there are three operations for each, low, medium, and high. There is one **action variable (water jet flow rate)** which also takes three choices low, medium, and high.

Considering individual effect of each control variable separately, there are total of $4 * 4 - 1 = 15$ combinations of control variables possible as shown in the below Table.

Humidity	Temperature			
	Low	Medium	High	No action
Low	High	Medium	-	Medium
Medium	Low	-	Medium	Medium
High	Medium	High	-	-
No action	-	-	High	-

Fig. Action variables for string representing fuzzy rule base

Thus, finding an optimal rule base is equivalent to finding the four operations (fourth operation-not known) or the action variable for each combination of the control variables.

A GA with a string length of 15 and with a ternary coding can be used to represent the rule base for this problem. Considering real values to 4 for representation as **1-Low, 2-Medium, 3-High, 4-No action**, thereby signifying the absence of the corresponding combination of action variables in the rule base. Table shows the rule base and this can be represented by the following string.

312424344243224

Although this rule base may not be the optimal one, GA can process a population of such rule bases and finally find the optimal rule base. Once the rows present in the rule base are determined from the string user, defined fixed membership functions can be used to simulate the underlying process. **The objective function can be evaluated and the usual single point cross over and a mutation operator (one allele mutating to one of three other alleles) can be used with this coding GA can find the optimal number of rules to solve the problem.**

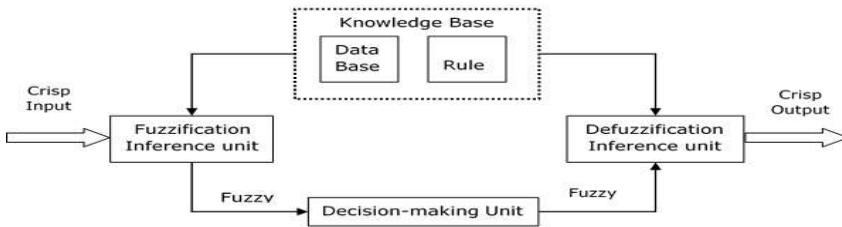
If one wants to use binary strings instead of ternary strings and two bits are used to represent each of four operations, **a total of 30 bits** is necessary to represent a rule base. This kind of technique has been used to design fuzzy logic controller for mobile robot navigation among dynamic obstacles (Deb et al., 1998).

Both optimum membership function determination and optimal rule base tasks can be achieved simultaneously by using a concatenation of two codings mentioned above. A part of the overall string will represent the abscissa of the

		control variables and the rest of the string will represent the rules present in the rule base. Fitness is calculated as explained above		
	b	Describe the fuzzy rule base with neat architecture.	[L2][CO6]	[6M]
		<p>Rule Base is a component used for storing the set of rules and the If-Then conditions given by the experts are used for controlling the decision-making systems. There are so many updates that come in the Fuzzy theory recently, which offers effective methods for designing and tuning of fuzzy controllers. These updates or developments decreases the number of fuzzy set of rules. In a fuzzy system, all rules fire to some extent, i.e., fire partially. If the antecedent is true to some degree of membership, then the consequent is also true to that same degree.</p> <p>Fuzzy IF-THEN Rules</p> <p>Fuzzy rules take the form IF (conditions) and THEN (actions), where conditions and actions are linguistic variables, respectively. An example of Fuzzy IF-THEN rule is given below.</p> <ul style="list-style-type: none"> ➤ Increase interest rates slightly if unemployment is low and inflation is moderate. ➤ Increase interest rates sharply if unemployment is low and inflation is moderate but rising sharply. ➤ Decrease interest rates slightly if unemployment is low but increasing and inflation rate is low and stable. <p>The primary format of IF-THEN rules is given in Fig.</p> <p>Fig format of IF THEN rules</p>		
9	a	Explain various Soft Computing Tools.	[L2][CO6]	[6M]
		<p>Fuzzy Logic as a Soft Computing Tool</p> <p>The realization of uncertainty, vagueness and ambiguity in the world has led to the concept of fuzziness.</p> <p>In the last three decades, significant progress has been made in the development of fuzzy sets, fuzzy logic theory, and their use in engineering applications.</p> <p>The successful application of fuzzy sets and fuzzy logic can be attributed to the fact that fuzzy theory reflects the true situation in the real world, where human thinking is dominated by approximate reasoning logic.</p> <p>This is suited for applications where the ability to model real-world problems in precise mathematical forms is difficult.</p> <p>In a traditional optimization techniques, the constraints must strictly be satisfied. However, it is not reasonable to discard those designs that slightly violate one or more constraints during the early design stage.</p>		

		<p>These complexities and uncertainties encountered in the optimization and design of real structures provide the main motivation for the fuzzy integrated system. Hence, fuzzy logic can be considered as a soft computing tool.</p> <p>Genetic Algorithm as a Soft Computing Tool</p> <p>A system to be optimized is represented by a binary string which encodes the parameters of the system.</p> <p>A population of strings with initial random parameters is used. A number of generations are simulated with operators representing the major elements of evolution such as competition, fitness based selection, recombination, and mutation. The whole process is highly random.</p> <p>However, the evolutionary process lead to filter individuals in the population closer to satisfying the objective function of the optimization problem.</p> <p>Genetic algorithms have all the characteristics of soft computing. The methodology is highly robust and imprecision tolerant.</p> <p>If a unique optimum exists, the procedure approaches it through gradual improvement of the fitness and if the optimum is not unique, the method will approach one of the optimum solutions.</p>		
	b	Compare Sequential , Auxillary and Embedded hybrid systems.	[L5][CO6]	[6M]
		<p>In <i>Sequential hybrid system</i>, the technologies are used in pipelining fashion.</p> <p>In <i>Auxiliary hybrid system</i>, one technology calls the other technology as subroutine.</p> <p>In <i>Embedded hybrid system</i>, the technologies participating appear to be fused totally.</p> <p>In Sequential hybrid system, the technologies are used in pipelining fashion. Thus, one technology's output becomes another technology's input and it goes on. However, this is one of the weakest form of hybridization since an integrated combination of technologies is not present.</p> <p>Example: A Genetic algorithm preprocessor obtains the optimal parameters for different instances of a problem and hands over the preprocessed data to a neural network for further processing.</p> <p>In Auxiliary hybrid system, one technology calls the other technology as subroutine to process or manipulate information needed. The second technology processes the information provided by the first and hands it over for further use. This type of hybridization is better than the sequential hybrids.</p> <p>Example : A neuron-genetic system in which a neural network employs a genetic algorithm to optimize its structural parameters that defines its architecture.</p> <p>In Embedded hybrid system, the technologies participating are integrated in such a manner that they appear intertwined. The fusion is so complete that it would appear that no technology can be used without the others for solving the problem.</p> <p>Example : A NN-FL hybrid system may have an NN which receives fuzzy inputs, processes it and extracts fuzzy outputs as well.</p>		
10	a	Explain the various components in Fuzzy Rule Base System.	[L2][CO6]	[6M]
		<p>The following five functional blocks will help you understand the construction of FIS –</p> <ul style="list-style-type: none"> • Rule Base – It contains fuzzy IF-THEN rules. • Database – It defines the membership functions of fuzzy sets used in 		

- fuzzy rules.
- **Decision-making Unit** – It performs operation on rules.
 - **Fuzzification Interface Unit** – It converts the crisp quantities into fuzzy quantities.
 - **Defuzzification Interface Unit** – It converts the fuzzy quantities into crisp quantities. Following is a block diagram of fuzzy inference system.



b Compare Neuro processing and Fuzzy Processing.

[L4][CO6] [6M]

Comparison of Fuzzy Systems with Neural Networks:

When neural networks are concerned, if one problem is expressed by sufficient number of observed examples then only it can be used. These observations are used to train the black box. Though no prior knowledge about the problem is needed extracting comprehensible rules from a neural network's structure is very difficult.

A fuzzy system, on the other hand, does not need learning examples as prior knowledge, rather linguistic rules are required. Moreover, linguistic description of the input and output variables should be given. If the knowledge is incomplete, wrong or contradictory, then the fuzzy system must be tuned. This is a time consuming process.

Neural processing	Fuzzy processing
Mathematical model not necessary	Mathematical model not necessary
Learning can be done from search	Apriori knowledge is needed
There are several learning algorithms	Learning is not possible
Black-box behavior	Simple interpretation And implementation