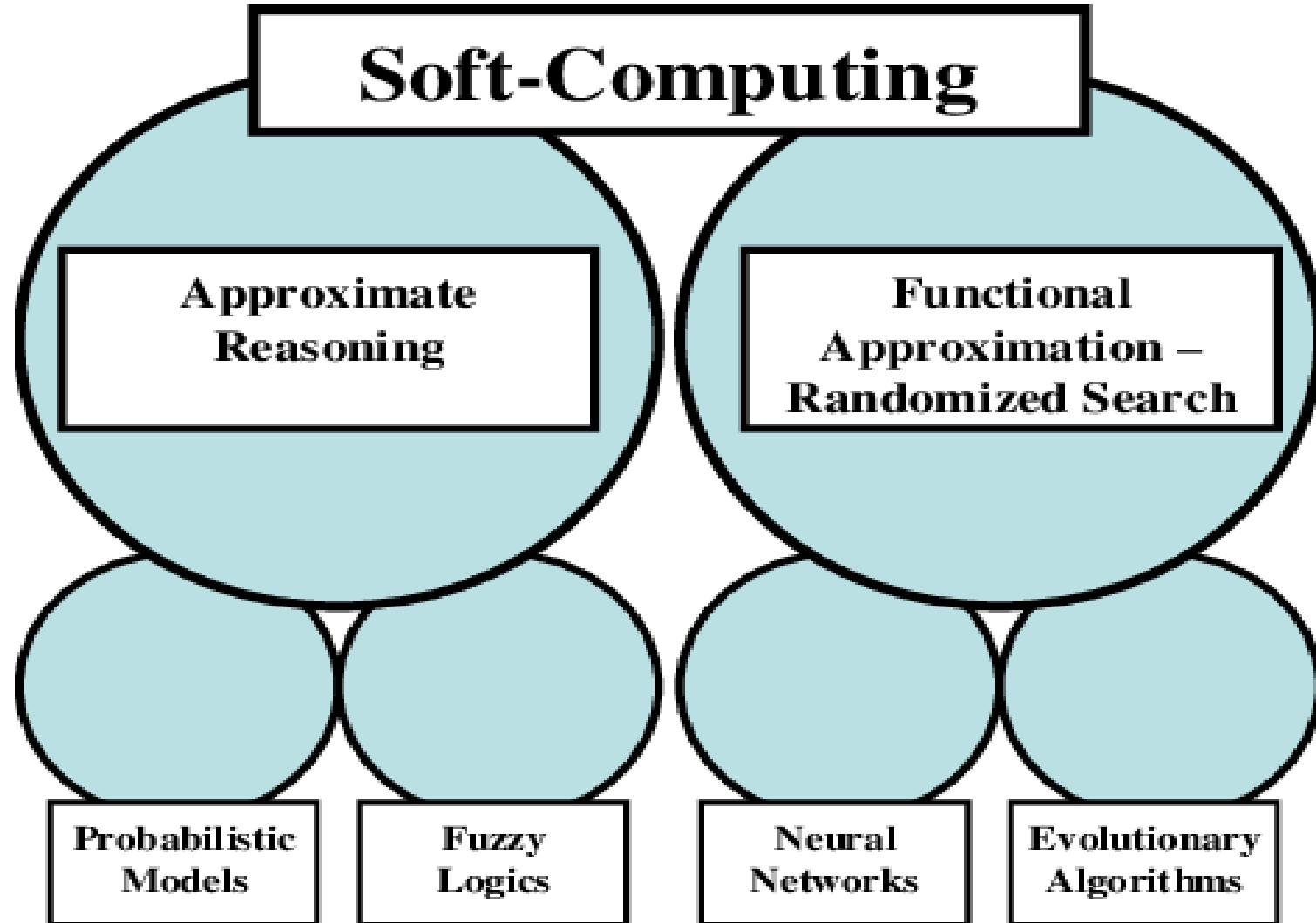


Introduction to Artificial Neural Network (ANN)

Dr. Shruti Mishra
School of Computer Sc. & Engg. (SCOPE)

Computing: Soft Computing Vs Hard Computing



Soft Computing

- Soft Computing (SC) is an emerging area in Computer Science that is tolerant:
 - to imprecise and uncertain problems with partial truth to achieve an approximate, robust and low cost optimal solution.
- **Soft Computing** consists of numerous techniques that study:
 - the biological processes such as reasoning, genetic evolution, survival of the creatures and human nervous system.
- SC solves tremendous number of complex real world problems in different sections such as:
 - stock market predictions in business,
 - computer aided diagnosis in medical,
 - handwriting recognition in fraud detection ,
 - image retrieval,
 - biometric application in image processing etc. and the list go on.

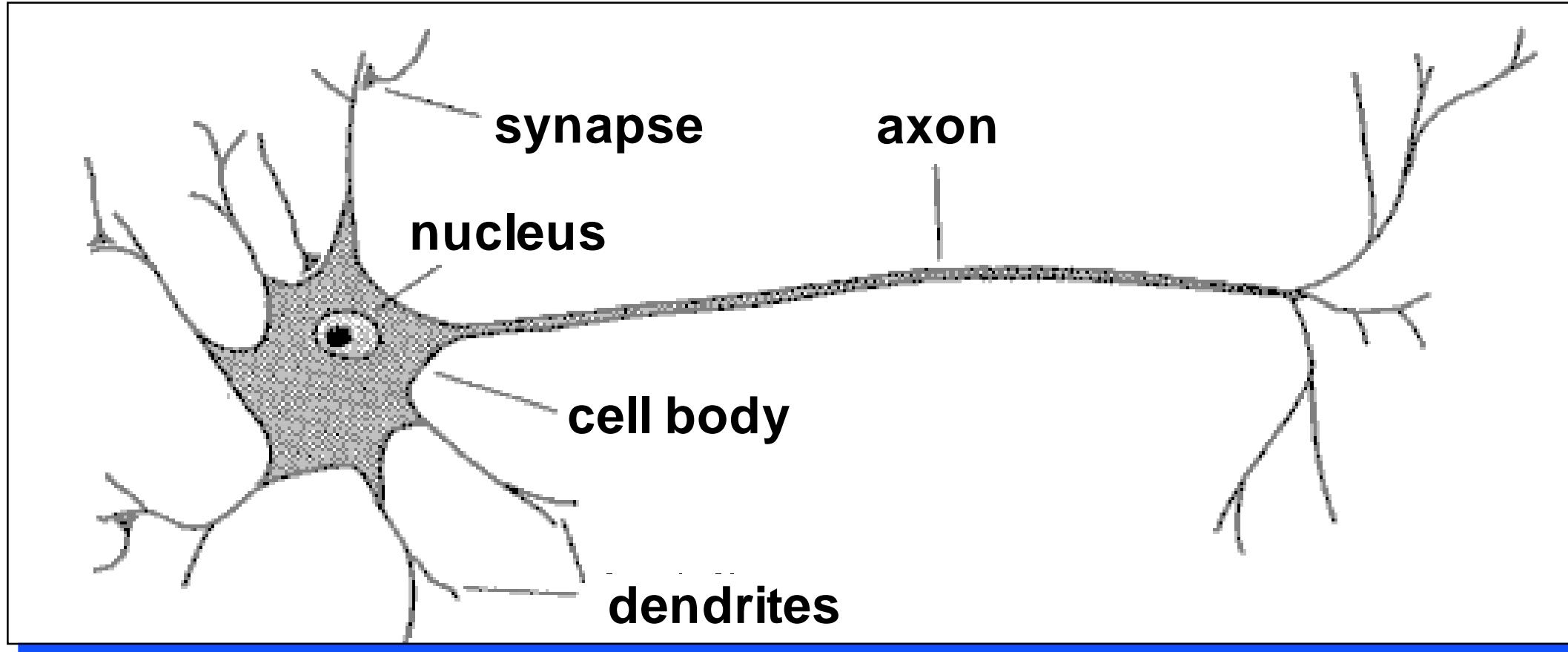
Hard Computing

- The conventional algorithms, also termed as Hard Computing algorithms, follow *mathematical methodologies strictly* which make it inefficient to solve real world problems by taking more computation time.
- The conventional algorithms require *exact input data, use a precise methodology and generate a precise output* which make it a crisp system.
- It fails when the input is not exact.
- Examples of conventional algorithms are merge sort, quick sort, binary search, greedy algorithm, dynamic programming etc. which are deterministic.

Biological Nervous System

- Biological nervous system is the most important part of many living things, in particular, human beings.
- There is a part called **brain** at the center of human nervous system.
- In fact, any biological nervous system consists of a large number of interconnected processing units called **neurons**.
- Each neuron is approximately 10m long and they can operate in parallel.
- Typically, a human brain consists of approximately 10^{11} neurons communicating with each other with the help of **electrical impulses**.

Neuron: Basic unit of nervous system



Neuron and its working

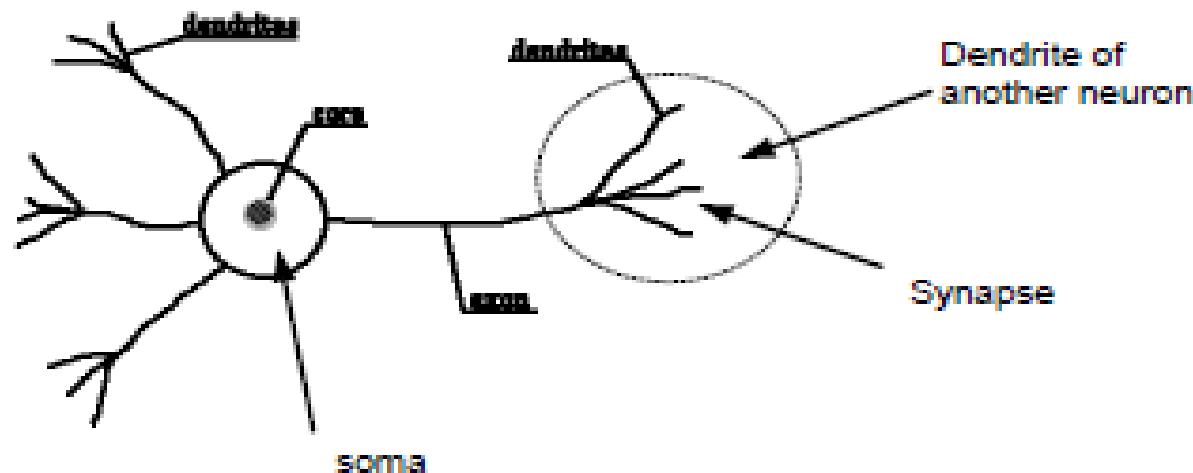


Figure shows a schematic of a biological neuron. There are different parts in it : dendrite, soma, axon and synapse.

Dendrite : A bush of very thin fibre.

Axon : A long cylindrical fibre.

Soma : It is also called a cell body, and just like as a nucleus of cell.

Synapse : It is a junction where axon makes contact with the dendrites of neighboring dendrites/ The point of interconnection of one neuron with other neurons. The amount of signal transmitted depend upon the strength (synaptic weights) of the connections.

Contd..

- There is a chemical in each neuron called neurotransmitter.
- A signal (also called sense) is transmitted across neurons by this chemical.
- That is, all inputs from other neuron arrive to a neurons through dendrites.
- These signals are accumulated at the synapse of the neuron and then serve as the output to be transmitted through the neuron.
- An action may produce an electrical impulse, which usually lasts for about a millisecond.
- Note that this pulse generated due to an incoming signal and all signal may not produce pulses in axon unless it crosses a **threshold value**.
- Also, note that an action signal in axon of a neuron is commutative signals arrive at dendrites which summed up at soma.

Terminology Relationships Between Biological and Artificial Neuron

Biological Neuron	Artificial Neuron
Cell	Neuron
Dendrites/ Synapse	Weights or interconnection
Soma	Net Input
Axon	Output

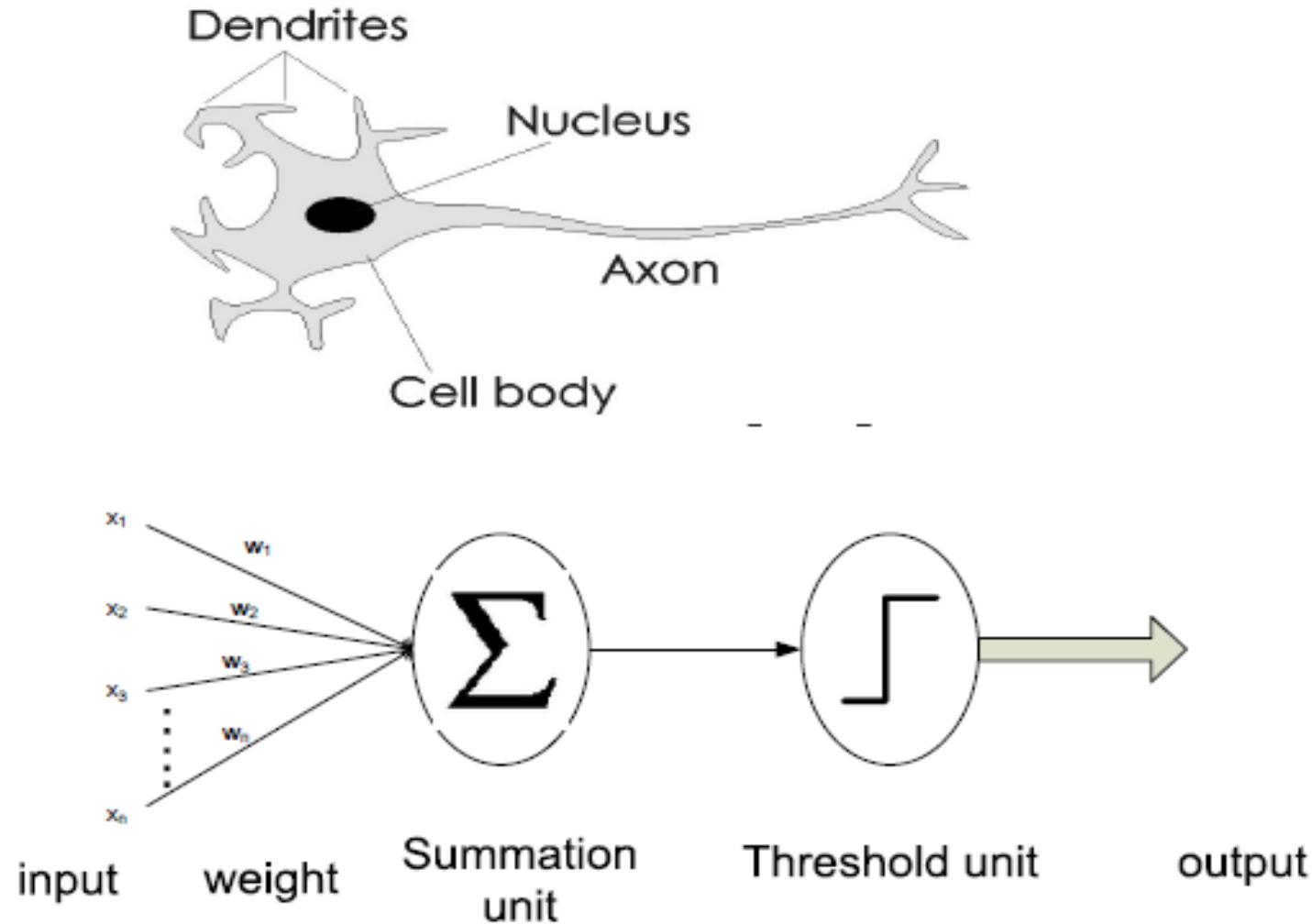
Origin of Neural Network

- Human brain has many incredible characteristics such as massive parallelism, distributed representation and computation, learning ability, generalization ability, adaptivity, which seems simple but is really complicated.
- It has been always a dream for computer scientist to create a computer which could solve complex perceptual problems this fast.
- ANN models was an effort to apply the same method as human brain uses to solve perceptual problems.
- Three periods of development for ANN:
 - 1940:Mcculloch and Pitts: Initial works
 - 1960: Rosenblatt: perceptron convergence theorem
 - Minsky and Papert: work showing the limitations of a simple perceptron
 - 1980: Hopfield/Werbos and Rumelhart: Hopfield's energy approach/back-propagation learning algorithm.

Artificial Neural Network

- In fact, the human brain is a highly complex structure viewed as a massive, highly interconnected network of simple processing elements called **neurons**.
- Artificial neural networks (ANNs) or simply we refer it as neural network (NNs), which are simplified models (i.e. imitations) of the biological nervous system, and obviously, therefore, have been motivated by the kind of computing performed by the human brain.
- The behaviour of a biological neural network can be captured by a simple model called artificial neural network.

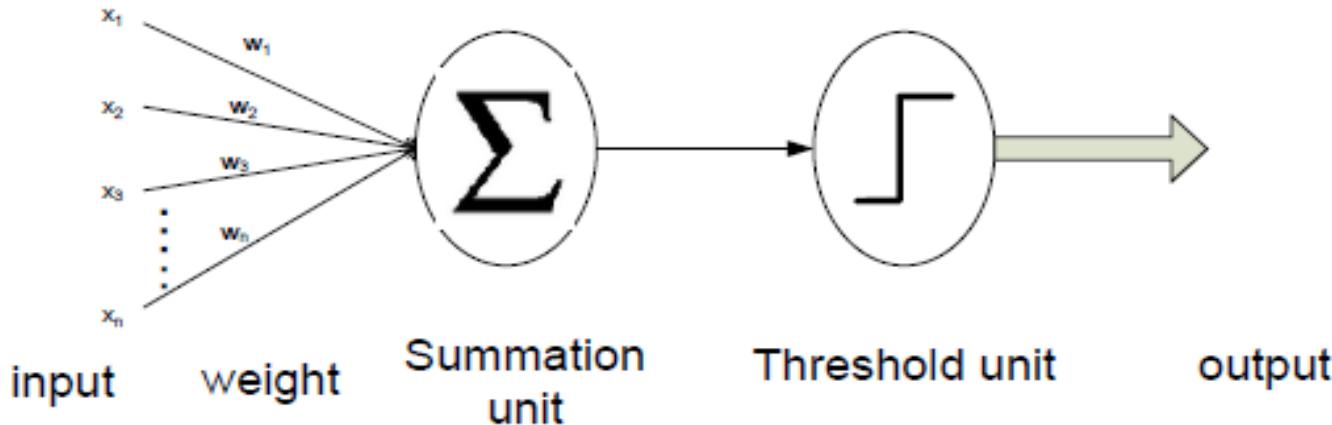
Analogy between BNN and ANN



Artificial Neural Network

- We may note that a neuron is a part of an interconnected network of nervous system and serves the following.
 - Compute input signals
 - Transportation of signals (at a very high speed)
 - Storage of information
 - Perception, automatic training and learning
- We also can see the analogy between the biological neuron and artificial neuron. Truly, every component of the model (i.e. artificial neuron) bears a direct analogy to that of a biological neuron. It is this model which forms the basis of neural network (i.e. artificial neural network).

Artificial Neural Network



- Here, $x_1; x_2; \dots; x_n$ are the n inputs to the artificial neuron
- $w_1; w_2; \dots; w_n$ are weights attached to the input links.

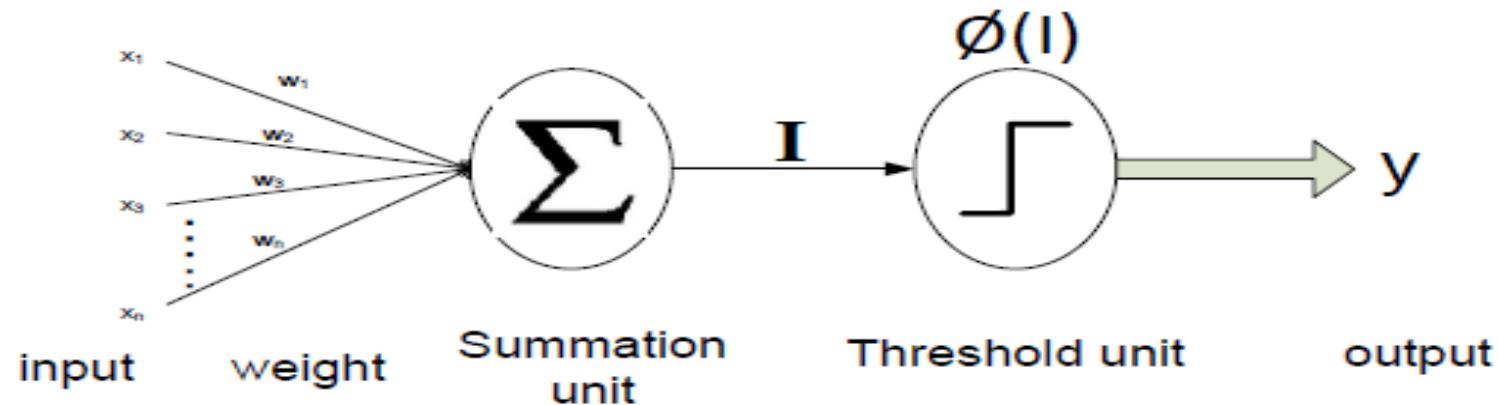
Contd..

- Hence, the total input say I received by the soma of the artificial neuron is:

$$I = w_1x_1 + w_2x_2 + \dots + w_nx_n = \sum_{i=1}^n w_i x_i$$

- To generate the final output y , the sum is passed to a filter Φ called transfer function or Activation function or Squash function, which releases the output.

That is, $y = \phi(I)$



Contd..

- A very commonly known transfer function is the **thresholding function**.
- In this thresholding function, sum (i.e. I) is compared with a threshold value θ .
- If the value of I is greater than θ , then the output is 1 else it is 0 (this is just like a simple linear filter).
- In other words,

$$y = \phi(\sum_{i=1}^n w_i x_i - \theta)$$

where

$$\phi(I) = \begin{cases} 1 & , \text{ if } I > \theta \\ 0 & , \text{ if } I \leq \theta \end{cases}$$

Such a ϕ is called **step function** (also known as **Heaviside function**).

Transformation Function

- **Hard-limit transfer function** : The transformation we have just discussed is called hard-limit transfer function. It is generally used in perception neuron.

In other words,

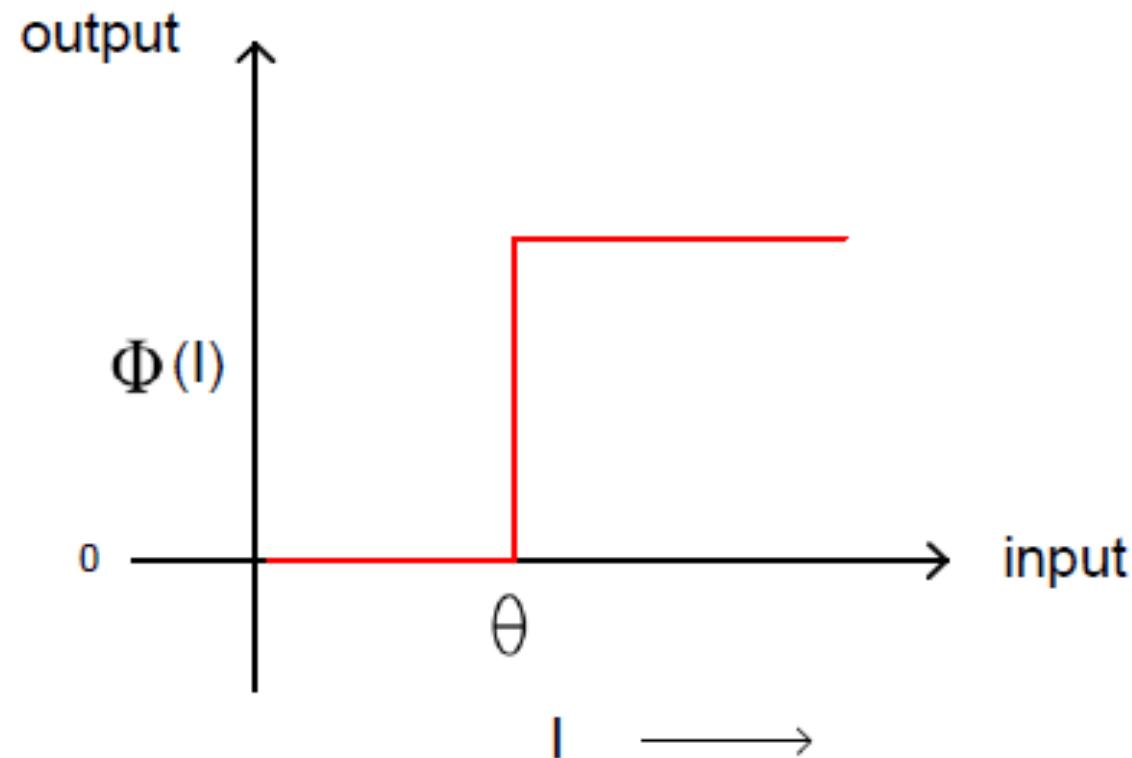
$$\phi(I) = \begin{cases} 1 & , \text{ if } I > \theta \\ 0 & , \text{ if } I \leq \theta \end{cases}$$

- **Linear transfer function** : The output of the transfer function is made equal to its input (normalized) and its lies in the range of –1.0 to +1.0. It is also known as Signum or Quantizer function and it defined as

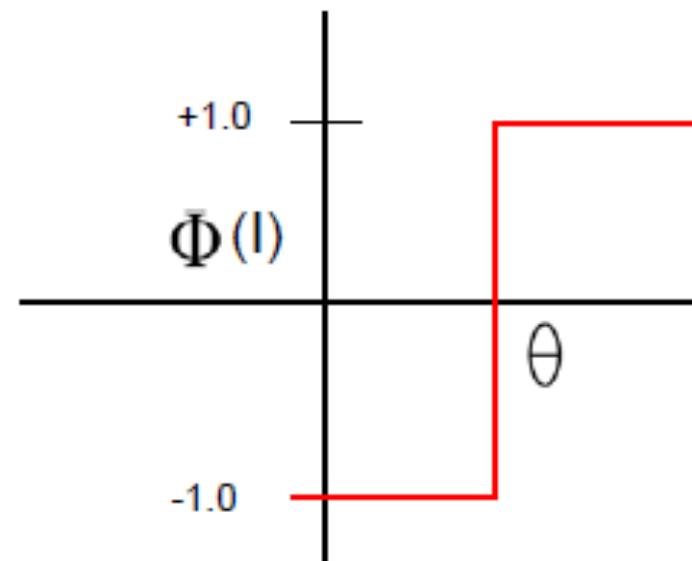
$$\phi(I) = \begin{cases} +1 & , \text{ if } I > \theta \\ -1 & , \text{ if } I \leq \theta \end{cases}$$

Contd..

Following figures illustrates two simple thresholding functions.



(a) Hard-limit transfer function



(b) Signum transfer function

Contd..

- **Sigmoid transfer function** : This function is a continuous function that varies gradually between the asymptotic values 0 and 1 (called log-sigmoid) or -1 and +1 (called Tan-sigmoid) threshold function and is given by

$$\phi(I) = \frac{1}{1+e^{-\alpha I}} \text{ [log-Sigmoid]}$$

$$\phi(I) = \tanh(I) = \frac{e^{\alpha I} - e^{-\alpha I}}{e^{\alpha I} + e^{-\alpha I}} \text{ [tan-Sigmoid]}$$

Here, α is the coefficient of transfer function.



Also, Learning Rate

Contd..

- *Binary sigmoid function:*

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

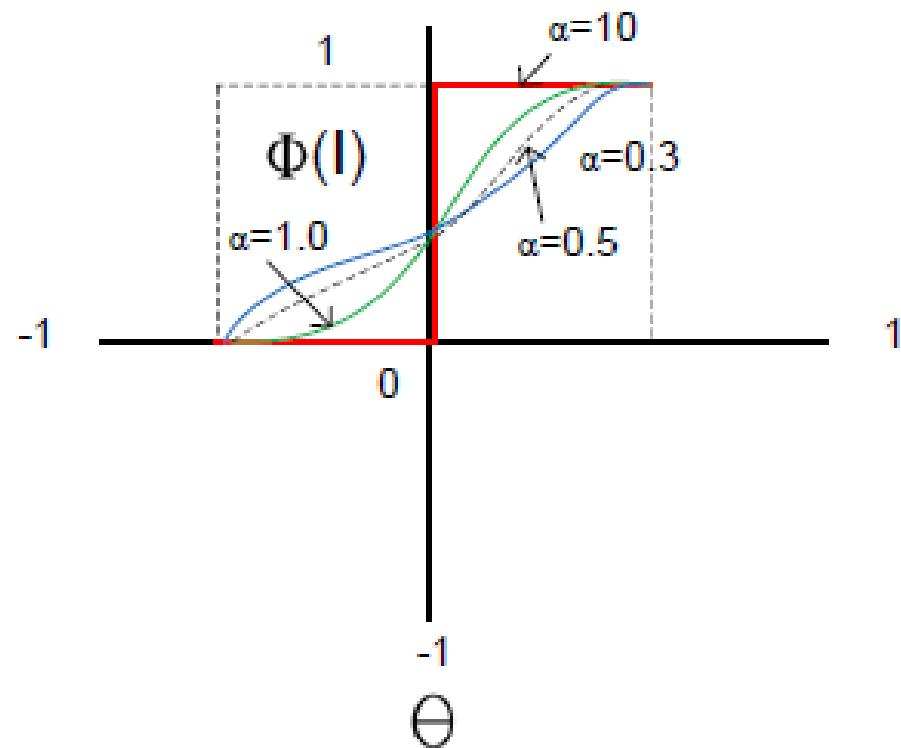
The derivative of this function as: $f'(x) = \lambda f(x)[1 - f(x)]$

Bipolar sigmoid function: This function is defined as

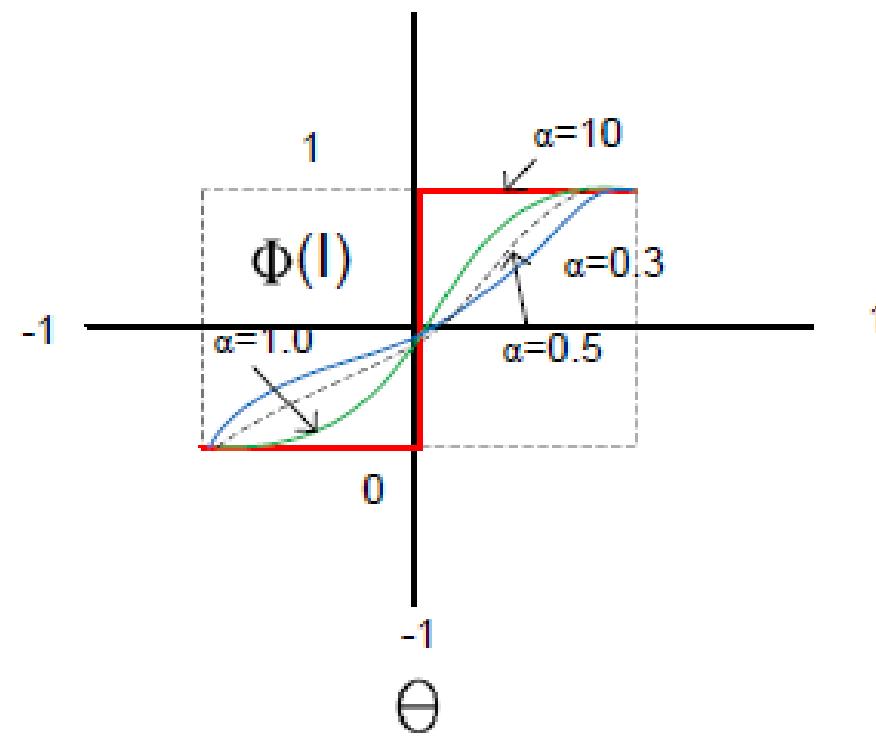
$$f(x) = \frac{2}{1 + e^{-\lambda x}} - 1 = \frac{1 - e^{-\lambda x}}{1 + e^{-\lambda x}}$$

The derivative of this function as: $f'(x) = \frac{\lambda}{2}[1 + f(x)][1 - f(x)]$

Contd..



(a) Log-Sigmoid transfer function



(b) Tan-Sigmoid transfer function

Neural Network Architecture

There are three fundamental classes of ANN architectures:

- Single layer feed forward architecture
- Multilayer feed forward architecture
- Recurrent networks architecture

Single Layer Feedforward Neural Network

- We see, a layer of n neurons constitutes a single layer feed forward neural network.
- This is so called because, it contains a single layer of artificial neurons.
- Note that the input layer and output layer, which receive input signals and transmit output signals are although called layers, they are actually boundary of the architecture and hence truly not layers.
- The only layer in the architecture is the synaptic links carrying the weights connect every input to the output neurons.

Modelling SLFFNN

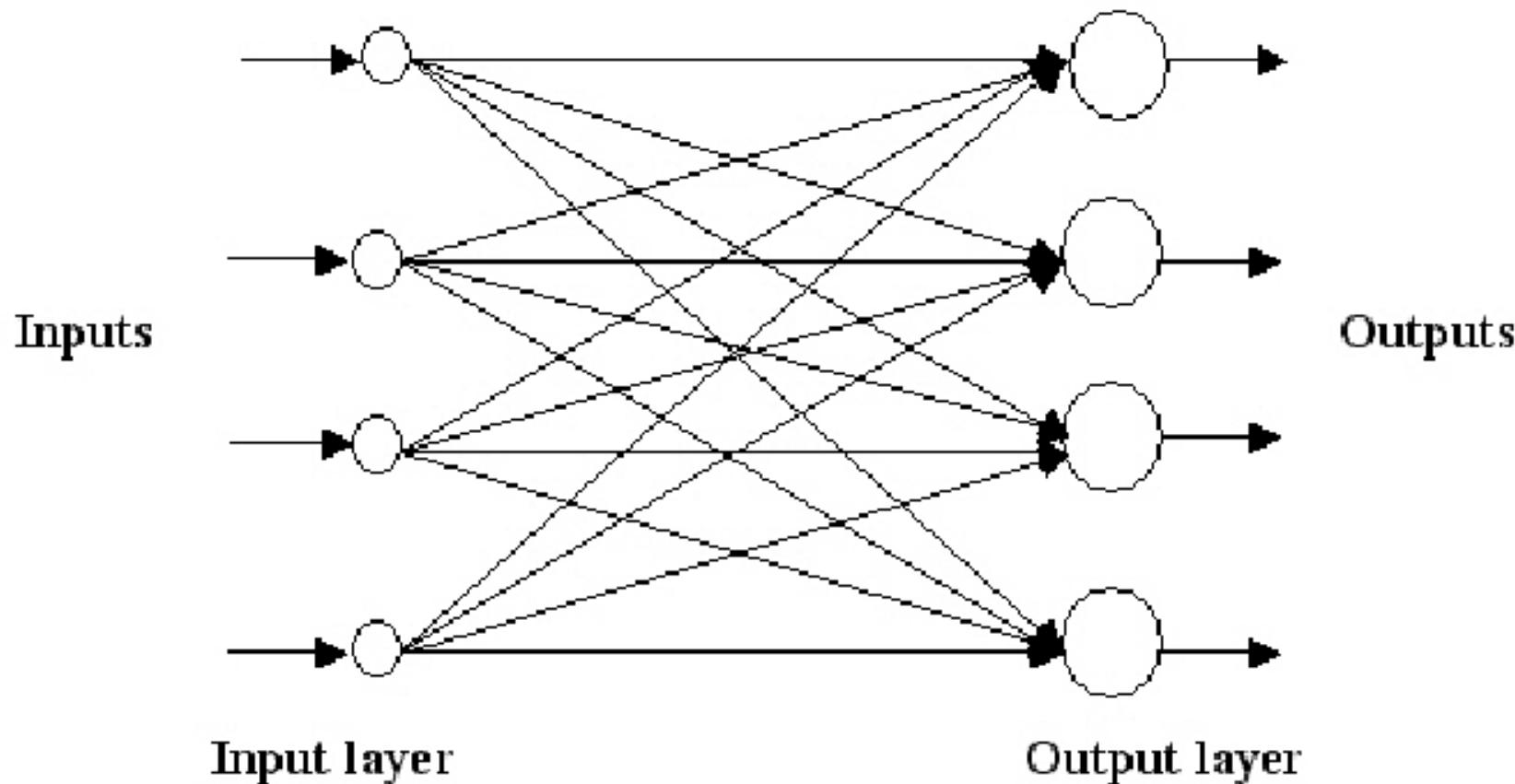
In a single layer neural network, the inputs x_1, x_2, \dots, x_m are connected to the layers of neurons through the weight matrix W . The weight matrix $W_{m \times n}$ can be represented as follows.

$$W = \begin{vmatrix} w_{11} & w_{12} & w_{13} & \cdots & w_{1n} \\ w_{21} & w_{22} & w_{23} & \cdots & w_{2n} \\ \vdots & \vdots & \vdots & & \vdots \\ w_{m1} & w_{m2} & w_{m3} & \cdots & w_{mn} \end{vmatrix} \quad (1)$$

The output of any k -th neuron can be determined as follows.

$$O_k = f_k (\sum_{i=1}^m (w_{ik} x_i) + \theta_k)$$

where $k = 1, 2, 3, \dots, n$ and θ_k denotes the threshold value of the k -th neuron. Such network is feed forward in type or acyclic in nature and hence the name.

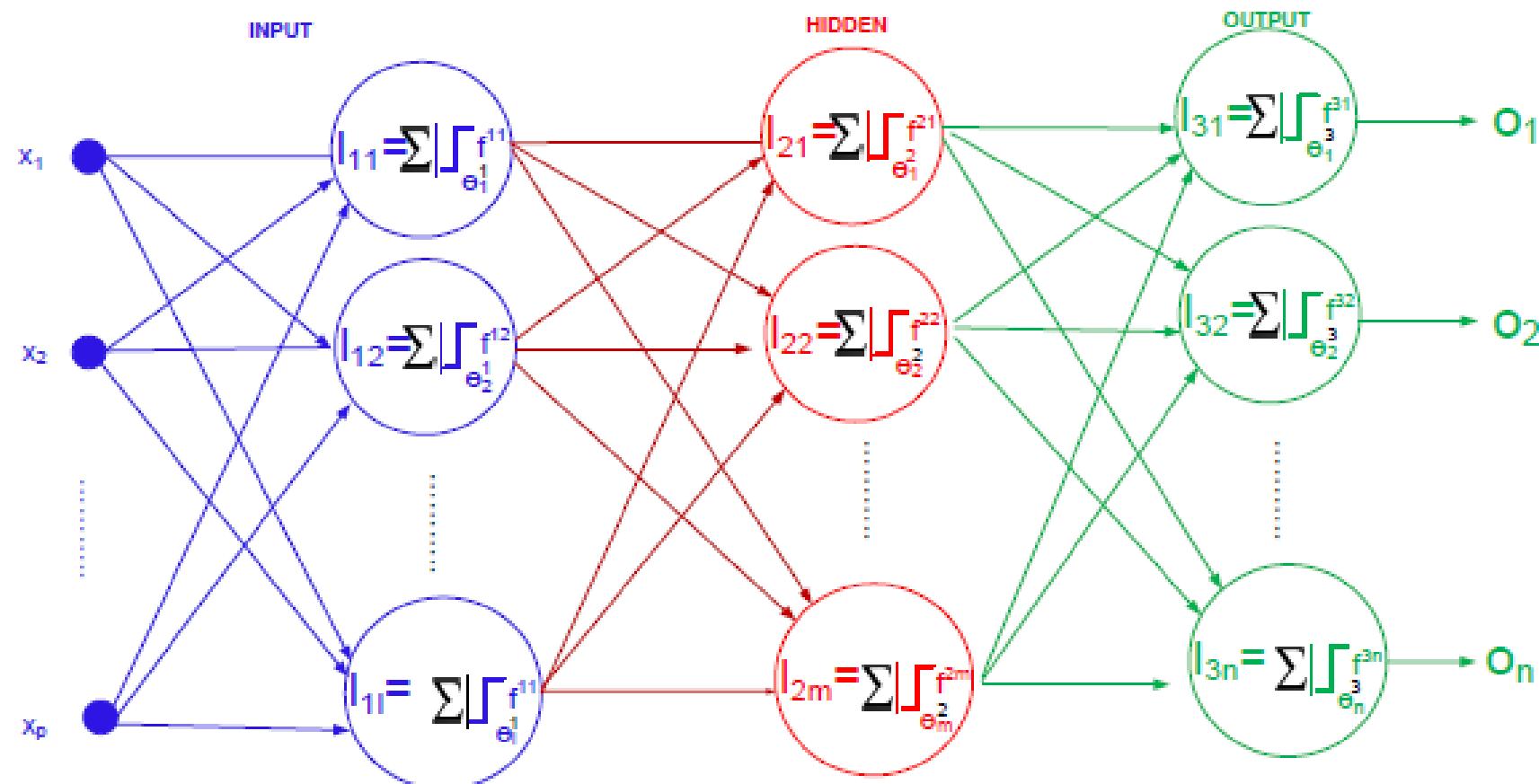


Multilayer Feedforward Neural Network

- This network, as its name indicates is made up of multiple layers.
- Thus architectures of this class besides processing an input and an output layer also have one or more intermediary layers called **hidden layers**.
- The hidden layer(s) aid in performing useful intermediary computation before directing the input to the output layer.
- A multilayer feed forward network with l input neurons (number of neuron at the first layer), m_1, m_2, \dots, m_p number of neurons at i -th hidden layer ($i = 1, 2, \dots, p$) and n neurons at the last layer (it is the output neurons) is written as $l - m_1 - m_2 - \dots - m_p - n$ MLFFNN.

Contd..

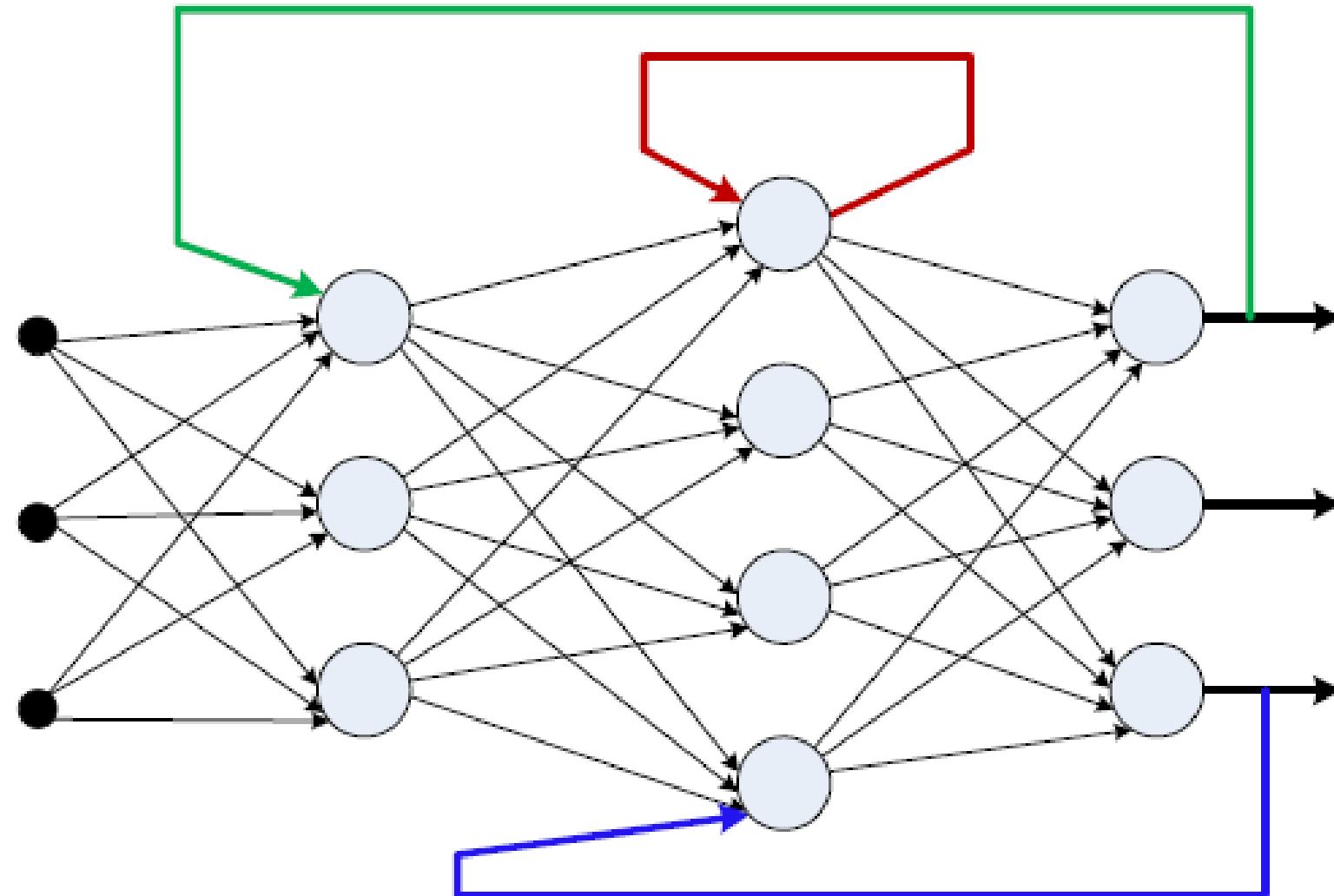
Figure shows a schematic diagram of multilayer feed forward neural network with a configuration of $l - m - n$.



Recurrent Neural Network

- The networks differ from feedback network architectures in the sense that there is at least one "feedback loop".
- Thus, in these networks, there could exist one layer with feedback connection.
- There could also be neurons with self-feedback links, that is, the output of a neuron is fed back into itself as input.

Contd..

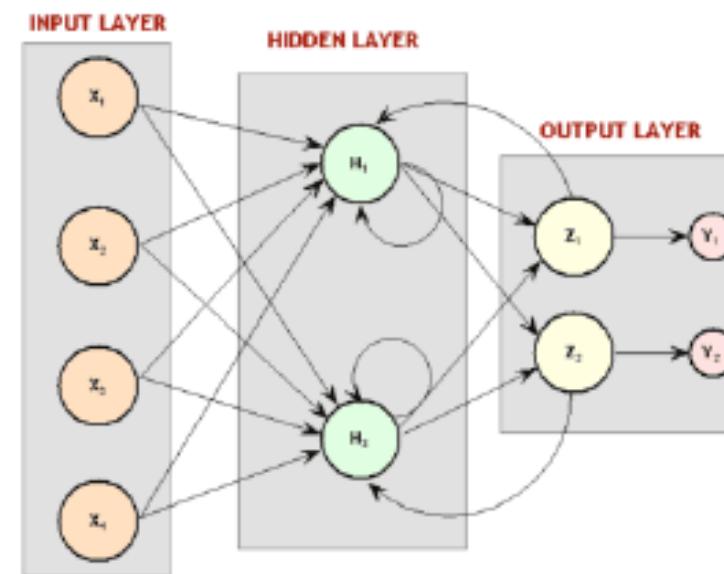
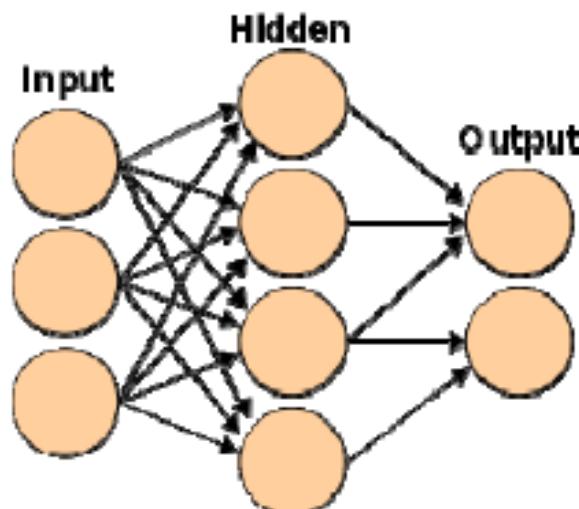


NN Architecture Summary

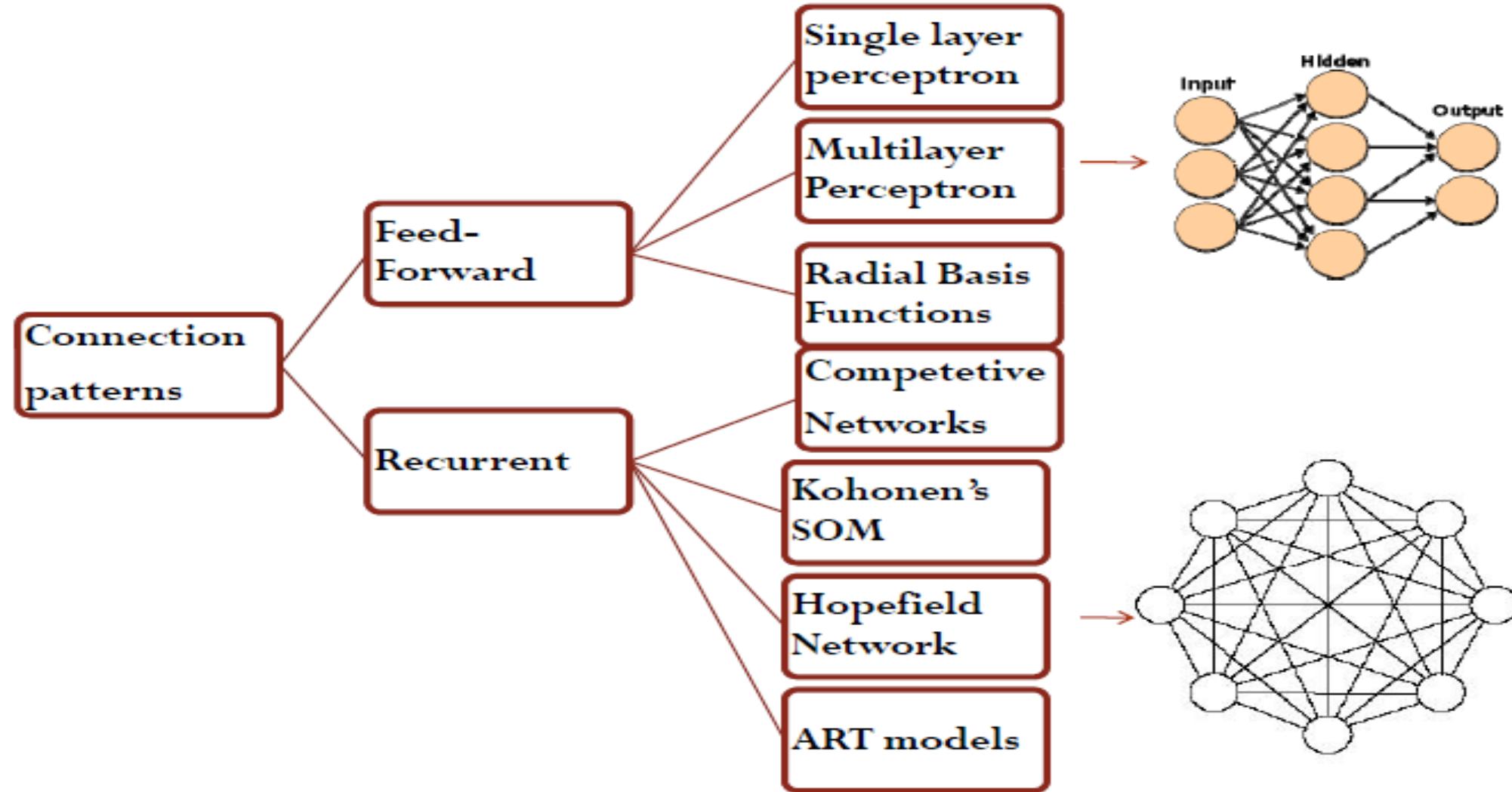
Connection
patterns

Feed-
Forward

Recurrent



Contd..



Learning

1. Supervised

- The correct answer is provided for the network for every input pattern
- Weights are adjusted regarding the correct answer
- In reinforcement learning only a critique of correct answer is provided

2. Unsupervised

- Does not need the correct output
- The system itself recognize the correlation and organize patterns into categories accordingly

3. Hybrid

- A combination of supervised and unsupervised
- Some of the weights are provided with correct output while the others are automatically corrected.

Learning Rules

- There are four basic types of learning rules:
 - *Error correction rules*
 - *Boltzmann*
 - *Hebbian*
 - *Competitive learning*
- Each of these can be trained with or without a teacher
- Have a particular architecture and learning algorithm

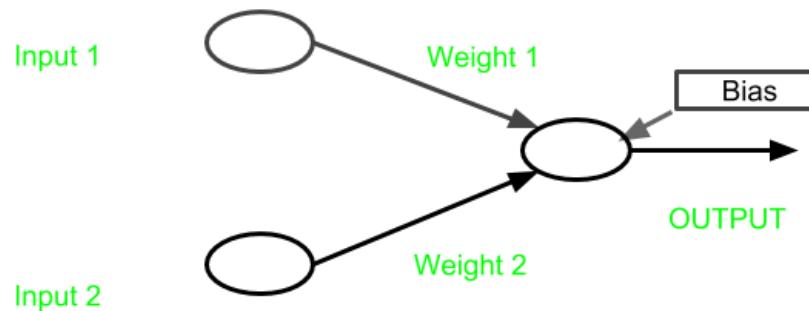
Terminologies

- ❖ **Bias:** The bias included in the n/w has an impact in calculating the net input.
- It is an additional parameter in the Neural Network which is used to adjust the output along with the weighted sum of the inputs to the neuron.
- Bias serves two functions within the neural network – as a specific neuron type, called Bias Neuron, and a statistical concept for assessing models before training.
- **Bias** is a constant which helps the model in a way that it can fit best for the given data.

$$\text{output} = \text{sum}(\text{weights} * \text{inputs}) + \text{bias}$$

Contd..

- Bias is just like a intercept added in a linear equation.



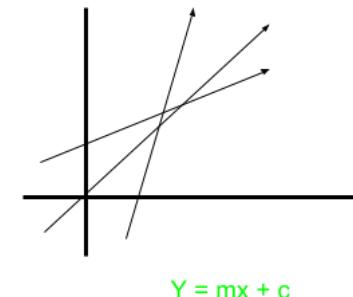
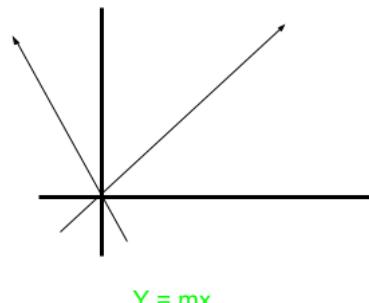
$$\text{Output} = \text{weight1} * \text{input1} + \text{weight2} * \text{input2} + \text{bias}$$

Need of bias

In the figure let us say: $y=mx+c$

Where, m = weight and c = bias

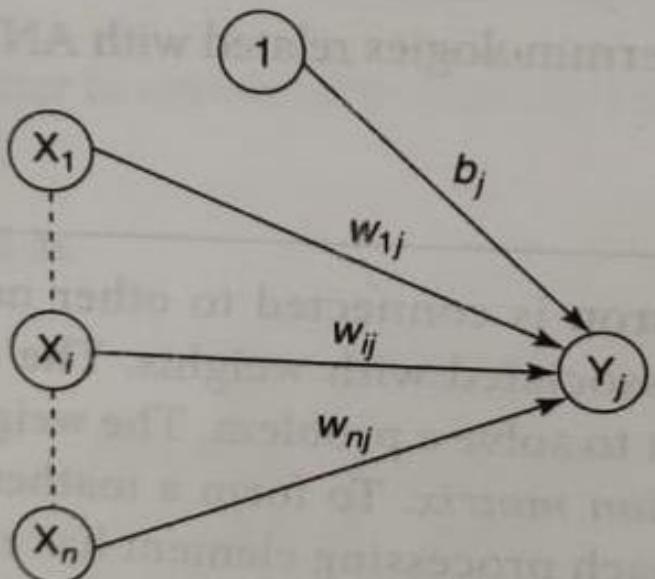
Now, Suppose if c was absent, then the graph will be formed like this:



$$y_{inj} = \sum_{i=0}^n x_i w_{ij} = x_0 w_{0j} + x_1 w_{1j} + x_2 w_{2j} + \dots + x_n w_{nj}$$

$$= w_{0j} + \sum_{i=1}^n x_i w_{ij} \quad [\because x_0 = 1]$$

$$y_{inj} = b_j + \sum_{i=1}^n x_i w_{ij} \quad [\because w_{0j} = b_j]$$



The bias can be of two types:

- 1. Positive bias:** Helps in increasing the net input of the n/w.
- 2. Negative Bias:** Helps in decreasing the net input of the n/w.

Result of the bias effect is the output of the network can be varied.

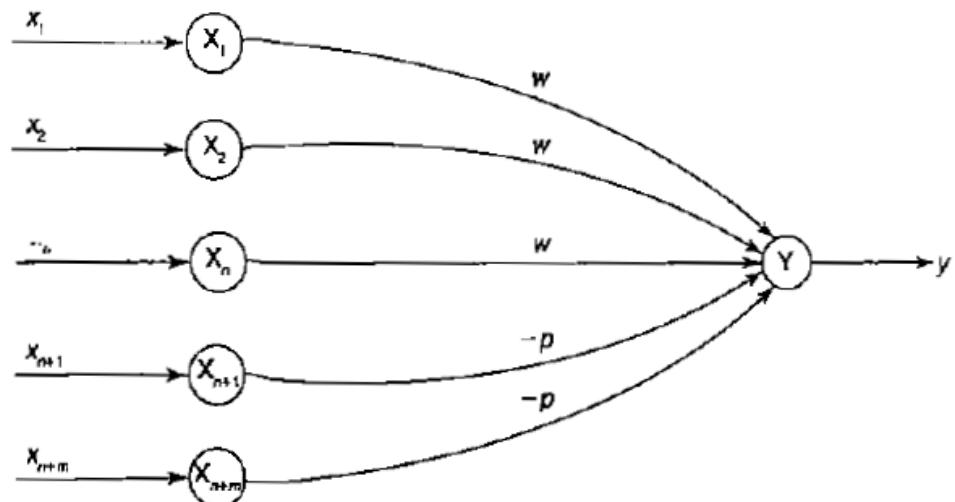
Contd..

- ❖ **Learning Rate:** Denoted by “ α ”.
- It is used to control the amount of weight adjustment at each step of training.
- The range is 0 to 1 and it determines the rate of learning at each step.

Notations

- x_i : Activation of unit X_i , input signal.
- y_j : Activation of unit Y_j , $y_j = f(y_{nj})$
- w_{ij} : Weight on connection from unit X_i to unit Y_j
- b_j : Bias acting on unit j . Bias has a constant activation of 1.
- W : Weight matrix, $W = \{w_{ij}\}$
- y_{nj} : Net input to unit Y_j given by $y_{nj} = b_j + \sum_i x_i w_{ij}$
- $\|x\|$: Norm of magnitude vector X .
- θ_j : Threshold for activation of neuron Y_j .
- S : Training input vector, $S = (s_1, \dots, s_i, \dots, s_n)$
- T : Training output vector, $T = (t_1, \dots, t_j, \dots, t_n)$
- X : Input vector, $X = (x_1, \dots, x_i, \dots, x_n)$
- Δw_{ij} : Change in weights given by $\Delta w_{ij} = w_{ij}(\text{new}) - w_{ij}(\text{old})$
- α : Learning rate; it controls the amount of weight adjustment at each step of training.
- ρ : Vigilance parameter, it controls the number of clusters in unsupervised networks.

McCulloch-Pitts Neuron



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

- The McCulloch-Pitts neuron was the earliest neural network discovered in 1943. It is usually called as **M-P neuron**.
The M-P neurons are connected by directed weighted paths.
It should be noted that the activation of a M-P neuron is binary, that is, at *any* time step the neuron may fire or may not fire.
- The weights associated with the communication links may be excitatory (weight is positive) or inhibitory (weight is negative).

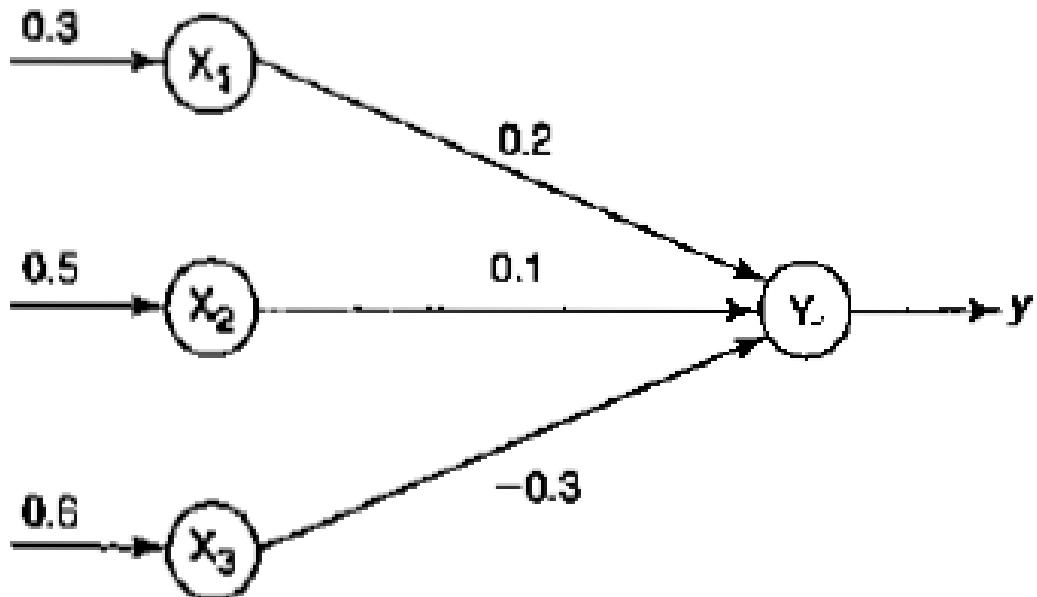
- All the excitatory connected weights entering into a particular neuron will have same weights.
- Also, it should be noted that any nonzero inhibitory input would prevent the neuron from firing.
- The M-P neurons are most widely used in the case of logic function.
- Since the firing of the output neuron is based upon the threshold, the activation function here is defined as:

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

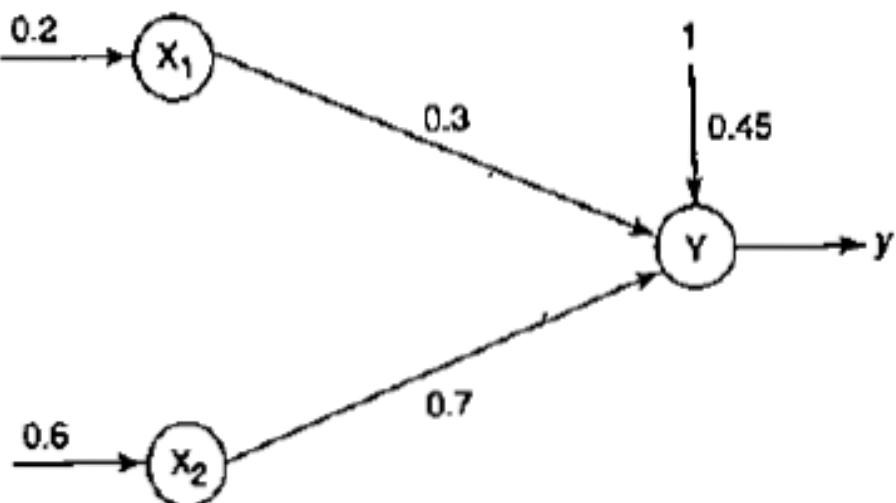
- For inhibition to be absolute, the threshold with the activation function should satisfy the following condition:

$$\theta > nw - p$$

For the network shown in Figure 1, calculate the net input to the output neuron.

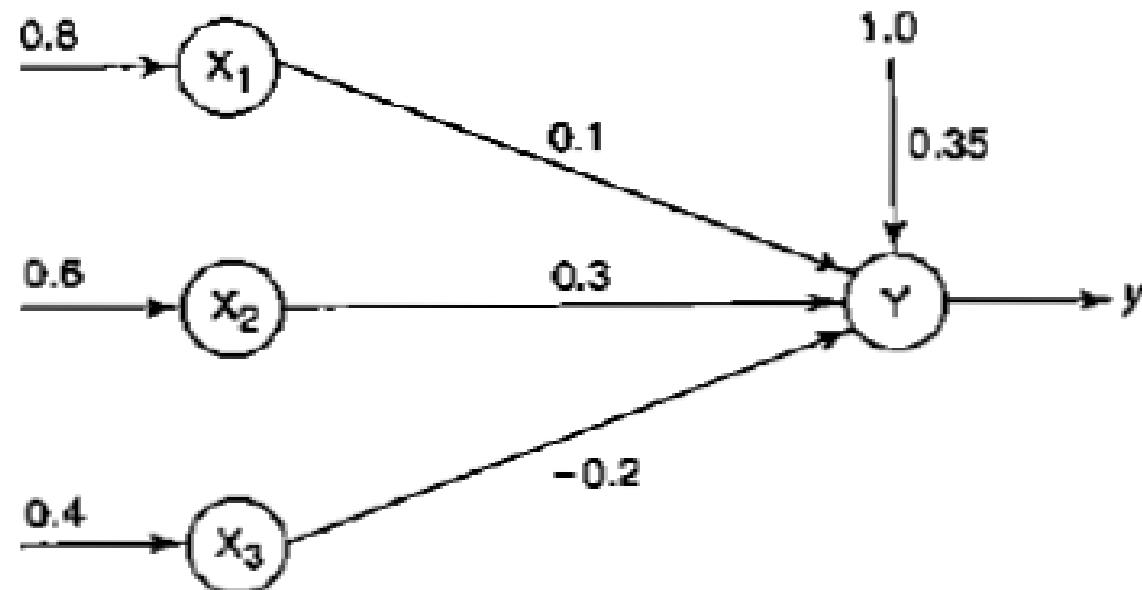


Calculate the net input for the network shown in Figure 2 with bias included in the network.



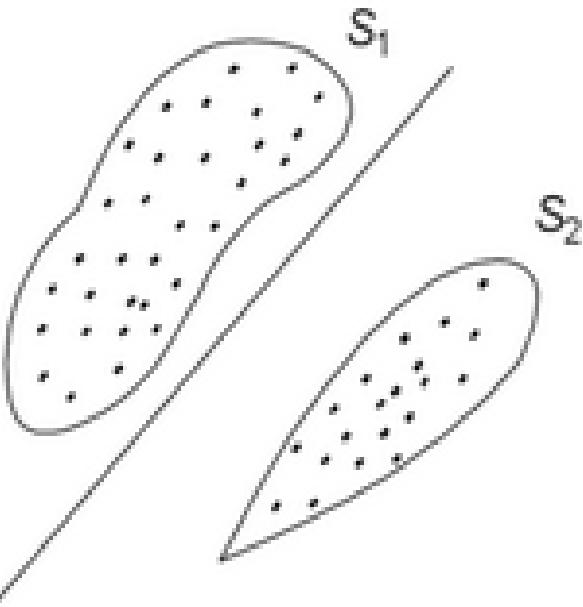
Obtain the output of the neuron Y for the network shown in Figure 3 using activation functions as: (i) binary sigmoidal and (ii) bipolar sigmoidal.

Implement AND function using McCulloch-Pitts neuron (take binary data).

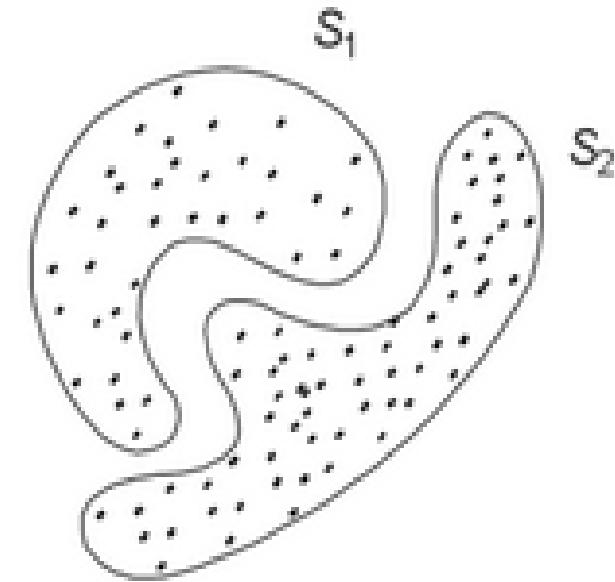


Linear Separability

- ANN does not give an exact solution for a nonlinear problem. However, it provides possible approximate solutions nonlinear problems.
- Linear separability, is the concept wherein the separation of the input space into regions is based on whether the network response is positive or negative.
- A decision line is drawn to separate positive and negative response.
- It is called as *decision making line or decision support line or linear separable line.*



(a) Linearly separable patterns



(b) Non-linearly separable patterns

The necessity of linear separability concept was felt to classify the patterns based upon their output responses.

Contd..

- The net input is usually calculated as : $y_{in} = b + \sum_{i=1}^n x_i w_i$
- There exist a clear boundary between the regions $y_{in} > 0$ and $y_{in} < 0$. This region is called *decision boundary* and is determined by the relation:

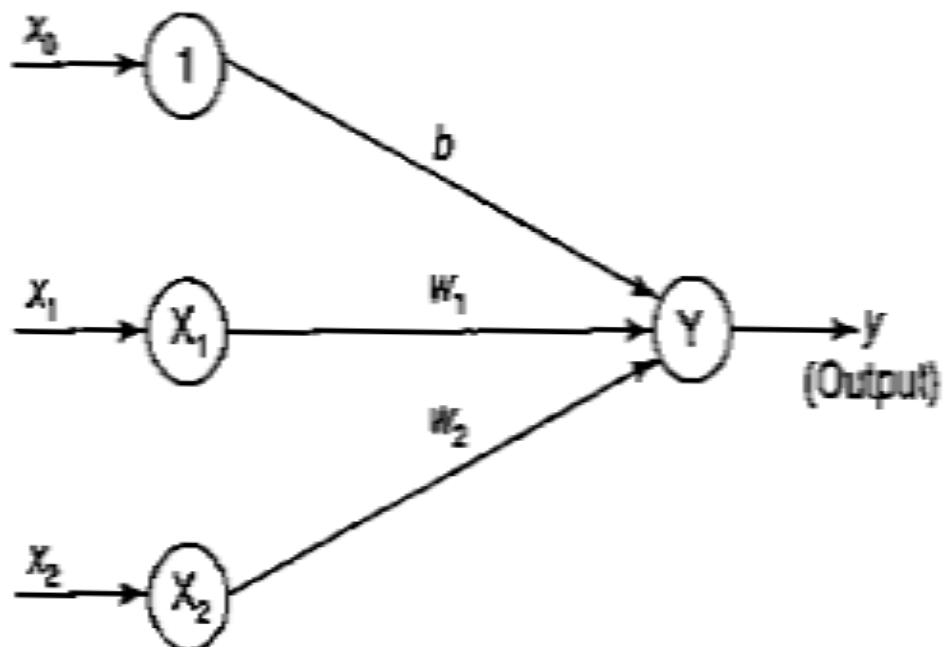
$$b + \sum_{i=1}^n x_i w_i = 0$$

- On the basis of the number of i/p units in the n/w, we can represent the above equation as line, hyperplane or plane.
- Hence, the linear separability of the n/w lies in the decision boundary line., i.e. +response lie in one side of the boundary and -ve on the other side.

Contd..

Consider a single layer n/w with bias included. The net input of the n/w is as follow:

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



The separating line for which the boundary lies between the values X_1 and X_2 so that the net gives a positive response on one side and negative response on other side, is given as:

$$b + x_1 w_1 + x_2 w_2 = 0$$

If weight W_2 is not equal to 0 then we get:

$$x_2 = -\frac{w_1}{w_2}x_1 - \frac{b}{w_2}$$

Thus, the requirement for the positive response of the net is:

$$b+x_1 w_1+x_2 w_2 > 0$$

Contd..

- During training process, the values of $w_1 > w_2$ and b , are determined so that the net will produce a positive (correct) response for the training data.
- If on the other hand, threshold value is being used, then the condition for obtaining the positive response from output unit is:

Net input received $> \theta$ (threshold)

$$y_{in} > \theta$$

$$x_1w_1 + x_2w_2 > \theta$$

The separating line equation will then be

$$x_1w_1 + x_2w_2 = \theta$$

$$x_2 = -\frac{w_1}{w_2}x_1 + \frac{\theta}{w_2} \quad (\text{with } w_2 \neq 0)$$

Hebb Network

- Donald Hebb stated in 1949 that in the brain, the learning is performed by the change in the Synaptic gap.
- Hebb explained it: "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 cell firing B is increased".
- According to Hebb rule, the weight vector is found increase proportionately to the product of the input and the learning signal.
- Here, the learning signal is equal to the neuron's o/p.

Contd..

- In Hebb learning, if two interconnected neurons are 'on' simultaneously then the weights associated with these neurons can be increased by the modification made in their synaptic gap (strength). The weight update in Hebb rule is given by:

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

- The Hebb rule is more suited for **bipolar data than binary data.**
- It has some limitations for binary data.

Algorithm

- 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: $y_1 = t$.
- Step 4: Weight adjustments and bias adjustment are performed:

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



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

Problems

- Design a Hebb net to implement logical AND function (use bipolar inputs and targets).
- Using the Hebb rule, find the weights required to perform the following classifications of the given input patterns as shown in the figure below. The pattern is a 3×3 matrix form in the squares. The “+” symbols represent the value “1” and empty sequences indicate “-1”. Consider “I” belongs to the members of class (so has target value 1) and “O” does not belong to the members of class (so has target value -1).

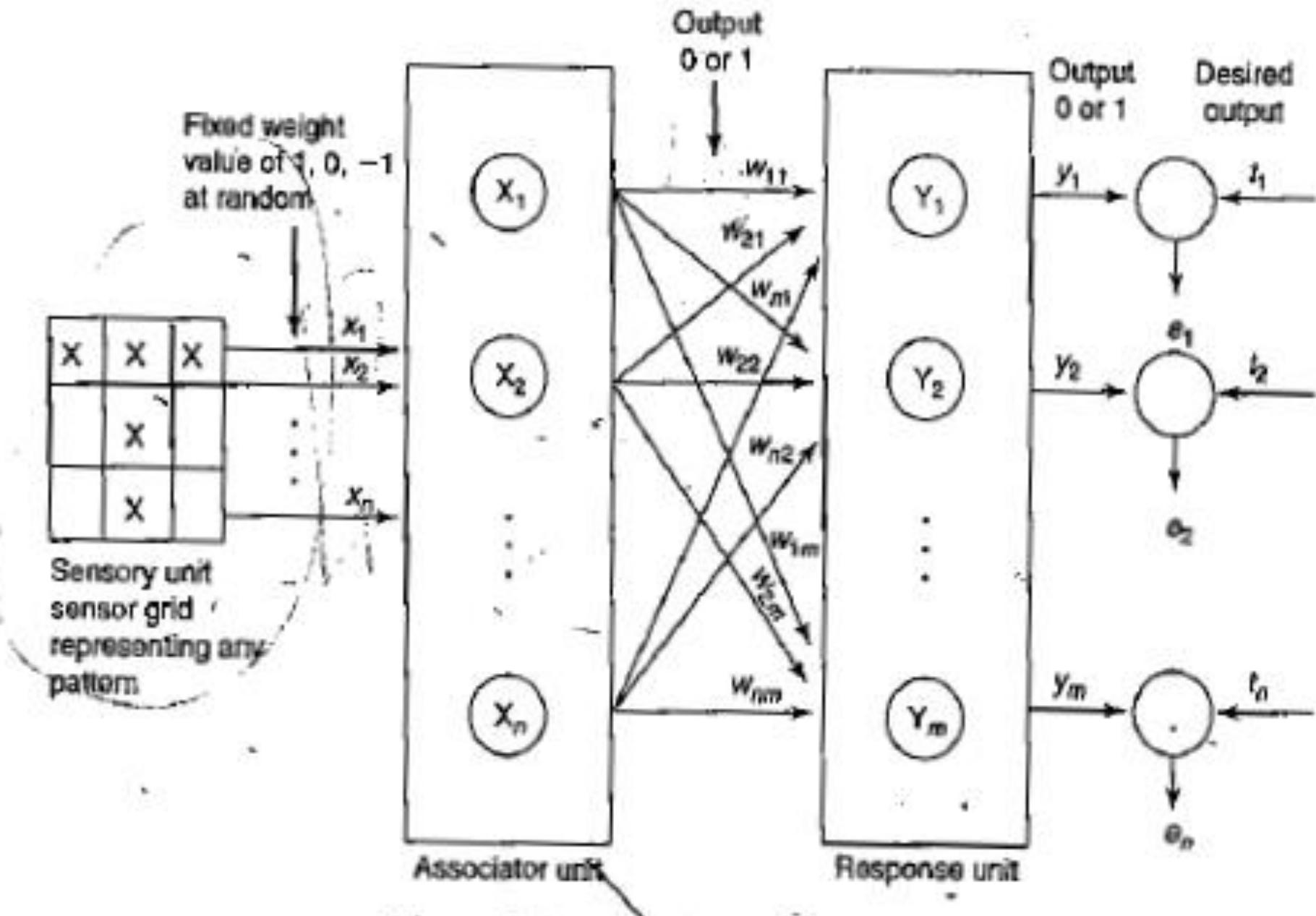
<table border="1"><tr><td>+</td><td>+</td><td>+</td></tr><tr><td></td><td>+</td><td></td></tr><tr><td>+</td><td>+</td><td>+</td></tr></table>	+	+	+		+		+	+	+	I
+	+	+								
	+									
+	+	+								
<table border="1"><tr><td>+</td><td>+</td><td>+</td></tr><tr><td>+</td><td></td><td>+</td></tr><tr><td>+</td><td>+</td><td>+</td></tr></table>	+	+	+	+		+	+	+	+	O
+	+	+								
+		+								
+	+	+								

Perceptron Networks and Backpropagation Network

Dr. Shruti Mishra
School of Computer Sc. & Engg. (SCOPE)

Introduction

- Frank Rosenblatt, an American psychologist, proposed the *classical perceptron* model in 1958. Further refined and carefully analyzed by Minsky and Papert (1969) — their model is referred to as the *perceptron* model.
- *Perceptron* networks come under single-layer feed-forward networks and are also called *simple perceptrons*.
- The key points to be noted in a perceptron network are:
- The perceptron network consists of three units, namely, sensory unit (input unit), associator unit (hidden unit), response unit (output unit).
- The sensory units are connected to associator units with fixed weights having values 1, 0 or -1, which are assigned at random.
- The binary activation function is used in sensory unit and associator unit.
- The response unit has an activation of 1, 0 or -1.



Contd..

- A **Perceptron** is an algorithm for supervised learning of binary classifiers.
- There are two types of **Perceptrons**: Single layer and Multilayer.
- Single layer **Perceptrons** can learn only linearly separable patterns.
- Multilayer Perceptrons or feedforward neural networks with two or more layers have the greater processing power.
- The Perceptron algorithm learns the weights for the input signals in order to draw a linear decision boundary.
- This enables you to distinguish between the two linearly separable classes +1 and -1.

Perceptron Learning Rule

- In case of the perceptron learning rule, the learning signal is the difference between the desired and the actual response of a neuron.
- Perceptron Learning Rule states that the algorithm would automatically learn the optimal weight coefficients.

Algorithm for Single Output Classes

Step 0: Initialize the weights and the bias (for easy calculation they can be set to zero). Also initialize the learning rate $\alpha (0 < \alpha \leq 1)$. For simplicity α is set to 1.

Step 1: Perform Steps 2–6 until the final stopping condition is false.

Step 2: Perform Steps 3–5 for each training pair indicated by *s,t*.

Step 3: The input layer containing input units is applied with identity activation functions:

$$x_i = s_i$$

Step 4: Calculate the output of the network. To do so, first obtain the net input:

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

where “*n*” is the number of input neurons in the input layer. Then apply activations over the net input calculated to obtain the output:

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

Step 5: Weight and bias adjustment: Compare the value of the actual (calculated) output and desired (target) output.

If $y \neq t$, then

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

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

else, we have

$$w_i(\text{new}) = w_i(\text{old})$$

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

Step 6: Train the network until there is no weight change. This is the stopping condition for the network. If this condition is not met, then start again from Step 2.

Algorithm for Multiple Output Classes

Step 0: Initialize the weights, biases and learning rate suitably.

Step 1: Check for stopping condition; if it is false, perform Steps 2–6.

Step 2: Perform Steps 3–5 for each bipolar or binary training vector pair $s:t$.

Step 3: Set activation (identity) of each input unit $i = 1$ to n :

$$x_i := s_i$$

Step 4: Calculate output response of each output unit $j = 1$ to m . First, the net input is calculated as

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

Then activations are applied over the net input to calculate the output response:

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

Step 5: Make adjustment in weights and bias for $j = 1$ to m and $i = 1$ to n .

If $t_j \neq y_j$, then

$$w_{ij}(\text{new}) = w_{ij}(\text{old}) + \alpha t_j x_i$$

$$b_j(\text{new}) = b_j(\text{old}) + \alpha t_j$$

else, we have

$$w_{ij}(\text{new}) = w_{ij}(\text{old})$$

$$b_j(\text{new}) = b_j(\text{old})$$

Step 6: Test for the stopping condition, i.e., if there is no change in weights then stop the training process, else start again from Step 2.

Exercise

- Implement AND function using perceptron networks for bipolar inputs and targets.

For the first input pattern, $x_1 = 1$, $x_2 = 1$ and $t = 1$, with weights and bias, $w_1 = 0$, $w_2 = 0$ and $b = 0$:

- Calculate the net input

$$\begin{aligned}y_n &= b + x_1 w + x_2 w \\&= 0 + 1 \times 0 + 1 \times 0 = 0\end{aligned}$$

- The output y is computed by applying activations over the net input calculated:

$$y = f(y_n) = \begin{cases} 1 & \text{if } y_n > 0 \\ 0 & \text{if } y_n = 0 \\ -1 & \text{if } y_n < 0 \end{cases}$$

Here we have taken $\theta = 0$. Hence, when, $y_n = 0$, $y = 0$.

Check whether $t = y$. Here, $t = 1$ and $y = 0$, so $t \neq y$, hence weight updation takes place:

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

$$w_1(\text{new}) = w_1(\text{old}) + \alpha \alpha_1 = 0 + 1 \times 1 \times 1 = 1$$

$$w_2(\text{new}) = w_2(\text{old}) + \alpha \alpha_2 = 0 + 1 \times 1 \times 1 = 1$$

$$b(\text{new}) = b(\text{old}) + \alpha t = 0 + 1 \times 1 = 1$$

Here, the change in weights are

$$\Delta w_1 = \alpha \alpha_1;$$

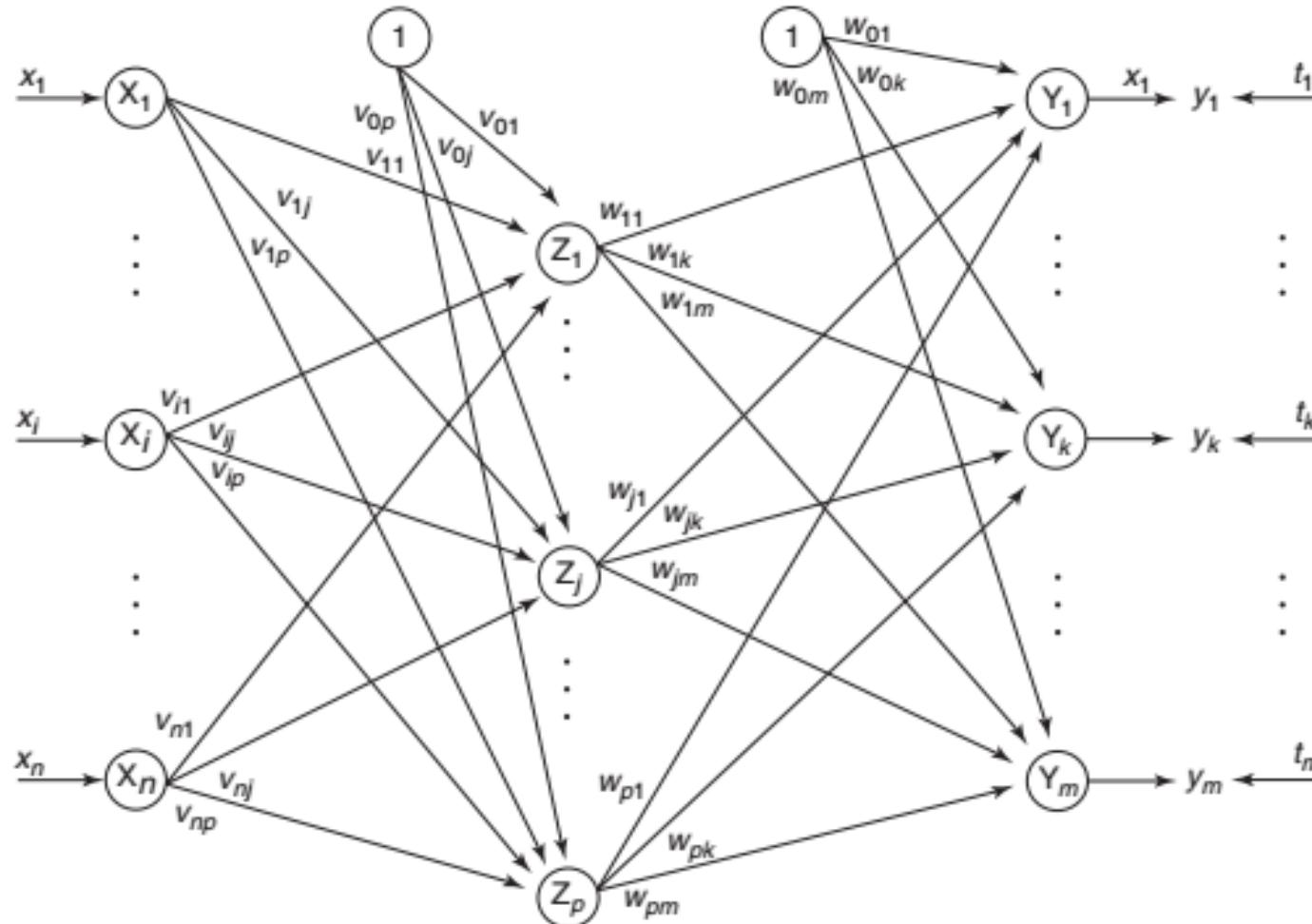
$$\Delta w_2 = \alpha \alpha_2;$$

$$\Delta b = \alpha t$$

Input			Target (t)	Net input (y_{in})	Calculated output (y)	Weight changes			Weights		
x_1	x_2	1				Δw_1	Δw_2	Δb	w_1	w_2	b
EPOCH-1											
1	1	1	1	0	0	1	1	1	1	1	1
1	-1	1	-1	1	1	-1	1	-1	0	2	0
-1	1	1	-1	2	1	+1	-1	-1	1	1	-1
-1	-1	1	-1	-3	-1	0	0	0	1	1	-1
EPOCH-2											
1	1	1	1	1	1	0	0	0	1	1	-1
1	-1	1	-1	-1	-1	0	0	0	1	1	-1
-1	1	1	-1	-1	-1	0	0	0	1	1	-1
-1	-1	1	-1	-3	-1	0	0	0	1	1	-1

Backpropagation Algorithm

- It is a widely used algorithm in training feedforward neural networks for supervised learning.



Contd..

x = input training vector ($x_1, \dots, x_i, \dots, x_n$)

t = target output vector ($t_1, \dots, t_k, \dots, t_m$)

α = learning rate parameter

x_i = input unit i . (Since the input layer uses identity activation function, the input and output signals here are same.)

v_{0j} = bias on j th hidden unit

w_{0k} = bias on k th output unit

z_j = hidden unit j . The net input to z_j is

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

and the output is

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

y_k = output unit k . The net input to y_k is

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

and the output is

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

δ_k = error correction weight adjustment for w_{jk} that is due to an error at output unit y_k , which is back-propagated to the hidden units that feed into unit y_k

δ_j = error correction weight adjustment for v_{ij} that is due to the back-propagation of error to the hidden unit z_j .

Training Algorithm

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 (Phase I):

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:

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

Contd..

and apply the activation function to compute output signal

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

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:

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

The derivative $f'(y_{ik})$ can be calculated as in Section 2.3.3. 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.

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

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

Contd..

Step 7: Each hidden unit ($z_j, j = 1 \text{ to } p$) 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 discussed in Section 2.3.3 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:

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

Contd..

Weight and bias updation (Phase III):

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

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

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

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

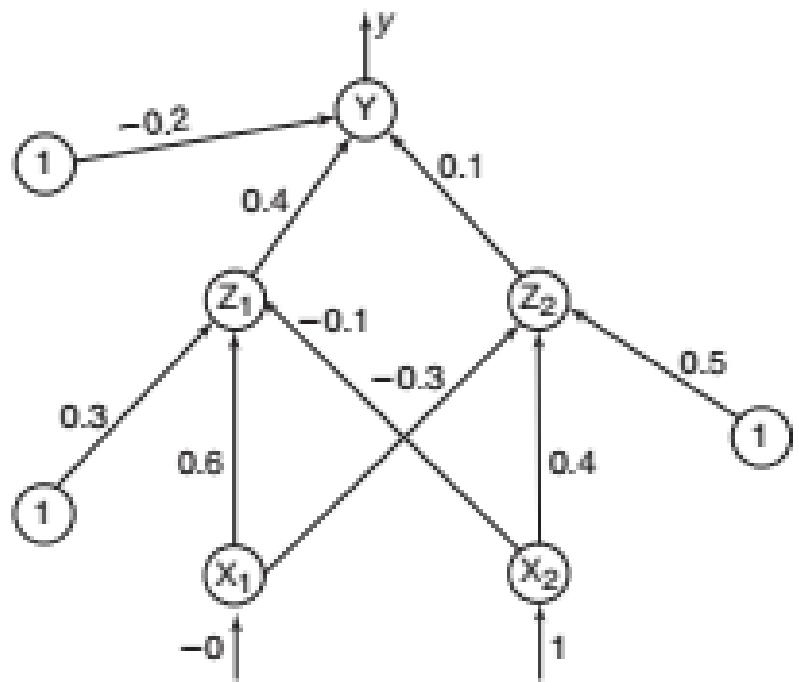
$$v_{ij}(\text{new}) = v_{ij}(\text{old}) + \Delta v_{ij}$$

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

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.

Example

Using back-propagation network, find the new weights for the net shown in Figure 12. It is presented with the input pattern $[0, 1]$ and the target output is 1. Use a learning rate $\alpha = 0.25$ and binary sigmoidal activation function.



Solution:

The initial weights are:

$$[v_{11} \ v_{21} \ v_{01}] = [0.6 \ -0.1 \ 0.3]$$

$$[v_{12} \ v_{22} \ v_{02}] = [-0.3 \ 0.4 \ 0.5]$$

$$[w_1 \ w_2 \ w_0] = [0.4 \ 0.1 \ -0.2]$$

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

Contd..

Given the output sample $[x_1, x_2] = [0, 1]$ and target $t=1$,

- Calculate the net input: For z_1 layer

$$\begin{aligned} z_{in1} &= v_{01} + x_1 v_{11} + x_2 v_{21} \\ &= 0.3 + 0 \times 0.6 + 1 \times -0.1 = 0.2 \end{aligned}$$

For z_2 layer

$$\begin{aligned} z_{in2} &= v_{02} + x_1 v_{12} + x_2 v_{22} \\ &= 0.5 + 0 \times -0.3 + 1 \times 0.4 = 0.9 \end{aligned}$$

Applying activation to calculate the output, we obtain

$$z_1 = f(z_{in1}) = \frac{1}{1+e^{-z_{in1}}} = \frac{1}{1+e^{-0.2}} = 0.5498$$

$$z_2 = f(z_{in2}) = \frac{1}{1+e^{-z_{in2}}} = \frac{1}{1+e^{-0.9}} = 0.7109$$

- Calculate the net input entering the output layer.
For y layer

$$\begin{aligned} y_{in} &= w_0 + z_1 w_1 + z_2 w_2 \\ &= -0.2 + 0.5498 \times 0.4 + 0.7109 \times 0.1 \\ &= 0.09101 \end{aligned}$$

Applying activations to calculate the output, we obtain

$$y = f(y_{in}) = \frac{1}{1+e^{-y_{in}}} = \frac{1}{1+e^{-0.09101}} = 0.5227$$

- Compute the error portion δ_k :

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

Now

$$f'(y_{in}) = f(y_{in})[1 - f(y_{in})] = 0.5227[1 - 0.5227]$$

$$f'(y_{in}) = 0.2495$$

This implies

$$\delta_1 = (1 - 0.5227)(0.2495) = 0.1191$$

Find the changes in weights between hidden and output layer:

$$\begin{aligned} \Delta w_1 &= \alpha \delta_1 z_1 = 0.25 \times 0.1191 \times 0.5498 \\ &= 0.0164 \end{aligned}$$

Contd..

$$\begin{aligned}\Delta w_2 &= \alpha \delta_1 z_2 = 0.25 \times 0.1191 \times 0.7109 \\ &= 0.02117\end{aligned}$$

$$\Delta w_0 = \alpha \delta_1 = 0.25 \times 0.1191 = 0.02978$$

- Compute the error portion δ_j between input and hidden layer ($j=1$ to 2):

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

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

$$\delta_{inj} = \delta_1 w_{j1} \quad [\because \text{only one output neuron}]$$

$$\Rightarrow \delta_{in1} = \delta_1 w_{11} = 0.1191 \times 0.4 = 0.04764$$

$$\Rightarrow \delta_{in2} = \delta_1 w_{21} = 0.1191 \times 0.1 = 0.01191$$

$$\text{Error, } \delta_1 = \delta_{in1} f'(z_{in1})$$

$$\begin{aligned}f'(z_{in1}) &= f(z_{in1}) [1 - f(z_{in1})] \\ &= 0.5498 [1 - 0.5498] = 0.2475\end{aligned}$$

$$\begin{aligned}\delta_1 &= \delta_{in1} f'(z_{in1}) \\ &= 0.04764 \times 0.2475 = 0.0118\end{aligned}$$

$$\text{Error, } \delta_2 = \delta_{in2} f'(z_{in2})$$

$$\begin{aligned}f'(z_{in2}) &= f(z_{in2}) [1 - f(z_{in2})] \\ &= 0.7109 [1 - 0.7109] = 0.2055 \\ \delta_2 &= \delta_{in2} f'(z_{in2}) \\ &= 0.01191 \times 0.2055 = 0.00245\end{aligned}$$

Now find the changes in weights between input and hidden layer:

$$\Delta v_{11} = \alpha \delta_1 x_1 = 0.25 \times 0.0118 \times 0 = 0$$

$$\Delta v_{21} = \alpha \delta_1 x_2 = 0.25 \times 0.0118 \times 1 = 0.00295$$

$$\Delta v_{01} = \alpha \delta_1 = 0.25 \times 0.0118 = 0.00295$$

$$\Delta v_{12} = \alpha \delta_2 x_1 = 0.25 \times 0.00245 \times 0 = 0$$

$$\Delta v_{22} = \alpha \delta_2 x_2 = 0.25 \times 0.00245 \times 1 = 0.0006125$$

$$\Delta v_{02} = \alpha \delta_2 = 0.25 \times 0.00245 = 0.0006125$$

Compute the final weights of the network:

$$v_{11}(\text{new}) = v_{11}(\text{old}) + \Delta v_{11} = 0.6 + 0 = 0.6$$

$$v_{12}(\text{new}) = v_{12}(\text{old}) + \Delta v_{12} = -0.3 + 0 = -0.3$$

$$v_{21}(\text{new}) = v_{21}(\text{old}) + \Delta v_{21}$$

Contd..

$$= -0.1 + 0.00295 = -0.09705$$

$$\begin{aligned}v_{22}(\text{new}) &= v_{22}(\text{old}) + \Delta v_{22} \\&= 0.4 + 0.0006125 = 0.4006125\end{aligned}$$

$$\begin{aligned}w_1(\text{new}) &= w_1(\text{old}) + \Delta w_1 = 0.4 + 0.0164 \\&= 0.4164\end{aligned}$$

$$\begin{aligned}w_2(\text{new}) &= w_2(\text{old}) + \Delta w_2 = 0.1 + 0.02117 \\&= 0.12117\end{aligned}$$

$$\begin{aligned}v_{01}(\text{new}) &= v_{01}(\text{old}) + \Delta v_{01} = 0.3 + 0.00295 \\&= 0.30295\end{aligned}$$

$$\begin{aligned}v_{02}(\text{new}) &= v_{02}(\text{old}) + \Delta v_{02} \\&= 0.5 + 0.0006125 = 0.5006125\end{aligned}$$

$$\begin{aligned}w_0(\text{new}) &= w_0(\text{old}) + \Delta w_0 = -0.2 + 0.02978 \\&= -0.17022\end{aligned}$$

Associative Memory

Dr. Shruti Mishra
School of Computer Sc. & Engg. (SCOPE)

Associate Memory Network

- These kinds of neural networks work on the basis of pattern association, which means they can store different patterns and at the time of giving an output they can produce one of the stored patterns by matching them with the given input pattern.
- These types of memories are also called **Content-Addressable Memory (CAM)**.
- Associative memory makes a parallel search with the stored patterns as data files.
- Following are the two types of associative memories we can observe –
 - Auto Associative Memory
 - Hetero Associative memory

Contd..

If there exist vectors, say, $x = (x_1, x_2, \dots, x_n)^T$ and $x' = (x'_1, x'_2, \dots, x'_n)^T$, then the hamming distance (HD) is defined as the number of mismatched components of x and x' vectors, i.e.,

$$\text{HD}(x, x') = \begin{cases} \sum_{i=1}^n |x_i - x'_i| & \text{if } x_i, x'_i \in [0, 1] \\ \frac{1}{2} \sum_{i=1}^n |x_i - x'_i| & \text{if } x_i, x'_i \in [-1, 1] \end{cases}$$

Training Algorithms for Pattern Association: Hebb Rule

Step 0: Set all the initial weights to zero, i.e.,

$$w_{ij} = 0 \quad (i = 1 \text{ to } n, j = 1 \text{ to } m)$$

Step 1: For each training target input output vector pairs $s:t$, perform Steps 2–4.

Step 2: Activate the input layer units to current training input,

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

Step 3: Activate the output layer units to current target output,

$$y_j = t_j \quad (\text{for } j = 1 \text{ to } m)$$

Step 4: Start the weight adjustment

$$w_{ij}(\text{new}) = w_{ij}(\text{old}) + x_i y_j \quad (\text{for } i = 1 \text{ to } n, j = 1 \text{ to } m)$$

Training Algorithms for Pattern Association: Outer Products Rule

Outer products rule is an alternative method for finding weights of an associative net. This is depicted as follows:

$$\text{Input} \Rightarrow s = (s_1, \dots, s_i, \dots, s_n)$$

$$\text{Output} \Rightarrow t = (t_1, \dots, t_j, \dots, t_m)$$

For storing a set of associations, $s(p): t(p)$, $p = 1$ to P , wherein,

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

$$t(p) = (t_1(p), \dots, t_j(p), \dots, t_m(p))$$

the weight matrix $W = \{w_{ij}\}$ can be given as

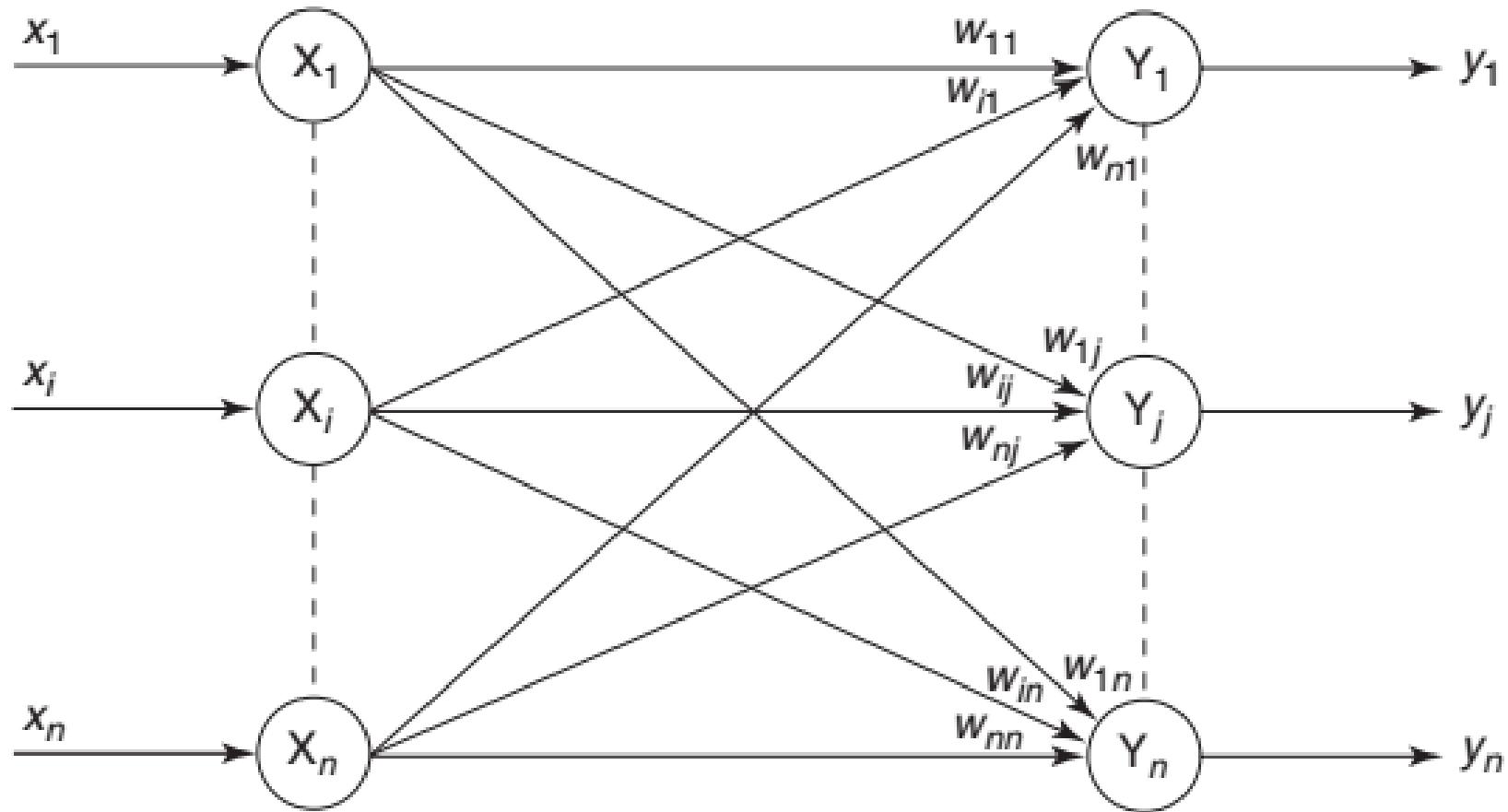
$$w_{ij} = \sum_{p=1}^P s_i^T(p)t_j(p)$$

This can also be rewritten as

$$W = \sum_{p=1}^P s^T(p)t(p)$$

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.
- For training, this network is using the Hebb learning rule and outer product rule.
- Testing algorithm, shows whether the given input vector is a “known” vector or “unknown” vector.
- It is known if the network produces a pattern of activation on the o/p units which is same as one of the vectors stored in it.
- In case of this network, *the weights on the diagonal can be set to zero*. This is called as **auto-associative network with no self-connection**.



Training Algorithm

Step 0: Initialize all the weights to zero,

$$w_{ij} = 0 \quad (i = 1 \text{ to } n, j = 1 \text{ to } n)$$

Step 1: For each of the vector that has to be stored perform Steps 2–4.

Step 2: Activate each of the input unit,

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

Step 3: Activate each of the output unit,

$$y_j = s_j \quad (j = 1 \text{ to } n)$$

Step 4: Adjust the weights,

$$w_{ij}(\text{new}) = w_{ij}(\text{old}) + x_i y_j$$

The weights can also be determined by the formula

$$W = \sum_{p=1}^P s^T(p) s(p)$$

Testing Algorithm

- Step 0:** Set the weights obtained for Hebb's rule or outer products.
- Step 1:** For each of the testing input vector presented perform Steps 2–4.
- Step 2:** Set the activations of the input units equal to that of input vector.
- Step 3:** Calculate the net input to each output unit $j = 1$ to n :

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

- Step 4:** Calculate the output by applying the activation over the net input:

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

Train the autoassociative network for input vector $[-1 \ 1 \ 1 \ 1]$ and also test the network for the same input vector. Test the autoassociative network with one missing, one mistake, two missing and two mistake entries in test vector.

Solution: The input vector is $x = [-1 \ 1 \ 1 \ 1]$. The weight vector is

$$W = \sum s^T(p)s(p) = \begin{bmatrix} -1 \\ 1 \\ 1 \\ 1 \end{bmatrix}_{4 \times 1} [-1 \ 1 \ 1 \ 1]_{1 \times 4}$$

$$= \begin{bmatrix} 1 & -1 & -1 & -1 \\ -1 & 1 & 1 & 1 \\ -1 & 1 & 1 & 1 \\ -1 & 1 & 1 & 1 \end{bmatrix}_{4 \times 4}$$

Testing the network with same input vector: The test input is $[-1 \ 1 \ 1 \ 1]$. The weight obtained above is used as the initial weight here. Computing the input, we get

$$y_{inj} = x \cdot W = [-1 \ 1 \ 1 \ 1] \begin{bmatrix} 1 & -1 & -1 & -1 \\ -1 & 1 & 1 & 1 \\ -1 & 1 & 1 & 1 \\ -1 & 1 & 1 & 1 \end{bmatrix} = [-4 \ 4 \ 4 \ 4]$$

Applying activations over the net input to calculate the output, we have

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

Hence the correct response is obtained.

Testing the network with one missing entry

- Test input $x = [0 \ 1 \ 1 \ 1]$. Computing the input, we get

$$y_{inj} = x \cdot W = [0 \ 1 \ 1 \ 1] \begin{bmatrix} 1 & -1 & -1 & -1 \\ -1 & 1 & 1 & 1 \\ -1 & 1 & 1 & 1 \\ -1 & 1 & 1 & 1 \end{bmatrix} = [-3 \ 3 \ 3 \ 3]$$

Applying the activations, we get $y_j = [-1 \ 1 \ 1 \ 1]$ which is the correct response.

- Test input $x = [-1 \ 1 \ 0 \ 1]$. Computing net input, we obtain

$$y_{inj} = x \cdot W$$

$$\begin{aligned} &= [-1 \ 1 \ 0 \ 1] \begin{bmatrix} 1 & -1 & -1 & -1 \\ -1 & 1 & 1 & 1 \\ -1 & 1 & 1 & 1 \\ -1 & 1 & 1 & 1 \end{bmatrix} \\ &= [-3 \ 3 \ 3 \ 3] \end{aligned}$$

Applying the activations, we get $y_j = [-1 \ 1 \ 1 \ 1]$ which is the correct response.

Testing the network with one mistake entry

- Test input $x = [-1 \ -1 \ 1 \ 1]$. Computing net input, we get

$$y_{inj} = x \cdot W$$

$$\begin{aligned} &= [-1 \ -1 \ 1 \ 1] \begin{bmatrix} 1 & -1 & -1 & -1 \\ -1 & 1 & 1 & 1 \\ -1 & 1 & 1 & 1 \\ -1 & 1 & 1 & 1 \end{bmatrix} \\ &= [-2 \ 2 \ 2 \ 2] \end{aligned}$$

Applying the activations, we get $y_j = [-1 \ 1 \ 1 \ 1]$ which is the correct response.

- Test input $x = [1 \ 1 \ 1 \ 1]$. Computing net input, we get

$$\begin{aligned} y_{inj} &= x \cdot W = [1 \ 1 \ 1 \ 1] \begin{bmatrix} 1 & -1 & -1 & -1 \\ -1 & 1 & 1 & 1 \\ -1 & 1 & 1 & 1 \\ -1 & 1 & 1 & 1 \end{bmatrix} \\ &= [-2 \ 2 \ 2 \ 2] \end{aligned}$$

Applying the activations, we get $y_j = [-1 \ 1 \ 1 \ 1]$ which is the correct response.

Testing the network with two missing entry

- Test input $x = [0 \ 0 \ 1 \ 1]$. Computing net input, we get

$$y_{\text{inj}} = x \cdot W = [0 \ 0 \ 1 \ 1] \begin{bmatrix} 1 & -1 & -1 & -1 \\ -1 & 1 & 1 & 1 \\ -1 & 1 & 1 & 1 \\ -1 & 1 & 1 & 1 \end{bmatrix} = [-2 \ 2 \ 2 \ 2]$$

Applying the activations, we get $y_j = [-1 \ 1 \ 1 \ 1]$ which is the correct response.

- Test input $x = [-1 \ 0 \ 0 \ 1]$. Computing net input, we obtain

$$y_{\text{inj}} = x \cdot W = [-1 \ 0 \ 0 \ 1] \begin{bmatrix} 1 & -1 & -1 & -1 \\ -1 & 1 & 1 & 1 \\ -1 & 1 & 1 & 1 \\ -1 & 1 & 1 & 1 \end{bmatrix} = [-2 \ 2 \ 2 \ 2]$$

Applying the activations, we get $y_j = [-1 \ 1 \ 1 \ 1]$ which is the correct response.

Testing the network with two mistaken entry

- Test input $x = [-1 \ -1 \ -1 \ 1]$. Computing net input, we obtain

$$\begin{aligned} y_{\text{inj}} &= x \cdot W \\ &= [-1 \ -1 \ -1 \ 1] \begin{bmatrix} 1 & -1 & -1 & -1 \\ -1 & 1 & 1 & 1 \\ -1 & 1 & 1 & 1 \\ -1 & 1 & 1 & 1 \end{bmatrix} \\ &= [0 \ 0 \ 0 \ 0] \end{aligned}$$

Applying the activations over the net input, we get $y_j = [0 \ 0 \ 0 \ 0]$ which is the incorrect response. Thus, the network with two mistakes is not recognized.

Hetroassociative 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 operation

Testing Algorithm

Step 0: Initialize the weights from the training algorithm.

Step 1: Perform Steps 2–4 for each input vector presented.

Step 2: Set the activation for input layer units equal to that of the current input vector given, x_i .

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

$$y_{inj} = \sum_{i=1}^n x_i w_{ij} \quad (j = 1 \text{ to } m)$$

Step 4: Determine the activations of the output units over the calculated net input:

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

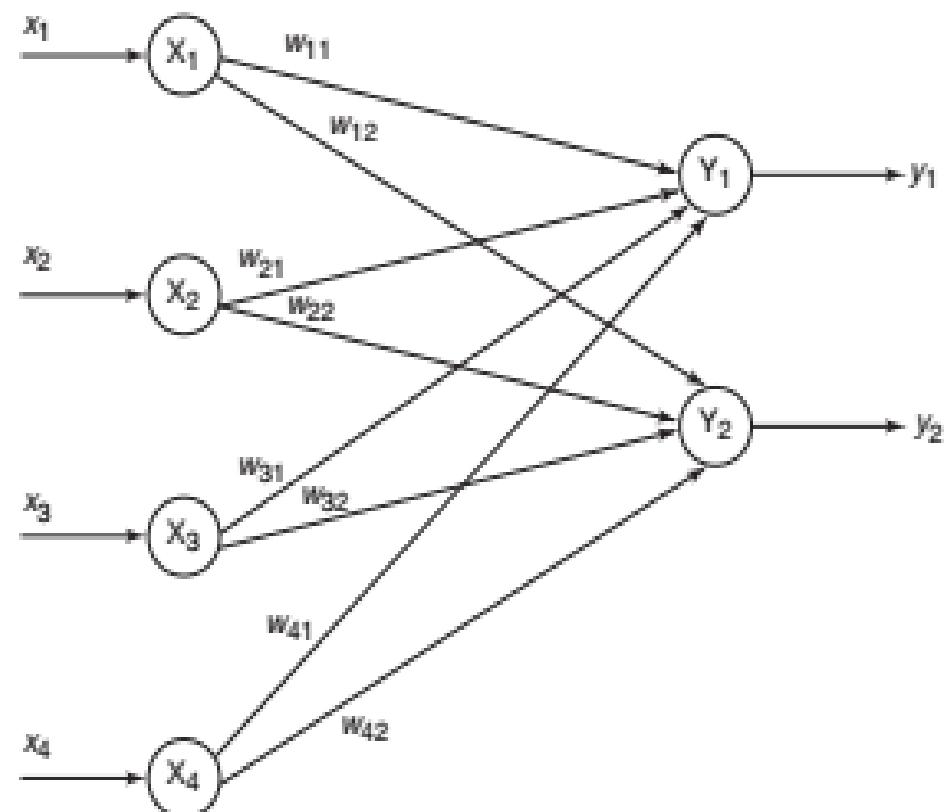
Thus, the output vector y obtained gives the pattern associated with the input vector x .

Note: Heteroassociative memory is not an iterative memory network. If the responses of the net are binary, then the activation function to be used is

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

Train a heteroassociative memory network using Hebb rule to store input row vector $s = (s_1, s_2, s_3, s_4)$ to the output row vector $t = (t_1, t_2)$. The vector pairs are given in Table 1.

Input targets	s_1	s_2	s_3	s_4	t_1	t_2
1 st	1	0	1	0	1	0
2 nd	1	0	0	1	1	0
3 rd	1	1	0	0	0	1
4 th	0	0	1	1	0	1



For 1st input vector:

Step 0: Initialize the weights, the initial weights are taken as zero.

Step 1: For first pair $(1, 0, 1, 0):(1, 0)$

Step 2: Set the activations of input units:

$$x_1 = 1, \quad x_2 = 0, \quad x_3 = 1, \quad x_4 = 0$$

Step 3: Set the activations of output unit:

$$y_1 = 1, \quad y_2 = 0$$

Step 4: Update the weights,

$$w_{\bar{y}}(\text{new}) = w_{\bar{y}}(\text{old}) + x_i y_j$$

$$w_{11}(\text{new}) = w_{11}(\text{old}) + x_1 y_1 = 0 + 1 \times 1 = 1$$

$$w_{21}(\text{new}) = w_{21}(\text{old}) + x_2 y_1 = 0 + 0 \times 1 = 0$$

$$w_{31}(\text{new}) = w_{31}(\text{old}) + x_3 y_1 = 0 + 1 \times 1 = 1$$

$$w_{41}(\text{new}) = w_{41}(\text{old}) + x_4 y_1 = 0 + 0 \times 1 = 0$$

$$w_{12}(\text{new}) = w_{12}(\text{old}) + x_1 y_2 = 0 + 1 \times 0 = 0$$

$$w_{22}(\text{new}) = w_{22}(\text{old}) + x_2 y_2 = 0 + 0 \times 0 = 0$$

$$w_{32}(\text{new}) = w_{32}(\text{old}) + x_3 y_2 = 0 + 1 \times 0 = 0$$

$$w_{42}(\text{new}) = w_{42}(\text{old}) + x_4 y_2 = 0 + 0 \times 0 = 0$$

For 2nd input vector:

The input-output vector pair is $(1, 0, 0, 1):(1, 0)$

$$x_1 = 1, \quad x_2 = 0, \quad x_3 = 0, \quad x_4 = 1,$$

$$y_1 = 1, \quad y_2 = 0$$

The final weights obtained for the input vector pair is used as initial weight here:

$$w_{11}(\text{new}) = w_{11}(\text{old}) + x_1 y_1 = 1 + 1 \times 1 = 2$$

$$w_{41}(\text{new}) = w_{41}(\text{old}) + x_4 y_1 = 0 + 1 \times 1 = 1$$

Since $x_2 = x_3 = y_2 = 0$, the other weights remains the same.

The final weights after second input vector is presented are

$$w_{11} = 2, w_{21} = 0, w_{31} = 1, w_{41} = 1$$

$$w_{12} = 0, w_{22} = 0, w_{32} = 0, w_{42} = 0$$

For 3rd input vector:

The input-output vector pair is $(1, 1, 0, 0):(0, 1)$

$$x_1 = 1, x_2 = 1, x_3 = 0, x_4 = 0, y_1 = 0, y_2 = 1$$

Training, using Hebb rule, evolves the final weights as follows:

Since $y_1 = 0$, the weights of y_1 are going to the same.
Computing the weights of y_2 unit, we obtain

$$w_{12}(\text{new}) = w_{12}(\text{old}) + x_1 y_2 = 0 + 1 \times 1 = 1$$

$$w_{22}(\text{new}) = w_{22}(\text{old}) + x_2 y_2 = 0 + 1 \times 1 = 1$$

$$w_{32}(\text{new}) = w_{32}(\text{old}) + x_3 y_2 = 0 + 0 \times 1 = 0$$

$$w_{42}(\text{new}) = w_{42}(\text{old}) + x_4 y_2 = 0 + 0 \times 1 = 0$$

The final weights after presenting third input vector are

$$w_{11} = 2, w_{21} = 0, w_{31} = 1, w_{41} = 1$$

$$w_{12} = 1, w_{22} = 1, w_{32} = 0, w_{42} = 0$$

For 4th input vector:

The input-output vector pair is $(0, 0, 1, 1):(0, 1)$

$$x_1 = 0, x_2 = 0, x_3 = 1, x_4 = 1, y_1 = 0, y_2 = 1$$

The weights are given by

$$w_{32}(\text{new}) = w_{32}(\text{old}) + x_3 y_2 = 0 + 1 \times 1 = 1$$

$$w_{42}(\text{new}) = w_{42}(\text{old}) + x_4 y_2 = 0 + 1 \times 1 = 1$$

Since, $x_1 = x_2 = y_1 = 0$, the other weights remains the same. The final weights after presenting the fourth input vector are

$$w_{11} = 2, w_{21} = 0, w_{31} = 1, w_{41} = 1$$

$$w_{12} = 1, w_{22} = 1, w_{32} = 1, w_{42} = 1$$

Thus, the weight matrix in matrix form is

$$W = \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \\ w_{41} & w_{42} \end{bmatrix} = \begin{bmatrix} 2 & 1 \\ 0 & 1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix}$$

Using Outer Products Rule

$$W = \sum_{p=1}^P s^T(p) t(p)$$

For 1st pair: The input and output vectors are $s = (1 0 1 0)$, $t = (1 0)$. For $p = 1$,

$$s^T(p) t(p) = s^T(1) t(1)$$

$$= \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}_{4 \times 1} [1 \ 0]_{1 \times 2} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 0 \end{bmatrix}_{4 \times 2}$$

For 2nd pair: The input and output vectors are $s = (1 0 0 1)$, $t = (1 0)$. For $p = 2$,

$$s^T(p) t(p) = s^T(2) t(2)$$

$$= \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}_{4 \times 1} [1 \ 0]_{1 \times 2} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \end{bmatrix}_{4 \times 2}$$

For 3rd pair: The input and output vectors are $s = (1 1 0 0)$, $t = (0 1)$. For $p = 3$,

$$s^T(p) t(p) = s^T(3) t(3)$$

$$= \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}_{4 \times 1} [0 \ 1]_{1 \times 2} = \begin{bmatrix} 0 & 1 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}_{4 \times 2}$$

For 4th pair: The input and output vectors are $s = (0 0 1 1)$, $t = (0 1)$. For $p = 4$,

$$s^T(p) t(p) = s^T(4) t(4)$$

$$= \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}_{4 \times 1} [0 \ 1]_{1 \times 2} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}_{4 \times 2}$$

The final weight matrix is the summation of all the individual weight matrices obtained for each pair.

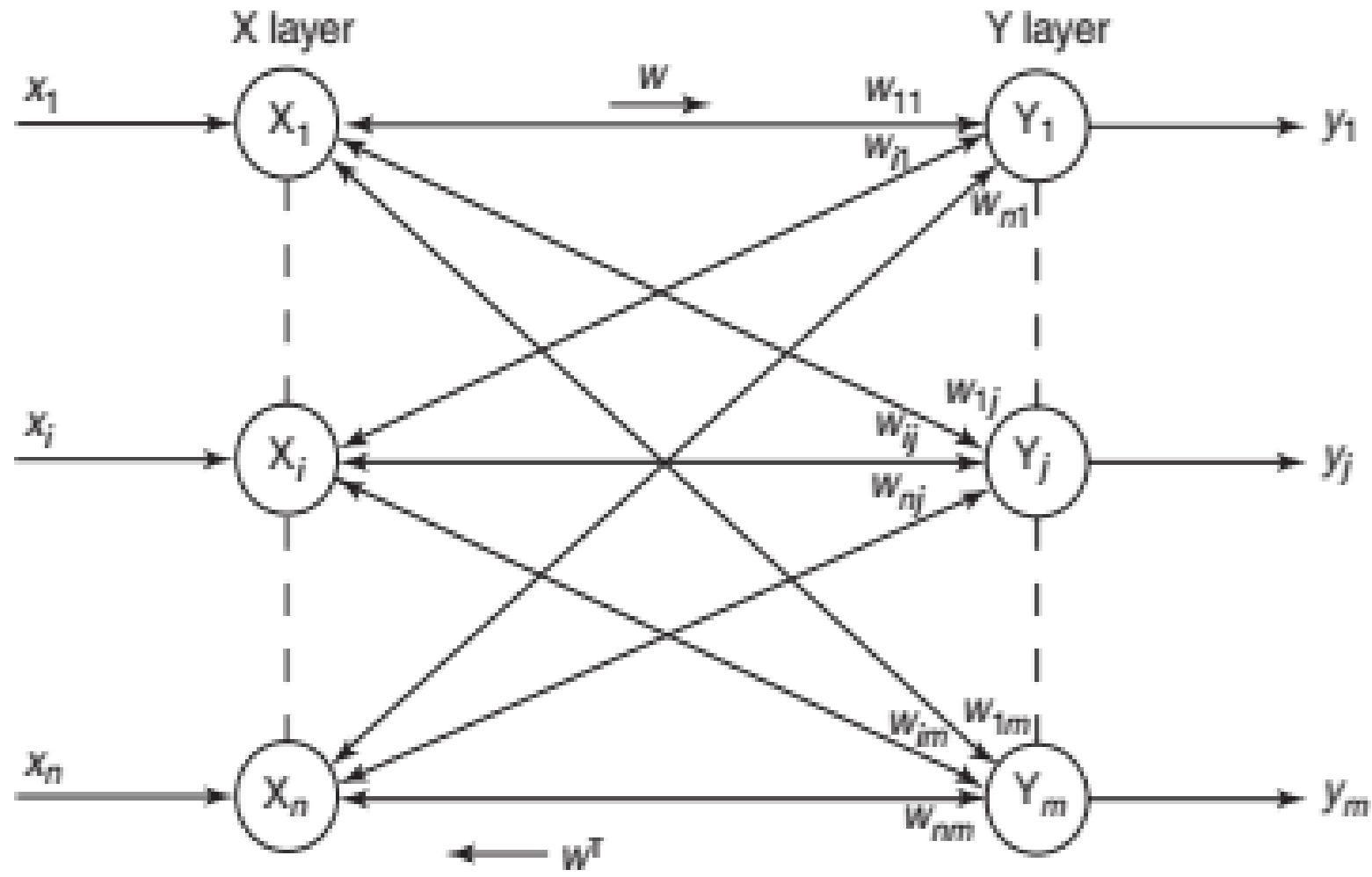
$$\begin{aligned} W &= \sum_{p=1}^4 s^T(p)t(p) \\ &= s^T(1)t(1) + s^T(2)t(2) + s^T(3)t(3) + s^T(4)t(4) \end{aligned}$$

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

$$W = \begin{bmatrix} 2 & 1 \\ 0 & 1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix}$$

Bidirectional Associative Memory (BAM)

- BAM is a *type of recurrent neural network* and also it is a *hetero-associative*, meaning given a *pattern it can return another pattern which is potentially of a different size*.
- A *bidirectional associative memory stores a set of pattern associations by summing bipolar correlation matrices* (an n by m outer product matrix for each pattern to be stored).
- The architecture of the net consists of *two layers of neurons, connected by directional weighted connection paths*.



Bidirectional Associative Memory (BAM)

- The net iterates, sending signals back and forth between the two layers until all neurons reach equilibrium (i.e., until each neuron's activation remains constant for several steps).
- Bidirectional associative memory neural nets can respond to input to either layer.
- Because *the weights are bidirectional and the algorithm alternates between updating the activations for each layer, we shall refer to the layers as the X-layer and the Y-layer* (rather than the input and output layers).

Types of BAM

- There exist two types of BAM:
 - Discrete BAM
 - Continuous BAM

Discrete BAM: Determination of Weights

Let the input vectors be denoted by $s(p)$ and target vectors by $t(p)$, $p = 1, \dots, P$. Then the weight matrix to store a set of input and target vectors, where

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

$$t(p) = (t_1(p), \dots, t_j(p), \dots, t_m(p))$$

Hebb's Learning rule is used.

Case 1: Input vectors are *binary*, the weight matrix is given by:

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

Case 2: Input vectors are *bipolar*, the weight matrix is given by:

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

The weights matrix in both the cases are going to be in bipolar form irrespective of the form of the input vectors. The formulas mentioned above can be directly applied to the determination of weights of a BAM.

Discrete BAM: Activation function for BAM

- Step activation with threshold (non-zero) is used as the activation function for discrete BAM.
- The activation function also used is one the basis that input target vector pair are binary or bipolar

- The activation function **for Y layer**:

1. with binary input vectors is

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

2. with bipolar input vectors is

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

- The activation function **for X layer**:

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

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

Discrete BAM: Testing Algorithm

Step 0: Initialize the weights to store p vectors. Also initialize all the activations to zero.

Step 1: Perform Steps 2–6 for each testing input.

Step 2: Set the activations of X layer to current input pattern, i.e., presenting the input pattern x to X layer and similarly presenting the input pattern y to Y layer. Even though, it is bidirectional memory, at one time step, signals can be sent from only one layer. So, either of the input patterns may be the zero vector.

Step 3: Perform Steps 4–6 when the activations are not converged.

Step 4: Update the activations of units in Y layer. Calculate the net input,

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

Applying the activations (as in Section 4.5.3.2), we obtain

$$y_j = f(y_{ij})$$

Send this signal to the X layer.

Step 5: Update the activations of units in X layer. Calculate the net input,

$$x_{int} = \sum_{j=1}^m y_j w_{ij}$$

Apply the activations over the net input,

$$x_i = f(x_{int})$$

Send this signal to the Y layer.

Step 6: Test for convergence of the net. The convergence occurs if the activation vectors x and y reach equilibrium. If this occurs then stop, otherwise, continue.

Continuous BAM

If the input vectors are binary, $(s(p), t(p))$, $p = 1$ to P the weights are determined using the formula

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

i.e., even though the input vectors are binary, the weight matrix is bipolar. The activation function used here is the logistic sigmoidal function. If it is binary logistic function, then the activation function is

$$f(y_{inj}) = \frac{1}{1 + e^{-y_{inj}}}$$

If the activation function used is a bipolar logistic function, then the function is defined as

$$f(y_{inj}) = \frac{2}{1 + e^{-y_{inj}}} - 1 = \frac{1 - e^{-y_{inj}}}{1 + e^{-y_{inj}}}$$

These activation functions are applied over the net input to calculate the output. The net input can be calculated with a bias included, i.e.,

$$y_{inj} = b_j + \sum_i x_i w_{ij}$$

and all these formulas apply for the units in X layer also.

Construct and test a BAM network to associate letters E and F with simple bipolar input-output vectors. The target output for E is $(-1, 1)$ and for F is $(1, 1)$. The display matrix size is 5×3 . The input patterns are

$$\begin{array}{ccc}
 * & * & * & * & * & * \\
 * & \bullet & \bullet & * & * & * \\
 * & * & * & * & \bullet & \bullet \\
 * & \bullet & \bullet & * & \bullet & \bullet \\
 * & * & * & * & \bullet & \bullet \\
 \text{"E"} & & & \text{"F"} & &
 \end{array}$$

Target output $(-1, 1)$ $(1, 1)$

The inputs are

Input pattern	Inputs	Targets	Weights
E	$[1 \ 1 \ 1 \ 1 -1 -1 \ 1 \ 1 \ 1 \ 1 -1 -1 \ 1 \ 1 \ 1]$	$[-1, \ 1]$	W_1
F	$[1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 -1 -1 \ 1 -1 -1 \ 1 -1 -1]$	$[1 \ 1]$	W_2

(i) X vectors as input: The weight matrix is obtained by

$$W = \sum s^T(p) t(p)$$

$$W_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ -1 \\ -1 \\ 1 \\ 1 \\ 1 \\ 1 \\ -1 \\ -1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix} = \begin{bmatrix} -1 & 1 \\ -1 & 1 \\ -1 & 1 \\ -1 & 1 \\ 1 & -1 \\ 1 & -1 \\ -1 & 1 \\ -1 & 1 \\ -1 & 1 \\ -1 & 1 \\ 1 & -1 \\ 1 & -1 \\ -1 & 1 \\ -1 & 1 \\ -1 & 1 \end{bmatrix}$$

$$w_2 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ -1 \\ -1 \\ 1 \\ -1 \\ -1 \\ -1 \\ 1 \\ -1 \\ -1 \end{bmatrix} [1 \quad 1] = \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ -1 & -1 \\ -1 & -1 \\ 1 & 1 \\ -1 & -1 \\ -1 & -1 \\ -1 & -1 \\ 1 & 1 \\ -1 & -1 \\ -1 & -1 \end{bmatrix}$$

Testing the network with test vectors "E" and "F".

The total weight matrix is

$$W = W_1 + W_2 = \begin{bmatrix} -1 & 1 \\ -1 & 1 \\ -1 & 1 \\ -1 & 1 \\ 1 & -1 \\ 1 & -1 \\ -1 & 1 \\ -1 & 1 \\ -1 & 1 \\ 1 & -1 \\ 1 & -1 \\ -1 & 1 \\ -1 & 1 \\ -1 & 1 \end{bmatrix} + \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ -1 & -1 \\ -1 & -1 \\ -1 & -1 \\ -1 & -1 \\ 1 & 1 \\ -1 & -1 \\ -1 & -1 \end{bmatrix} = \begin{bmatrix} 0 & 2 \\ 0 & 2 \\ 0 & 2 \\ 0 & 2 \\ 2 & 0 \\ 2 & 0 \\ 0 & 2 \\ 0 & 2 \\ 0 & 2 \\ 0 & -2 \\ 0 & -2 \\ 0 & 2 \\ 0 & -2 \\ -2 & 0 \end{bmatrix}$$

- For test pattern E, computing net input we get

$$y_{in} = [1 \ 1 \ 1 \ 1 \ -1 \ -1 \ 1 \ 1 \ 1 \ 1 \ -1 \ -1 \ 1 \ 1 \ 1]_{1 \times 15} \begin{bmatrix} 0 & 2 \\ 0 & 2 \\ 0 & 2 \\ 0 & 2 \\ 2 & 0 \\ 2 & 0 \\ -2 & 0 \\ 0 & 2 \\ 0 & 2 \\ -2 & 0 \\ -2 & 0 \\ 0 & 2 \\ 0 & -2 \\ 0 & -2 \\ 0 & 2 \\ -2 & 0 \\ -2 & 0 \end{bmatrix}_{15 \times 2}$$

$$= [-12 \ 18]_{1 \times 2}$$

Applying activations, we get $y = [-1 \ 1]$, hence correct response is obtained.

- For test pattern F. Computing net input, we get

$$y_{in} = [1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ -1 \ -1 \ 1 \ -1 \ -1 \ -1 \ 1 \ -1 \ -1] \begin{bmatrix} 0 & 2 \\ 0 & 2 \\ 0 & 2 \\ 0 & 2 \\ 2 & 0 \\ 2 & 0 \\ 0 & 2 \\ -2 & 0 \\ -2 & 0 \\ 0 & 2 \\ 0 & -2 \\ 0 & -2 \\ 0 & 2 \\ -2 & 0 \\ -2 & 0 \end{bmatrix} = [12 \ 18]$$

Applying activations over the net input, to calculate output, we get $y = [1 \ 1]$, hence correct response is obtained.

(ii) Y vectors as input: The weight matrix when Y vectors are used as input is obtained as the transpose of the weight matrix when X vectors were presented as input, i.e.,

$$W^T = \begin{bmatrix} 0 & 0 & 0 & 0 & 2 & 2 & 0 & -2 & -2 & 0 & 0 & 0 & 0 & -2 & -2 \\ 2 & 2 & 2 & 2 & 0 & 0 & 2 & 0 & 0 & 2 & -2 & -2 & 2 & 0 & 0 \end{bmatrix}$$

Testing the network

(a) For test pattern E, now the input is $[-11]$. Computing net input, we have

$$\begin{aligned} y_{in} = x \cdot W^T &= [-11] \cdot \begin{bmatrix} 0 & 0 & 0 & 0 & 2 & 2 & 0 & -2 & -2 & 0 & 0 & 0 & 0 & -2 & -2 \\ 2 & 2 & 2 & 2 & 0 & 0 & 2 & 0 & 0 & 2 & -2 & -2 & 2 & 0 & 0 \end{bmatrix} \\ &= [2 \ 2 \ 2 \ 2 \ -2 \ -2 \ 2 \ 2 \ 2 \ -2 \ -2 \ 2 \ 2 \ 2] \end{aligned}$$

Applying the activation functions, we get

$$y = [1 \ 1 \ 1 \ 1 \ -1 \ -1 \ 1 \ 1 \ 1 \ 1 \ -1 \ -1 \ 1 \ 1 \ 1]$$

which is the correct response.

(b) For test pattern F, now the input is [1, 1]. Computing net input, we have

$$y_{in} = x \cdot W^T = [1 \ 1] \begin{bmatrix} 0 & 0 & 0 & 0 & 2 & 2 & 0 & -2 & -2 & 0 & 0 & 0 & -2 & -2 \\ 2 & 2 & 2 & 2 & 0 & 0 & 2 & 0 & 0 & 2 & -2 & -2 & 2 & 0 & 0 \end{bmatrix}$$
$$= [2 \ 2 \ 2 \ 2 \ 2 \ 2 \ -2 \ -2 \ 2 \ -2 \ -2 \ 2 \ -2 \ -2]$$

Applying the activation functions, we get

$$y = [1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ -1 \ -1 \ 1 \ -1 \ -1 \ 1 \ -1 \ -1]$$

which is the correct response. Thus, a BAM network has been constructed and tested in both the directions from X to Y and Y to X.

Learning Vector Quantization

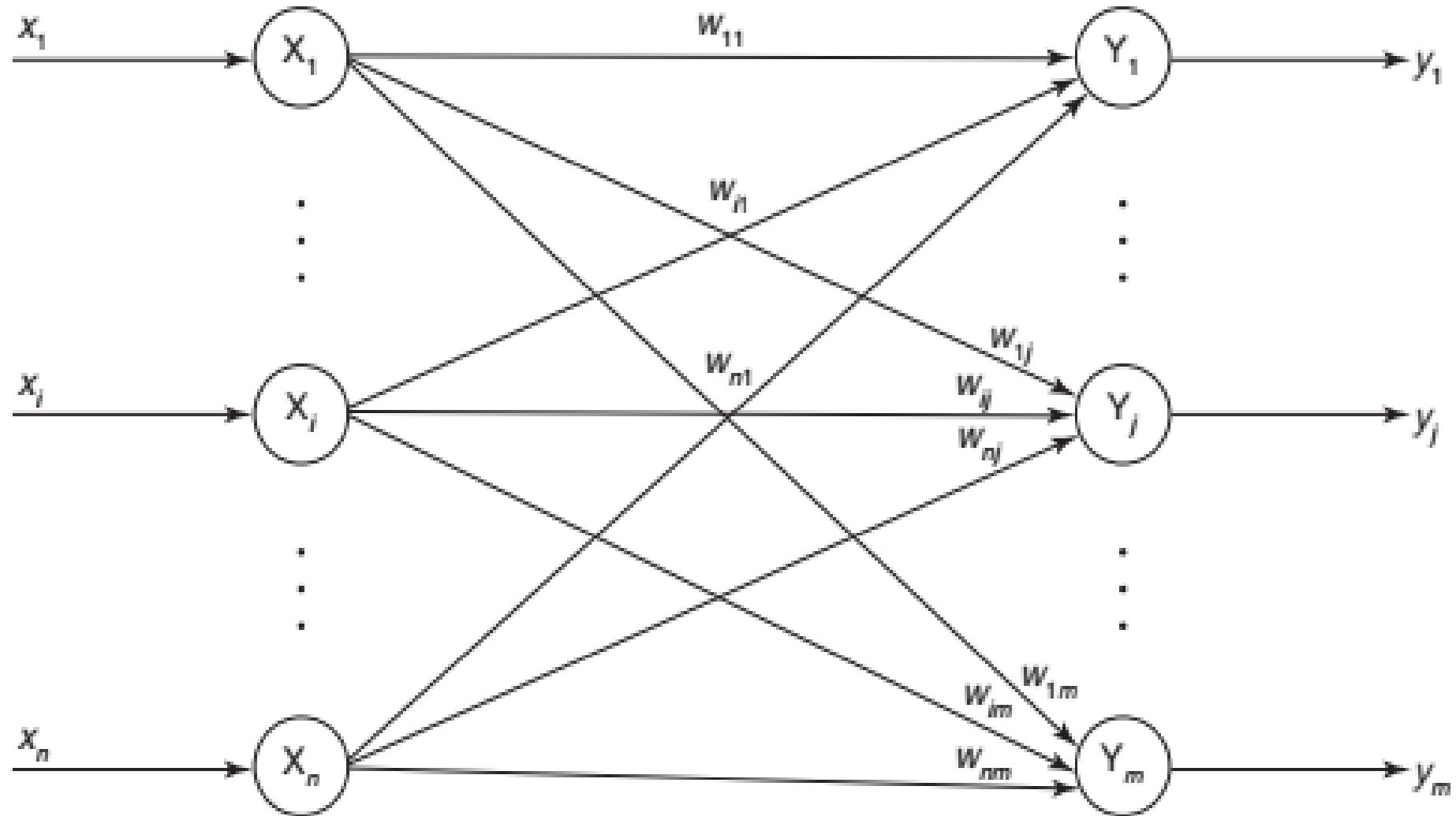
&

Adaptive Resonance Theory Network

Dr. Shruti Mishra
School of Computer Sc. & Engg. (SCOPE)

Introduction

- It's a kind of ANN.
- LVQ is a process of classifying the patterns, wherein each *output unit represents a particular class.*
- The O/P weight vector is called the **reference vector or code book vector** for the class which the unit represents.
- Input to LVQ: Set of known training patterns with know classification is given to the n/w.
- After training process LVQ net will classify an i/p vector by assigning it to the same class as that of the o/p unit, which has very close weight vector to that of the input vector.



Training Algorithm

The parameters used for the training process of a LVQ include the following:

x = training vector ($x_1, \dots, x_i, \dots, x_n$)

T = category or class for the training vector x

w_j = weight vector for j th output unit ($w_{1j}, \dots, w_{ij}, \dots, w_{nj}$)

c_j = cluster or class or category associated with j th output unit.

Objective of the algorithm is to find the output unit that is closest to the input vector.

Step 0: Initialize the reference vectors. This can be done using the following steps.

- From the given set of training vectors, take the first “ m ” (number of clusters) training vectors and use them as weight vectors, the remaining vectors can be used for training.
- Assign the initial weights and classifications randomly.
- K-means clustering method.

Set initial learning rate α .

Step 1: Perform Steps 2–6 if the stopping condition is false.

Step 2: Perform Steps 3–4 for each training input vector x .

Step 3: Calculate the Euclidean distance; for $i = 1$ to n , $j = 1$ to m ,

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

Find the winning unit index J , when $D(J)$ is minimum.

Step 4: Update the weights on the winning unit, w_j , using the following conditions.

$$\text{If } T = c_j, \text{ then } w_j(\text{new}) = w_j(\text{old}) + \alpha[x - w_j(\text{old})]$$

$$\text{If } T \neq c_j, \text{ then } w_j(\text{new}) = w_j(\text{old}) - \alpha[x - w_j(\text{old})]$$

Step 5: Reduce the learning rate α .

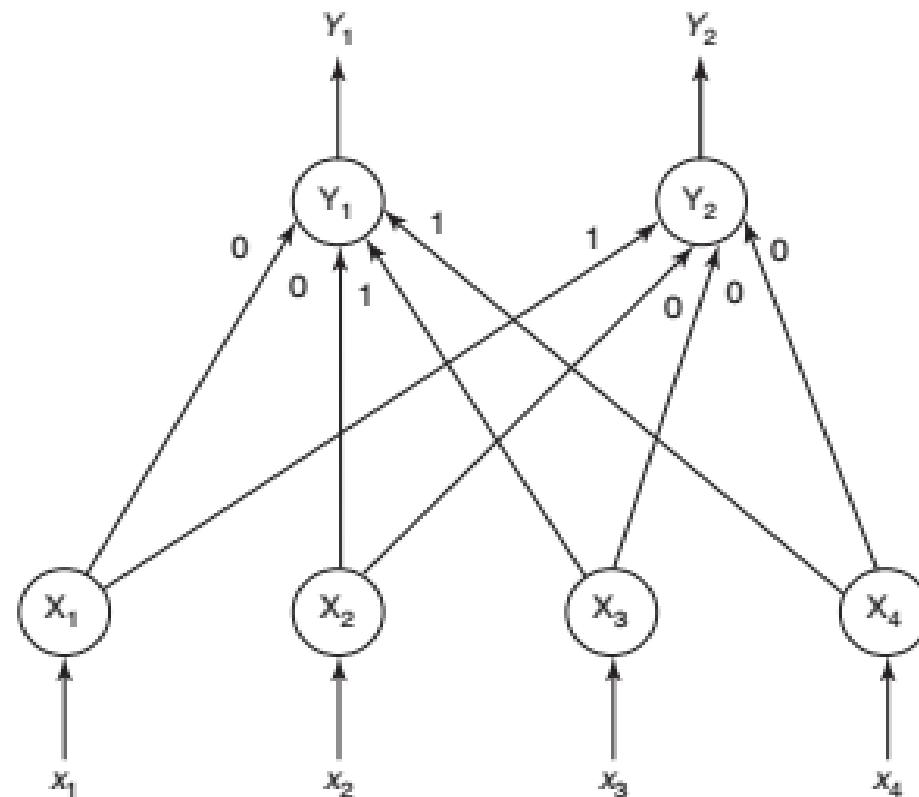
Step 6: Test for the stopping condition of the training process. (The stopping conditions may be fixed number of epochs or if learning rate has reduced to a negligible value.)

Problem

Construct and test an LVQ net with five vectors assigned to two classes. The given vectors along with the classes are as shown in Table 1.

Vector	Class
[0 0 1 1]	1
[1 0 0 0]	2
[0 0 0 1]	2
[1 1 0 0]	1
[0 1 1 0]	1

Solution: In the given five vectors, first two vectors are used as initial weight vectors and the remaining three vectors are used as input vectors. Based on this, LVQ net is shown in Figure 6, along with initial weights.



Initialize the reference weight vectors as

$$w_1 = [0 \ 0 \ 1 \ 1]; \quad w_2 = [1 \ 0 \ 0 \ 0]$$

Let the learning rate be $\alpha = 0.1$.

First input vector

For $[0 \ 0 \ 0 \ 1]$ with $T = 2$, calculate the square of the Euclidean distance, i.e.,

$$D(j) = \sum_{i=1}^4 (w_{ij} - x_i)^2$$

For $j = 1$ to 2,

$$D(1) = (0-0)^2 + (0-0)^2 + (1-0)^2$$

$$+ (1-1)^2 = 1$$

$$D(2) = (1-0)^2 + (0-0)^2 + (0-0)^2$$

$$+ (0-1)^2 = 2$$

Since $D(1) < D(2)$, $D(1)$ is minimum; hence the winner unit index is $J = 1$. Now that $T \neq J$, the weight updation is performed as

$$w_j(\text{new}) = w_j(\text{old}) - \alpha [x - w_j(\text{old})]$$

$$w_{11}(n) = w_{11}(0) - \alpha [x_1 - w_{11}(0)]$$

$$= 0 - 0.1(0 - 0) = 0$$

$$w_{21}(n) = w_{21}(0) - \alpha [x_2 - w_{21}(0)]$$

$$= 0 - 0.1(0 - 0) = 0$$

$$w_{31}(n) = w_{31}(0) - \alpha [x_3 - w_{31}(0)]$$

$$= 1 - 0.1(0 - 1) = 1.1$$

$$w_{41}(n) = w_{41}(0) - \alpha [x_4 - w_{41}(0)]$$

$$= 1 - 0.1(1 - 1) = 1$$

After the presentation of first input pattern, the weight matrix becomes

$$W = \begin{bmatrix} 0 & 1 \\ 0 & 0 \\ 1.1 & 0 \\ 1 & 0 \end{bmatrix}$$

Second input vector

For $[1 \ 1 \ 0 \ 0]$ with $T=1$, calculate the square of the Euclidean distance, i.e.,

$$D(j) = \sum_{i=1}^4 (w_{ij} - x_i)^2$$

For $j=1$ to 2,

$$\begin{aligned} D(1) &= (0-1)^2 + (0-1)^2 + (1.1-0)^2 \\ &\quad + (1-0)^2 = 4.21 \end{aligned}$$

$$\begin{aligned} D(2) &= (1-1)^2 + (0-1)^2 + (0-0)^2 \\ &\quad + (0-0)^2 = 1 \end{aligned}$$

Since $D(2) < D(1)$, $D(2)$ is minimum; hence the winner unit index is $J=2$. Again since $T \neq J$, the weight updation is performed as

$$w_j(\text{new}) = w_j(\text{old}) - \alpha [x - w_j(\text{old})]$$

$$\begin{aligned} w_{12}(n) &= w_{12}(0) - \alpha [x_1 - w_{12}(0)] \\ &= 1 - 0.1(1-1) = 1 \end{aligned}$$

$$\begin{aligned} w_{22}(n) &= w_{22}(0) - \alpha [x_2 - w_{22}(0)] \\ &= 0 - 0.1(1-0) = -0.1 \end{aligned}$$

$$\begin{aligned} w_{32}(n) &= w_{32}(0) - \alpha [x_3 - w_{32}(0)] \\ &= 0 - 0.1(0-0) = 0 \end{aligned}$$

$$\begin{aligned} w_{42}(n) &= w_{42}(0) - \alpha [x_4 - w_{42}(0)] \\ &= 0 - 0.1(0-0) = 0 \end{aligned}$$

After the presentation of second input pattern, the weight matrix becomes

$$W = \begin{bmatrix} 0 & 1 \\ 0 & -0.1 \\ 1.1 & 0 \\ 1 & 0 \end{bmatrix}$$

Third input vector

For $[0 \ 1 \ 1 \ 0]$ with $T = 1$, calculate the square of the Euclidean distance as

$$D(j) = \sum_{i=1}^4 (w_{ij} - x_i)^2$$

For $j = 1$ to 2,

$$\begin{aligned} D(1) &= (0-0)^2 + (0-1)^2 + (1.1-1)^2 \\ &\quad + (1-0)^2 = 2.01 \end{aligned}$$

$$\begin{aligned} D(2) &= (1-0)^2 + (-0.1-1)^2 + (0-1)^2 \\ &\quad + (0-0)^2 = 3.21 \end{aligned}$$

Since $D(1) < D(2)$, $D(1)$ is minimum; hence the winner unit index is $J = 1$. Now that $T = J$, the weight updation is performed as

$$w_j(\text{new}) = w_j(\text{old}) + \alpha [x - w_j(\text{old})]$$

Updating the weights on the winner unit, we obtain

$$\begin{aligned} w_{11}(n) &= w_{11}(0) + \alpha [x_1 - w_{11}(0)] \\ &= 0 + 0.1(0 - 0) = 0 \end{aligned}$$

$$\begin{aligned} w_{21}(n) &= w_{21}(0) + \alpha [x_2 - w_{21}(0)] \\ &= 0 + 0.1(1 - 0) = 0.1 \end{aligned}$$

$$\begin{aligned} w_{31}(n) &= w_{31}(0) + \alpha [x_3 - w_{31}(0)] \\ &= 1.1 + 0.1(1 - 1.1) = 1.09 \end{aligned}$$

$$\begin{aligned} w_{41}(n) &= w_{41}(0) + \alpha [x_4 - w_{41}(0)] \\ &= 1 + 0.1(0 - 1) = 0.9 \end{aligned}$$

After the presentation of third input pattern, the weight matrix becomes

$$W = \begin{bmatrix} 0 & -1 \\ 0.1 & -0.1 \\ 1.09 & -0 \\ 0.9 & -0 \end{bmatrix}$$

Adaptive Resonance Theory (ART)

Introduction

- **Adaptive resonance theory (ART)** is a theory developed by *Stephen Grossberg and Gail Carpenter* on aspects of **how the brain processes information**.
- It describes a number of *neural network models which use supervised and unsupervised learning methods*, and *address problems such as pattern recognition and prediction*.
- The primary intuition behind the ART model is that object identification and recognition generally occur as a result of the interaction of '**top-down**' observer expectations with '**bottom-up**' sensory information.

Adaptive Resonance Theory *ART* networks, as the name suggests, is always open to new learning *adaptive* without losing the old patterns *resonance*

- The model postulates that 'top-down' expectations take the form of a memory template or prototype that is then compared with the actual features of an object as detected by the senses.
- This comparison gives rise to a measure of category belongingness.
- As long as this difference between sensation and expectation does not exceed a set threshold called the 'vigilance parameter', the sensed object will be considered a member of the expected class.
- The system thus offers a solution to the 'plasticity/stability' problem, i.e. the problem of acquiring new knowledge without disrupting existing knowledge that is also called incremental learning.

Learning model

- The basic ART system is an unsupervised learning model.
- It typically consists of a *comparison field and a recognition field composed of neurons, a vigilance parameter (threshold of recognition), and a reset module.*
- The **comparison field** takes an *input vector (a one-dimensional array of values) and transfers it to its best match in the recognition field.*
- Its best match is the single neuron whose set of weights (weight vector) most closely matches the input vector.

- After the **input vector** is classified, the reset module compares the strength of the recognition match to the vigilance parameter.
- If the **vigilance parameter is overcome** (i.e. the input vector is within the normal range seen on previous input vectors) **training commences**.
- If the match level is below the **vigilance parameter** (i.e. the input vector's match is outside the normal expected range for that neuron) the *winning recognition neuron is inhibited and a search procedure is carried out vectors*), **training commences**.

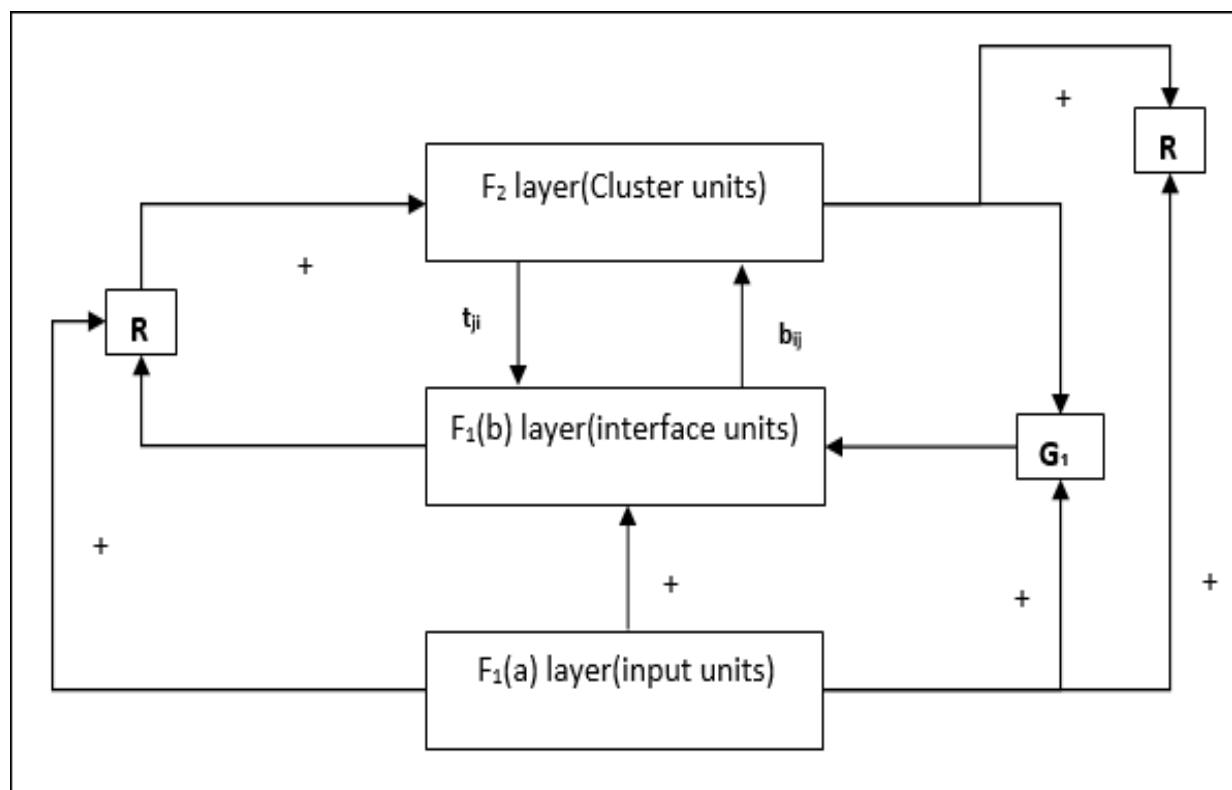
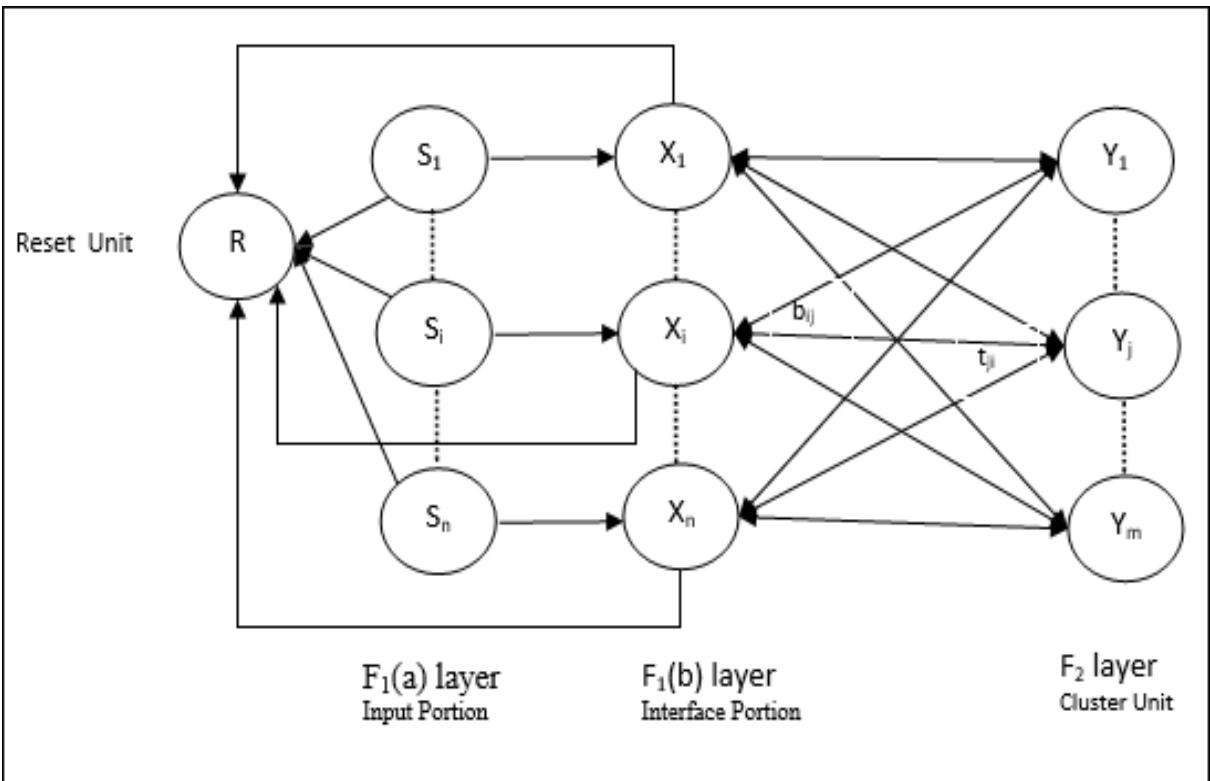
Summary of the Learning Model

- **Recognition phase** – The input vector is compared with the classification presented at every node in the output layer. The output of the neuron becomes “1” if it best matches with the classification applied, otherwise it becomes “0”.
- **Comparison phase** – In this phase, a comparison of the input vector to the comparison layer vector is done. **The condition for reset is that the degree of similarity would be less than vigilance parameter.**
- **Search phase** – In this phase, the network will search for reset as well as the match done in the above phases. Hence, **if there would be no reset and the match is quite good, then the classification is over. Otherwise, the process would be repeated and the other stored pattern must be sent to find the correct match.**

ART1 & Architecture of ART1

- It is a type of ART, which is designed to cluster binary vectors.
- It consists of the following two units –
- **Computational Unit** – It is made up of the following –
 - **Input unit (F_1 layer)** – It further has the following two portions:
 - 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.

- **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.



Parameters Used

- Following parameters are used –
 - n – Number of components in the input vector
 - m – Maximum number of clusters that can be formed
 - b_{ij} – Weight from F_1 layer to F_2 layer, i.e. bottom-up weights
 - t_{ji} – Weight from F_2 to F_1 layer, i.e. top-down weights
 - ρ – Vigilance parameter
 - $\|x\|$ – Norm of vector x

Algorithm

Step 1 – Initialize the learning rate, the vigilance parameter, and the weights as follows –

$$\alpha > 1 \text{ and } 0 < \rho \leq 1$$

$$0 < b_{ij}(0) < \frac{\alpha}{\alpha - 1 + n} \text{ and } t_{ij}(0) = 1$$

$$b_{ij}(0) = 1/(1+n)$$

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

Step 3 – Continue step 4-6 for every training input.

Step 4 – Set activations of all F_1 a and F_1 units as follows

$$F_2 = 0 \text{ and } F_1 \ a = \text{input vectors}$$

Step 5 – Input signal from F_1 a to F_1 b layer must be sent like

$$s_i = x_i$$

Step 6 – For every inhibited F_2 node

$$y_j = \sum_i b_{ij}x_i \quad \text{the condition is } y_j \neq -1$$

Step 7 – Perform step 8-10, when the reset is true.

Step 8 – Find J for $y_J \geq y_j$ for all nodes j

Step 9 – Again calculate the activation on F_1 b as follows

$$x_i = s_i t_{Ji}$$

$$x_i = s_i t_{Ji}$$

Step 10 – Now, after calculating the norm of vector x and vector s , we need to check the reset condition as follows –

If $\|x\|/\|s\| <$ vigilance parameter p , then inhibit node J and go to step 7

Else If $\|x\|/\|s\| \geq$ vigilance parameter p , then proceed further.

Step 11 – Weight updating for node **J** can be done as follows –

$$b_{ij}(\text{new}) = \frac{\alpha x_i}{\alpha - 1 + \|x\|}$$

$$t_{ij}(\text{new}) = x_i$$

Step 12 – The stopping condition for algorithm must be checked and it may be as follows –

- Do not have any change in weight.
- Reset is not performed for units.
- Maximum number of epochs reached.

Problem

Construct an ART 1 network for clustering four input vectors with low vigilance parameter of 0.4 into three clusters. The four input vectors are [0 0 0 1], [0 1 0 1], [0 0 1 1] and [1 0 0 0]. Assume the necessary parameters needed.

Solution: The values assumed in this case are $\rho = 0.4$, $\alpha = 2$. Also it can be noted that $n = 4$ and $m = 3$. Hence,

Bottom-up-weights, $b_{ij}(0) = 1/1+n = 1/1+4 = 0.2$.

Top-down-weights $t_{ji}(0) = 1$.

For $i = 1$ to 4 and $j = 1$ to 2,

$$b_{ij} = \begin{bmatrix} 0.2 & 0.2 & 0.2 \\ 0.2 & 0.2 & 0.2 \\ 0.2 & 0.2 & 0.2 \\ 0.2 & 0.2 & 0.2 \end{bmatrix}_{4 \times 3}$$

$$\text{and } t_{ji} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}_{3 \times 4}$$

Step 0: Initialize the parameters:

$$\rho = 0.4; \quad \alpha = 2$$

Initialize weights:

$$b_{ij}(0) = 0.2; \quad t_{ji}(0) = 1$$

Step 1: Start computation.

Step 2: For the first input vector [0 0 0 1], perform Steps 3–12.

Step 3: Set activations of all F_2 units to zero.
Set activations of F_1 (a) units to input vector $s = [0 0 0 1]$.

Step 4: Compute norm of s :

$$\|s\| = 0 + 0 + 0 + 1 = 1$$

Step 5: Compute activations for each node in the F_1 layer:

$$x = [0 0 0 1]$$

Step 6: Compute net input to each node in the F_2 layer:

$$y_j = \sum_{i=1}^4 b_{ij} x_i$$

For $j = 1$ to 3,

$$\begin{aligned}y_1 &= 0.2(0) + 0.2(0) + 0.2(0) + 0.2(1) \\&= 0.2\end{aligned}$$

$$\begin{aligned}y_2 &= 0.2(0) + 0.2(0) + 0.2(0) + 0.2(1) \\&= 0.2\end{aligned}$$

$$\begin{aligned}y_3 &= 0.2(0) + 0.2(0) + 0.2(0) + 0.2(1) \\&= 0.2\end{aligned}$$

Step 7: When reset is true, perform Steps 8–11.

Step 8: Since all the inputs pose same net input, there exists a tie and the unit with the smallest index is the winner, i.e., $J=1$.

Step 9: Recompute the F_i activations (for $J=1$):

$$x_i = s_i t_{ji}$$

$$x_1 = s_1 t_{j1} = [0 \ 0 \ 0 \ 1][1 \ 1 \ 1 \ 1]$$

$$x_1 = [0 \ 0 \ 0 \ 1]$$

Step 10: Calculate norm of x :

$$\|x\| = 1$$

Step 11: Test for reset condition:

$$\frac{\|x\|}{\|s\|} = \frac{1}{1} = 1.0 \geq 0.4(\rho)$$

Hence reset is false. Proceed to Step 12.

Step 12: • Update bottom-up weights for $\alpha = 2$:

$$\begin{aligned}b_{ij}(\text{new}) &= \frac{\alpha x_i}{\alpha - 1 + \|x\|} \\&= \frac{2x_i}{2 - 1 + \|x\|} = \frac{2x_i}{1 + \|x\|}\end{aligned}$$

$$b_{11} = \frac{2 \times 0}{1 + 1} = 0;$$

$$b_{21} = \frac{2 \times 0}{1 + 1} = 0;$$

$$b_{31} = \frac{2 \times 0}{1 + 1} = 0;$$

$$b_{41} = \frac{2 \times 1}{1 + 1} = \frac{2}{2} = 1$$

Therefore, the bottom-up weight matrix b_{ij} becomes

$$b_{ij} = \begin{bmatrix} 0 & 0.2 & 0.2 \\ 0 & 0.2 & 0.2 \\ 0 & 0.2 & 0.2 \\ 1 & 0.2 & 0.2 \end{bmatrix}$$

- Update the top-down weights:

$$t_{ji}(\text{new}) = x_i$$

$$t_{ji} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

Steps 0 and 1 remain the same.

Step 2: For the second input vector [0 1 0 1], perform Steps 3–12.

Step 3: Set activations of all F_2 units to zero.
Set activations of F_1 (a) units to input vector $s = [0 1 0 1]$.

Step 4: Compute norm of s :

$$\|s\| = 0 + 1 + 0 + 1 = 2$$

Step 5: Compute activations of each node in the F_1 layer:

$$x = [0 1 0 1]$$

Step 6: Compute net input to each node in the F_2 layer:

$$y_j = \sum_{i=1}^4 b_{ij} x_i$$

For $j = 1$ to 3,

$$y_1 = 0(0) + 0(1) + 0(0) + 1(1) = 1$$

$$\begin{aligned} y_2 &= 0.2(0) + 0.2(1) + 0.2(0) + 0.2(1) \\ &= 0.4 \end{aligned}$$

$$\begin{aligned} y_3 &= 0.2(0) + 0.2(1) + 0.2(0) + 0.2(1) \\ &= 0.4 \end{aligned}$$

Step 7: When reset is true, perform Steps 8–11.

Step 8: The unit with largest net input is the winner, i.e., $J = 1$.

Step 9: Recompute F_i activations (for $J = 1$):

$$x_i = s_i t_F = [0101][0001] = [0001]$$

Step 10: Calculate norm of x :

$$\|x\| = 0 + 0 + 0 + 1 = 1$$

Step 11: Test for reset condition:

$$\frac{\|x\|}{\|s\|} = \frac{1}{2} = 0.5 \geq 0.4 (\rho)$$

Hence reset is false. Proceed to Step 12.

Step 12: Update bottom-up weights for $\alpha = 2$:

$$b_{ij}(\text{new}) = \frac{2x_i}{1 + \|x\|}$$

$$b_{11} = \frac{2 \times 0}{1 + 1} = 0;$$

$$b_{21} = \frac{2 \times 0}{1 + 1} = 0;$$

$$b_{31} = \frac{2 \times 0}{1 + 1} = 0;$$

$$b_{41} = \frac{2 \times 1}{1 + 1} = \frac{2}{2} = 1$$

Therefore, the bottom-up weight matrix b_{ij} becomes

$$b_y = \begin{bmatrix} 0 & 0.2 & 0.2 \\ 0 & 0.2 & 0.2 \\ 0 & 0.2 & 0.2 \\ 1 & 0.2 & 0.2 \end{bmatrix}$$

- Update the top-down weights:

$$t_k(\text{new}) = x_i$$

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

Step 2: For the third input vector [0 0 1 1], perform Steps 3–12.

Step 3: Set activations of all F_2 units to zero.
Set activations of F_1 (a) units to input vector $s = [0 0 1 1]$.

Step 4: Compute norm of s :

$$\|s\| = 0 + 0 + 1 + 1 = 2$$

Step 5: Compute activations of each node in the F_1 layer:

$$x = [0 0 1 1]$$

Step 6: Compute net input to each node in the F_2 layer:

$$y_j = \sum_{i=1}^4 b_{ij} x_i$$

For $j = 1$ to 3,

$$y_1 = 0(0) + 0(0) + 0(1) + 1(1) = 1$$

$$\begin{aligned} y_2 &= 0.2(0) + 0.2(0) + 0.2(1) + 0.2(1) \\ &= 0.4 \end{aligned}$$

$$\begin{aligned} y_3 &= 0.2(0) + 0.2(0) + 0.2(1) + 0.2(1) \\ &= 0.4 \end{aligned}$$

Step 12: • Update bottom-up weights for $\alpha = 2$:

$$b_{ij}(\text{new}) = \frac{2x_i}{1 + \|x\|}$$

$$b_{11} = \frac{2 \times 0}{1+1} = 0;$$

$$b_{21} = \frac{2 \times 0}{1+1} = 0;$$

$$b_{31} = \frac{2 \times 0}{1+1} = 0;$$

$$b_{41} = \frac{2 \times 1}{1+1} = \frac{2}{2} = 1$$

Therefore, the bottom-up weight matrix b_{ij} becomes

$$b_{ij} = \begin{bmatrix} 0 & 0.2 & 0.2 \\ 0 & 0.2 & 0.2 \\ 0 & 0.2 & 0.2 \\ 1 & 0.2 & 0.2 \end{bmatrix}$$

Step 7: When reset is true, perform Steps 8–11.

Step 8: The unit with largest net input is the winner, i.e., $J = 1$.

Step 9: Recompute F_i activations (for $J = 1$):

$$\begin{aligned} x_i = s_i t_{ji} &= [0 \ 0 \ 1 \ 1][0 \ 0 \ 0 \ 1] \\ &= [0 \ 0 \ 0 \ 1] \end{aligned}$$

Step 10: Calculate norm of x :

$$\|x\| = 0 + 0 + 0 + 1 = 1$$

Step 11: Test for reset condition:

$$\frac{\|x\|}{\|s\|} = \frac{1}{2} = 0.5 \geq 0.4 (\rho)$$

Hence reset is false. Proceed to Step 12.

• Update the top-down weights:

$$t_{ji}(\text{new}) = x_i$$

$$t_{ji} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

Steps 0 and 1 remain the same.

Step 2: For the fourth input vector [1 0 0 0], perform Steps 3–12.

Step 3: Set activations of all F_2 units to zero.
Set activations of F_1 (a) units to input vector $s = [1 0 0 0]$.

Step 4: Compute norm of s :

$$\|s\| = 1 + 0 + 0 + 0 = 1$$

Step 5: Compute activations of each node in the F_1 layer:

$$x = [1 0 0 0]$$

Step 6: Compute net input to each node in the F_2 layer:

$$y_j = \sum_{i=1}^4 b_{ij} x_i$$

For $j = 1$ to 3,

$$y_1 = 0(1) + 0(0) + 0(0) + 1(0) = 0$$

$$\begin{aligned} y_2 &= 0.2(1) + 0.2(0) + 0.2(0) + 0.2(0) \\ &= 0.2 \end{aligned}$$

$$\begin{aligned} y_3 &= 0.2(1) + 0.2(0) + 0.2(0) + 0.2(0) \\ &= 0.2 \end{aligned}$$

Step 7: When reset is true, perform Steps 8–11.

Step 8: The unit with largest net input is the winner, i.e., $J = 2$.

Step 9: Recompute F_1 activations:

$$\begin{aligned} x_i &= s_i t_R = [1 \ 0 \ 0 \ 0][1 \ 1 \ 1 \ 1] \\ &= [1 \ 0 \ 0 \ 0] \end{aligned}$$

Step 10: Calculate norm of x :

$$\|x\| = 1 + 0 + 0 + 0 = 1$$

Step 11: Test for reset condition:

$$\frac{\|x\|}{\|s\|} = \frac{1}{1} = 1 \geq 0.4 (\rho)$$

Hence reset is false. Proceed to Step 12.

Step 12: • Update bottom-up weights for $\alpha = 2$:

$$b_{ij}(\text{new}) = \frac{2x_i}{1 + \|x\|}$$

$$b_{11} = \frac{2 \times 1}{1 + 1} = \frac{2}{2} = 1;$$

$$b_{21} = \frac{2 \times 0}{1 + 1} = 0;$$

$$b_{31} = \frac{2 \times 0}{1 + 1} = 0;$$

$$b_{41} = \frac{2 \times 0}{1 + 1} = 0$$

Therefore, the bottom-up weight matrix b_g becomes

$$b_g = \begin{bmatrix} 0 & 1 & 0.2 \\ 0 & 0 & 0.2 \\ 0 & 0 & 0.2 \\ 1 & 0 & 0.2 \end{bmatrix}$$

• Update the top-down weights:

$$t_n(\text{new}) = x_i$$

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

Step 13: Test for stopping condition. (This completes one epoch of training.)

The network may be trained for a particular number of epochs on the basis of the stopping condition.

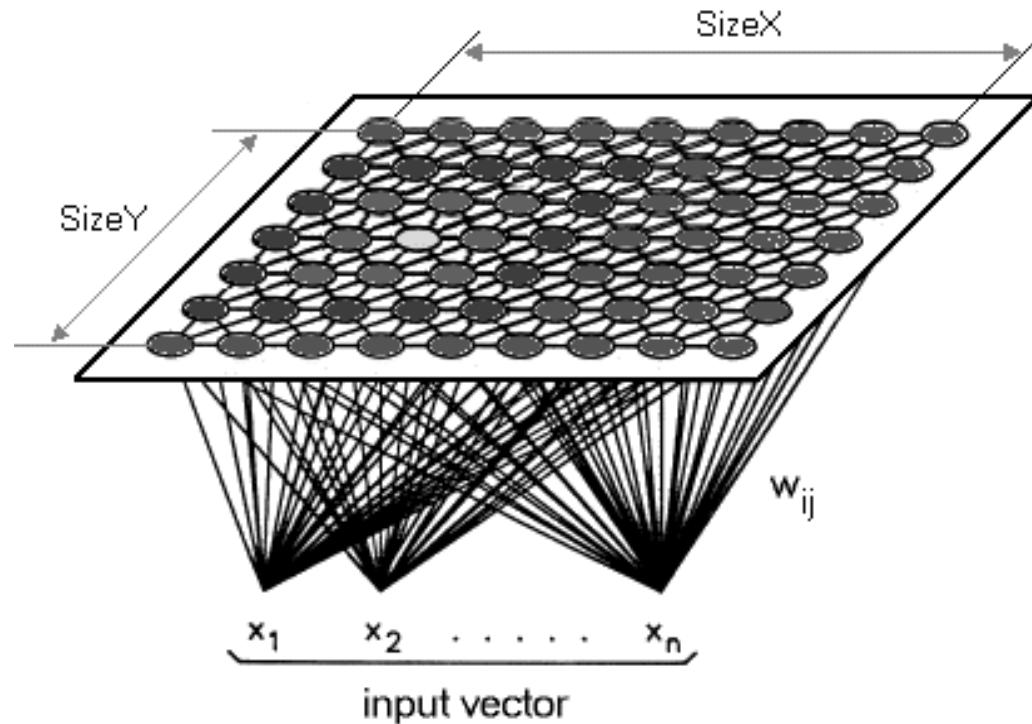
Unsupervised Learning Networks

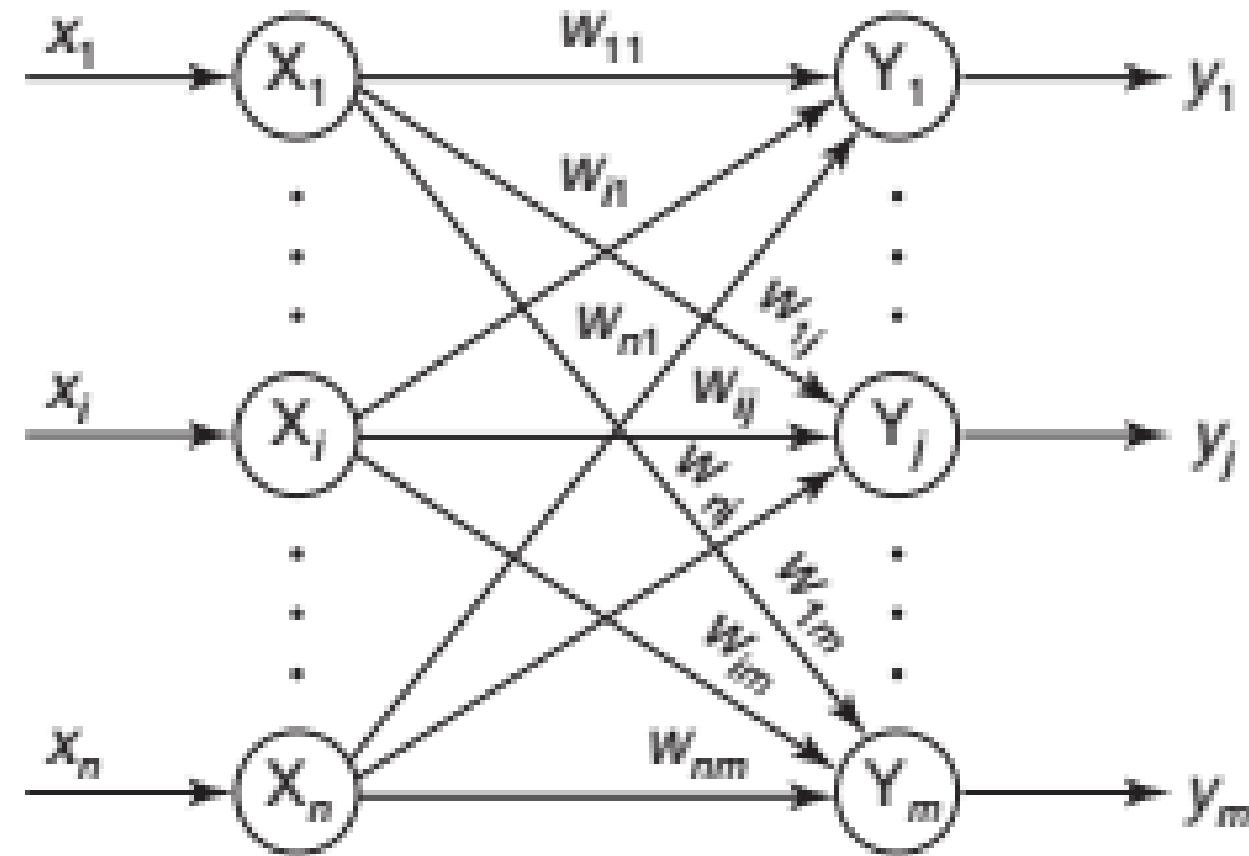
Dr. Shruti Mishra
School of Computer Sc. & Engg. (SCOPE)

Kohonen Self-Organizing Feature Maps (KSOFM)

- A self-organising map is an **unsupervised learning model**, intended for applications in **which maintaining a topology between input and output spaces.**
- It is in essence a method for **dimensionality reduction**, as it maps high-dimension inputs to a low (typically two) dimensional discretised representation and conserves the underlying structure of its input space.

- The entire **learning occurs without supervision** i.e. the nodes are *self-organising*.
- They are also called *feature maps*, as they are essentially retraining the features of the input data, and simply grouping themselves according to the *similarity* between one another.





- Step 0:**
- Initialize the weights w_{ij} : Random values may be assumed. They can be chosen as the same range of values as the components of the input vector. If information related to distribution of clusters is known, the initial weights can be taken to reflect that prior knowledge.
 - Set topological neighborhood parameters: As clustering progresses, the radius of the neighborhood decreases.
 - Initialize the learning rate α : It should be a slowly decreasing function of time.

Step 1: Perform Steps 2–8 when stopping condition is false.

Step 2: Perform Steps 3–5 for each input vector x .

Step 3: Compute the square of the Euclidean distance, i.e., for each $j = 1$ to m ,

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

Step 4: Find the winning unit index J , so that $D(J)$ is minimum. (In Steps 3 and 4, dot product method can also be used to find the winner, which is basically the calculation of net input, and the winner will be the one with the largest dot product.)

Step 5: For all units j within a specific neighborhood of J and for all i , calculate the new weights:

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

or

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

Step 6: Update the learning rate α using the formula $\alpha(t+1) = 0.5\alpha(t)$.

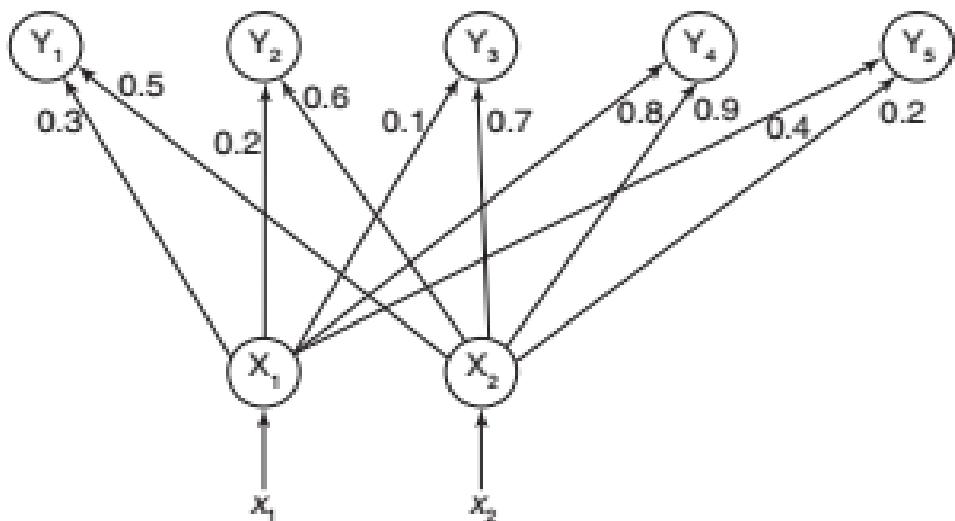
Step 7: Reduce radius of topological neighborhood at specified time intervals.

Step 8: Test for stopping condition of the network.

For a given Kohonen self-organizing feature map with weights shown in Figure 4: (a) Use the square of the Euclidean distance to find the cluster unit Y_j closest to the input vector $(0.2, 0.4)$. Using a learning rate of 0.2, find the new weights for unit Y_j . (b) For the input vector $(0.6, 0.6)$ with learning rate 0.1, find the winning cluster unit and its new weights.

Solution: (a) For the input vector $(0.2, 0.4) = (x_1, x_2)$ and $\alpha = 0.2$, the weight vector W is given by

$$W = \begin{bmatrix} 0.3 & 0.2 & 0.1 & 0.8 & 0.4 \\ 0.5 & 0.6 & 0.7 & 0.9 & 0.2 \end{bmatrix}$$



Now we find the winner unit using square of Euclidean distance, i.e.,

$$D(j) = \sum_{i=1}^2 (w_{ij} - x_i)^2 = (w_{1j} - x_1)^2 + (w_{2j} - x_2)^2$$

For $j = 1$ to 5

$$\begin{aligned} D(1) &= (0.3 - 0.2)^2 + (0.5 - 0.4)^2 \\ &= 0.01 + 0.01 = 0.02 \end{aligned}$$

$$\begin{aligned} D(2) &= (0.2 - 0.2)^2 + (0.6 - 0.4)^2 \\ &= 0 + 0.04 = 0.04 \end{aligned}$$

$$\begin{aligned} D(3) &= (0.1 - 0.2)^2 + (0.7 - 0.4)^2 \\ &= 0.01 + 0.09 = 0.1 \end{aligned}$$

$$\begin{aligned} D(4) &= (0.8 - 0.2)^2 + (0.9 - 0.4)^2 \\ &= 0.36 + 0.25 = 0.61 \end{aligned}$$

$$\begin{aligned} D(5) &= (0.4 - 0.2)^2 + (0.2 - 0.4)^2 \\ &= 0.04 + 0.04 = 0.08 \end{aligned}$$

Since $D(1)=0.02$ is the minimum value, the winner unit is $J=1$. We now update the weights on the winner unit $J=1$. The weight updation formula is given by

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

Substituting $J=1$ in the equation above, we obtain

$$w_{i1}(\text{new}) = w_{i1}(\text{old}) + \alpha [x_i - w_{i1}(\text{old})]$$

For $i=1$ to 2,

$$\begin{aligned} w_{11}(n) &= w_{11}(0) + \alpha [x_1 - w_{11}(0)] \\ &= 0.3 + 0.2(0.2 - 0.3) = 0.28 \end{aligned}$$

$$\begin{aligned} w_{21}(n) &= w_{21}(0) + \alpha [x_2 - w_{21}(0)] \\ &= 0.5 + 0.2(0.4 - 0.5) = 0.48 \end{aligned}$$

The updated weight matrix is given by

$$W = \begin{bmatrix} 0.28 & 0.2 & 0.1 & 0.8 & 0.4 \\ 0.48 & 0.6 & 0.7 & 0.9 & 0.2 \end{bmatrix}$$

- For the input vector $(x_1, x_2) = (0.6, 0.6)$ and $\alpha = 0.1$, the weight matrix is initialized from Figure 4 as

$$W = \begin{bmatrix} 0.3 & 0.2 & 0.1 & 0.8 & 0.4 \\ 0.5 & 0.6 & 0.7 & 0.9 & 0.2 \end{bmatrix}$$

Now we find the winner unit using square of Euclidean distance, i.e.,

$$D(j) = \sum_{i=1}^2 (w_{ij} - x_i)^2 = (w_{1j} - x_1)^2 + (w_{2j} - x_2)^2$$

For $j = 1$ to 5

$$\begin{aligned}D(1) &= (0.3 - 0.6)^2 + (0.5 - 0.6)^2 \\&= 0.09 + 0.01 = 0.1\end{aligned}$$

$$\begin{aligned}D(2) &= (0.2 - 0.6)^2 + (0.6 - 0.6)^2 \\&= 0.08 + 0 = 0.08\end{aligned}$$

$$\begin{aligned}D(3) &= (0.1 - 0.6)^2 + (0.7 - 0.6)^2 \\&= 0.25 + 0.01 = 0.26\end{aligned}$$

$$\begin{aligned}D(4) &= (0.8 - 0.6)^2 + (0.9 - 0.6)^2 \\&= 0.04 + 0.09 = 0.13\end{aligned}$$

$$\begin{aligned}D(5) &= (0.4 - 0.6)^2 + (0.2 - 0.6)^2 \\&= 0.04 + 0.16 = 0.2\end{aligned}$$

Since $D(2) = 0.08$ is the minimum value, the winner unit is $J = 2$. We now update the weights on the winner unit with $\alpha = 0.1$. The weight updation formula is given by

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

Substituting $J = 2$ in the equation above, we obtain

$$w_{i2}(\text{new}) = w_{i2}(\text{old}) + \alpha [x_i - w_{i2}(\text{old})]$$

For $i = 1$ to 2,

$$\begin{aligned}w_{12}(n) &= w_{12}(0) + \alpha [x_1 - w_{12}(0)] \\&= 0.2 + 0.1(0.6 - 0.2) = 0.24\end{aligned}$$

$$\begin{aligned}w_{22}(n) &= w_{22}(0) + \alpha [x_2 - w_{22}(0)] \\&= 0.6 + 0.1(0.6 - 0.6) = 0.6\end{aligned}$$

The new weight matrix is given by

$$W = \begin{bmatrix} 0.3 & 0.24 & 0.1 & 0.8 & 0.4 \\ 0.5 & 0.6 & 0.7 & 0.9 & 0.2 \end{bmatrix}$$

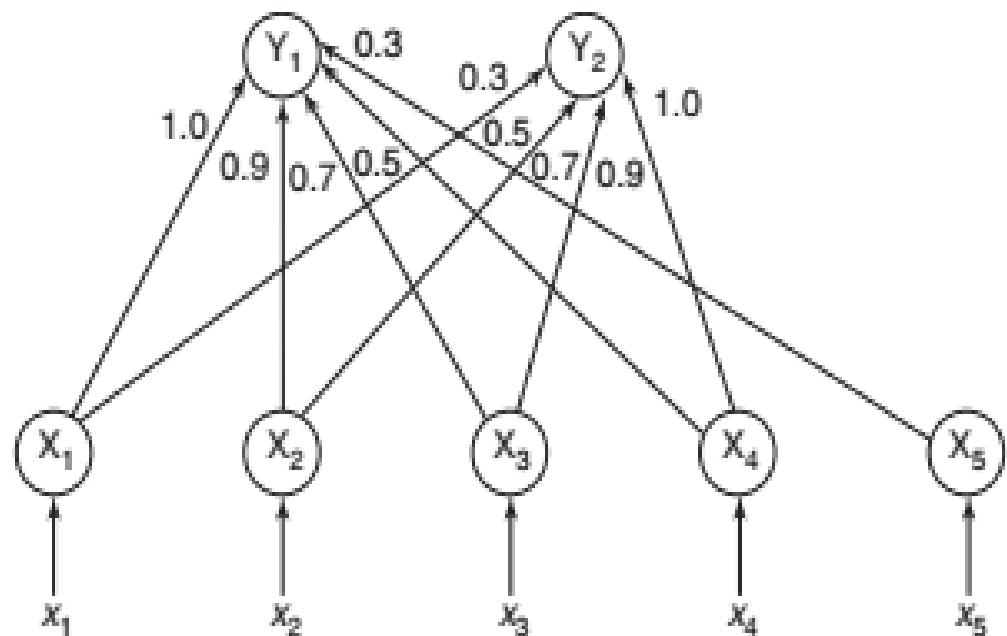
Practice Question

Consider a Kohonen self-organizing net with two cluster units and five input units. The weight vectors for the cluster units are given by

$$w_1 = [1.0 \ 0.9 \ 0.7 \ 0.5 \ 0.3]$$

$$w_2 = [0.3 \ 0.5 \ 0.7 \ 0.9 \ 1.0]$$

Use the square of the Euclidean distance to find the winning cluster unit for the input pattern $x = [0.0 \ 0.5 \ 1.0 \ 0.5 \ 0.0]$. Using a learning rate of 0.25, find the new weights for the winning unit.



Introduction to Fuzzy Logic

Dr. Shruti Mishra
School of Computer Sc. & Engg. (SCOPE)

Introduction

- The word **fuzzy** refers to things which are not clear or are vague.
- Fuzzy Logic resembles the human decision-making methodology.
- *It deals with vague and imprecise information.*
- The growth of fuzzy logic approach is to *handle ambiguity and uncertainty* existing in the complex problems.
- Fuzzy logic is *multivalued logic that deals with reasoning i.e. approximate rather than precise* (Antonym of this is Crisp set).

What is Fuzzy logic?

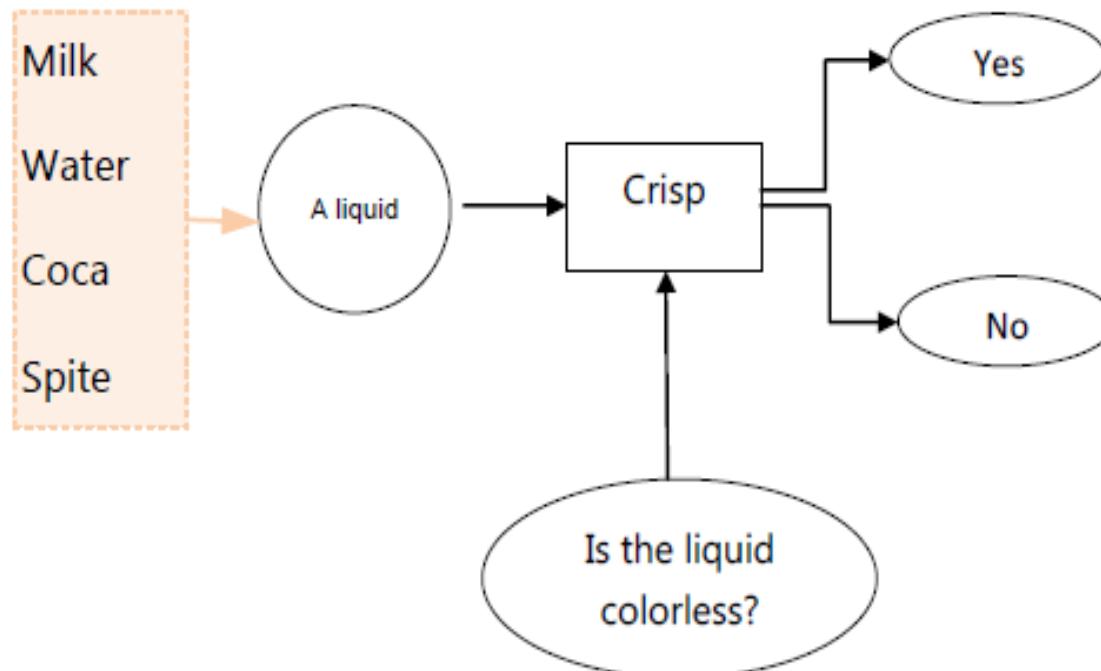
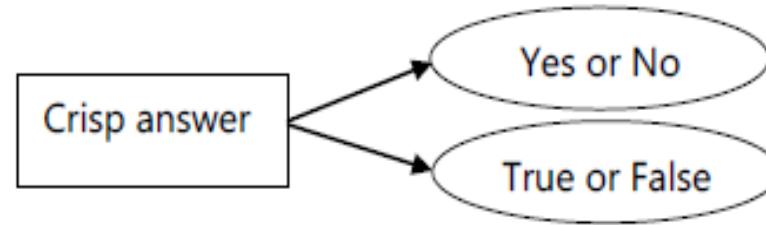
- Fuzzy logic is a mathematical language to express something.
- This means it has grammar, syntax, semantic like a language for communication.
- There are some other mathematical languages also known
 - **Relational algebra** (operations on sets)
 - **Boolean algebra** (operations on Boolean variables)
 - **Predicate logic** (operations on well formed formulae (wff), also called predicate propositions)
 - **Fuzzy logic** deals with **Fuzzy set**.

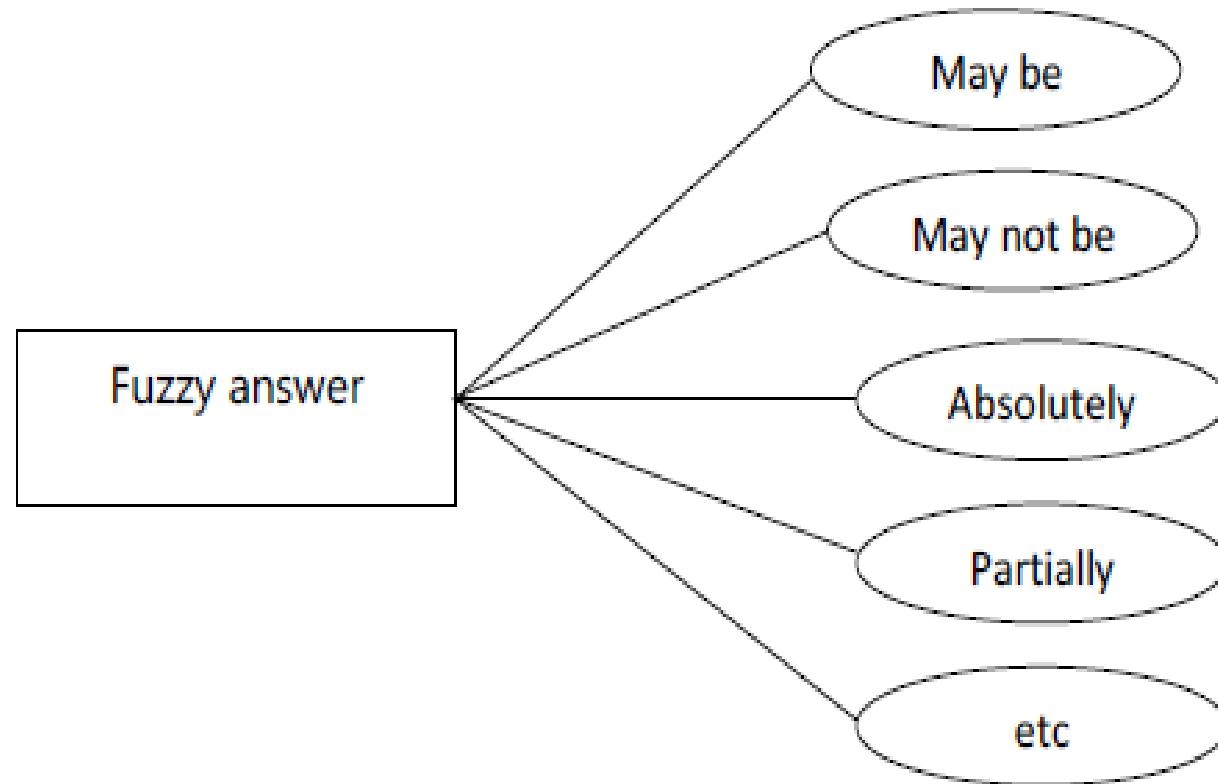
A brief history of Fuzzy Logic

- First time introduced by Lotfi Abdelli Zadeh (1965), University of California, Berkley, USA (1965).

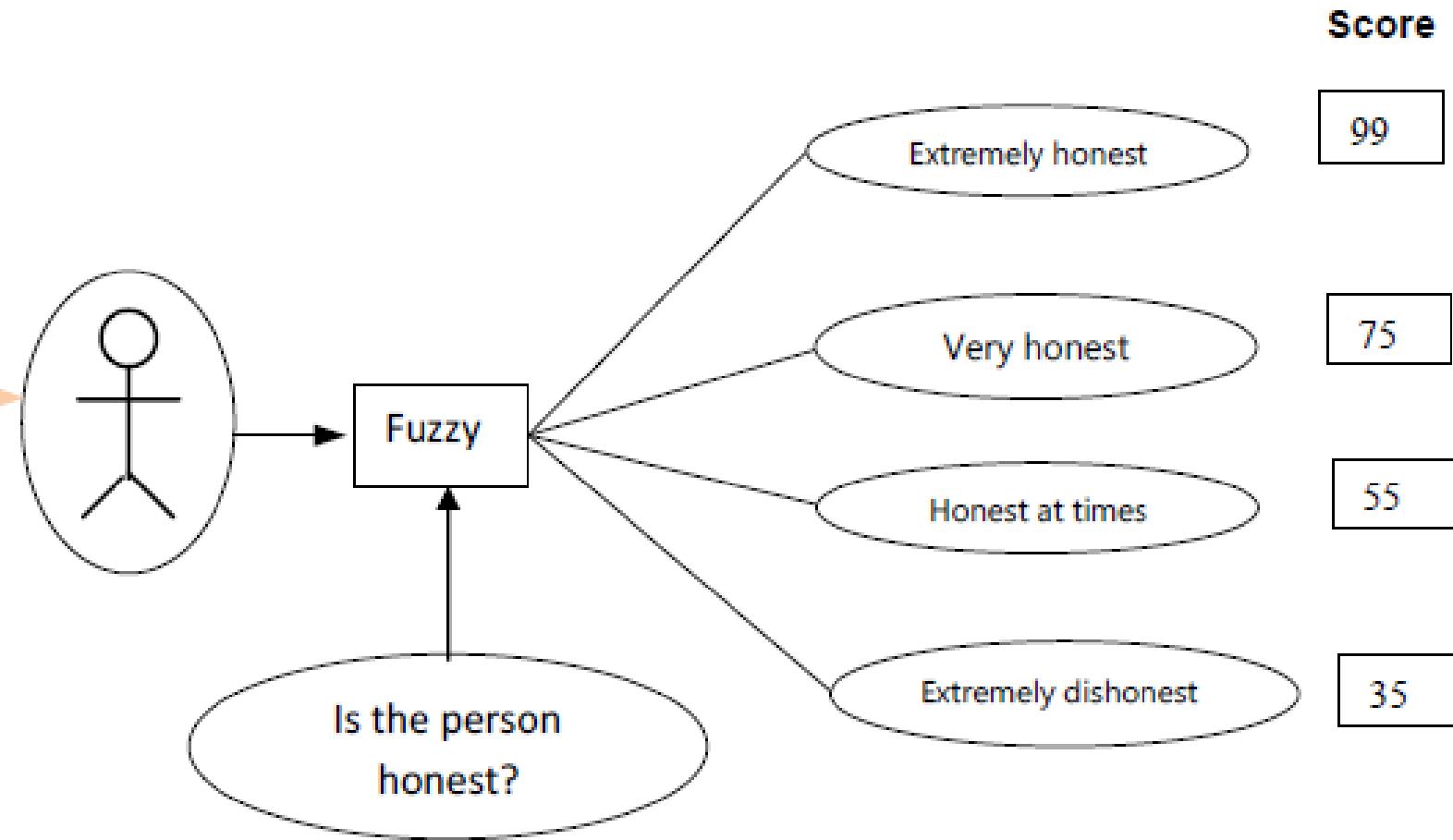
“ As the complexity of a system increases, it becomes more difficult and eventually impossible to make a precise statement about its behaviour, eventually at a point of complexity where the fuzzy logic method born in humans is the only way to get at the problem.”

Example : Fuzzy logic vs. Crisp logic





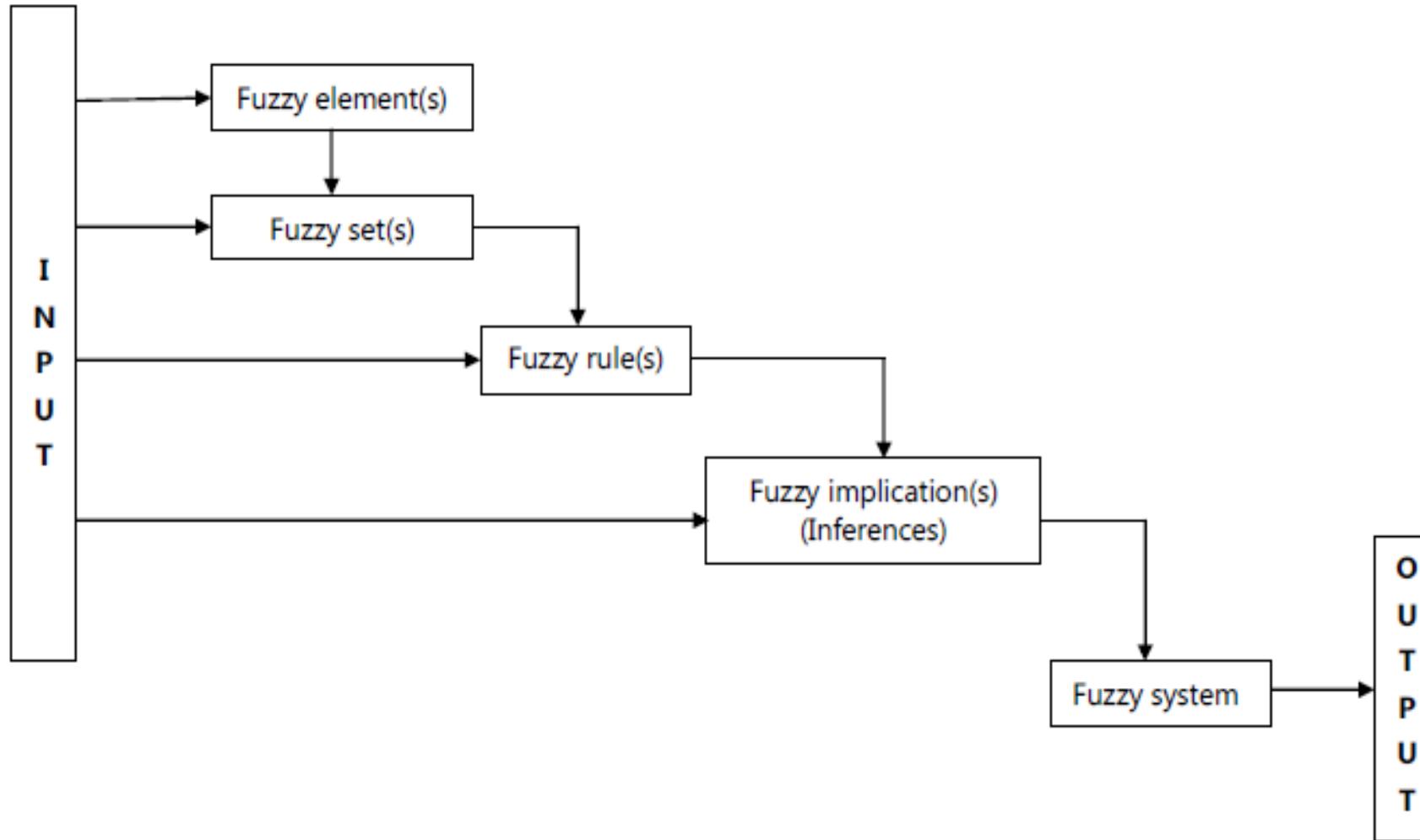
- Ankit
- Rajesh
- Santosh
- Kabita
- Salmon





**Our world is better
described with
fuzzily!**

Concept of Fuzzy System



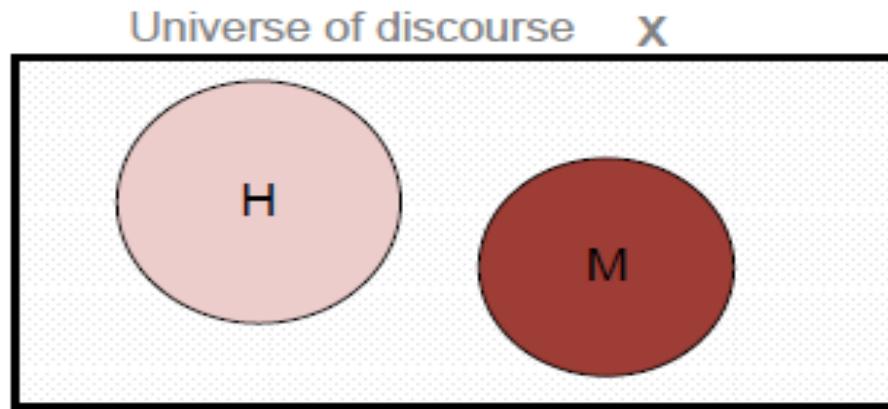
Concept of fuzzy set

To understand the concept of **fuzzy set** it is better, if we first clear our idea of **crisp set**.

X = The entire population of India.

H = All Hindu population = { $h_1, h_2, h_3, \dots, h_L$ }

M = All Muslim population = { $m_1, m_2, m_3, \dots, m_N$ }



Here, All are the sets of finite numbers of individuals.

Such a set is called **crisp set**.

Example of fuzzy set

Let us discuss about fuzzy set.

X = All students in IT60108.

S = All **Good students**.

S = { (s, g) | s ∈ X } and g(s) is a measurement of goodness of the student s.

Example:

S = { (Rajat, 0.8), (Kabita, 0.7), (Salman, 0.1), (Ankit, 0.9) } etc.

Fuzzy set vs. Crisp set

Crisp Set	Fuzzy Set
1. $S = \{ s \mid s \in X \}$	1. $F = (s, \mu) \mid s \in X$ and $\mu(s)$ is the degree of s .
2. It is a collection of elements.	2. It is collection of ordered pairs.
3. Inclusion of an element $s \in X$ into S is crisp, that is, has strict boundary yes or no .	3. Inclusion of an element $s \in X$ into F is fuzzy, that is, if present, then with a degree of membership .

Fuzzy set vs. Crisp set

Note: A crisp set is a fuzzy set, but, a fuzzy set is not necessarily a crisp set.

Example:

$$H = \{ (h_1, 1), (h_2, 1), \dots, (h_L, 1) \}$$

$$\text{Person} = \{ (p_1, 1), (p_2, 0), \dots, (p_N, 1) \}$$

In case of a crisp set, the elements are with extreme values of degree of membership namely either 1 or 0.

How to decide the degree of memberships of elements in a fuzzy set?

City	Bangalore	Bombay	Hyderabad	Kharagpur	Madras	Delhi
DoM	0.95	0.90	0.80	0.01	0.65	0.75

How the cities of comfort can be judged?

Few examples of fuzzy set

- High Temperature
- Low Pressure
- Color of Apple
- Sweetness of Orange
- Weight of Mango

Note: Degree of membership values lie in the range [0...1].

Fuzzy Sets

- It is an extension and generalization of crisp sets.
- **Important Property:** It allows partial membership.
- It has a degree of membership between 1 and 0.

A fuzzy set \underline{A} in the universe of discourse U can be defined as a set of ordered pairs and it is given by

$$\underline{A} = \{(x, \mu_{\underline{A}}(x)) \mid x \in U\}$$

where $\mu_{\underline{A}}(x)$ is the degree of membership of x in \underline{A} and it indicates the degree that x belongs to \underline{A} . The degree of membership $\mu_{\underline{A}}(x)$ assumes values in the range from 0 to 1, i.e., the membership is set to unit interval $[0, 1]$ or $\mu_{\underline{A}}(x) \in [0, 1]$.

Contd..

There are other ways of representation of fuzzy sets; all representations allow partial membership to be expressed. When the universe of discourse U is discrete and finite, fuzzy set \underline{A} is given as follows:

$$\underline{A} = \left\{ \frac{\mu_{\underline{A}}(x_1)}{x_1} + \frac{\mu_{\underline{A}}(x_2)}{x_2} + \frac{\mu_{\underline{A}}(x_3)}{x_3} + \dots \right\} = \left\{ \sum_{i=1}^n \frac{\mu_{\underline{A}}(x_i)}{x_i} \right\}$$

where “ n ” is a finite value. When the universe of discourse U is continuous and infinite, fuzzy set \underline{A} is given by

$$\underline{A} = \left\{ \int \frac{\mu_{\underline{A}}(x)}{x} \right\}$$

Note: The summation sign is an indication of discrete theoretic sign and not addition; similarly, the horizontal bar act as delimiter and not as quotient.

The collection of all fuzzy sets and fuzzy subsets on universe U is called **fuzzy power set** $\underline{P}(U)$. Since all the fuzzy sets can overlap, the cardinality of the fuzzy power set, $n_{\underline{P}(U)}$ is infinite, i.e., $n_{\underline{P}(U)} = \infty$.

On the basis of the above discussion we have

$$\underline{A} \subseteq U \Rightarrow \mu_{\underline{A}}(x) \leq \mu_U(u)$$

Also, for all $x \in U$

$$\mu_\emptyset(x) = 0; \quad \mu_U(x) = 1$$

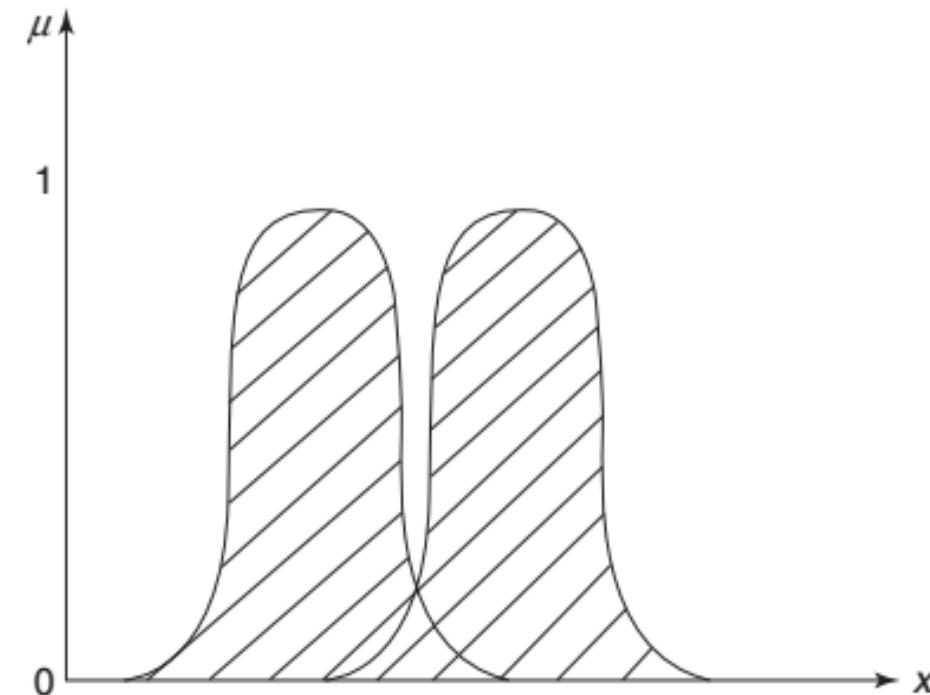
Fuzzy Set Operations

1) Union

The union of fuzzy sets \underline{A} and \underline{B} , denoted by $\underline{A} \cup \underline{B}$, is defined as

$$\mu_{\underline{A} \cup \underline{B}}(x) = \max[\mu_{\underline{A}}(x), \mu_{\underline{B}}(x)] = \mu_{\underline{A}}(x) \vee \mu_{\underline{B}}(x) \quad \text{for all } x \in U$$

where \vee indicates max operation. The Venn diagram for union operation of fuzzy sets \underline{A} and \underline{B} :

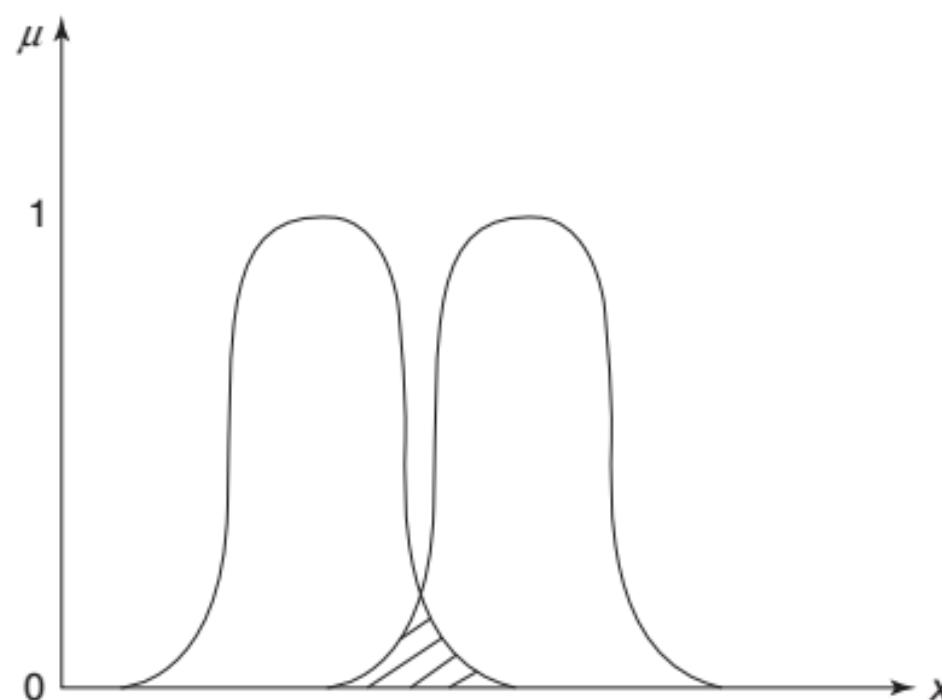


2) Intersection

The intersection of fuzzy sets \underline{A} and \underline{B} , denoted by $\underline{A} \cap \underline{B}$, is defined by

$$\mu_{\underline{A} \cap \underline{B}}(x) = \min[\mu_{\underline{A}}(x), \mu_{\underline{B}}(x)] = \mu_{\underline{A}}(x) \wedge \mu_{\underline{B}}(x) \quad \text{for all } x \in U$$

where \wedge indicates min operator. The Venn diagram for intersection operation of fuzzy sets \underline{A} and \underline{B}

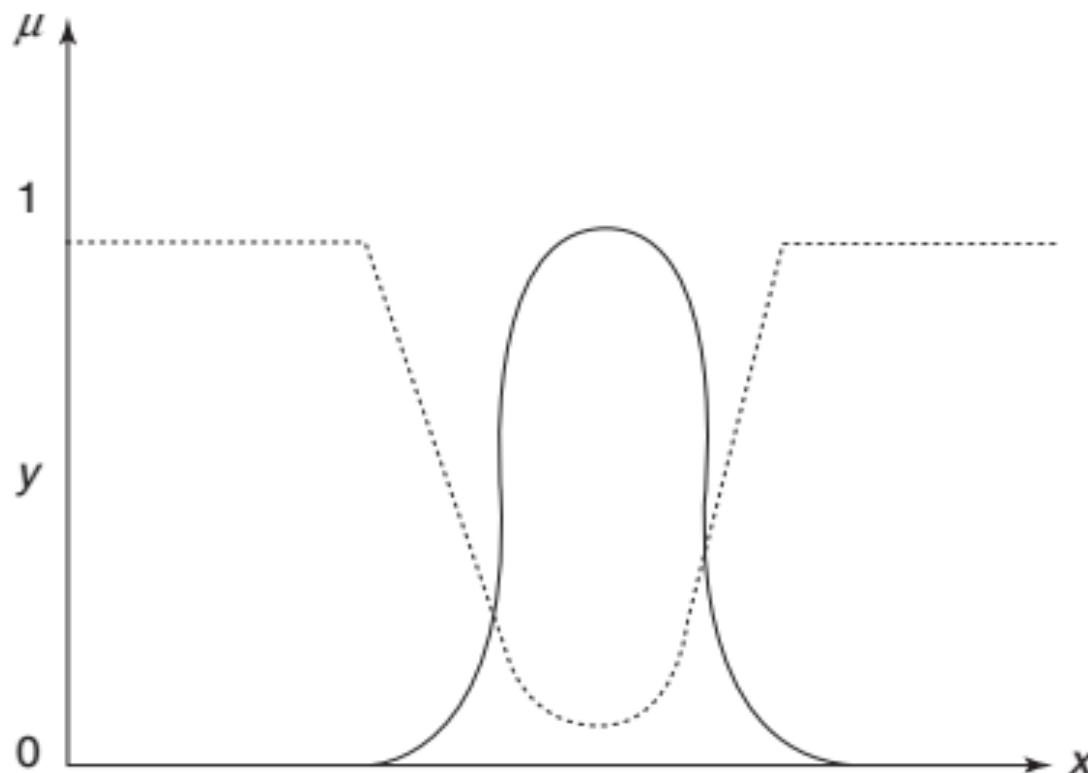


3) Complement

When $\mu_{\tilde{A}}(x) \in [0, 1]$, the complement of \tilde{A} , denoted as $\overline{\tilde{A}}$ is defined by

$$\mu_{\overline{\tilde{A}}}(x) = 1 - \mu_{\tilde{A}}(x) \quad \text{for all } x \in U$$

The Venn diagram for complement operation of fuzzy set \tilde{A} is shown in Figure 10-12.



Other Operations on Fuzzy Sets

1. *Algebraic sum:* The algebraic sum ($\tilde{A} + \tilde{B}$) of fuzzy sets, fuzzy sets \tilde{A} and \tilde{B} is defined as

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

2. *Algebraic product:* The algebraic product ($\tilde{A} \cdot \tilde{B}$) of two fuzzy sets \tilde{A} and \tilde{B} is defined as

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

3. *Bounded sum:* The bounded sum ($\tilde{A} \oplus \tilde{B}$) of two fuzzy sets \tilde{A} and \tilde{B} is defined as

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

4. *Bounded difference:* The bounded difference ($\tilde{A} \odot \tilde{B}$) of two fuzzy sets \tilde{A} and \tilde{B} is defined as

$$\mu_{\tilde{A} \odot \tilde{B}}(x) = \max\{0, \mu_{\tilde{A}}(x) - \mu_{\tilde{B}}(x)\}$$

Properties of Fuzzy Sets

Fuzzy sets follow the same properties as crisp sets except for the law of excluded middle and law of contradiction. That is, for fuzzy set \underline{A}

$$\underline{A} \cup \overline{\underline{A}} \neq U; \quad \underline{A} \cap \overline{\underline{A}} \neq \emptyset$$

Frequently used properties of fuzzy sets are given as follows:

1. Commutativity

$$\underline{A} \cup \underline{B} = \underline{B} \cup \underline{A}; \quad \underline{A} \cap \underline{B} = \underline{B} \cap \underline{A}$$

2. Associativity

$$\underline{A} \cup (\underline{B} \cup \underline{C}) = (\underline{A} \cup \underline{B}) \cup \underline{C}$$

$$\underline{A} \cap (\underline{B} \cap \underline{C}) = (\underline{A} \cap \underline{B}) \cap \underline{C}$$

3. Distributivity

$$\underline{A} \cup (\underline{B} \cap \underline{C}) = (\underline{A} \cup \underline{B}) \cap (\underline{A} \cup \underline{C})$$

$$\underline{A} \cap (\underline{B} \cup \underline{C}) = (\underline{A} \cap \underline{B}) \cup (\underline{A} \cap \underline{C})$$

4. Idempotency

$$\underline{A} \cup \underline{A} = \underline{A}; \quad \underline{A} \cap \underline{A} = \underline{A}$$

5. Identity

$$\underline{A} \cup \phi = \underline{A} \quad \text{and} \quad \underline{A} \cup U = U \text{(universal set)}$$

$$\underline{A} \cap \phi = \phi \quad \text{and} \quad \underline{A} \cap U = \underline{A}$$

6. Involution (double negation)

$$\overline{\underline{\underline{A}}} = \underline{A}$$

7. Transitivity

If $\underline{A} \subseteq \underline{B} \subseteq \underline{C}$, then $\underline{A} \subseteq \underline{C}$

8. De Morgan's law

$$\overline{\underline{A} \cup \underline{B}} = \underline{\overline{A}} \cap \underline{\overline{B}}; \quad \overline{\underline{A} \cap \underline{B}} = \underline{\overline{A}} \cup \underline{\overline{B}}$$

Problems

Consider two given fuzzy sets

$$\underline{A} = \left\{ \frac{1}{2} + \frac{0.3}{4} + \frac{0.5}{6} + \frac{0.2}{8} \right\}$$

$$\underline{B} = \left\{ \frac{0.5}{2} + \frac{0.4}{4} + \frac{0.1}{6} + \frac{1}{8} \right\}$$

Perform union, intersection, difference and complement over fuzzy sets \underline{A} and \underline{B} .

Solution: For the given fuzzy sets we have the following

(a) Union

$$\begin{aligned}\underline{A} \cup \underline{B} &= \max\{\mu_{\underline{A}}(x), \mu_{\underline{B}}(x)\} \\ &= \left\{ \frac{1}{2} + \frac{0.4}{4} + \frac{0.5}{6} + \frac{1}{8} \right\}\end{aligned}$$

(b) Intersection

$$\begin{aligned}\underline{A} \cap \underline{B} &= \min\{\mu_{\underline{A}}(x), \mu_{\underline{B}}(x)\} \\ &= \left\{ \frac{0.5}{2} + \frac{0.3}{4} + \frac{0.1}{6} + \frac{0.2}{8} \right\}\end{aligned}$$

(c) Complement

$$\underline{A}^c = 1 - \mu_{\underline{A}}(x) = \left\{ \frac{0}{2} + \frac{0.7}{4} + \frac{0.5}{6} + \frac{0.8}{8} \right\}$$

$$\underline{B}^c = 1 - \mu_{\underline{B}}(x) = \left\{ \frac{0.5}{2} + \frac{0.6}{4} + \frac{0.9}{6} + \frac{0}{8} \right\}$$

(d) Difference

$$\underline{A} | \underline{B} = \underline{A} \cap \underline{B}^c = \left\{ \frac{0.5}{2} + \frac{0.3}{4} + \frac{0.5}{6} + \frac{0}{8} \right\}$$

$$\underline{B} | \underline{A} = \underline{B} \cap \underline{A}^c = \left\{ \frac{0}{2} + \frac{0.4}{4} + \frac{0.1}{6} + \frac{0.8}{8} \right\}$$

Contd..

Given the two fuzzy sets

$$\tilde{B}_1 = \left\{ \frac{1}{1.0} + \frac{0.75}{1.5} + \frac{0.3}{2.0} + \frac{0.15}{2.5} + \frac{0}{3.0} \right\}$$

$$\tilde{B}_2 = \left\{ \frac{1}{1.0} + \frac{0.6}{1.5} + \frac{0.2}{2.0} + \frac{0.1}{2.5} + \frac{0}{3.0} \right\}$$

find the following:

- (a) $\tilde{B}_1 \cup \tilde{B}_2$; (b) $\tilde{B}_1 \cap \tilde{B}_2$; (c) $\overline{\tilde{B}_1}$;
- (d) $\overline{\tilde{B}_2}$; (e) $\tilde{B}_1 | \tilde{B}_2$; (f) $\overline{\tilde{B}_1 \cup \tilde{B}_2}$;
- (g) $\overline{\tilde{B}_1 \cap \tilde{B}_2}$; (h) $\tilde{B}_1 \cap \overline{\tilde{B}_1}$; (i) $\tilde{B}_1 \cup \overline{\tilde{B}_1}$;
- (j) $\tilde{B}_2 \cap \overline{\tilde{B}_2}$; (k) $\tilde{B}_2 \cup \overline{\tilde{B}_2}$

Consider two fuzzy sets

$$\tilde{A} = \left\{ \frac{0.2}{1} + \frac{0.3}{2} + \frac{0.4}{3} + \frac{0.5}{4} \right\}$$

$$\tilde{B} = \left\{ \frac{0.1}{1} + \frac{0.2}{2} + \frac{0.2}{3} + \frac{1}{4} \right\}$$

Find the algebraic sum, algebraic product, bounded sum and bounded difference of the given fuzzy sets.

Classical Relations and Fuzzy Relations

Dr. Shruti Mishra
School of Computer Sc. & Engg. (SCOPE)

Introduction

- Relationships between objects are the basic concepts involved in decision making and other dynamic system applications.
- Classical binary relation represents *the presence or absence of a connection or interaction or association between the elements of two sets.*
- Fuzzy binary relations are a *generalization of crisp binary relations and they allow various degrees of relationship between elements.*

Fuzzy Relation

A fuzzy relation between two sets X and Y is called binary fuzzy relation and is denoted by $R(X, Y)$. /

Let

$$\underline{X} = \{x_1, x_2, \dots, x_n\} \quad \text{and} \quad \underline{Y} = \{y_1, y_2, \dots, y_m\}$$

Fuzzy relation $\underline{R} = (\underline{X}, \underline{Y})$ can be expressed by an $n \times m$ matrix as follows:

$$\underline{R}(\underline{X}, \underline{Y}) = \begin{bmatrix} \mu_{\underline{R}}(x_1, y_1) & \mu_{\underline{R}}(x_1, y_2) & \mu_{\underline{R}}(x_1, y_m) \\ \mu_{\underline{R}}(x_2, y_1) & \mu_{\underline{R}}(x_2, y_2) & \mu_{\underline{R}}(x_2, y_m) \\ \vdots & \vdots & \vdots \\ \mu_{\underline{R}}(x_n, y_1) & \mu_{\underline{R}}(x_n, y_2) & \mu_{\underline{R}}(x_n, y_m) \end{bmatrix}$$

Domain and Range of Fuzzy Relation

The domain of a binary fuzzy relation $R(X, Y)$ is the fuzzy set, $\text{dom } R(X, Y)$, having the membership function as

$$\mu_{\text{domain } R}(x) = \max_{y \in Y} \mu_R(x, y) \quad \forall x \in X$$

The range of a binary fuzzy relation $R(X, Y)$ is the fuzzy set, $\text{ran } R(X, Y)$, having the membership function as

$$\mu_{\text{range } R}(y) = \max_{x \in X} \mu_R(x, y) \quad \forall y \in Y$$

Let

$$\underline{X} = \{x_1, x_2, x_3, x_4\} \quad \text{and} \quad \underline{Y} = \{y_1, y_2, y_3, y_4\}$$

Let \underline{R} be a relation from \underline{X} to \underline{Y} given by

$$\underline{R} = \frac{0.2}{(x_1, y_3)} + \frac{0.4}{(x_1, y_2)} + \frac{0.1}{(x_2, y_2)} + \frac{0.6}{(x_2, y_3)} + \frac{1.0}{(x_3, y_3)} + \frac{0.5}{(x_3, y_1)}$$

The corresponding fuzzy matrix for relation \underline{R} is

$$\underline{R} = \begin{bmatrix} & y_1 & y_2 & y_3 \\ x_1 & 0 & 0.4 & 0.2 \\ x_2 & 0 & 0.1 & 0.6 \\ x_3 & 0.5 & 0 & 1.0 \end{bmatrix}$$

Cardinality and Operations of Fuzzy Relation

The cardinality of fuzzy sets on any universe is infinity; hence the cardinality of a fuzzy relation between two or more universes is also infinity. This is mainly a result of the occurrence of partial membership in fuzzy sets and fuzzy relations.

The basic operations on fuzzy sets also apply on fuzzy relations. Let \underline{R} and \underline{S} be fuzzy relations on the Cartesian space $X \times Y$. The operations that can be performed on these fuzzy relations are described below:

1. Union

$$\mu_{\underline{R} \cup \underline{S}}(x, y) = \max[\mu_{\underline{R}}(x, y), \mu_{\underline{S}}(x, y)]$$

2. Intersection

$$\mu_{\underline{R} \cap \underline{S}}(x, y) = \min[\mu_{\underline{R}}(x, y), \mu_{\underline{S}}(x, y)]$$

3. Complement

$$\mu_{\underline{\underline{R}}}(x, y) = 1 - \mu_{\underline{R}}(x, y)$$

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

$$\mu_{[R \downarrow Y]}(x, y) = \max_x \mu_R(x, y)$$

The projection concept can be extended to an n -ary relation $R(x_1, x_2, \dots, x_n)$.

Fuzzy Composition

$$\underline{R} \subset X \times Y$$

The membership function of fuzzy relation is given by

$$\mu_{\underline{R}}(x, y) = \mu_{A \times B}(x, y) = \min[\mu_A(x), \mu_B(y)]$$

The Cartesian product is not an operation similar to arithmetic product. Cartesian product $\underline{R} = \underline{A} \times \underline{B}$ is obtained in the same way as the cross-product of two vectors. For example, for a fuzzy set \underline{A} that has three elements (hence column vector of size 3×1) and a fuzzy set \underline{B} that has four elements (hence row vector of size 1×4), the resulting fuzzy relation \underline{R} will be represented by a matrix of size 3×4 , i.e., \underline{R} will have three rows and four columns.

Fuzzy Composition Techniques

1. Fuzzy max-min composition
2. Fuzzy max-product composition

There also exists fuzzy min-max composition method, but the most commonly used technique is fuzzy max-min composition. Let \underline{R} be fuzzy relation on Cartesian space $X \times Y$, and \underline{S} be fuzzy relation on Cartesian space $Y \times Z$.

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

$$\begin{aligned}\mu_T(x, z) &= \mu_{\underline{R} \circ \underline{S}}(x, z) = \max_{y \in Y} \{\min[\mu_R(x, y), \mu_S(y, z)]\} \\ &= \vee_{y \in Y} [\mu_R(x, y) \wedge \mu_S(y, z)] \quad \forall x \in X, z \in Z\end{aligned}$$

The min-max composition of $R(X, Y)$ and $S(Y, Z)$, denoted as $R(X, Y) \circ S(Y, Z)$, is defined by $T(X, Z)$ as

$$\mu_T(x, z) = \mu_{\underline{R} \circ \underline{S}}(x, z) = \min_{y \in Y} \{\max[\mu_R(x, y), \mu_S(y, z)]\} = \wedge_{y \in Y} [\mu_R(x, y) \vee \mu_S(y, z)] \quad \forall x \in X, z \in Z$$

From the above definitions it can be noted that

$$\overline{R(X, Y) \circ S(Y, Z)} = \overline{R(X, Y)} \circ \overline{S(Y, Z)}$$

The max-product composition of $R(X, Y)$ and $S(Y, Z)$, denoted as $R(X, Y) \cdot S(Y, Z)$, is defined by

$T(X, Z)$ as

$$\begin{aligned}\mu_{\underline{T}}(x, z) &= \mu_{\underline{R} \cdot \underline{S}}(x, z) = \max_{y \in Y} [\mu_{\underline{R}}(x, y) \cdot \mu_{\underline{S}}(y, z)] \\ &= \vee_{y \in Y} [\mu_{\underline{R}}(x, y) \cdot \mu_{\underline{S}}(y, z)]\end{aligned}$$

The properties of fuzzy composition can be given as follows:

$$\begin{aligned}\underline{R} \circ \underline{S} &\neq \underline{S} \circ \underline{R} \\ (\underline{R} \circ \underline{S})^{-1} &= \underline{S}^{-1} \circ \underline{R}^{-1} \\ (\underline{R} \circ \underline{S}) \circ \underline{M} &= \underline{R} \circ (\underline{S} \circ \underline{M})\end{aligned}$$

Problems

Consider the following two fuzzy sets:

$$\underline{A} = \left\{ \frac{0.3}{x_1} + \frac{0.7}{x_2} + \frac{1}{x_3} \right\}$$

$$\text{and } \underline{B} = \left\{ \frac{0.4}{y_1} + \frac{0.9}{y_2} \right\}$$

Perform the Cartesian product over these given fuzzy sets.

Solution: The fuzzy Cartesian product performed over fuzzy sets \underline{A} and \underline{B} results in fuzzy relation \underline{R} given by $\underline{R} = \underline{A} \times \underline{B}$. Hence

$$\underline{R} = \begin{bmatrix} 0.3 & 0.3 \\ 0.4 & 0.7 \\ 0.4 & 0.9 \end{bmatrix}$$

The calculation for \underline{R} is as follows:

$$\begin{aligned}\mu_{\underline{R}}(x_1, y_1) &= \min[\mu_{\underline{A}}(x_1), \mu_{\underline{B}}(y_1)] \\ &= \min(0.3, 0.4) = 0.3\end{aligned}$$

$$\begin{aligned}\mu_{\underline{R}}(x_1, y_2) &= \min[\mu_{\underline{A}}(x_1), \mu_{\underline{B}}(y_2)] \\ &= \min(0.3, 0.9) = 0.3\end{aligned}$$

$$\begin{aligned}\mu_{\underline{R}}(x_2, y_1) &= \min[\mu_{\underline{A}}(x_2), \mu_{\underline{B}}(y_1)] \\ &= \min(0.7, 0.4) = 0.4\end{aligned}$$

$$\begin{aligned}\mu_{\underline{R}}(x_2, y_2) &= \min[\mu_{\underline{A}}(x_2), \mu_{\underline{B}}(y_2)] \\ &= \min(0.7, 0.9) = 0.7\end{aligned}$$

$$\begin{aligned}\mu_{\underline{R}}(x_3, y_1) &= \min[\mu_{\underline{A}}(x_3), \mu_{\underline{B}}(y_1)] \\ &= \min(1, 0.4) = 0.4\end{aligned}$$

$$\begin{aligned}\mu_{\underline{R}}(x_3, y_2) &= \min[\mu_{\underline{A}}(x_3), \mu_{\underline{B}}(y_2)] \\ &= \min(1, 0.9) = 0.9\end{aligned}$$

Thus, the Cartesian product between fuzzy sets \underline{A} and \underline{B} are obtained.

Two fuzzy relations are given by

$$\underline{R} = \begin{matrix} & y_1 & y_2 \\ x_1 & 0.6 & 0.3 \\ x_2 & 0.2 & 0.9 \end{matrix}$$

and

$$\underline{S} = \begin{matrix} & z_1 & z_2 & z_3 \\ y_1 & 1 & 0.5 & 0.3 \\ y_2 & 0.8 & 0.4 & 0.7 \end{matrix}$$

Obtain fuzzy relation \underline{T} as a composition between the fuzzy relations.

Solution: The composition between two given fuzzy relations is performed in two ways as

- (a) Max-min composition
- (b) Max-product composition
- (a) Max-min composition

$$\underline{T} = \underline{R} \circ \underline{S} = \begin{matrix} & z_1 & z_2 & z_3 \\ x_1 & 0.6 & 0.5 & 0.3 \\ x_2 & 0.8 & 0.4 & 0.7 \end{matrix}$$

The calculations for obtaining \underline{T} are as follows:

$$\mu_{\underline{T}}(x_1, z_1) = \max\{\min[\mu_{\underline{R}}(x_1, y_1), \mu_{\underline{S}}(y_1, z_1)]\},$$

$$\min[\mu_{\underline{R}}(x_1, y_2), \mu_{\underline{S}}(y_2, z_1)]\}$$

$$= \max[\min(0.6, 1), \min(0.3, 0.8)]$$

$$= \max(0.6, 0.3) = 0.6$$

$$\mu_{\underline{T}}(x_1, z_2) = \max[\min(0.6, 0.5), \min(0.3, 0.4)]$$

$$= \max(0.5, 0.3) = 0.5$$

$$\mu_{\underline{T}}(x_1, z_3) = \max[\min(0.6, 0.3), \min(0.3, 0.7)]$$

$$= \max(0.3, 0.3) = 0.3$$

$$\mu_{\underline{T}}(x_2, z_1) = \max[\min(0.2, 1), \min(0.9, 0.8)]$$

$$= \max(0.2, 0.8) = 0.8$$

$$\mu_{\underline{T}}(x_2, z_2) = \max[\min(0.2, 0.5), \min(0.9, 0.4)]$$

$$= \max(0.2, 0.4) = 0.4$$

$$\mu_{\underline{T}}(x_2, z_3) = \max[\min(0.2, 0.3), \min(0.9, 0.7)]$$

$$= \max(0.2, 0.7) = 0.7$$

(b) Max-product composition

$$\underline{T} = R \cdot S$$

Calculations for \underline{T} are as follows:

$$\begin{aligned}\mu_{\underline{T}}(x_1, z_1) &= \max\{\mu_R(x_1, y_1) \cdot \mu_S(y_1, z_1), \\ &\quad [\mu_R(x_1, y_2) \cdot \mu_S(y_2, z_1)]\} \\ &= \max(0.6, 0.24) = 0.6\end{aligned}$$

$$\begin{aligned}\mu_{\underline{T}}(x_1, z_2) &= \max[(0.6 \times 0.5), (0.3 \times 0.4)] \\ &= \max(0.3, 0.12) = 0.3\end{aligned}$$

$$\begin{aligned}\mu_{\underline{T}}(x_1, z_3) &= \max[(0.6 \times 0.3), (0.3 \times 0.7)] \\ &= \max(0.18, 0.21) = 0.21\end{aligned}$$

$$\begin{aligned}\mu_{\underline{T}}(x_2, z_1) &= \max[(0.2 \times 1), (0.9 \times 0.8)] \\ &= \max(0.2, 0.72) = 0.72\end{aligned}$$

$$\begin{aligned}\mu_{\underline{T}}(x_2, z_2) &= \max[(0.2 \times 0.5), (0.9 \times 0.4)] \\ &= \max(0.1, 0.36) = 0.36\end{aligned}$$

$$\begin{aligned}\mu_{\underline{T}}(x_2, z_3) &= \max[(0.2 \times 0.3), (0.9 \times 0.7)] \\ &= \max(0.06, 0.63) = 0.63\end{aligned}$$

The fuzzy relation, \underline{T} by max-product composition is given as

$$\underline{T} = \begin{matrix} & z_1 & z_2 & z_3 \\ \begin{matrix} x_1 \\ x_2 \end{matrix} & \left[\begin{matrix} 0.6 & 0.3 & 0.21 \\ 0.72 & 0.36 & 0.63 \end{matrix} \right] \end{matrix}$$

For a speed control of DC motor, the membership functions of series resistance, armature current and speed are given as follows:

$$\underline{R_s} = \left\{ \frac{0.4}{30} + \frac{0.6}{60} + \frac{1.0}{100} + \frac{0.1}{120} \right\}$$

$$\underline{I_a} = \left\{ \frac{0.2}{20} + \frac{0.3}{40} + \frac{0.6}{60} + \frac{0.8}{80} + \frac{1.0}{100} + \frac{0.2}{120} \right\}$$

$$\underline{N} = \left\{ \frac{0.35}{500} + \frac{0.67}{1000} + \frac{0.97}{1500} + \frac{0.25}{1800} \right\}$$

Compute relation \underline{T} for relating series resistance to motor speed, i.e., $\underline{R_s}$ to \underline{N} . Perform max-min composition only.

Solution: For relating series resistance to motor-speed, i.e., \underline{R}_{se} to \underline{N} , we have to perform the following operations – two fuzzy cross-products and one fuzzy composition (max-min):

$$\underline{R} = \underline{R}_{se} \times \underline{I}_a$$

$$\underline{S} = \underline{I}_a \times \underline{N}$$

$$\underline{T} = \underline{R} \circ \underline{S}$$

Relation \underline{R} is obtained as the Cartesian product of \underline{R}_{se} and \underline{I}_a , i.e.,

$$\underline{R} = \underline{R}_{se} \times \underline{I}_a$$

$$= \begin{matrix} & 20 & 40 & 60 & 80 & 100 & 120 \\ \begin{bmatrix} 30 \\ 60 \\ 100 \\ 120 \end{bmatrix} & \begin{bmatrix} 0.2 & 0.3 & 0.4 & 0.4 & 0.4 & 0.2 \\ 0.2 & 0.3 & 0.6 & 0.6 & 0.6 & 0.2 \\ 0.2 & 0.3 & 0.6 & 0.8 & 1.0 & 0.2 \\ 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 \end{bmatrix} \end{matrix}$$

Relation \underline{S} is obtained as the Cartesian product of \underline{I}_a and \underline{N} , i.e.,

$$\underline{S} = \underline{I}_a \times \underline{N} = \begin{matrix} & 500 & 1000 & 1500 & 1800 \\ \begin{bmatrix} 20 \\ 40 \\ 60 \\ 80 \\ 100 \\ 200 \end{bmatrix} & \begin{bmatrix} 0.2 & 0.2 & 0.2 & 0.2 \\ 0.3 & 0.3 & 0.3 & 0.25 \\ 0.35 & 0.6 & 0.6 & 0.25 \\ 0.35 & 0.67 & 0.8 & 0.25 \\ 0.35 & 0.67 & 0.97 & 0.25 \\ 0.2 & 0.2 & 0.2 & 0.2 \end{bmatrix} \end{matrix}$$

Relation \underline{T} is obtained as the composition between relations \underline{R} and \underline{S} , i.e.,

$$\underline{T} = \underline{R} \circ \underline{S} = \begin{matrix} & 500 & 1000 & 1500 & 1800 \\ \begin{bmatrix} 30 \\ 60 \\ 100 \\ 120 \end{bmatrix} & \begin{bmatrix} 0.35 & 0.4 & 0.4 & 0.25 \\ 0.35 & 0.6 & 0.6 & 0.25 \\ 0.35 & 0.67 & 0.97 & 0.25 \\ 0.1 & 0.1 & 0.1 & 0.1 \end{bmatrix} \end{matrix}$$

Consider two fuzzy sets given by

$$\underline{A} = \left\{ \frac{1}{\text{low}} + \frac{0.2}{\text{medium}} + \frac{0.5}{\text{high}} \right\}$$

$$\underline{B} = \left\{ \frac{0.9}{\text{positive}} + \frac{0.4}{\text{zero}} + \frac{0.9}{\text{negative}} \right\}$$

- (a) Find the fuzzy relation for the Cartesian product of \underline{A} and \underline{B} , i.e., $\underline{R} = \underline{A} \times \underline{B}$.
- (b) Introduce a fuzzy set \underline{C} given by

$$\underline{C} = \left\{ \frac{0.1}{\text{low}} + \frac{0.2}{\text{medium}} + \frac{0.7}{\text{high}} \right\}$$

Find the relation between \underline{C} and \underline{B} using

Cartesian product, i.e., find $\underline{S} = \underline{C} \times \underline{B}$.

- (c) Find $\underline{C} \circ \underline{R}$, using max-min composition.
- (d) Find $\underline{C} \circ \underline{S}$, using max-min composition.

- (a) The Cartesian product between \underline{A} and \underline{B} is obtained as

$$\begin{aligned} \underline{R} &= \underline{A} \times \underline{B} = \min[\mu_{\underline{A}}(x), \mu_{\underline{B}}(y)] \\ &\quad \text{positive zero negative} \\ &\quad \begin{array}{ccc} \text{low} & \left[\begin{array}{ccc} 0.9 & 0.4 & 0.9 \end{array} \right] \\ \text{medium} & \left[\begin{array}{ccc} 0.2 & 0.2 & 0.2 \end{array} \right] \\ \text{high} & \left[\begin{array}{ccc} 0.5 & 0.4 & 0.5 \end{array} \right] \end{array} \end{aligned}$$

- (b) The new fuzzy set is

$$\underline{C} = \left\{ \frac{0.1}{\text{low}} + \frac{0.2}{\text{medium}} + \frac{0.7}{\text{high}} \right\}$$

The Cartesian product between \underline{C} and \underline{B} is obtained as

$$\begin{aligned} \underline{S} &= \underline{C} \times \underline{B} = \min[\mu_{\underline{C}}(x), \mu_{\underline{B}}(y)] \\ &\quad \text{positive zero negative} \end{aligned}$$

$$\begin{aligned} &\quad \text{low} \left[\begin{array}{ccc} 0.1 & 0.1 & 0.1 \end{array} \right] \\ &= \text{medium} \left[\begin{array}{ccc} 0.2 & 0.2 & 0.2 \end{array} \right] \\ &\quad \text{high} \left[\begin{array}{ccc} 0.7 & 0.4 & 0.7 \end{array} \right] \end{aligned}$$

(c)

$$\begin{aligned} \underline{C} \circ \underline{R} &= [0.1 \ 0.2 \ 0.7]_{3 \times 3} \left[\begin{array}{ccc} 0.9 & 0.4 & 0.9 \\ 0.2 & 0.2 & 0.2 \\ 0.5 & 0.4 & 0.5 \end{array} \right]_{3 \times 3} \\ &= [0.5 \ 0.4 \ 0.5] \end{aligned}$$

For instance,

$$\begin{aligned} \mu_{\underline{C} \circ \underline{R}}(x_1, y_1) &= \max[\min(0.1, 0.9), \min(0.2, 0.2), \\ &\quad \min(0.7, 0.5)] \\ &= \max(0.1, 0.2, 0.5) = 0.5 \end{aligned}$$

(d)

$$\begin{aligned} \underline{C} \circ \underline{S} &= [0.1 \ 0.2 \ 0.7]_{3 \times 3} \left[\begin{array}{ccc} 0.1 & 0.1 & 0.1 \\ 0.2 & 0.2 & 0.2 \\ 0.7 & 0.4 & 0.7 \end{array} \right]_{3 \times 3} \\ &= [0.7 \ 0.4 \ 0.7] \end{aligned}$$

Hence max-min composition was used to find the relations.

Tolerance and Equivalence Relation

The three characteristic properties of relations are: *reflexivity, symmetry and transitivity.*

- 1 A relation is said to be *reflexive* if every vertex in the graph originates a single loop.
- 2 A relation is said to be *symmetric* if for every edge pointing from vertex i to vertex j , there is an edge pointing in the opposite direction, ie, from vertex j to i .
- 3 A relation is said to be *transitive* if for every pair of edges – one pointing from vertex i to vertex j and the other pointing from vertex j to vertex k , then there is an edge pointing from vertex i to vertex k .

PROPERTIES OF RELATIONS

- REFLEXIVITY

When a relation is reflexive, every vertex in the graph originates a single loop.



$$B = \{1, 2\}$$

$$R = \{(1, 1), (2, 2)\}$$

$$R = \{(x, x) \mid x \in B\}$$

- SYMMETRY

If a relation is symmetric, then in the graph for every edge pointing from vertex i to vertex j , there is an edge pointing in the opposite direction, that is, from vertex j to vertex i.

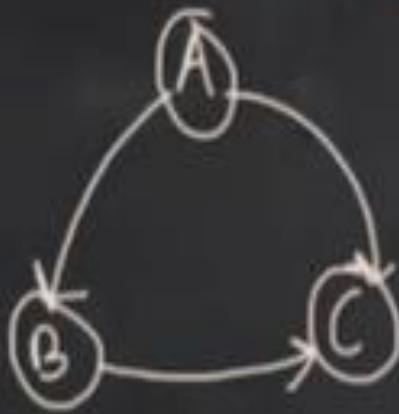


$$(x, y) \in R$$

$$(y, x) \in R$$

• TRANSITIVITY

When a relation is transitive, then for every pair of edges in the graph, one pointing from vertex i to vertex j and the other from vertex j to vertex k, there is an edge pointing from vertex i directly to vertex k.



$$(x, y) \in R$$

$$(y, z) \in R$$

$$(x, z) \in R$$

Classical Equivalence Relations

Let relation R on a universe X be a relation from X to X , is an *equivalence relation* if the following 3 properties are satisfied:

1 Reflexivity

$$\chi_R(x_i, x_i) = 1 \text{ or } (x_i, x_i) \in R$$

2 Symmetry

$$\begin{aligned}\chi_R(x_i, x_j) &= \chi_R(x_j, x_i) \\ ie, (x_i, x_j) \in R &\Rightarrow (x_j, x_i) \in R\end{aligned}$$

3 Transitivity

$$\chi_R(x_i, x_j) \text{ and } \chi_R(x_j, x_k) = 1, \text{ so, } \chi_R(x_i, x_k) = 1$$

Classical Tolerance Relations

A *tolerance relation* R_1 on universe X is one where the only properties of reflexivity and symmetry are satisfied.

It can also be called as *proximity relation*.

An equivalence relation, R , can be formed from tolerance relation R_1 by $(n - 1)$ compositions within itself:

$$R_1^{n-1} = R_1 \circ R_1 \circ \dots \circ R_1 = R$$

where n is the cardinality of the set that defines R_1 .

Example: The question below is to see whether equivalence relation exists or not. If not convert to equivalence relation.

	x_1	x_2	x_3	x_4	x_5
x_1	1	1	0	0	0
x_2	1	1	0	0	1
x_3	0	0	1	0	0
x_4	0	0	0	1	0
x_5	0	1	0	0	1

Step 1: The relation R is reflexive as the diagonal elements have unity or 1.

Step 2: To find if the relation R is symmetric or not, we need to find the transpose of R (R^T), and see that $R = R^T$.

Transpose of R \Rightarrow

	x_1	x_2	x_3	x_4	x_5
x_1	1	1	0	0	0
x_2	1	1	0	0	1
x_3	0	0	1	0	0
x_4	0	0	0	1	0
x_5	0	1	0	0	1

R is symmetric

Step 3: To find if the relation R is transitive or not.

$$\begin{aligned}\chi(a_1b) &= x & (x_1, x_2) &= 1 \\ \chi(b_1c) &= x & (x_2, x_5) &= 1 \\ \chi(a_1c) &= x & (x_1, x_5) &= 0\end{aligned}$$

This is not a transitive. Hence, this Relation R, is a tolerance or proximity relation but not equivalence relation.

We need to make it equivalence relation by performing composition of R.

Step 4: The relation R' is the composition.

$$R' = R \circ R$$
$$R = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 & x_5 \\ x_1 & 1 & 1 & 0 & 0 & 1 \\ x_2 & 1 & 1 & 0 & 0 & 1 \\ x_3 & 0 & 0 & 1 & 0 & 0 \\ x_4 & 0 & 0 & 0 & 1 & 0 \\ x_5 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

Step 5: The relation R is **reflexive** as the diagonal elements have unity or 1.

Step 6: The relation R is **symmetric** as the transpose elements of R' is equal to R' .

Step 7: The relation R is also **transitive in nature** as proved below:

$$\begin{aligned} x_1, x_2 &= 1 \\ x_2, x_5 &= 1 \\ x_1, x_5 &= 1 \end{aligned}$$



1) Reflexive
2) Symmetric
3) Transitive

Equivalence relation

Fuzzy Equivalence Relations

The *equivalence relation* can also be called *similarity relation*.

Let relation R on a universe X be a relation from X to X , is an *fuzzy equivalence relation* if the following 3 properties are satisfied:

1 Reflexivity

$$\mu_R(x_i, x_i) = 1 \forall x \in X$$

If this is not the case for few $x \in X$, then $R(X, X)$ is said to be *irreflexive*.

2 Symmetry

$$\mu_R(x_i, x_j) = \mu_R(x_j, x_i) \forall x_i, x_j \in X$$

If this is not the case for few $x_i, x_j \in X$, then $R(X, X)$ is called *asymmetric*.

3 Transitivity

$$\mu_R(x_i, x_j) = \lambda_1 \text{ and } \mu_R(x_j, x_k) = \lambda_2 \Rightarrow \mu_R(x_i, x_k) = \lambda \quad \text{where } \lambda \geq \min(\lambda_1, \lambda_2)$$

Fuzzy Tolerance Relations

A binary fuzzy relation that possesses the properties of reflexivity and symmetry is called *fuzzy tolerance relation*.

It can also be called as *resemblance relation*.

A fuzzy equivalence relation, R , can be formed from fuzzy tolerance relation R_1 by $(n - 1)$ compositions within itself:

$$R_1^{n-1} = R_1 \circ R_1 \circ \dots \circ R_1 = R$$

where n is the cardinality of the set that defines R_1 .

Example: The question below is to see whether equivalence relation exists or not. If not convert to equivalence relation.

$$R_1 = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 1 & 1 & 0.8 & 0 & 0.1 & 0.2 \\ 2 & 0.8 & 1 & 0.4 & 0 & 0.9 \\ 3 & 0 & 0.4 & 1 & 0 & 0 \\ 4 & 0.1 & 0 & 0 & 1 & 0.5 \\ 5 & 0.2 & 0.9 & 0 & 0.5 & 1 \end{bmatrix}$$

Step 1: The **relation R** is **reflexive** as the diagonal elements have unity or 1.

Step 2: To find if the **relation R** is **symmetric** or not, we need to find the transpose of **R** (R^T), and see that $R = R^T$.

Step 3: To find if the **relation R** is **transitive** or not.

$$\begin{array}{l} 1,2 \Rightarrow 0.8 \quad \gamma_1 \\ 2,5 \Rightarrow 0.9 \quad \gamma_2 \\ 1,5 \Rightarrow 0.2 \quad \gamma \\ \hline \end{array} \quad \gamma \geq \min(\gamma_1, \gamma_2) \quad 0.8 > 0.2$$

This is **not a transitive**. Hence, this Relation **R**, is a tolerance or proximity relation but **not equivalence relation**.

We need to make it **equivalence relation by performing composition of R**.

Step 4: The relation R' is the composition.

$$R' = R_1 \circ R_1$$

$$= \begin{matrix} & 1 & 2 & 3 & 4 & 5 \\ 1 & 1 & 0.8 & 0.4 & 0.2 & 0.8 \\ 2 & 0.8 & 1 & 0.4 & 0.5 & 0.9 \\ 3 & 0.4 & 0.4 & 1 & 0 & 0.4 \\ 4 & 0.2 & 0.5 & 0 & 1 & 0.5 \\ 5 & 0.8 & 0.9 & 0.4 & 0.5 & 1 \end{matrix}$$

Step 5: The relation R is reflexive as the diagonal elements have unity or 1.

Step 6: The relation R is symmetric as the transpose elements of R' is equal to R' .

Step 7: The relation R is also transitive in nature as proved below:

Consider X_1 and X_2 and X_3 and X_4

$$\lambda_1 = 0.8 \quad \lambda_2 = 0.5 \\ \lambda_3 = 0.2$$

But still its not transitive perform the composition again.

$$R'' = R' \circ R'_1$$

$$= \begin{matrix} & 1 & 2 & 3 & 4 & 5 \\ 1 & 1 & 0.8 & 0.4 & 0.5 & 0.8 \\ 2 & 0.8 & 1 & 0.4 & 0.5 & 0.9 \\ 3 & 0.4 & 0.4 & 1 & 0.4 & 0.4 \\ 4 & 0.5 & 0.5 & 0.4 & 1 & 0.5 \\ 5 & 0.8 & 0.9 & 0.4 & 0.5 & 1 \end{matrix}$$

Step 8: The relation R is reflexive as the diagonal elements have unity or 1.

Step 9: The relation R is symmetric as the transpose elements of R'' is equal to R'' .

Step 10: The relation R is transitive. If we consider x_1 and x_3 and x_3 and x_4 . Then x_1 and x_4 . The min is found to be less than λ

$$\lambda_1 = 0.4 \\ \lambda_2 = 0.4 \\ \lambda = 0.5$$

Membership Functions

Dr. Shruti Mishra
School of Computer Sc. & Engg. (SCOPE)

Introduction

- Membership Functions (MFs) defines the fuzziness in a fuzzy set irrespective of the elements in the set, which are discrete or continuous.
- **Features of MFs**

The membership function defines all the information contained in a fuzzy set; hence it is important to discuss the various features of the membership functions. A fuzzy set \underline{A} in the universe of discourse X can be defined as a set of ordered pairs:

$$\underline{A} = \{(x, \mu_{\underline{A}}(x)) \mid x \in X\}$$

where $\mu_{\underline{A}}(\cdot)$ is called membership function of \underline{A} . The membership function $\mu_{\underline{A}}(\cdot)$ maps X to the membership space M , i.e., $\mu_{\underline{A}} : X \rightarrow M$. 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.

Three Basic Features of MFs

1. Support: The **support** of a fuzzy set A is the set of all points x in X such that $\mu_A(x) > 0$:

$$\text{support}(A) = \{x | \mu_A(x) > 0\}.$$

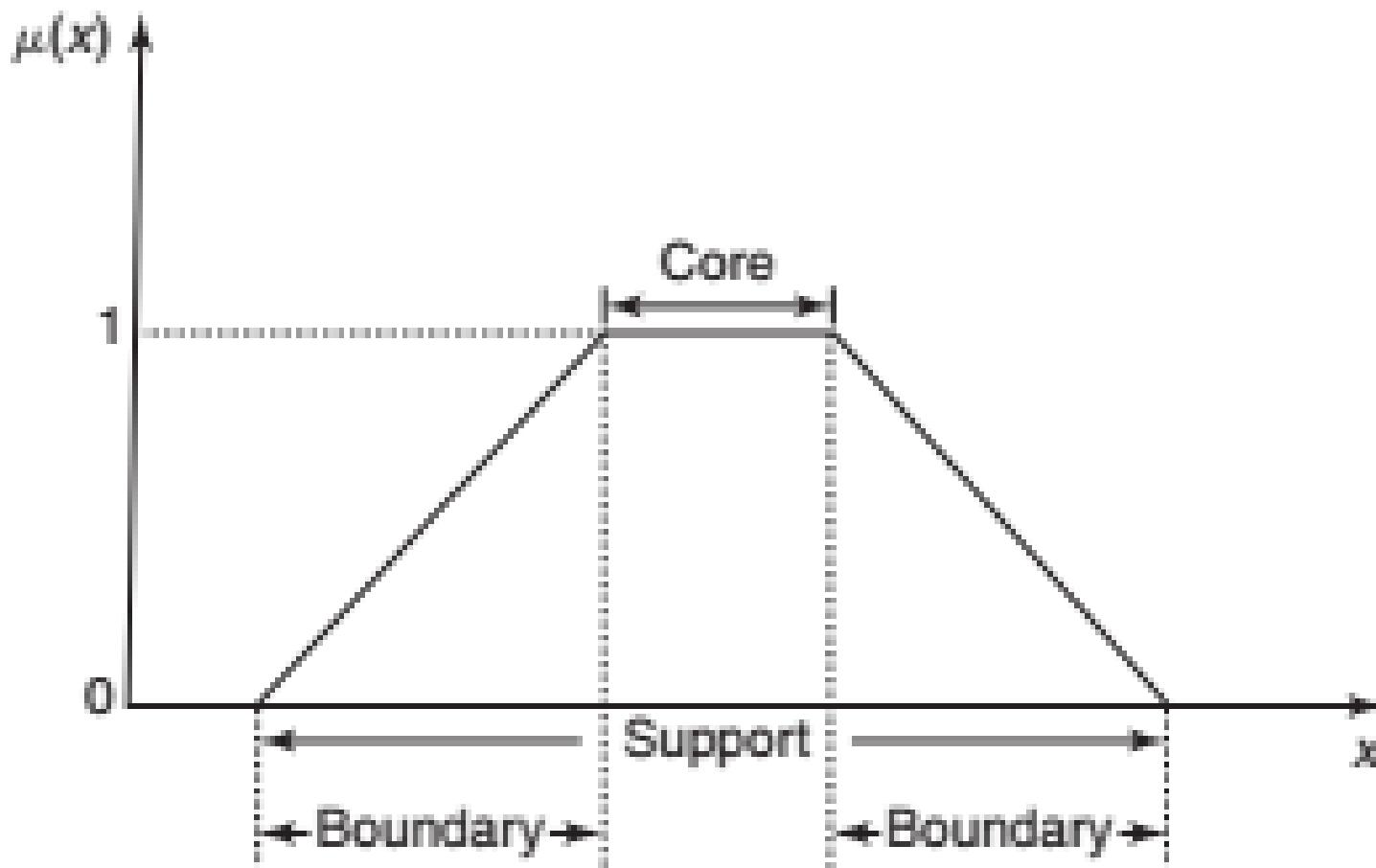
2. Core: The **core** of a fuzzy set A is the set of all points x in X such that $\mu_A(x) = 1$:

$$\text{core}(A) = \{x | \mu_A(x) = 1\}.$$

3. *Boundary:* The support of a membership function for a fuzzy set A is defined as that region of universe containing elements that have a nonzero but not complete membership. The boundary comprises those elements of x of the universe such that

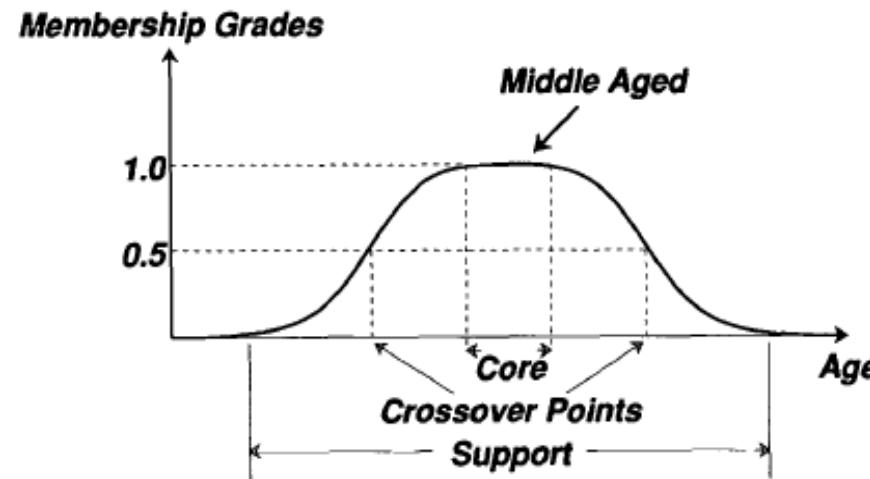
$$0 < \mu_{\underline{A}}(x) < 1$$

The boundary elements are those which possess partial membership in the fuzzy set A .

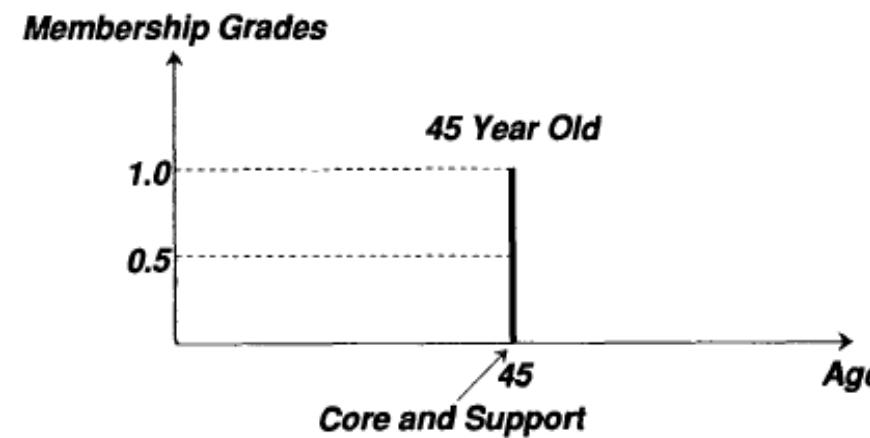


Other Nomenclatures

- Normality: A fuzzy set A is **normal** if its core is nonempty. In other words, we can always find a point $x \in X$ such that $\mu_A(x) = 1$.
- Crossover Points: A **crossover point** of a fuzzy set A is a point $x \in X$ at which $\mu_A(x) = 0.5$:
$$\text{crossover}(A) = \{x | \mu_A(x) = 0.5\}.$$
- Fuzzy Singleton: A fuzzy set whose support is a single point in X with $\mu_A(x) = 1$ is called a **fuzzy singleton**.



(a)



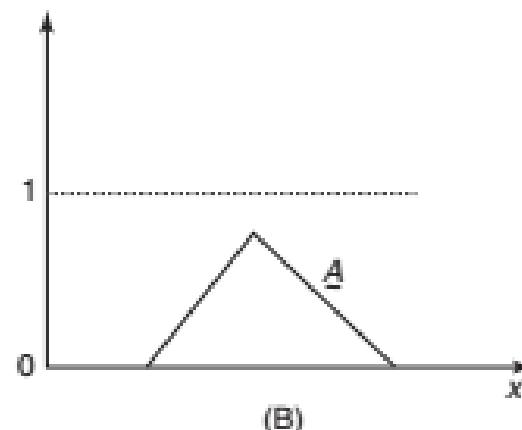
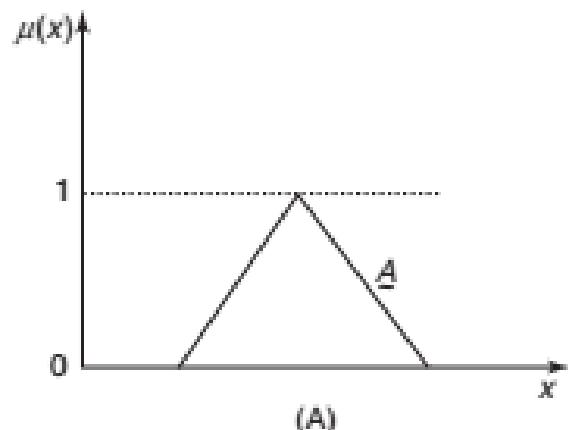
(b)

Cores, supports, and crossover points of (a) the fuzzy set “middle aged” and (b) the fuzzy singleton “45 years old.”

Types of Fuzzy sets

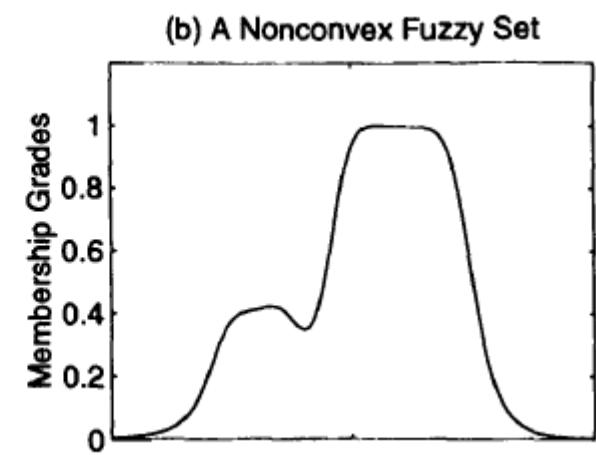
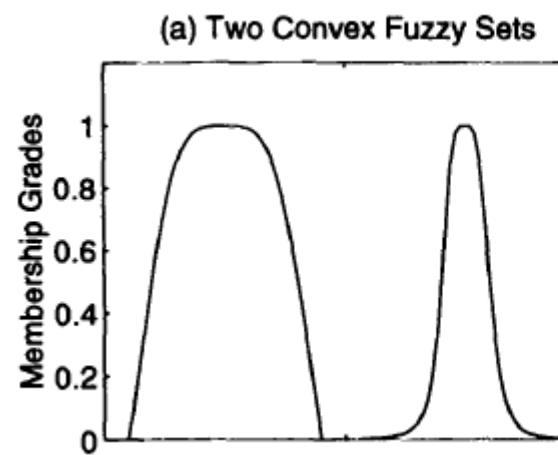
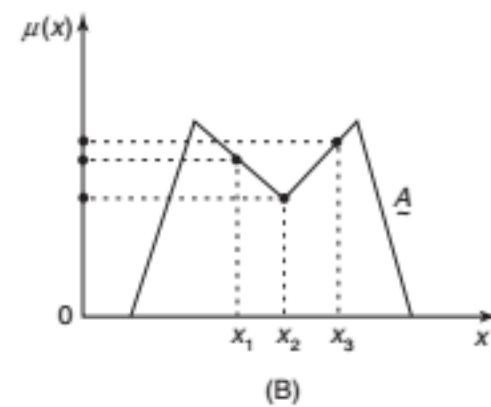
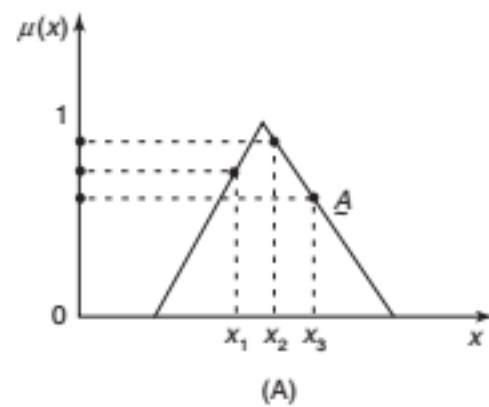
(A) Normal Fuzzy Set: A set whose fuzzy set whose MFs has at least one element ‘x’ in the universe whose membership value is unity is called Normal Fuzzy set. The element for which the membership is equal to 1 is called prototypical element.

(B) Subnormal Fuzzy set: A fuzzy set wherein no MF has its value equal to 1 is called subnormal fuzzy set.



- Convex & Non-convex Fuzzy set:

A *convex fuzzy set* has a membership function whose membership values are strictly monotonically increasing or strictly monotonically decreasing or strictly monotonically increasing than strictly monotonically decreasing with increasing values for elements in the universe. A fuzzy set possessing characteristics opposite to that of convex fuzzy set is called *nonconvex fuzzy set*, i.e., the membership values of the membership function are not strictly monotonically increasing or decreasing or strictly monotonically increasing than decreasing.



From Figure 12-3(A), the convex normal fuzzy set can be defined in the following way. For elements x_1 , x_2 and x_3 in a fuzzy set A , if the following relation between x_1 , x_2 and x_3 holds, i.e.,

$$\mu_A(x_2) \geq \min[\mu_A(x_1), \mu_A(x_3)]$$

then A is said to be a convex fuzzy set. The membership of the element x_2 should be greater than or equal to the membership of elements x_1 and x_3 . For a nonconvex fuzzy set, the constraint is not satisfied.

$$\mu_A(x_2) \geq \min[\mu_A(x_1), \mu_A(x_3)]$$

The intersection of two convex fuzzy sets is also a convex fuzzy set.

Fuzzification

Fuzzification is the process of transforming a crisp set to a fuzzy set or a fuzzy set to a fuzzier set, i.e., crisp quantities are converted to fuzzy quantities.

For a fuzzy set $\underline{A} = \{\mu_i/x_i | x_i \in X\}$, a common fuzzification algorithm is performed by keeping μ_i constant and x_i being transformed to a fuzzy set $Q(x_i)$ depicting the expression about x_i . The fuzzy set $Q(x_i)$ is referred to as the *kernel of fuzzification*. The fuzzified set \underline{A} can be expressed as

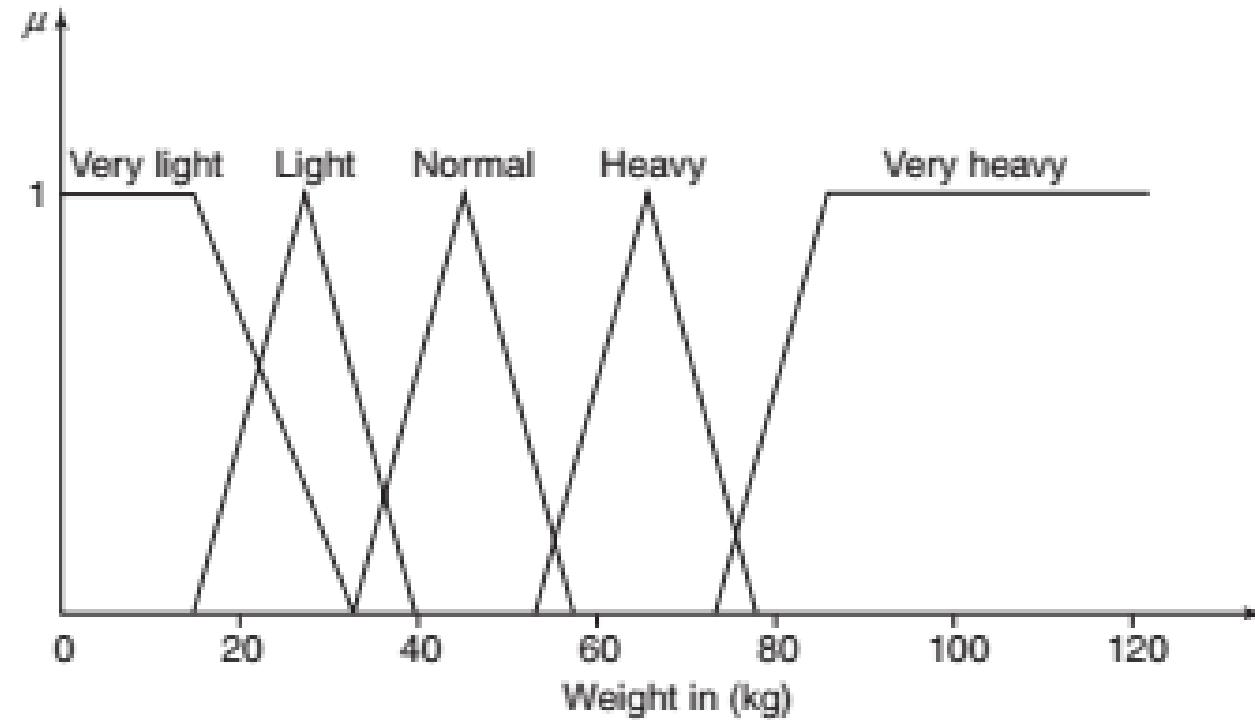
$$\underline{A} = \mu_1 Q(x_1) + \mu_2 Q(x_2) + \dots + \mu_n Q(x_n)$$

where the symbol \sim means fuzzified. This process of fuzzification is called support fuzzification (s-fuzzification). There is another method of fuzzification called *grade fuzzification* (g-fuzzification) where x_i is kept constant and μ_i is expressed as a fuzzy set. Thus, using these methods, fuzzification is carried out.

Methods of Membership Value Assignments

1. Intuition;
2. inference;
3. rank ordering;
4. angular fuzzy sets;
5. neural networks;
6. genetic algorithm;
7. inductive reasoning.

- **Intuition:** It is based upon human intelligence. Capacity of the human to develop the MF on the basis of their intelligence.



• Inference:

The inference method uses knowledge to perform deductive reasoning. Deduction achieves conclusion by means of forward inference. There are various methods for performing deductive reasoning. Here the knowledge of geometrical shapes and geometry is used for defining membership values. The membership functions may be defined by various shapes: triangular, trapezoidal, bell-shaped, Gaussian and so on. The inference method here is discussed via triangular shape.

Consider a triangle, where X, Y and Z are the angles, such that $X \geq Y \geq Z \geq 0$, and let U be the universe of triangles, i.e.,

$$U = \{(X, Y, Z) \mid X \geq Y \geq Z \geq 0; X + Y + Z = 180\}$$

There are various types of triangles available. Here a few are considered to explain inference methodology:

I = isosceles triangle (approximate)

E = equilateral triangle (approximate)

R = right-angle triangle (approximate)

IR = isosceles and right-angle triangle (approximate)

T = other triangles

The membership values of approximate isosceles triangle is obtained using the following definition, where $X \geq Y \geq Z \geq 0$ and $X + Y + Z = 180^\circ$:

$$\mu_l(X, Y, Z) = 1 - \frac{1}{60^\circ} \min(X - Y, Y - Z)$$

If $X = Y$ or $Y = Z$, the membership value of approximate isosceles triangle is equal to 1. On the other hand, if $X = 120^\circ, Y = 60^\circ$ and $Z = 0^\circ$, we get

$$\begin{aligned}\mu_l(X, Y, Z) &= 1 - \frac{1}{60^\circ} \min(120^\circ - 60^\circ, 60^\circ - 0^\circ) \\ &= 1 - \frac{1}{60^\circ} \min(60^\circ, 60^\circ) \\ &= 1 - \frac{1}{60^\circ} \times 60^\circ \\ &= 1 - 1 = 0\end{aligned}$$

The membership value of approximate right-angle triangle is given by

$$\mu_{\underline{R}}(X, Y, Z) = 1 - \frac{1}{90^\circ} |X - 90^\circ|$$

If $X = 90^\circ$, the membership value of a right-angle triangle is 1, and if $X = 180^\circ$, the membership value $\mu_{\underline{R}}$ becomes 0:

$$X = 90^\circ \Rightarrow \mu_{\underline{R}} = 1$$

$$X = 180^\circ \Rightarrow \mu_{\underline{R}} = 0$$

If $X = 90^\circ$, the membership value of a right-angle triangle is 1, and if $X = 180^\circ$, the membership value μ_R becomes 0:

$$X = 90^\circ \Rightarrow \mu_R = 1$$

$$X = 180^\circ \Rightarrow \mu_R = 0$$

The membership value of approximate isosceles right-angle triangle is obtained by taking the logical intersection of the approximate isosceles and approximate right-angle triangle membership functions, i.e.,

$$\underline{IR} = \underline{I} \cap \underline{R}$$

and it is given by

$$\begin{aligned}\mu_{\underline{IR}}(X, Y, Z) &= \min[\mu_I(X, Y, Z), \mu_R(X, Y, Z)] \\ &= 1 - \max \left[\frac{1}{60^\circ} \min(X - Y, Y - Z), \frac{1}{90^\circ} |X - 90^\circ| \right]\end{aligned}$$

The membership function for a fuzzy equilateral triangle is given by

$$\mu_E(X, Y, Z) = 1 - \frac{1}{180^\circ} |X - Z|$$

The membership function of other triangles, denoted by \underline{T} , is the complement of the logical union of \underline{I} , \underline{R} and \underline{E} , i.e.,

$$\underline{T} = \overline{\underline{I} \cup \underline{R} \cup \underline{E}}$$

By using De Morgan's law, we get

$$\underline{T} = \overline{\underline{I}} \cap \overline{\underline{R}} \cap \overline{\underline{E}}$$

The membership value can be obtained using the equation

$$\begin{aligned}\mu_{\underline{T}}(X, Y, Z) &= \min\{1 - \mu_I(X, Y, Z), 1 - \mu_E(X, Y, Z), 1 - \mu_R(X, Y, Z)\} \\ &= \frac{1}{180^\circ} \min\{3(X - Y), 3(Y - Z), 2|X - 90^\circ|, X - Z\}\end{aligned}$$

The inference method as discussed for triangular shape can be extended for trapezoidal shape and so on, on the basis of knowledge of geometry.

Question

Using the inference approach, find the membership values for the triangular shapes \underline{I} , \underline{R} , \underline{E} , \underline{IR} , and \underline{T} for a triangle with angles 45° , 55° and 80° .

Solution: Let the universe of discourse be

$$U = \{(X, Y, Z) : X = 80^\circ \geq Y = 55^\circ \geq Z = 45^\circ \text{ and } X + Y + Z = 80^\circ + 55^\circ + 45^\circ = 180^\circ\}$$

- Membership value of isosceles triangle, \underline{I} :

$$\begin{aligned}\mu_{\underline{I}} &= 1 - \frac{1}{60^\circ} \min(X - Y, Y - Z) \\ &= 1 - \frac{1}{60^\circ} \min(80^\circ - 55^\circ, 55^\circ - 45^\circ) \\ &= 1 - \frac{1}{60^\circ} \min(25^\circ, 10^\circ) \\ &= 1 - \frac{1}{60^\circ} \times 10^\circ \\ &= 1 - 0.1667 = 0.833\end{aligned}$$

- Membership value of right-angle triangle, \underline{R} :

$$\begin{aligned}\mu_{\underline{R}} &= 1 - \frac{1}{90^\circ} |X - 90^\circ| = 1 - \frac{1}{90^\circ} |80^\circ - 90^\circ| \\ &= 1 - \frac{1}{90^\circ} \times 10^\circ = 0.889\end{aligned}$$

- Membership value of equilateral triangle, \underline{E} :

$$\begin{aligned}\mu_{\underline{E}} &= 1 - \frac{1}{180^\circ} (X - Z) = 1 - \frac{1}{180^\circ} (80^\circ - 45^\circ) \\ &= 1 - \frac{1}{180^\circ} \times 35^\circ = 0.8056\end{aligned}$$

- Membership value of isosceles and right-angle triangle, \underline{IR} :

$$\begin{aligned}\mu_{\underline{IR}} &= \min[\mu_{\underline{I}}, \mu_{\underline{R}}] = \min[0.833, 0.889] \\ &= 0.833\end{aligned}$$

- Membership value of other triangles, \underline{T} :

$$\begin{aligned}\mu_{\underline{T}} &= \min[1 - \mu_{\underline{I}}, 1 - \mu_{\underline{E}}, 1 - \mu_{\underline{R}}] \\ &= \min[0.167, 0.1944, 0.111] = 0.111\end{aligned}$$

Thus the membership function is calculated for the triangular shapes.

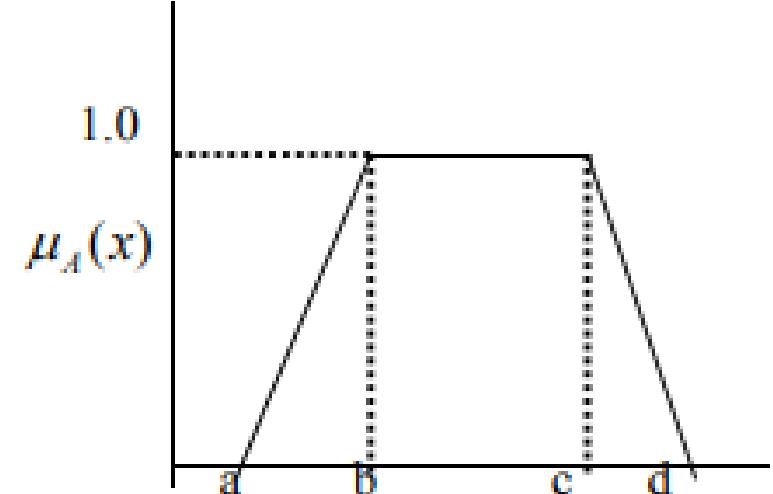
Try this

Using the inference approach, obtain the membership values for the triangular shapes (L, R, T) for a triangle with angles 40° , 60° and 80° .

Trapezoidal membership function:

Let a, b, c and d represents the x coordinates of the membership function. then

$$\begin{aligned}\text{Trapezoid}(x; a, b, c, d) &= 0 \text{ if } x \leq a; \\ &= (x-a)/(b-a) \text{ if } a \leq x \leq b \\ &= 1 \text{ if } b \leq x \leq c; \\ &= (d-x)/(d-c) \text{ if } c \leq x \leq d; \\ &= 0, \text{ if } d \leq x.\end{aligned}$$



$$\mu_{\text{trapezoid}} = \max\left(\min\left(\frac{x-a}{b-a}, 1, \frac{d-x}{d-c}\right), 0\right)$$

Defuzzification: λ -Cut Method

- It is the reverse process of fuzzification.
- We begin by considering a fuzzy set \underline{A} , then define a lambda- cut set, A_λ where $0 \leq \lambda \leq 1$.
- The set A_λ is a crisp set called lambda- cut set or alpha –cut set of the fuzzy set of \underline{A} , where $A_\lambda = \{x | \mu_{\underline{A}}(x) \geq \lambda\}$. Note that the λ -cut set A_λ does not have a tilde underscore; it is a crisp set derived from its parent fuzzy set, \underline{A} .
- Any particular fuzzy set \underline{A} can be transformed into an infinite number of λ -cut sets, because there are an infinite number of values λ on the interval $[0, 1]$.
- Any element $x \in \underline{A}$ belongs to \underline{A} with a grade of membership that is greater than or equal to the value λ .

- Example: Let us consider the discrete fuzzy set, using Zadeh's notation, defined on universe $X = \{a, b, c, d, e, f\}$,

$$\bar{A} = \left\{ \frac{1}{a} + \frac{0.9}{b} + \frac{0.6}{c} + \frac{0.3}{d} + \frac{0.01}{e} + \frac{0}{f} \right\}.$$

- We can reduce this fuzzy set into several λ -cut sets, all of which are crisp.
- For example, we can define λ -cut sets for the values of $\lambda = 1, 0.9, 0.6, 0.3, 0+$, and 0.

$$A_1 = \{a\}, \quad A_{0.9} = \{a, b\},$$

$$A_{0.6} = \{a, b, c\}, \quad A_{0.3} = \{a, b, c, d\},$$

$$A_{0+} = \{a, b, c, d, e\}, \quad A_0 = X.$$

- The quantity $\lambda = 0+$ is defined as a small “ ε ” value >0 , that is, a value just greater than zero.
- If we define a crisp set for $A_{0.9}$, then it will be as shown below:

$$A = \left\{ \frac{1}{a} + \frac{1}{b} + \frac{0}{c} + \frac{0}{d} + \frac{0}{e} + \frac{0}{f} \right\}$$

We can express λ -cut sets using Zadeh's notation. For example, λ -cut sets for the values $\lambda = 0.9$ and 0.25 are given here:

$$A_{0.9} = \left\{ \frac{1}{a} + \frac{1}{b} + \frac{0}{c} + \frac{0}{d} + \frac{0}{e} + \frac{0}{f} \right\} \quad \text{and} \quad A_{0.25} = \left\{ \frac{1}{a} + \frac{1}{b} + \frac{1}{c} + \frac{1}{d} + \frac{0}{e} + \frac{0}{f} \right\}.$$

λ -cut sets obey the following four very special properties:

1. $(\tilde{A} \cup \tilde{B})_\lambda = A_\lambda \cup B_\lambda$
2. $(\tilde{A} \cap \tilde{B})_\lambda = A_\lambda \cap B_\lambda$
3. $(\overline{\tilde{A}})_\lambda \neq \overline{A}_\lambda$ except for a value of $\lambda = 0.5$
4. For any $\lambda \leq \alpha$, where $0 \leq \alpha \leq 1$, it is true that $A_\alpha \subseteq A_\lambda$, where $A_0 = X$

λ -CUTS FOR FUZZY RELATIONS

λ -CUT FOR FUZZY SETS

Lets define $R_\lambda = \{(x, y) | \mu_{\tilde{R}}(x, y) \geq \lambda\}$ as a λ -cut relation of the fuzzy relation R . Since in this case R is a two-dimensional array defined on the universes X and Y , any pair $(x, y) \in R_\lambda$ belongs to \tilde{R} with a "strength" of relation greater than or equal to λ .

e.g. $\tilde{R} = \begin{bmatrix} 1 & 0.8 & 0.3 \\ 0.8 & 0.9 & 0.2 \\ 0.3 & 0.2 & 0 \end{bmatrix}$

$$\lambda = 0.8$$

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

$$\lambda = 0.3$$

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

λ -cuts on fuzzy relations obey certain properties, just as λ -cuts on fuzzy sets

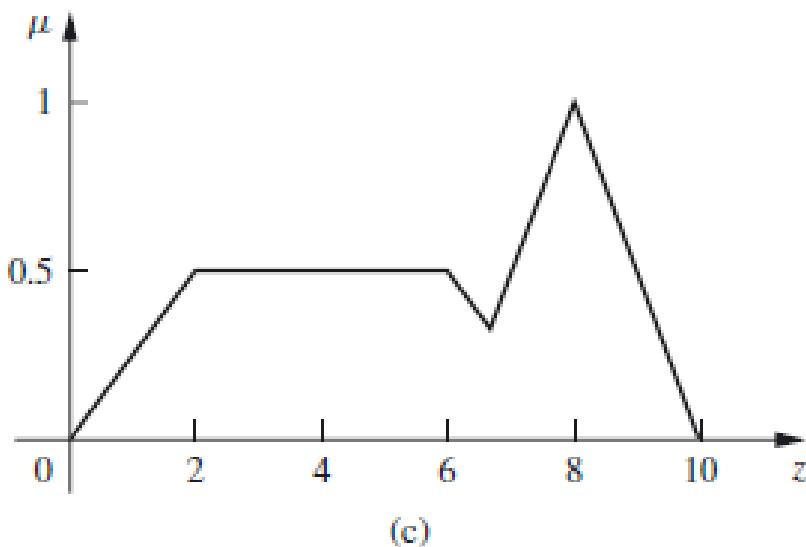
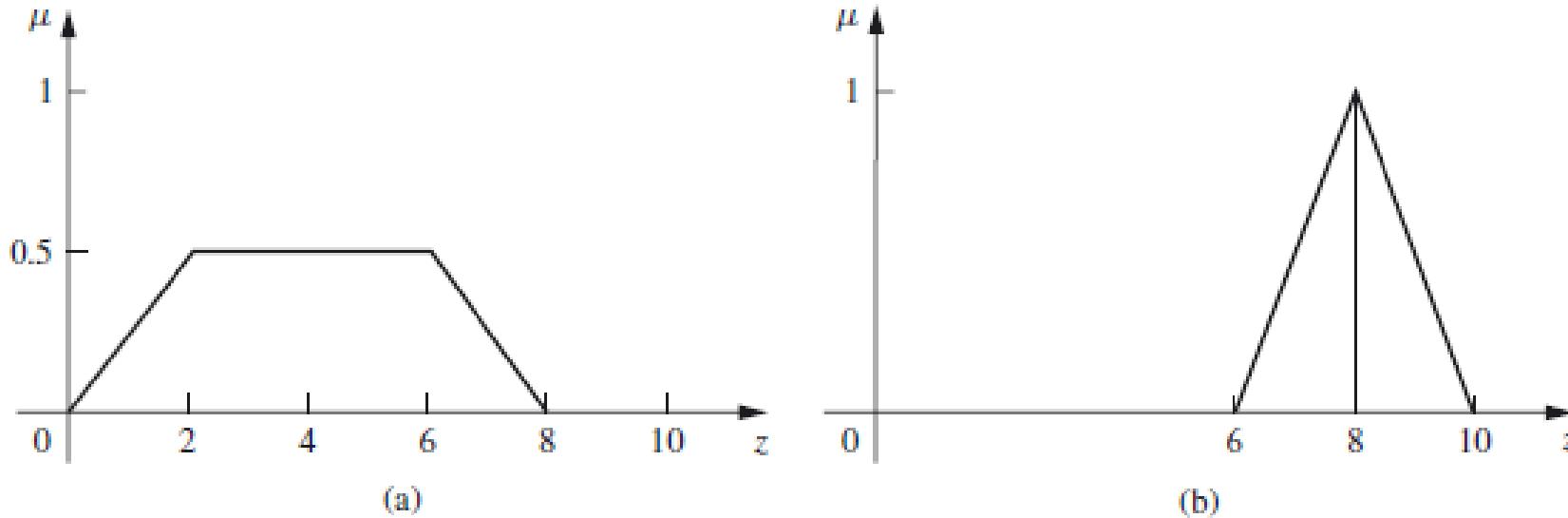
1. $(\tilde{R} \cup \tilde{S})_\lambda = R_\lambda \cup S_\lambda$
2. $(\tilde{R} \cap \tilde{S})_\lambda = R_\lambda \cap S_\lambda$
3. $(\tilde{R})_\lambda \neq \tilde{R}_\lambda$
4. For any $\lambda \leq \alpha, 0 \leq \alpha \leq 1$, then $R_\alpha \subseteq R_\lambda$

Defuzzification to Scalars

- Defuzzification is the conversion of a fuzzy quantity to a precise quantity, just as fuzzification is the conversion of a precise quantity to a fuzzy quantity.
- The output of a fuzzy process can be the logical union of two or more fuzzy membership functions defined on the universe of discourse of the output variable.
- For example, suppose a fuzzy output comprises two parts: (1) \tilde{C}_1 a trapezoidal shape and (2) \tilde{C}_2 a triangular membership shape.
- The union of these two membership functions, that is $\tilde{C} = \tilde{C}_1 \cup \tilde{C}_2$, involves the max operator.

A fuzzy output process may involve many output parts, and the membership function representing each part of the output can have any shape. The membership function of the fuzzy output need not always be normal. In general, we have

$$\tilde{C}_k = \bigcup_{i=1}^k \tilde{C}_i = \tilde{C}.$$



Typical fuzzy process output: (a) first part of fuzzy output; (b) second part of fuzzy output; and (c) union of both parts.

Defuzzification Methods

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.

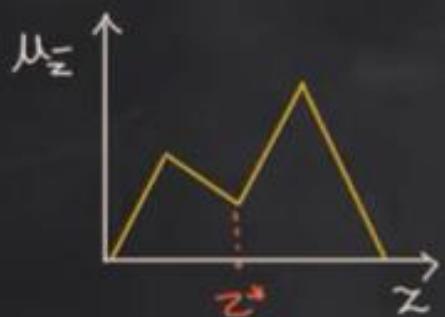
1) Centroid Method

This method is also known as center of mass, center of area or center of gravity method. It is the most commonly used defuzzification method. The defuzzified output x^* is defined as

$$x^* = \frac{\int \mu_C(x) \cdot x dx}{\int \mu_C(x) dx}$$

where the symbol \int denotes an algebraic integration.

Suppose we have a union of two fuzzy sets :

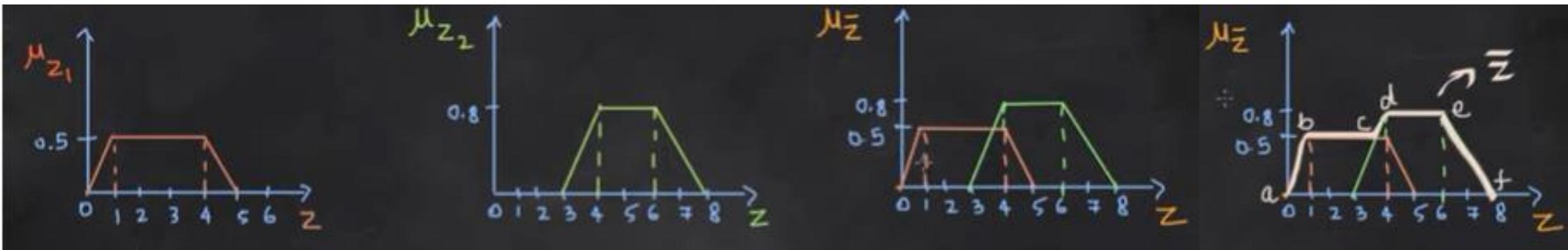


Centre of gravity

In centroid method, the defuzzified value z^* is given by:

$$z^* = \frac{\int \mu_{\bar{z}} \cdot z dz}{\int \mu_{\bar{z}} dz}$$

Example



Please note that I have used
 $\mu_z = y$ and $z = x$

ab

$$\frac{y - y_1}{x - x_1} = \frac{y_2 - y_1}{x_2 - x_1} \quad y_1 = 0 \quad x_1 = 0 \\ y_2 = 0.5 \quad x_2 = 1$$

$$y = 0.5x \quad 0 \text{ to } 1$$

$$\underline{bc} \\ y = 0.5 \quad 1 \text{ to } 3.5$$

cd

$$y_1 = 0.5 \quad x_1 = 3.5 \quad y_2 = 0.8 \quad x_2 = 4 \\ y = \frac{3}{5}x - 8/5 \quad 3.5 \text{ to } 4$$

de

$$y = 0.8 \quad 4 \text{ to } 6$$

$$\underline{ef} \quad 6 \text{ to } 9$$

$$y_1 = 0.8 \quad x_1 = 6 \\ y_2 = 0 \\ y = -0.8$$

$$\bar{z}^* = \frac{\int \mu_{\bar{z}} \cdot z dz}{\int \mu_{\bar{z}} \cdot dz}$$

$$= \frac{\int_0^{3.5} 0.5 z \cdot z dz + \int_1^{3.5} 0.5 z \cdot dz + \int_{3.5}^4 \left(\frac{3}{5}z - \frac{8}{5}\right) \cdot z dz + \int_4^6 0.8 z \cdot dz + \int_6^8 (-0.4z + 3.2) \cdot z dz}{\int_0^{3.5} 0.5 z dz + \int_1^{3.5} 0.5 dz + \int_{3.5}^4 \left(\frac{3}{5}z - \frac{8}{5}\right) dz + \int_4^6 0.8 dz + \int_6^8 (-0.4z + 3.2) dz}$$

$$\int_0^{3.5} 0.5 z dz + \int_1^{3.5} 0.5 dz + \int_{3.5}^4 \left(\frac{3}{5}z - \frac{8}{5}\right) dz + \int_4^6 0.8 dz + \int_6^8 (-0.4z + 3.2) dz$$

$$\bar{z} = 4.151$$

2) Max membership principle

- Also known as the *height method*, this scheme is limited to peaked output functions.

It gives us the maximum membership value of a fuzzy function.

Suppose we have a union of 2 fuzzy sets as:

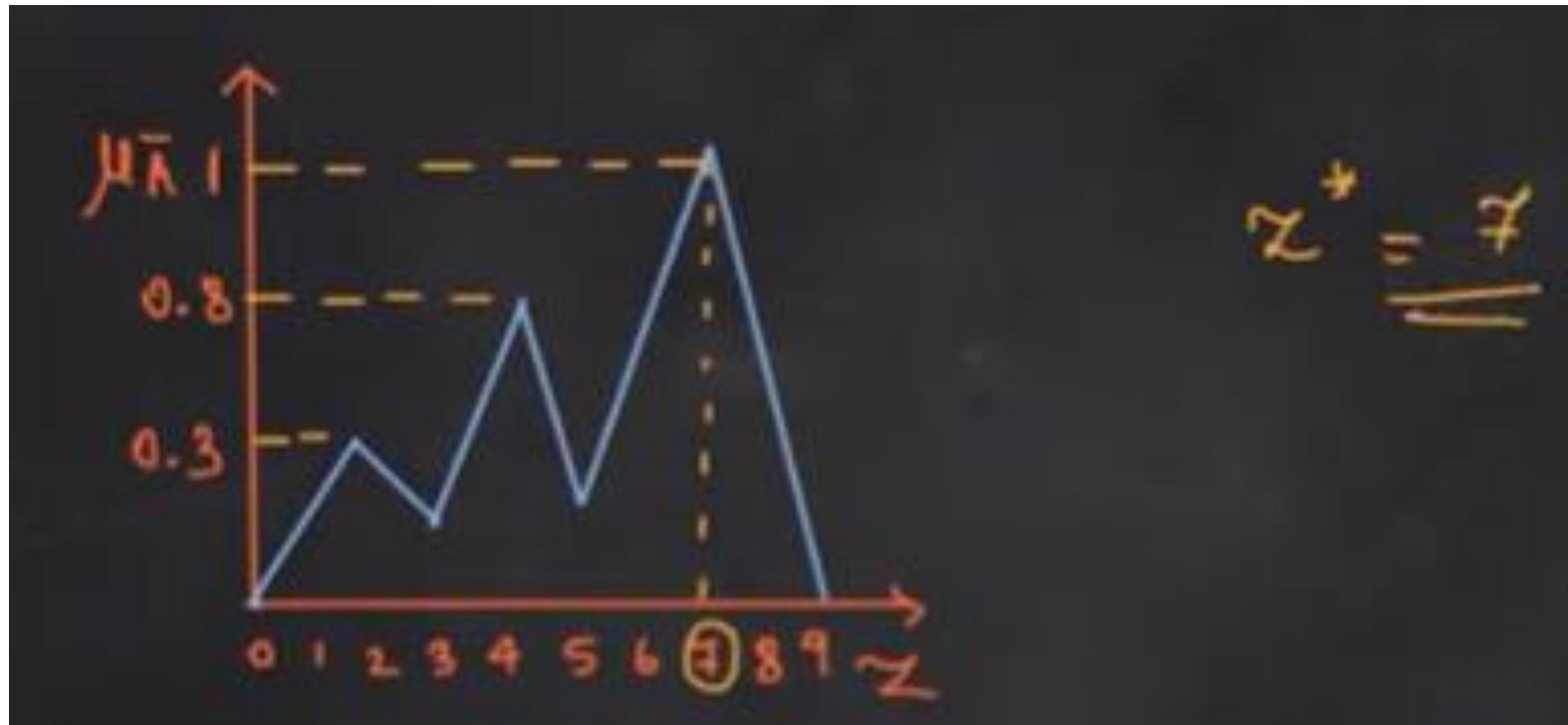


This principle is applicable only to peaked output functions

$$\mu_{\tilde{A}}(z^*) \geq \mu_{\tilde{A}}(z) \text{ for all } z \in Z$$

OR $\mu_{\tilde{C}}(z^*) \geq \mu_{\tilde{C}}(z)$, for all $z \in Z$ where z^* is the defuzzified value,

Example



$$x^* = \underline{x}$$

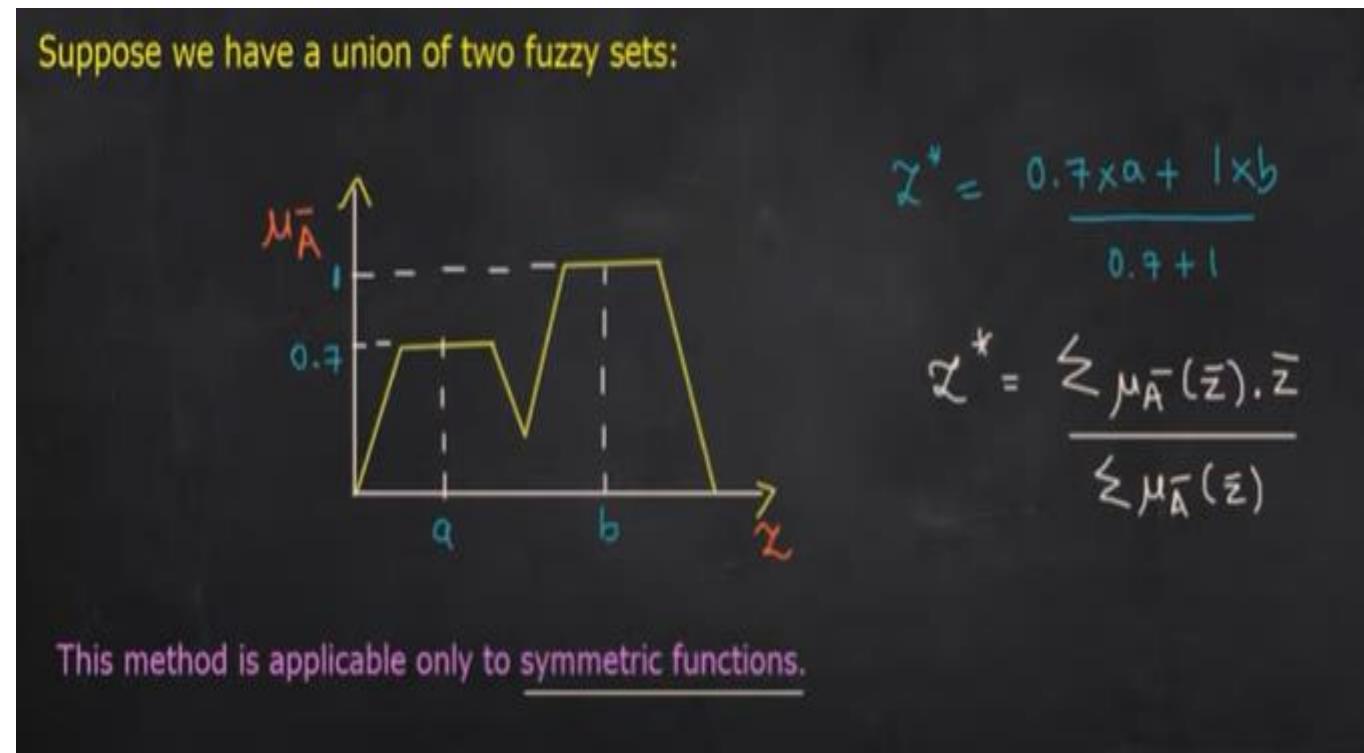
3) Weighted average method

- The weighted average method is the most frequently used in fuzzy applications since it is one of the more computationally efficient methods.
- Unfortunately, it is *usually* restricted to symmetrical output membership functions.

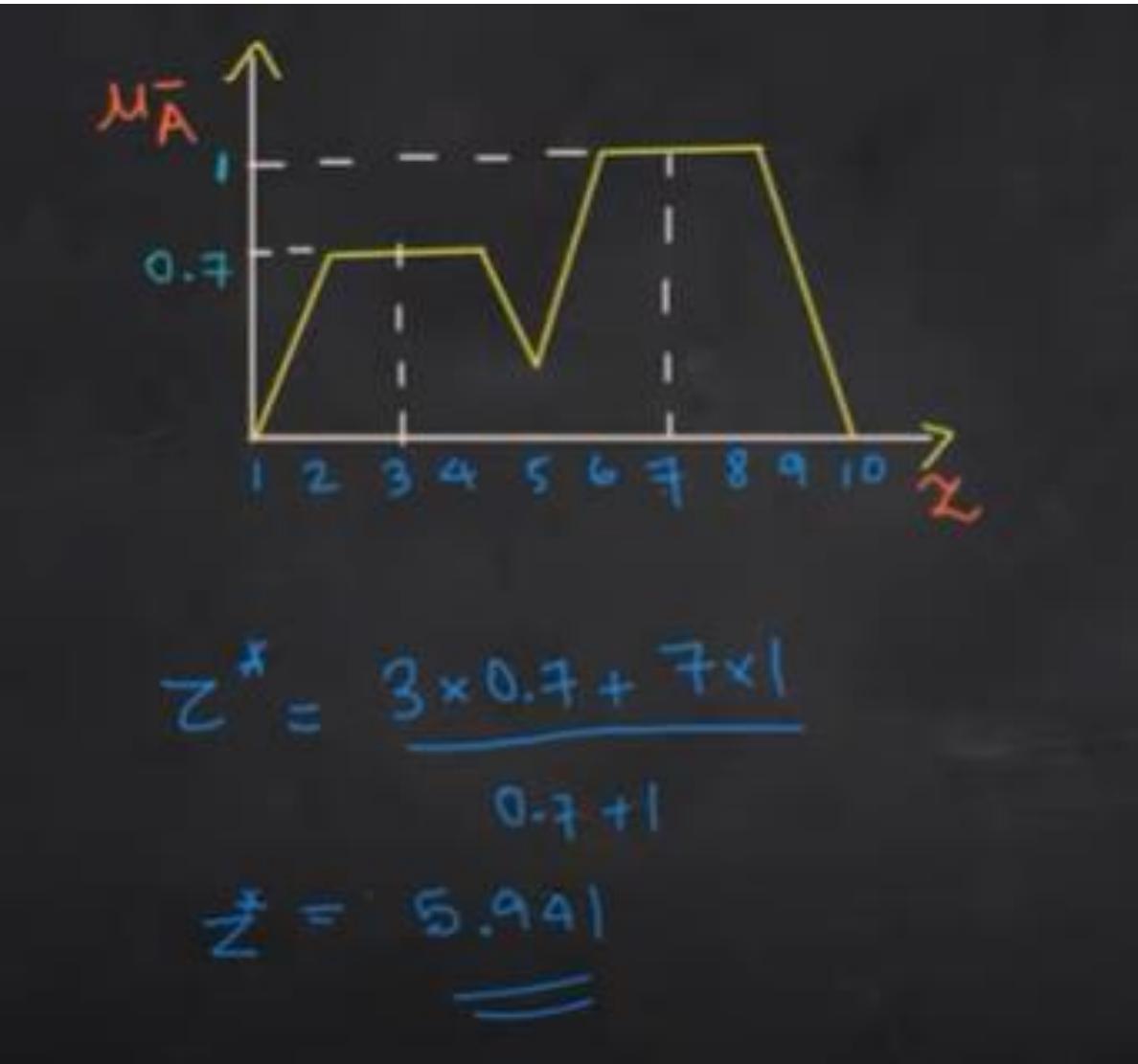
$$z^* = \frac{\sum \mu_{\bar{C}}(\bar{z}) \cdot \bar{z}}{\sum \mu_{\bar{C}}(\bar{z})},$$

where \sum denotes the algebraic sum and where \bar{z} is the centroid of each symmetric membership function.

Note: a & b are the centers of the two functions.



Example

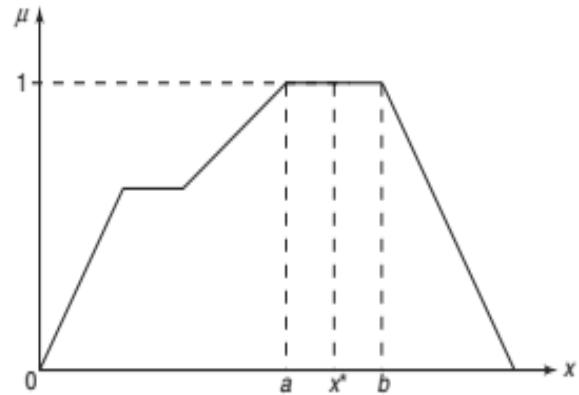


4.) Mean max membership

- This method (also called *middle-of-maxima*) is closely related to the first method, except that the locations of the maximum membership can be nonunique (i.e., the maximum membership can be a plateau rather than a single point).

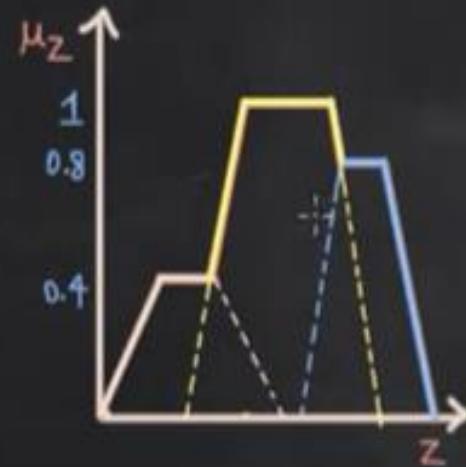
$$z^* = \frac{a + b}{2}$$

where a and b are as shown in the figure.



Note: We are basically finding the average of the max. membership value i.e a & b .

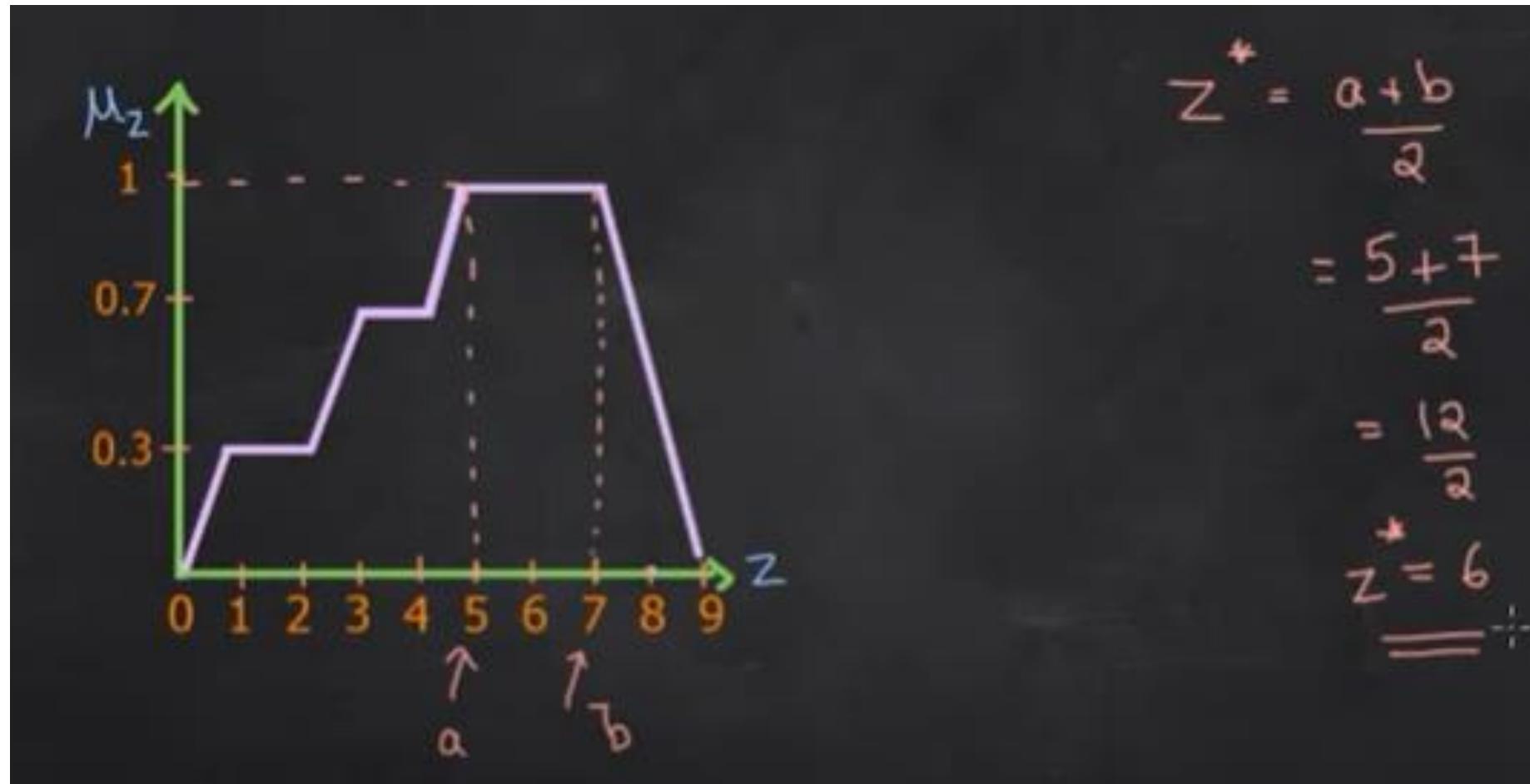
Consider a union of 3 fuzzy sets:



$$z^* = \frac{a + b}{2}$$

Note: This method is applicable to symmetric functions only.

Example



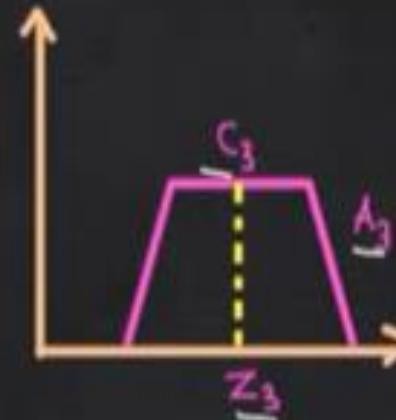
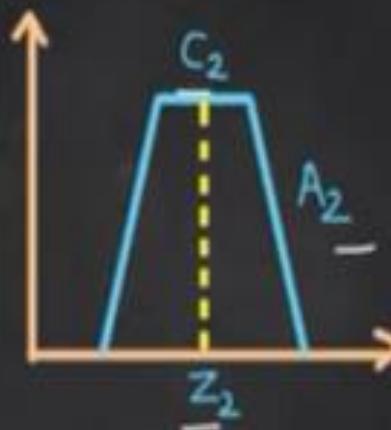
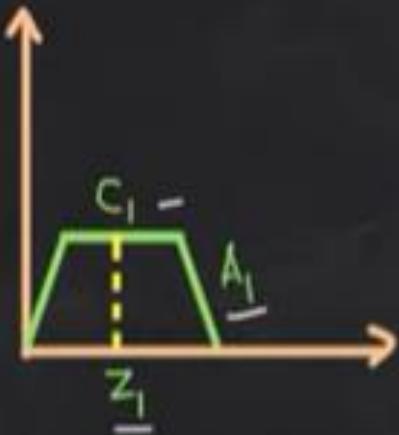
5.) Center of sums

- This method is similar to the weighted average method, except that in the center of sums method the weights are the areas of the respective membership functions whereas in the weighted average method the weights are individual membership values.

$$z^* = \frac{\sum_{k=1}^n \mu_{G_k}(z) \int_z \bar{z} dz}{\sum_{k=1}^n \mu_{G_k}(z) \int_z dz},$$

where the symbol \bar{z} is the distance to the centroid of each of the respective membership functions.

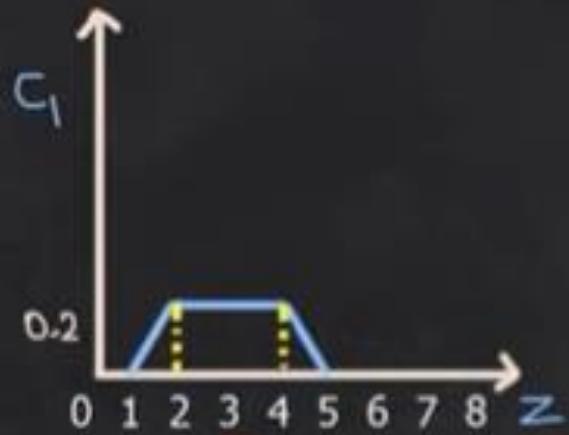
Consider three fuzzy sets:



$$z^* = \frac{\sum_{i=1}^n z_i A_{C_i}}{\sum_{i=1}^n A_{C_i}}$$

Note: C_1, C_2 & C_3 are the Fuzzy sets
 A_1, A_2, A_3 are the areas
 Z_1, Z_2 & Z_3 are the geometric centers

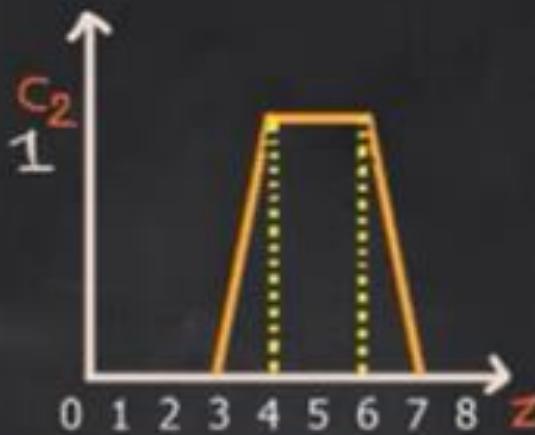
Example



$$A_1 \quad z_1 = 3$$

$$A_1 = \frac{1}{2}(b_1 + b_2)h \\ = \frac{1}{2}(2+4)0.2$$

$$A_1 = 0.6$$

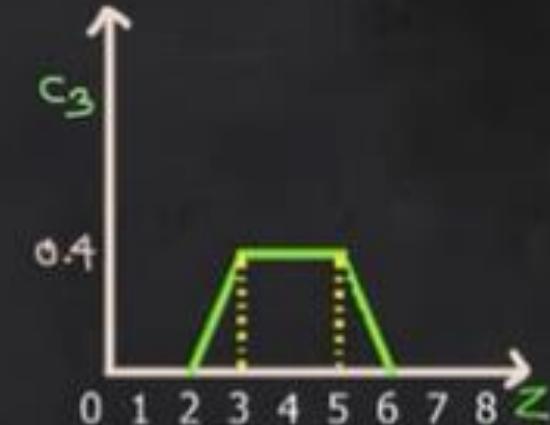
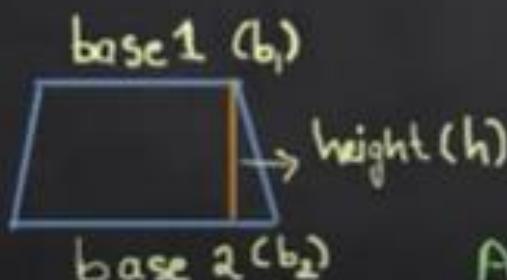


$$A_2 \quad z_2 = 5$$

$$A_{\text{trap}} = \frac{1}{2}(b_1 + b_2)h$$

$$A_2 = \frac{1}{2}(2+4)1$$

$$A_2 = 3$$



$$A_3 \quad z_3 = 4$$

$$A_3 = \frac{1}{2}(2+4)0.4$$

$$A_3 = 1.2$$



$$z^* = \frac{\sum_{i=1}^n z_i A_{C_i}}{\sum_{i=1}^n A_{C_i}}$$

$$\begin{aligned} z^* &= \frac{\sum_{i=1}^3 z_i A_{C_i}}{\sum_{i=1}^3 A_{C_i}} \\ &= \frac{z_1 A_1 + z_2 A_2 + z_3 A_3}{A_1 + A_2 + A_3} \\ &= \frac{(3 \times 0.6) + (5 \times 3) + (4 \times 1.2)}{0.6 + 3 + 1.2} \end{aligned}$$

$$z^* = \underline{\underline{4.5}}$$

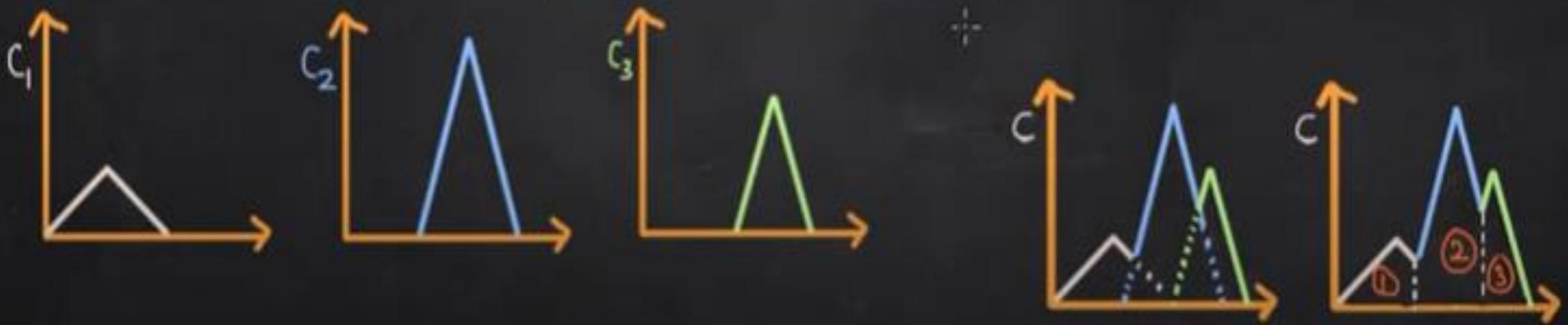
6) Center of largest area

If the fuzzy set has more than one subregion, then the center of gravity of the subregion with the largest area can be used to calculate the defuzzified value.

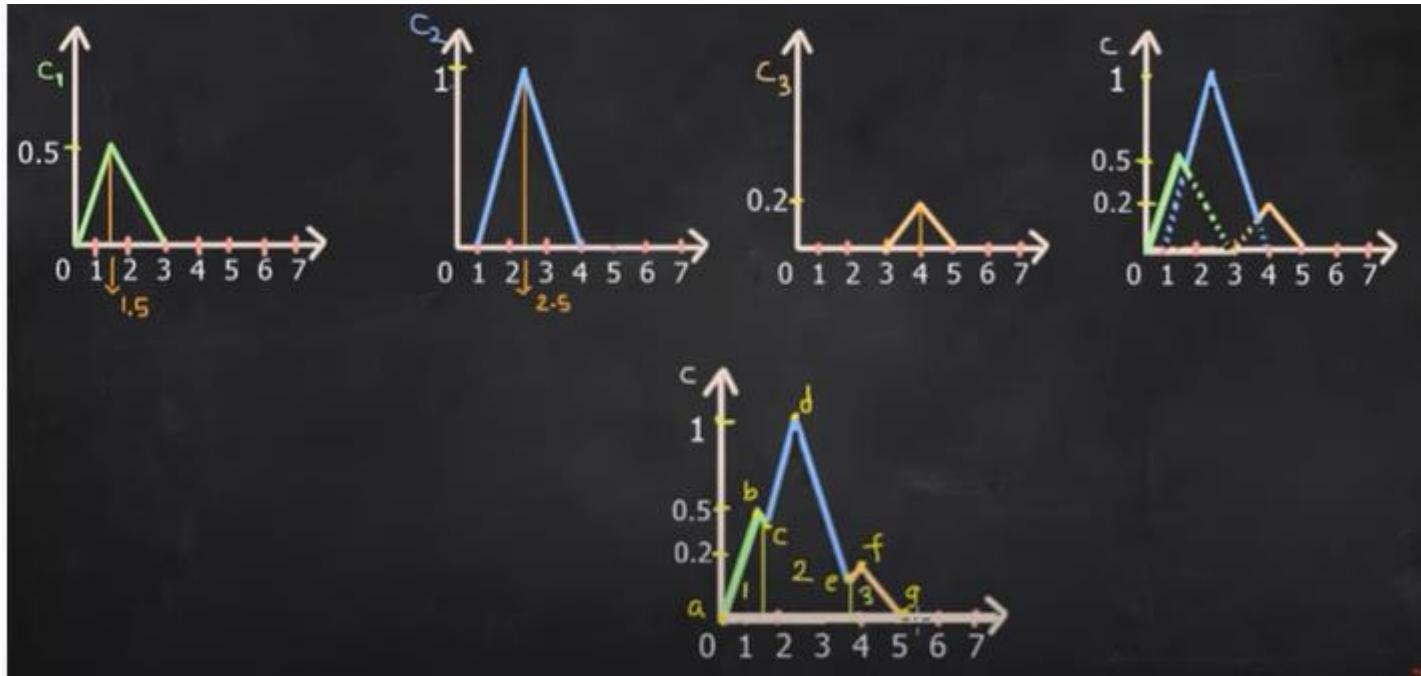
$$z^* = \frac{\int \mu_{C_m}(z) \cdot z \, dz}{\int \mu_{C_m}(z) \cdot dz}$$

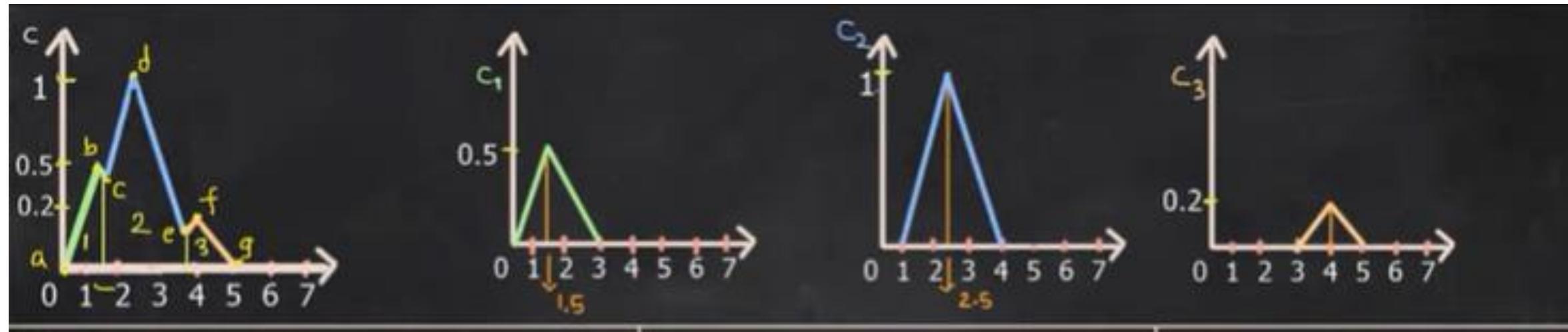
$C_m \Rightarrow$ Region with the largest area

$z \Rightarrow$ Center of gravity of C_m



Example





Area 1 = Area under ab + Area under bc

$$\frac{y - y_1}{x - x_1} = \frac{y_2 - y_1}{x_2 - x_1}$$

ab

$$x_1 = 0 \quad y_1 = 0$$

$$x_2 = 1.5 \quad y_2 = 0.5$$

$$y = 0.33x \quad [0, 1.5]$$

bc

$$x_1 = 1.5 \quad x_2 = 3$$

$$y_1 = 0.5 \quad y_2 = 0$$

$$y = -0.33x + 1 \quad [1.5, 1.7]$$

cd

$$x_1 = 1 \quad x_2 = 2.5$$

$$y_1 = 0 \quad y_2 = 1$$

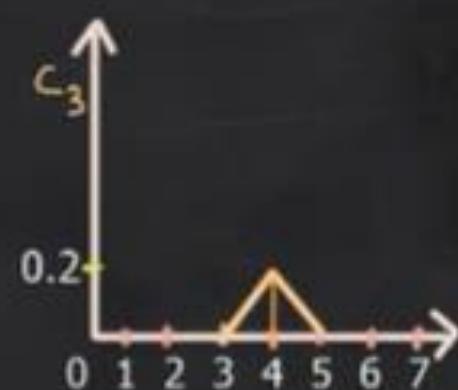
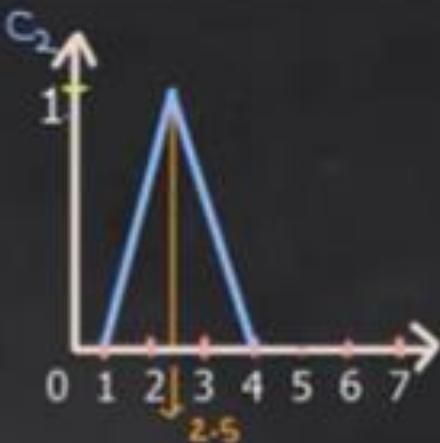
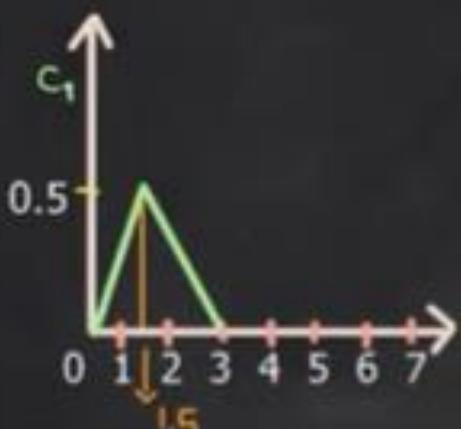
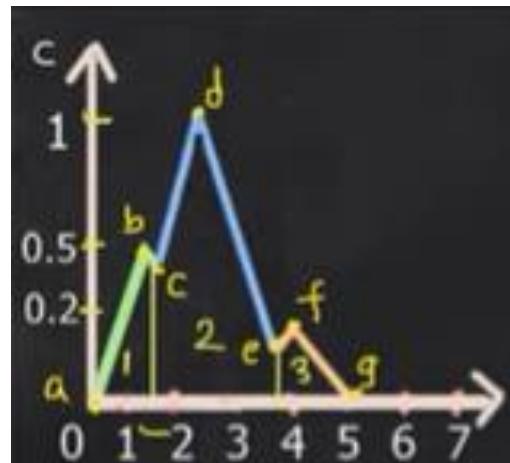
$$y = 0.67x - 0.67$$

$$c = 1.67$$

$$c \approx 1.7$$

$$\text{Area 1} = \int_0^{1.5} 0.33x dx + \int_{1.5}^1 f(0.33x + 1) dx$$

$$\text{Area 1} = 0.466$$



Area 2 = area under cd + area under de

cd

$$y = 0.67x - 0.67 \quad [1.7, 2.5]$$

de

$$x_1 = 2.5 \quad x_2 = 4$$

$$y_1 = 1 \quad y_2 = 0$$

$$y = -0.67x + 2.67 \quad [2.5, 3.8]$$

ef

$$x_1 = 3 \quad x_2 = 4$$

$$y_1 = 0 \quad y_2 = 0.2$$

$$y = 0.2x - 0.6$$

$$\text{E} = 3.758 \\ \approx 3.8$$

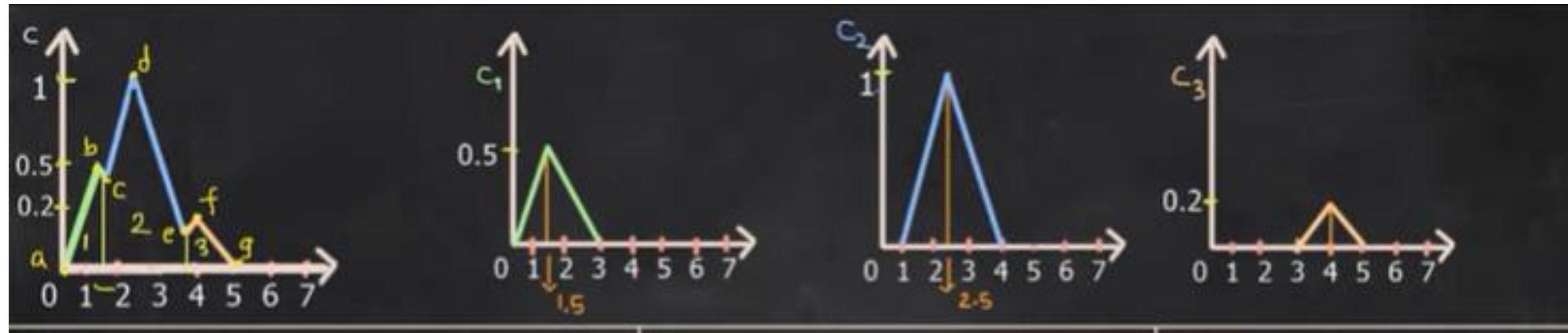
$$\text{Area 2} = \int_{1.7}^{2.5} (0.67x - 0.67) dx$$

$$+ \int_{2.5}^{3.8} (-0.67x +$$

$$2.67) dx$$

$$= 1.316$$

$$\text{Area 2} \approx 1.32$$



Area 3 = Area under ef + Area under fg

$$\text{ef} \\ y = 0.2x - 0.6 \quad [3.8, 4]$$

$$fg \\ x_1 = 4 \quad x_2 = 5$$

$$y_1 = 0.2 \quad y_2 = 0$$

$$y = -0.2x + 1 \quad [4, 5]$$

$$\text{Area 3} = \int_{3.8}^4 (0.2x - 0.6) dx + \int_{3.8}^5 (-0.2x + 1) dx$$

$$\text{Area 3} = 0.136$$

$$z^* = \frac{\int \mu_{C_m}(z) \cdot z dz}{\int \mu_{C_m}(z) \cdot dz}$$

$$z^* = \frac{\int_{1.7}^{2.5} (0.67x - 0.67) \cdot x dx + \int_{2.5}^4 (-0.67x + 2.67) \cdot x dx}{\int_{1.7}^4 (0.67x - 0.67) \cdot dx + \int_{2.5}^4 (-0.67x + 2.67) \cdot dx}$$

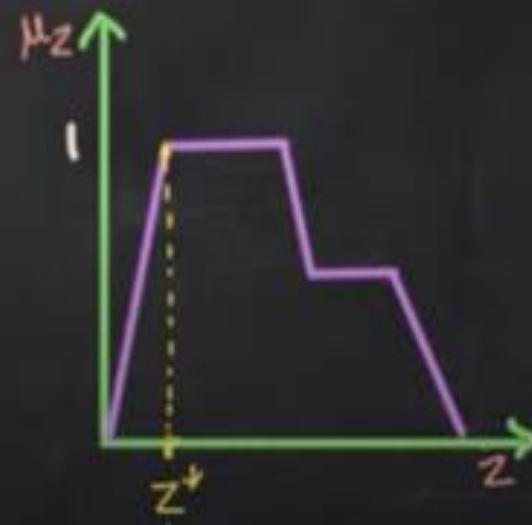
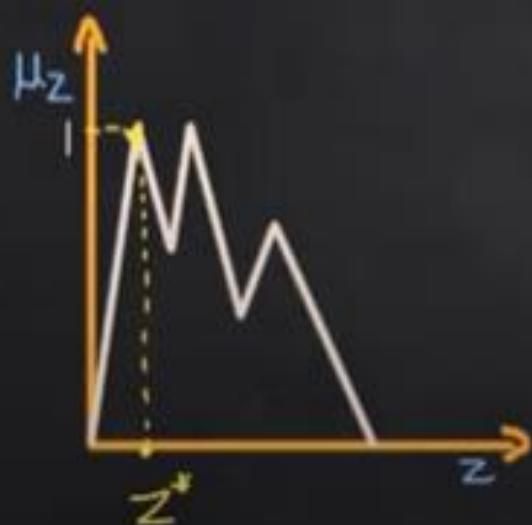
$$z^* = \frac{\int_{1.7}^{2.5} (0.67x - 0.67) \cdot dx + \int_{2.5}^4 (-0.67x + 2.67) \cdot dx}{\int_{1.7}^4 (0.67x - 0.67) \cdot dx + \int_{2.5}^4 (-0.67x + 2.67) \cdot dx}$$

7) First (or last) of maxima

The first of maxima is given by:

$$z^* = \inf_{z \in Z} \{z \in Z \mid \mu_{C_k} = hght(C_k)\}$$

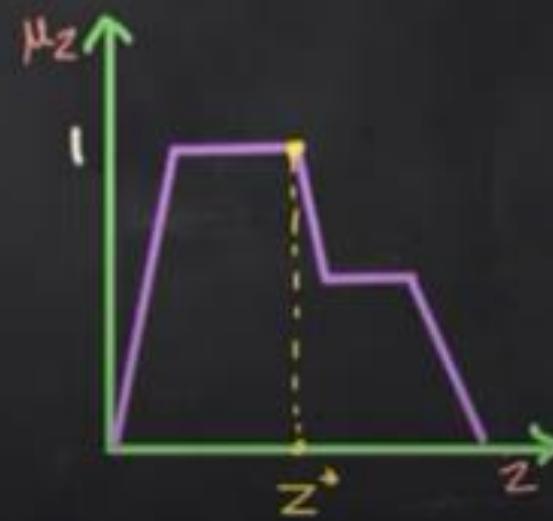
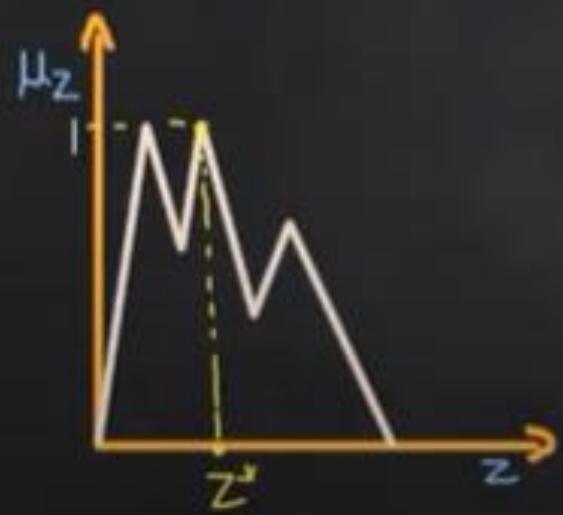
inf = infimum which is the greatest lower bound

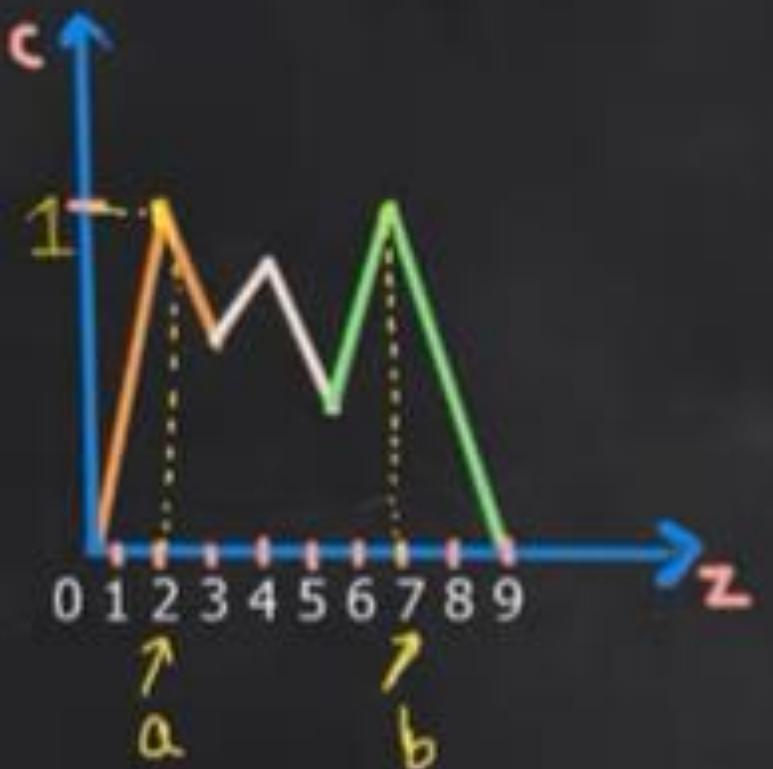


The last of maxima is given by:

$$z^* = \sup_{z \in Z} \{ z \in Z \mid \mu_{C_k} = hght(C_k) \}$$

sup = supremum which is the greatest upper bound





first of maxima

$$\vec{z}^* = a$$

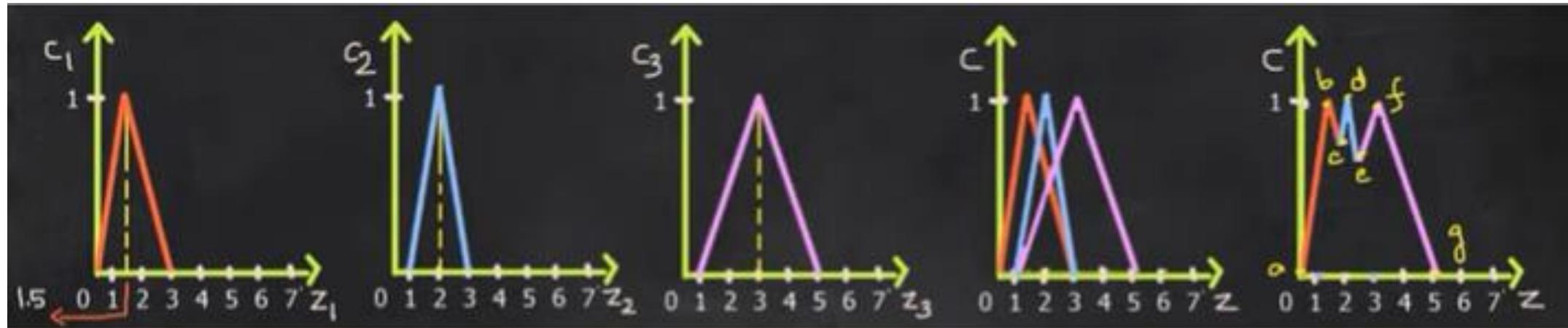
$$\vec{z}^* = 2$$

Last of maxima

$$\vec{z}^* = b$$

$$\vec{z}^* = 7$$

Solved Example



Centroid Method

$$ab \quad y = 0.67x \quad [0, 1.5]$$

$$bc \quad y = -0.67x + 2 \quad [1.5, 1.8]$$

$$cd \quad y = x - 1 \quad [1.8, 2]$$

$$de \quad y = -x + 3 \quad [2, 2.33]$$

$$ef \quad y = 0.5x - 0.5 \quad [2.33, 2.5]$$

$$fg \quad y = -0.5x + 2.5 \quad [2.5, 3]$$

$$z^* = \frac{\int \mu_z \cdot z dz}{\int \mu_z \cdot dz}$$

$$z^* = \frac{\int_0^{1.5} (0.67x) \cdot x dx + \int_{1.5}^{1.8} (-0.67x+2) \cdot x dx + \int_{1.8}^2 (x-1) \cdot x dx + \int_2^{2.33} (-x+3) \cdot x dx + \int_{2.33}^3 (0.5x-0.5) \cdot x dx}{\int_0^{1.5} (0.67x) \cdot dx + \int_{1.5}^{1.8} (-0.67x+2) \cdot dx + \int_{1.8}^2 (x-1) \cdot dx + \int_2^{2.33} (-x+3) \cdot dx + \int_{2.33}^3 (0.5x-0.5) \cdot dx}$$

$$z^* = \frac{\int_0^{1.5} (0.67x) \cdot dx + \int_{1.5}^{1.8} (-0.67x+2) \cdot dx + \int_{1.8}^2 (x-1) \cdot dx + \int_2^{2.33} (-x+3) \cdot dx + \int_{2.33}^3 (0.5x-0.5) \cdot dx}{\int_0^{1.5} (0.67x) \cdot dx + \int_{1.5}^{1.8} (-0.67x+2) \cdot dx + \int_{1.8}^2 (x-1) \cdot dx + \int_2^{2.33} (-x+3) \cdot dx + \int_{2.33}^3 (0.5x-0.5) \cdot dx}$$

$$z^* = \underline{\underline{2.5}}$$

Centre of Largest Area

Area 1=area under ab+area under bc

$$\text{Area 1} = \int_{1.8}^{1.5} (0.67x) \cdot dx + \int_0^{1.8} (-0.67x+2) \cdot dx$$

$$\text{Area 1} = \underline{\underline{1.02}}$$

Area 2=area under cd+area under de

$$\text{Area 2} = \int_{1.8}^2 (x-1) \cdot dx + \int_2^3 (-x+3) \cdot dx$$

$$\text{Area 2} = \underline{\underline{0.46}}$$

Area 3=area under ef+area under fg

$$\text{Area 3} = \int_{2.33}^3 (0.5x - 0.7) \cdot dx + \int_3^5 (-0.5x + 2.5) \cdot dx$$

$$\text{Area 3} = \underline{\underline{1.56}}$$

$$z^* = \frac{\int \mu_{C_m}(z) \cdot z \cdot dz}{\int \mu_{C_m}(z) \cdot dz}$$

$$z^* = \frac{\int_{2.33}^3 (0.5x - 0.5) \cdot x \cdot dx + \int_3^5 (-0.5x + 2.5) \cdot x \cdot dx}{\int_{2.33}^3 (0.5x - 0.5) \cdot dx + \int_3^5 (-0.5x + 2.5) \cdot dx}$$

$$z^* = 3.3$$

Weighted Average Method

$$z^* = \frac{\sum \mu_A(z) \cdot z}{\sum \mu_A(z)}$$

$$z^* = \frac{1.5 \times 1 + 2 \times 1 + 3 \times 1}{1 + 1 + 1}$$

$$z^* = \underline{\underline{2.25}}$$

Maximum Membership Principle

Centre Of Sums Method

$$z^* = \frac{\sum_{i=1}^n z_i A_{C_i}}{\sum_{i=1}^n A_{C_i}}$$

$$\text{Area of } C_1 = 2 \times \frac{1}{2} \times b \times h \\ = b \times h \\ \Rightarrow 1.5 \times 1$$

$$\text{Area of } C_2 \rightarrow 1.5$$

$$\text{Area of } C_3 = b \times h \\ = 1 \times 1$$

$$\text{Area of } C_4 = 1$$

$$\text{Area of } C_5 = b \times h \\ \Rightarrow 2 \times 1$$

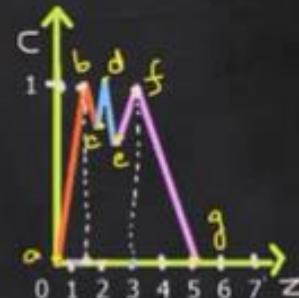
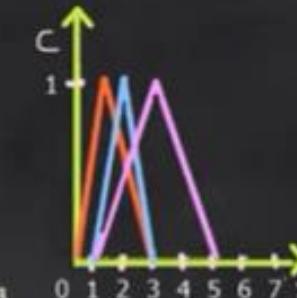
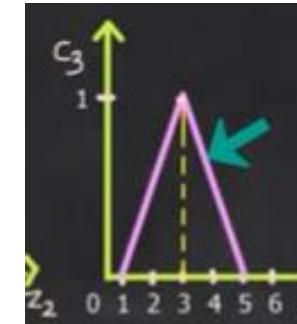
$$\text{Area of } C_6 \Rightarrow 2$$

$$z_1 = 1.5 \quad z_2 = 2 \quad z_3 = 3$$

$$z^* = \frac{z_1 A_1 + z_2 A_2 + z_3 A_3}{A_1 + A_2 + A_3} \\ \Rightarrow 1.5 \times 1.5 + 2 \times 1 + 3 \times 2 \\ \hline 1.5 + 1 + 2$$

$$z^* \Rightarrow 2.22$$

$$z^* \approx \underline{\underline{2.3}}$$



First (or) Last of Maxima

First of Maxima

$$z^* = 1.5$$

Last of Maxima

$$z^* = 3$$

Centroid Method

$$Z^* = 3.92$$

Centre of Largest Area

$$Z^* = 4$$

Maximum Membership Principle (Height Method)

Not Applicable

Weighted Average Method

$$Z^* = 3.97$$

Mean Max Membership (Middle of Maxima)

$$Z^* = 4$$

First Of Maxima

$$Z^* = 3$$

Last Of Maxima

$$Z^* = 5$$

Centre Of Sums

$$Z^* = 3.97$$

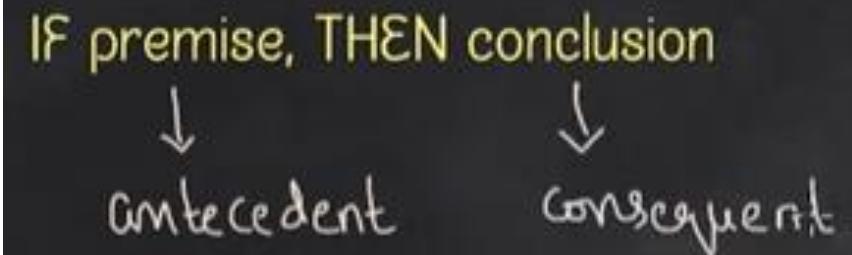
Fuzzy Rule Based System and Approximate Reasoning

Dr. Shruti Mishra
School of Computer Sc. & Engg. (SCOPE)

Introduction

- Fuzzy rule based system: Here, fuzzy logic and fuzzy sets are used as tools for representing different knowledge or logic about the problem.
- It is also used for modelling the interactions and its relations existing between the variables.

Contd..



Note: Antecedent is basically an hypothesis

These types of representation commonly refer to as **If-THEN rule based form**.

This *means that*, if we are aware of the fact or hypothesis then the conclusion can be derived.

Canonical Forms of Rule Based Systems

Rule 1: If condition C_1 , THEN restriction R_1

Rule 2: If condition C_2 , THEN restriction R_2

-

-

Rule n: If condition C_n , THEN restriction R_n

Rule 1: If condition C' , then restriction R'
a , c

Note: Restrictions are modelled by fuzzy sets and fuzzy relations, they are connected by linguistics statements are connected by AND, OR, ELSE. Similarly, we can have many conditions (antecedents) like conjunctive or disjunctive antecedents.

Decomposition of Rules (Compound Rules)

A compound rule is a collection of many simple rules combined together. Any compound rule structure may be decomposed and reduced to a number of simple canonical rule forms. The rules are generally based on natural language representations. The following are the methods used for decomposition of compound linguistic rules into simple canonical rules.

Multiple Conjunctive Antecedents

If x is \tilde{A}^1 and \tilde{A}^2 and $\tilde{A}^3 \dots \tilde{A}^L$ THEN y is \tilde{B}^S

x = input
 y = output
 $A_1, A_2 \dots A_L$ and B_S are fuzzy sets

↓
antecedent

Let us assume a new fuzzy subset \tilde{A}^S

$$\tilde{A}^S = \tilde{A}^1 \cap \tilde{A}^2 \cap \tilde{A}^3 \dots \tilde{A}^L - ①$$

$$\mu_{\tilde{A}^S}(x) = \min [\mu_{\tilde{A}^1}(x), \mu_{\tilde{A}^2}(x), \dots, \mu_{\tilde{A}^L}(x)]$$

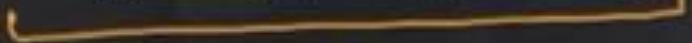
If x is \tilde{A}^S , THEN y is \tilde{B}^S

Rule

1

Multiple Disjunctive Antecedents

IF x is \tilde{A}^1 or \tilde{A}^2 or $\tilde{A}^3 \dots \tilde{A}^L$ THEN y is \tilde{B}^S




antecedent

Let us assume a new fuzzy subset \tilde{A}^S

$$\tilde{A}^S = \tilde{A}^1 \cup \tilde{A}^2 \cup \tilde{A}^3 \dots \tilde{A}^L - \textcircled{1}$$

$$\mu_{\tilde{A}^S}(x) = \max [\mu_{\tilde{A}^1}(x), \mu_{\tilde{A}^2}(x), \dots, \mu_{\tilde{A}^L}(x)]$$

IF x is \tilde{A}^S , THEN y is \tilde{B}^S

Rule

2

Conditional statements (with ELSE and UNLESS):

Statements of the kind

IF A_1 THEN (B_1 ELSE B_2)

can be decomposed into two simple canonical rule forms, connected by "OR":

IF A_1 THEN B_1

OR

IF NOT A_1 THEN B_2

IF A_1 (THEN B_1) UNLESS A_2

can be decomposed as

IF A_1 THEN B_1

OR

IF A_2 THEN NOT B_1

IF A_1 THEN (B_1) ELSE IF A_2 , THEN (B_2)

Rule

3

can be decomposed into the form

IF A_1 THEN B_1

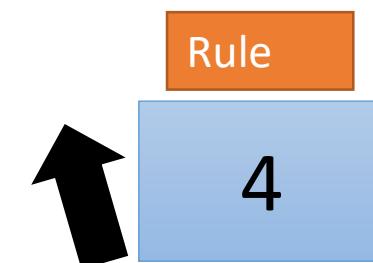
OR

IF NOT A_1 AND IF A_2 , THEN B_2

Nested-IF-THEN rules:

The rule "IF \underline{A}_1 THEN [IF \underline{A}_2 THEN (\underline{B}_1)]" can be of the form

IF \underline{A}_1 AND \underline{A}_2 THEN \underline{B}_1



Aggregation of Fuzzy Rules

1. Conjunctive System of Rules

In the case of a system of rules that must be jointly satisfied, the rules are connected by 'and' connectives.

In this case, the aggregated output, y , is found by the fuzzy intersection of all the individual rule consequents, y^i where $i = 1, 2, 3, \dots, r$

$$y = y^1 \text{ and } y^2 \text{ and } \dots \text{ and } y^r$$

or

$$y = y^1 \cap y^2 \cap \dots \cap y^r$$

$$\mu_y(y) = \min[\mu_{y^1}(y), \mu_{y^2}(y), \dots, \mu_{y^r}(y)]$$

2. Disjunctive System of Rules

In this rule system, we should satisfy atleast one rule.

The rules are connected by the 'or' connectives.

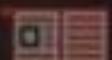
The aggregated output, y is found by the fuzzy union of all the individual rule consequents, y^i where $i = 1, 2, 3, \dots, r$

$$y = y^1 \text{ or } y^2 \text{ or } \dots \text{ or } y^r$$

or

$$y = y^1 \cup y^2 \cup \dots \cup y^r$$

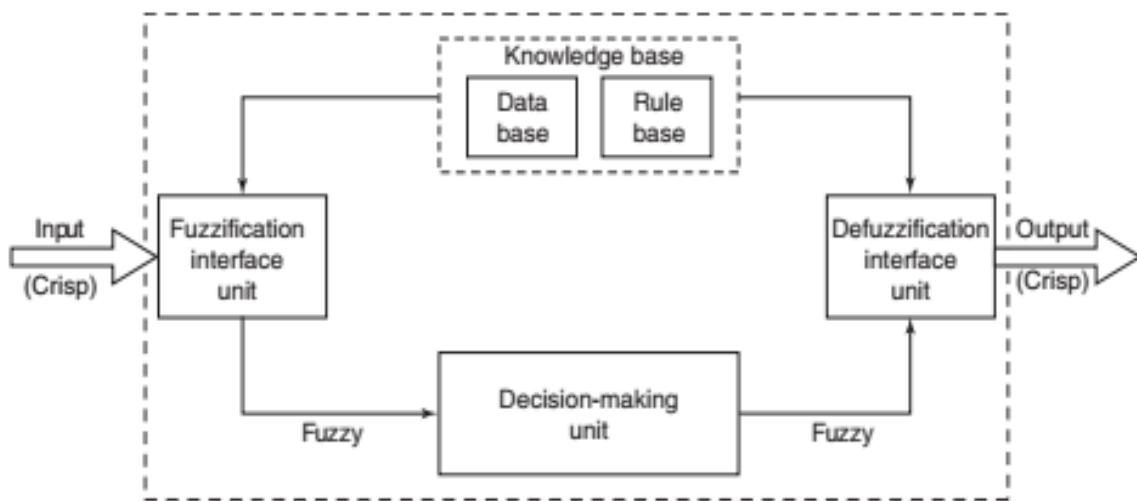
$$\mu_y(y) = \max[\mu_{y^1}(y), \mu_{y^2}(y), \dots, \mu_{y^r}(y)]$$



Fuzzy Inference System (FIS)

Fuzzy rule-based systems, fuzzy models, and fuzzy expert systems are generally known as fuzzy inference 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. When FIS is used as a controller, it is necessary to have crisp output. Hence, there should be a defuzzification unit for converting fuzzy variables into crisp variables along FIS. The entire FIS is discussed in detail in following subsections.

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 unit* that converts the fuzzy quantities into crisp quantities.



Block diagram of FIS.

Methods of FIS: Graphical Techniques of Inference

1. Mamdani Method



2. Sugeno Method

3. Tsukamoto Method

Mamdani Systems

IF x_1 is \tilde{A}_1 and x_2 is \tilde{A}_2 , THEN y is \tilde{B}

IF x_1 is \tilde{A}_1^k and x_2 is \tilde{A}_2^k , THEN y is \tilde{B}^k

where $k = 1, 2, \dots, r$

We will be considering a 2-input Mamdani systems.

There are two cases for the 2-input Mamdani Systems:

1. Max - Min Inference Method
2. Max Product Inference Method

Note: Inference to reach a particular conclusion based on some hypothesis or evidence associated with the logic.

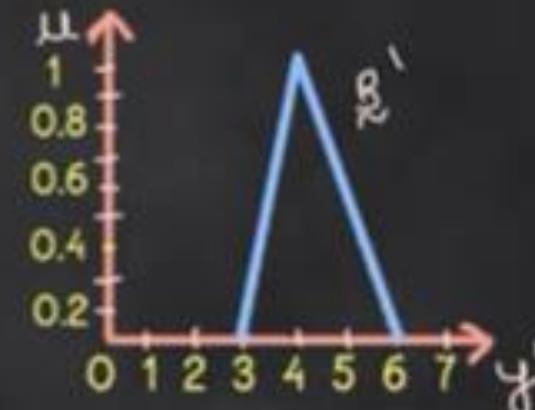
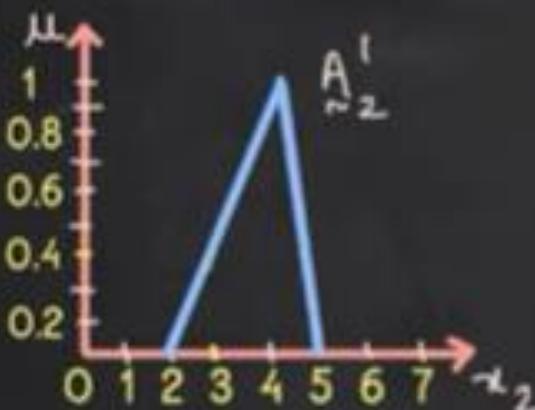
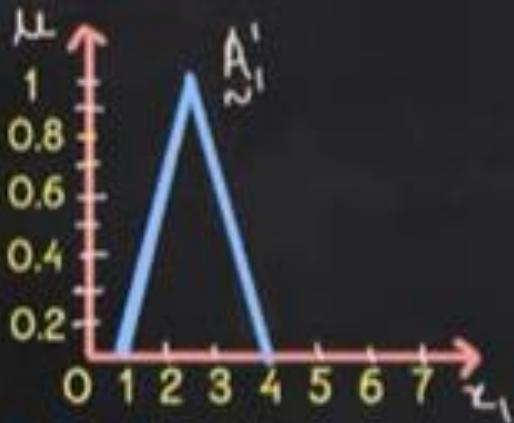
Max-Min Inference Method

Consider a simple 2 rule system where each rule has 2 antecedents and one consequent as follows:

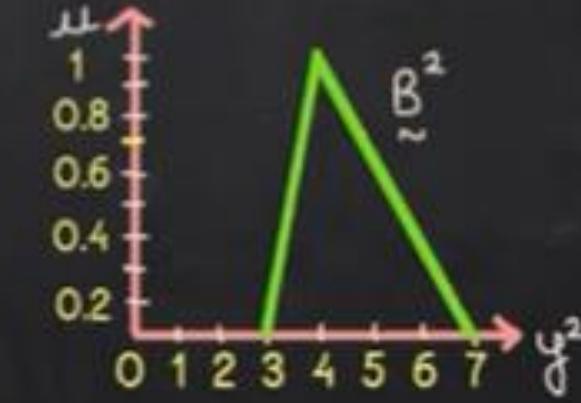
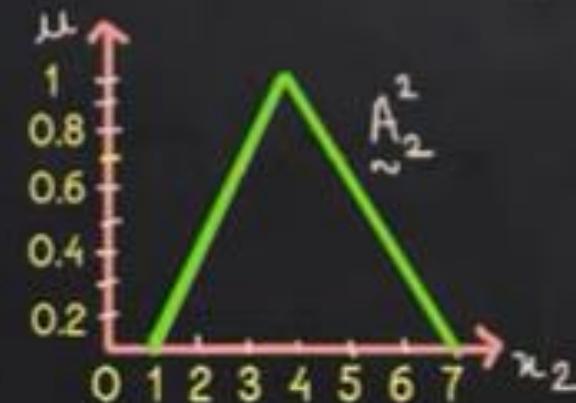
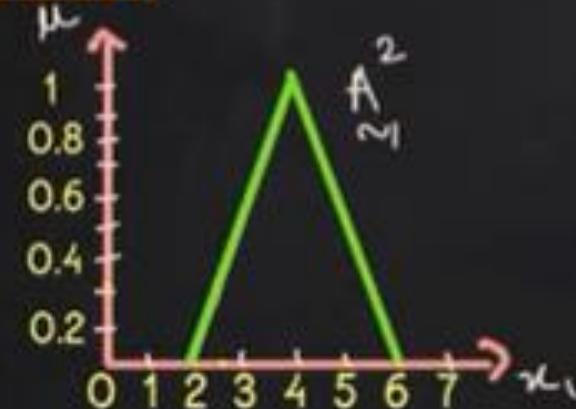
Rule 1: IF x_1 is A_1^1 and x_2 is A_2^1 , THEN y' is B^1
Rule 2: IF x_1 is $\sim A_1^2$ or x_2 is $\sim A_2^2$, THEN y' is $\sim B^2$

$$x_1 = 2.5 \quad x_2 = 3$$

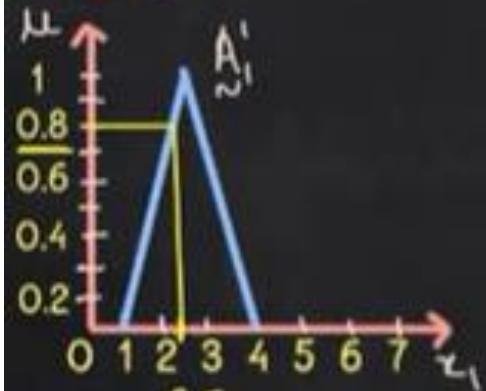
Rule 1:



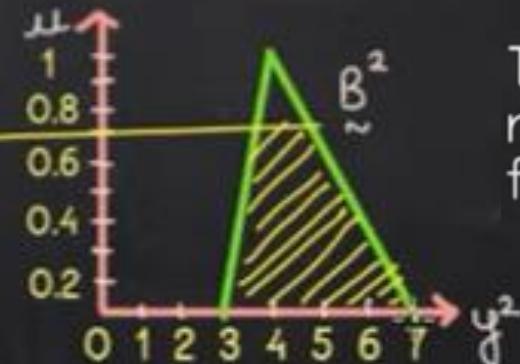
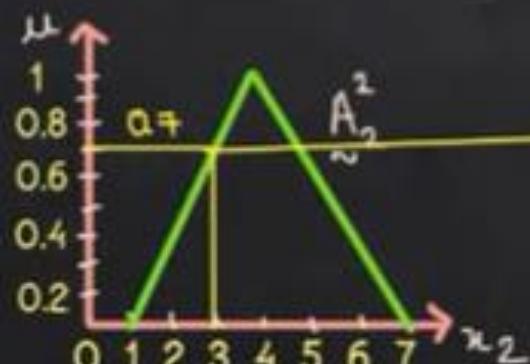
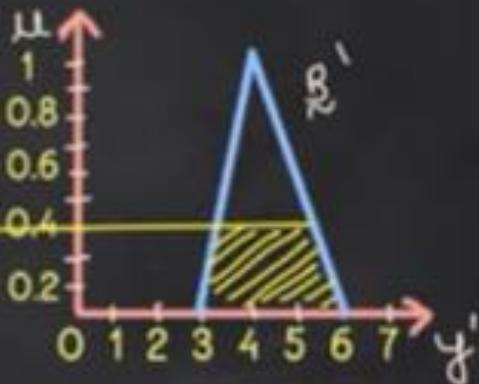
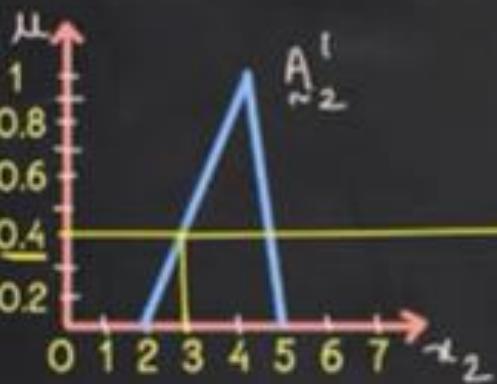
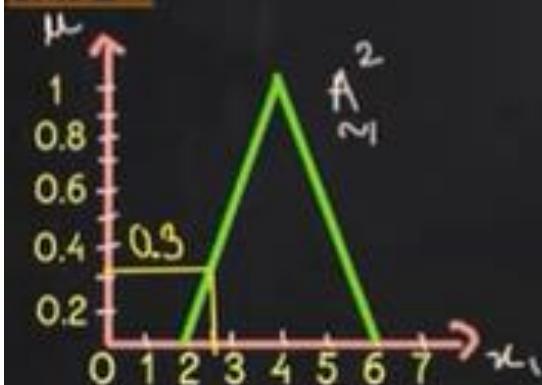
Rule 2:



Rule 1:

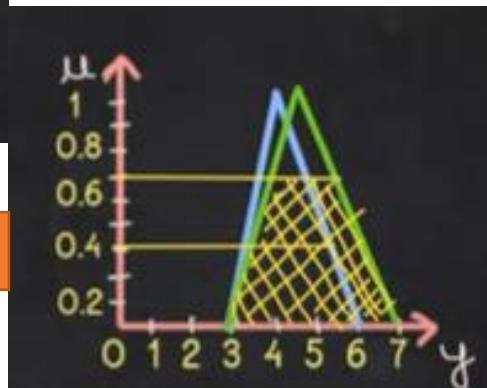


Rule 2:



Truncated
membership
function

Output



Whenever the antecedents are connected by 'and' operator, we take the minimum of the membership values of both x_1 and x_2 . If however they are connected by 'or' operator, we should take maximum of membership values of both x_1 and x_2 .

Next, we can apply any defuzzification method to find the crisp value

Max - Product Inference Method

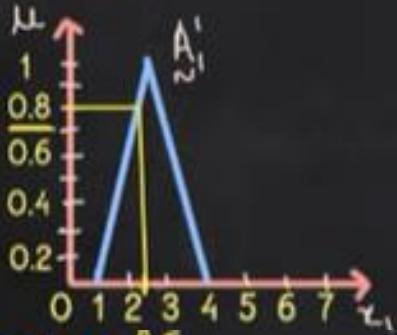
Consider a simple 2 rule system where each rule has 2 antecedents and one consequent as follows:

Rule 1: IF x_1 is A_1 and x_2 is A_2 , THEN y is B

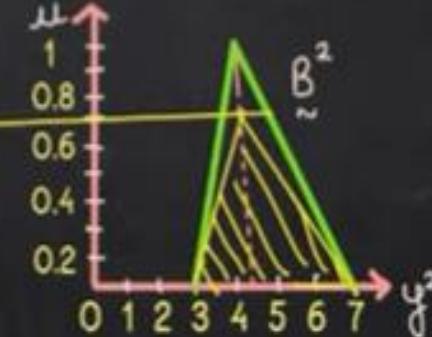
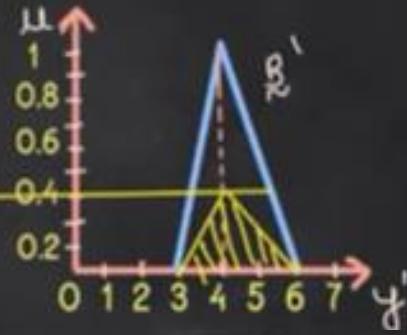
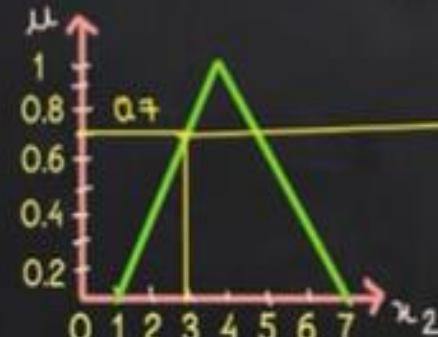
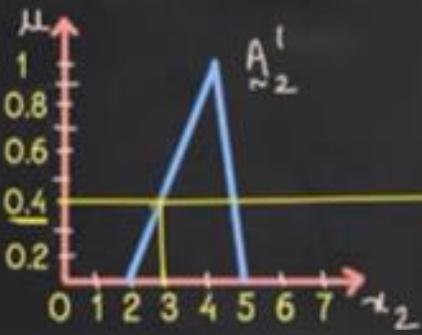
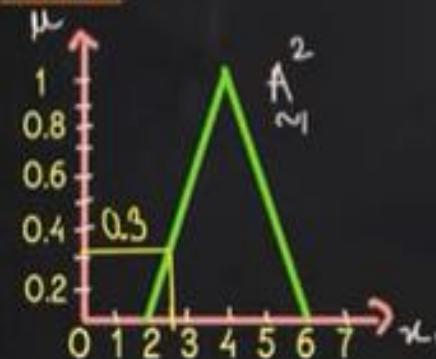
Rule 2: IF x_1 is A_1^2 or x_2 is A_2^2 , THEN y is B^2

$$x_1 = 2.5 \quad x_2 = 3$$

Rule 1:

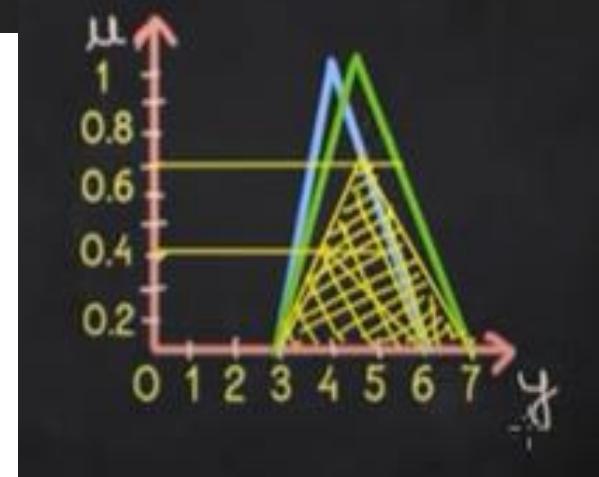


Rule 2:



Scaled
membership
function

Output



Next, we can apply any defuzzification method to find the crisp value

SUGENO SYSTEMS

Mamdani Systems : IF x_1 is \tilde{A}_1 and x_2 is \tilde{A}_2 , THEN y is \tilde{B}

Sugeno Systems : IF x_1 is \tilde{A}_1 and x_2 is \tilde{A}_2 , THEN y is $y = f(x_1, x_2)$

where $y = f(x_1, x_2)$ is a crisp function in the consequent.

The overall aggregated output will be obtained through weighted average defuzzification method.

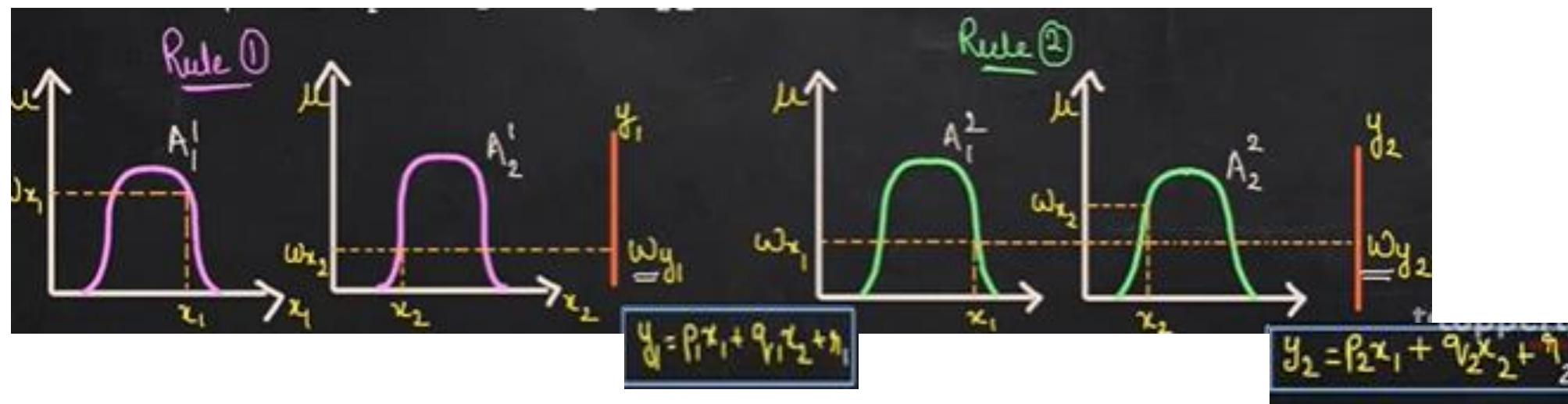
Rule 1: IF x_1 is \tilde{A}_1 and x_2 is \tilde{A}_2 , THEN y_1 is $y_1 = px_1 + qx_2 + r$,

Rule 2: IF x_1 is \tilde{A}_1 and x_2 is \tilde{A}_2 , THEN y_2 is $y_2 = p_2x_1 + q_2x_2 + r_2$

x_1 and x_2 - inputs
 y - output
 A_1, A_2 and B - Fuzzy sets

Note: In Sugeno systems, the antecedents are fuzzy sets or fuzzy numbers and the consequent is given as a function of the two inputs in the antecedent.

$$y^* = \frac{(\omega_{y_1} \cdot y_1) + (\omega_{y_2} \cdot y_2)}{\omega_{y_1} + \omega_{y_2}}$$



Consider a two input - single output Sugeno model with 4 rules as:

Rule 1 : IF x_1 is small and x_2 is small, THEN $y_1 = (-x_1) + x_2 + 1$

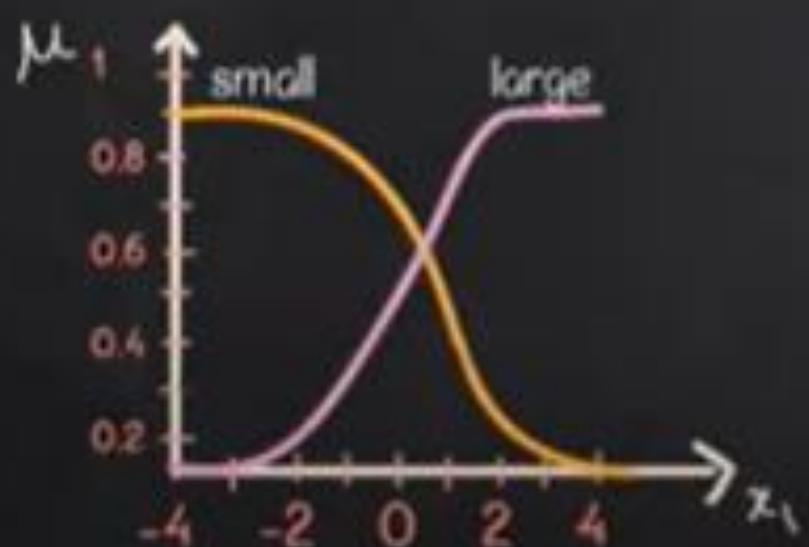
Rule 2 : IF x_1 is small and x_2 is large, THEN $y_2 = (-x_2) + 3$

Rule 3 : IF x_1 is large and x_2 is small, THEN $y_3 = (-x_1) + 3$

Rule 4 : IF x_1 is large and x_2 is large, THEN $y_4 = (-x_1) + x_2 + 2$

Find the output when $x_1 = 1.5$ and $x_2 = 2.5$.

The graphs for small and large fuzzy sets for x_1 and x_2 is given as:



Consider a two input - single output Sugeno model with 4 rules as:

Rule 1 : IF x_1 is small and x_2 is small, THEN $y_1 = (-x_1) + x_2 + 1$

Rule 2 : IF x_1 is small and x_2 is large, THEN $y_2 = (-x_2) + 3$

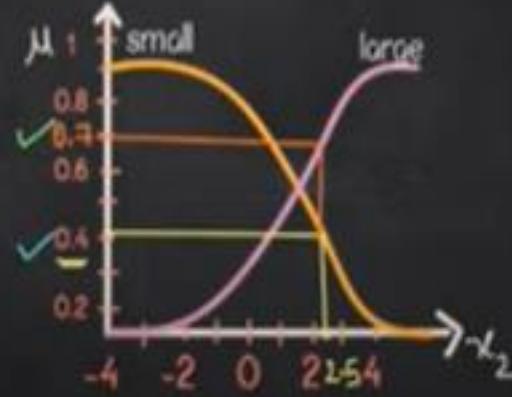
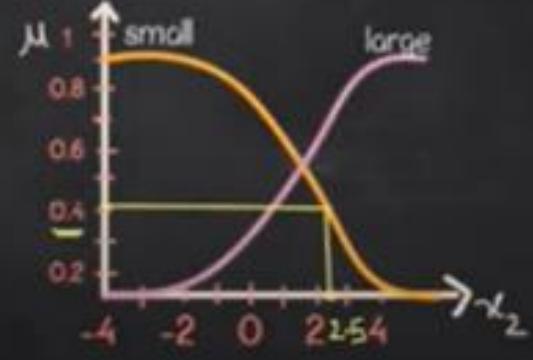
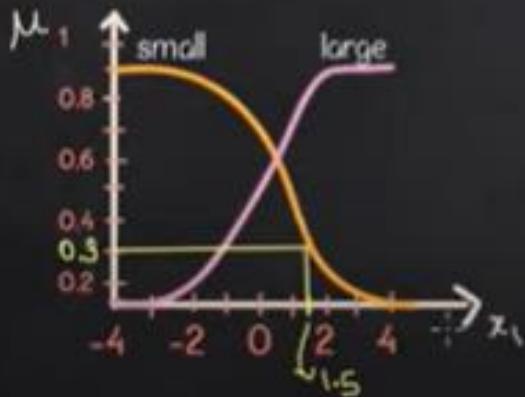
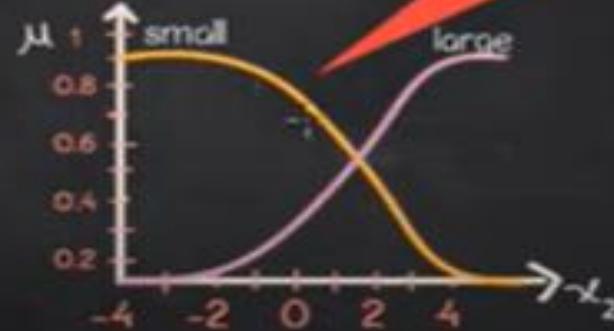
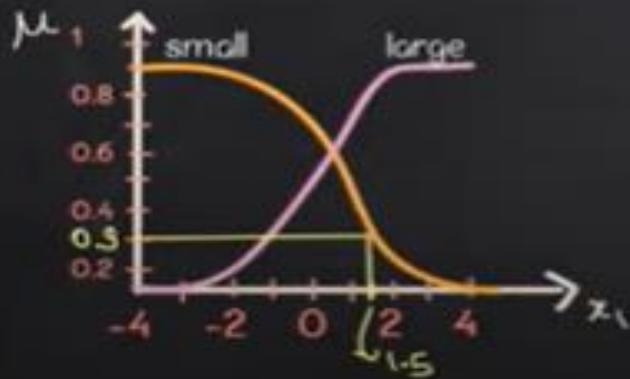
Rule 3 : IF x_1 is large and x_2 is small, THEN $y_3 = (-x_1) + 3$

Rule 4 : IF x_1 is large and x_2 is large, THEN $y_4 = (-x_1) + x_2 + 2$

Find the output when $x_1 = 1.5$ and $x_2 = 2.5$.

The graphs for small and large fuzzy sets for x_1 and x_2 is given as:

Since x_2 is given as
small in rule 1, we should
consider the small graph



Consider a two input - single output Sugeno model with 4 rules as:

Rule 1: IF x_1 is small and x_2 is small, THEN $y_1 = (-x_1) + x_2 + 1$

Rule 2: IF x_1 is small and x_2 is large, THEN $y_2 = (-x_2) + 3$

Rule 3: IF x_1 is large and x_2 is small, THEN $y_3 = (-x_1) + 3$

Rule 4: IF x_1 is large and x_2 is large, THEN $y_4 = (-x_1) + x_2 + 2$

Find the output when $x_1 = 1.5$ and $x_2 = 2.5$.

$$\omega_{y_1} = \min(\omega_{x_1}, \omega_{x_2}) = \min(0.3, 0.4)$$

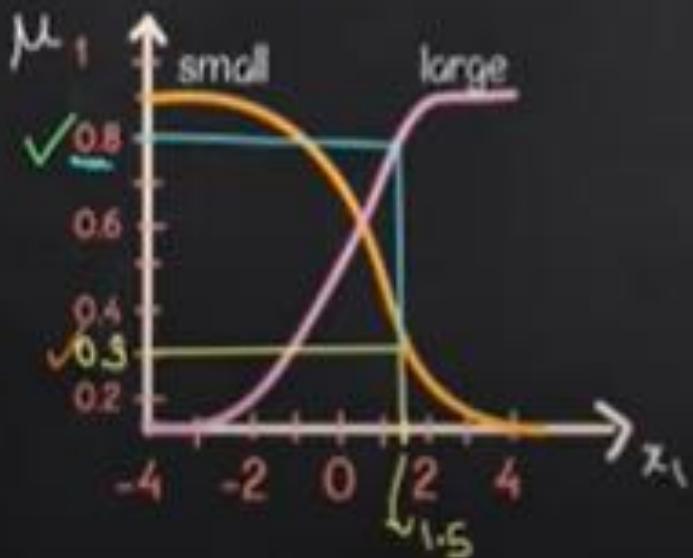
$$\omega_{y_1} = \underline{0.3}$$

$$\omega_{y_2} = \min(0.3, 0.7) = \underline{0.3}$$

$$\omega_{y_3} = \min(0.8, 0.4) = \underline{0.4}$$

$$\omega_{y_4} = \min(0.8, 0.7) = \underline{0.7}$$

The graphs for small and large fuzzy sets for x_1 and x_2 is given as:



$$y_1 = (-x_1) + x_2 + 1 = (-1.5) + 2.5 + 1$$

$$y_1 = \underline{3}$$

$$y_2 = \underline{0.5}$$

$$y_3 = \underline{1.5}$$

$$y_4 = \underline{6}$$

$$y^* = \frac{\omega_{y_1} y_1 + \omega_{y_2} y_2 + \omega_{y_3} y_3 + \omega_{y_4} y_4}{\omega_{y_1} + \omega_{y_2} + \omega_{y_3} + \omega_{y_4}}$$

$$y^* = \underline{3.264}$$

Note:

Sugeno method is better than Mamdani since it avoids the time consuming methods of defuzzification that are needed for Mamdani model.

TSUKAMOTO MODEL

Mamdoni Systems : IF x_1 is \tilde{A}_1 and x_2 is \tilde{A}_2 , THEN y is \tilde{B}

Sugeno Systems : IF x_1 is \tilde{A}_1 and x_2 is \tilde{A}_2 , THEN y is $y = f(x_1, x_2)$

Tsukamoto Systems : If x_1 is \tilde{A}_1 and x_2 is \tilde{A}_2 , THEN y is \tilde{B}

where the fuzzy set \tilde{B} has a monotonic membership function.

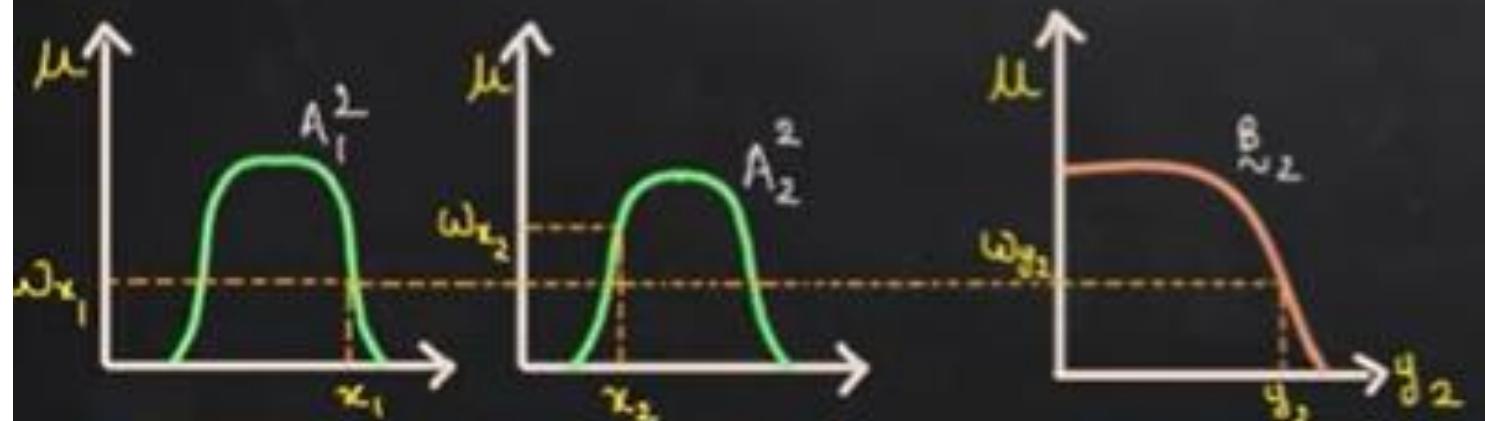
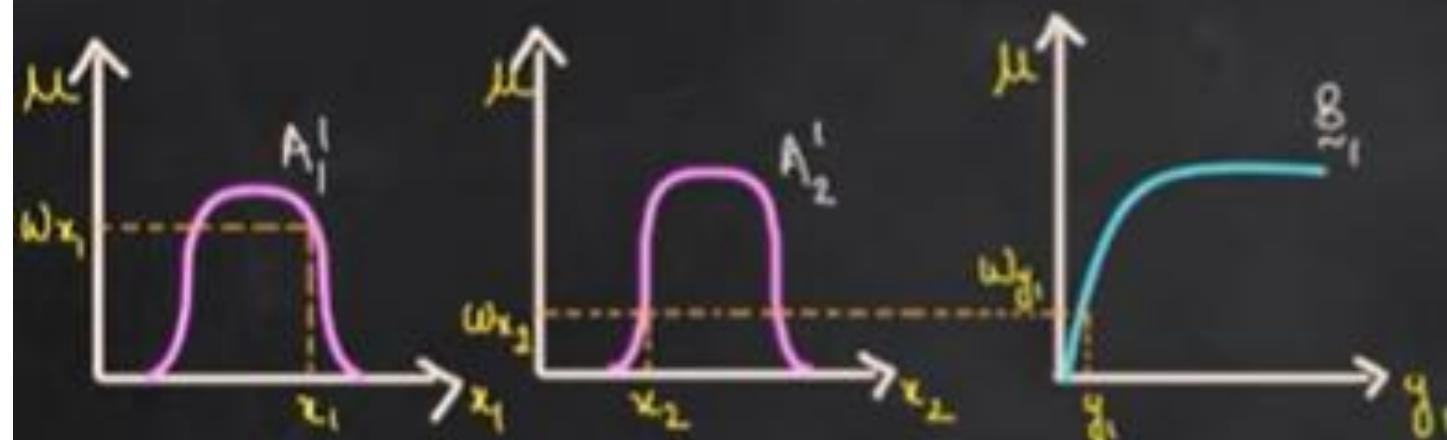
A monotonic function, also called as a shoulder function, is the one whose successive values are increasing, decreasing or constant.



The inferred output of each rule is defined as a crisp value induced by the membership value coming from the antecedent of the rule.

Rule 1: IF x_1 is A_1^1 and x_2 is A_2^1 THEN y_1 is B_1

Rule 2: IF x_1 is $\sim A_1^2$ and x_2 is $\sim A_2^2$ THEN y_2 is $\sim B_2$



The overall output will be calculated by the weighted average of each rule's output i.e weighted average of y_1 and y_2 as:

$$y^* = \frac{\omega_{y_1} \cdot y_1 + \omega_{y_2} \cdot y_2}{\omega_{y_1} + \omega_{y_2}}$$

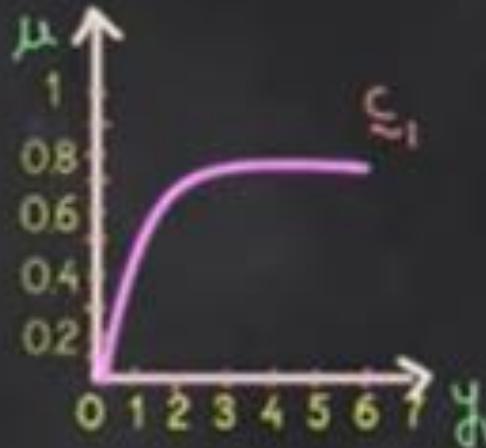
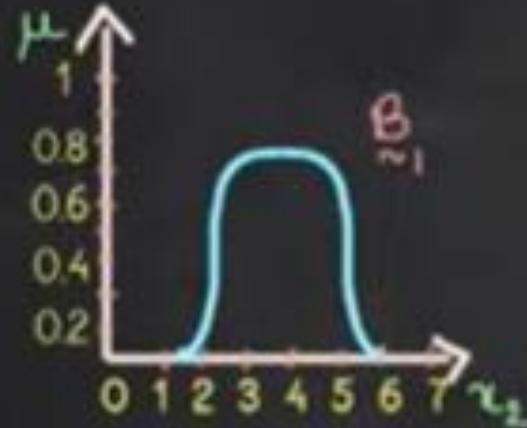
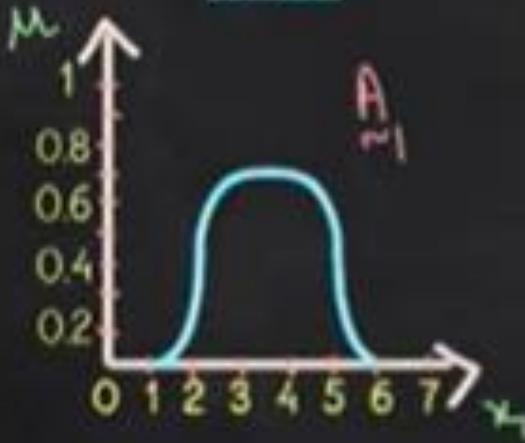
Q]

Rule 1 : IF x_1 is A_1 , and x_2 is B_1 , THEN y_1 is C_1 .

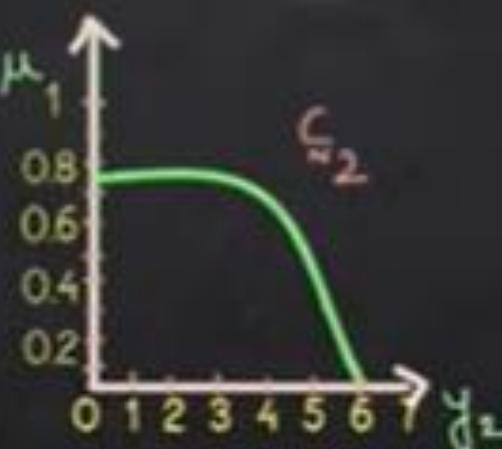
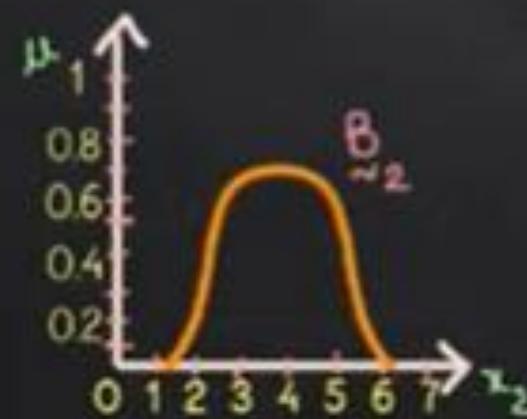
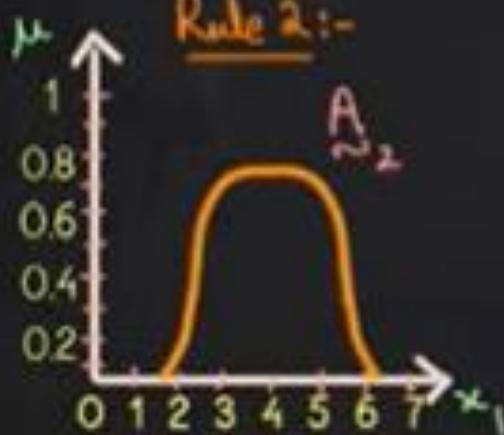
Rule 2 : IF x_1 is A_2 or x_2 is B_2 , THEN y_2 is C_2

$$x_1 = 2 \quad x_2 = 5$$

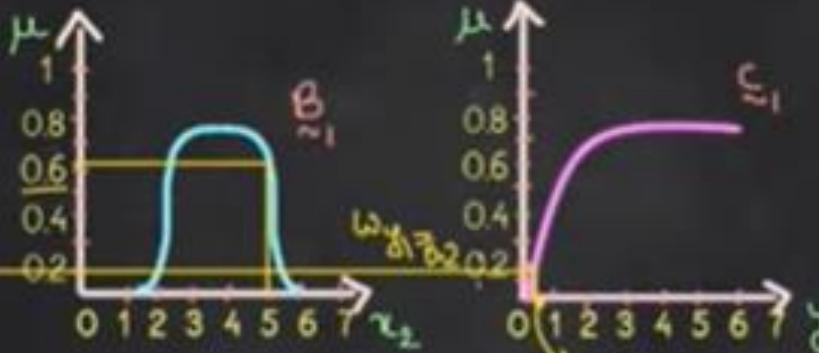
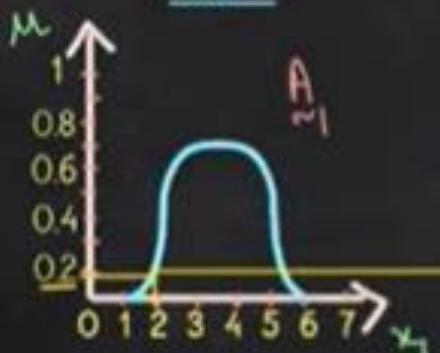
Rule 1 :-



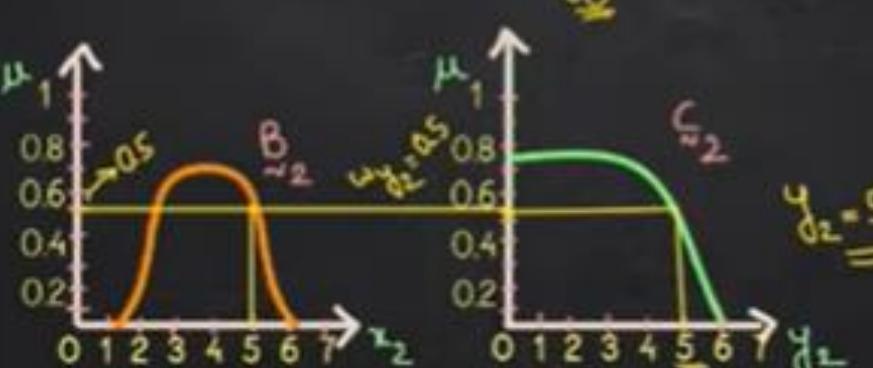
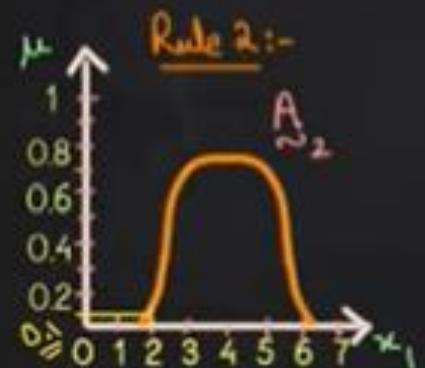
Rule 2 :-



Rule 1 :-



Rule 2 :-



$$y^* = \frac{\omega_{y_1} \cdot y_1 + \omega_{y_2} \cdot y_2}{\omega_{y_1} + \omega_{y_2}}$$

$$= \frac{0.2 \times 0.5 + 0.5 \times 5}{0.2 + 0.5}$$

$$= 3.714$$

$$y^* \approx 3.71$$

Advantages:

-:- Tsukamoto model avoids the time consuming process of defuzzification since each rule infers a crisp output and the overall output can be calculated using weighted average method.

Disadvantages:

Since the output membership function should be monotonic in nature, it's not as useful as a general approach and hence can only be used in specific situations.

Example

Consider a system with the following rules:

Rule 1: IF x is \tilde{A}_1 and y is \tilde{B}_1 , THEN z_1 is \tilde{C}_1 .

Rule 2: IF x is \tilde{A}_2 or y is \tilde{B}_2 , THEN z_2 is \tilde{C}_2 .

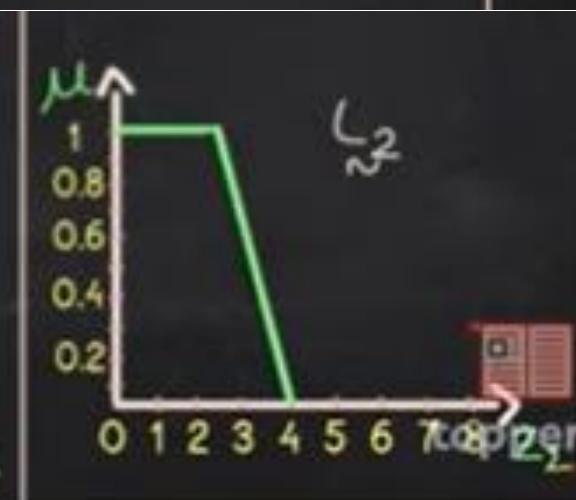
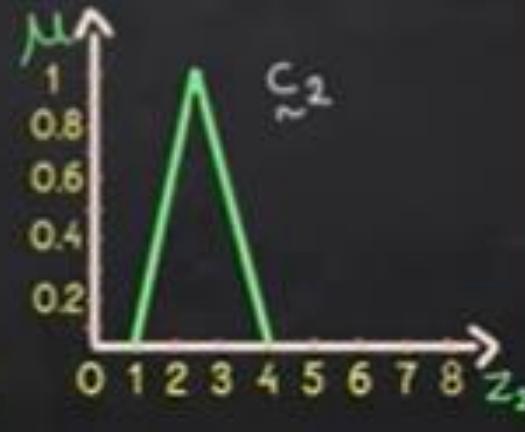
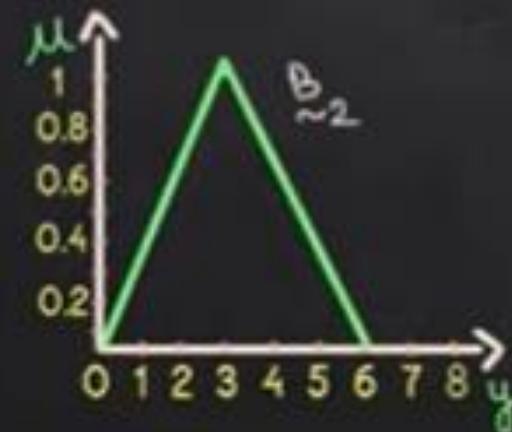
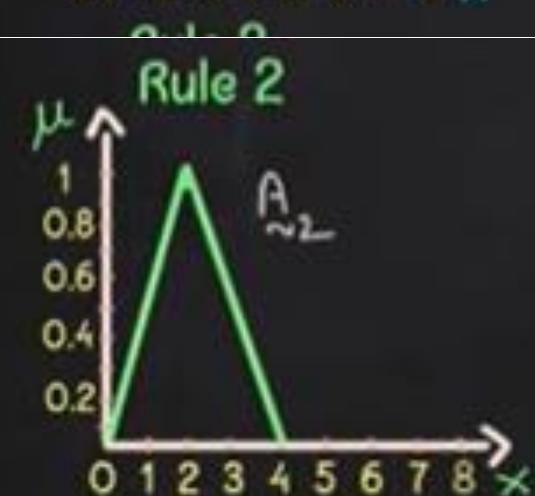
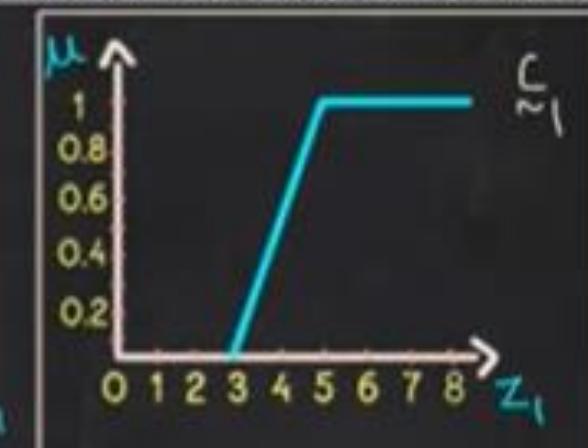
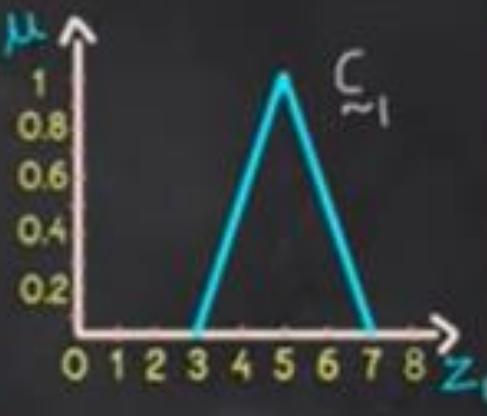
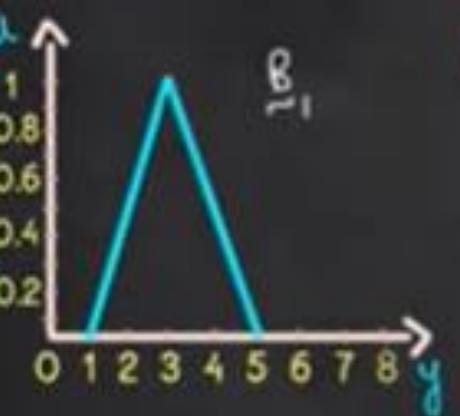
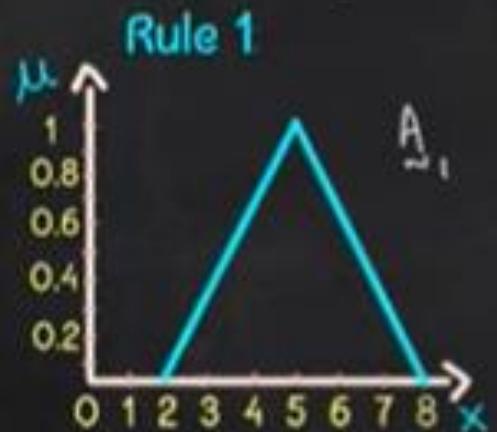
$$x = 3 \quad y = 4$$

The equations for z_1 and z_2 is given as:

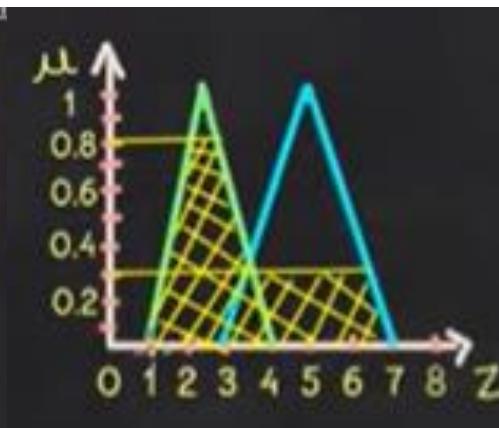
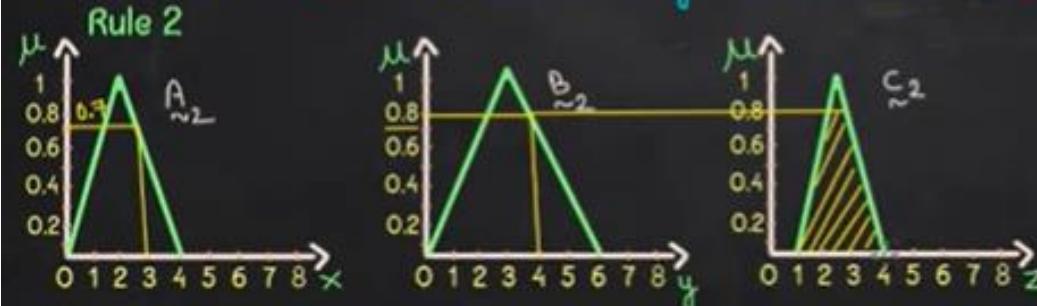
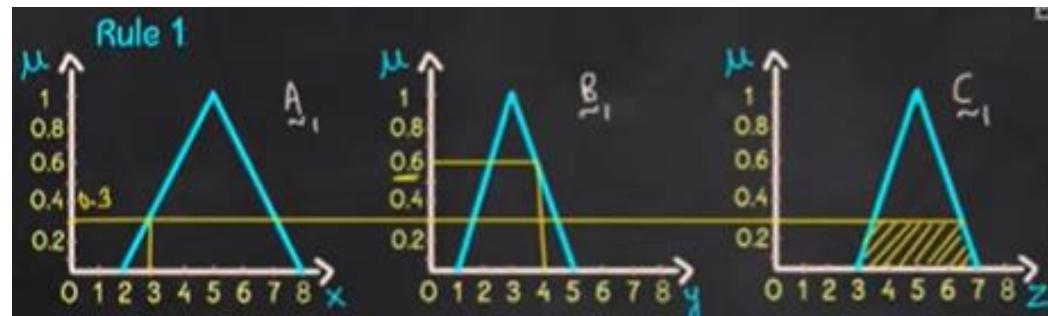
$$z_1 = x + 3y - 6$$

$$z_2 = 2x + y - 7$$

MONOTONIC MEMBERSHIP FUNCTIONS

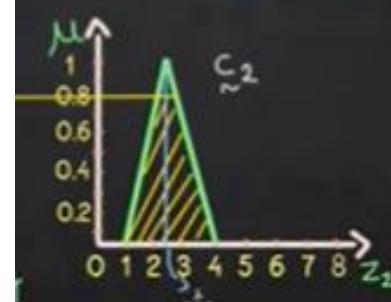
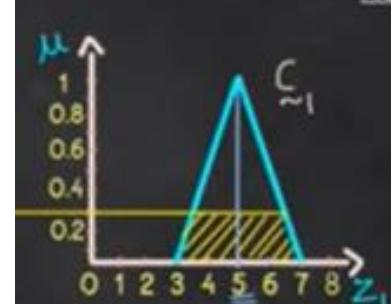


Mamdani: Max-Min inference system



$$Z^* = \frac{\omega_1 \bar{z}_1 + \omega_2 \bar{z}_2}{\omega_1 + \omega_2}$$

ω_1, ω_2 = Membership values of z_1 and z_2 respectively.
 \bar{z}_1, \bar{z}_2 = Centre of gravity of z_1 and z_2 respectively



$$\omega_1 = 0.3 \quad \omega_2 = 0.8 \quad \bar{z}_1 = 5 \quad \bar{z}_2 = 2.5$$
$$Z^* = \frac{(0.3 \times 5) + (0.8 \times 2.5)}{0.3 + 0.8} = 3.125$$
$$Z^* \approx 3.13$$

Sugeno Method

Sugeno Method: $x = 3 \quad y = 4$

$$z_1 = x + 3y - 6$$

$$z_2 = 2x + y - 7$$

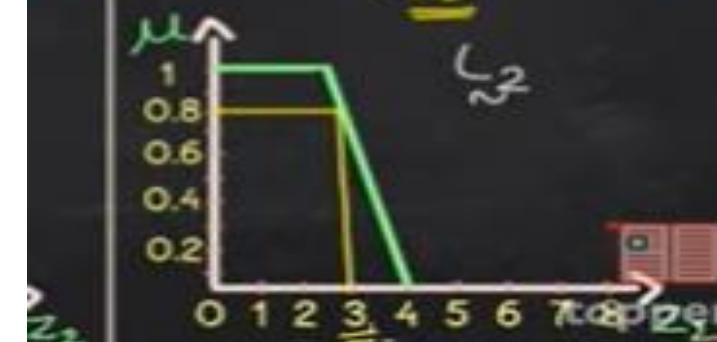
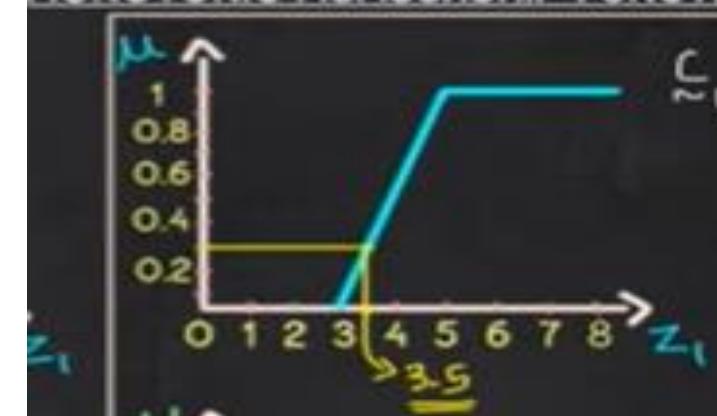
$$z_1 = 3 + 12 - 6 \Rightarrow z_1 = 9 \quad z_2 = 6 + 4 - 7 = 3$$

$$\omega_1 = 0.3 \quad \omega_2 = 0.8$$

$$z^* = \frac{\omega_1 z_1 + \omega_2 z_2}{\omega_1 + \omega_2} = \frac{(0.3 \times 9) + (0.8 \times 3)}{0.3 + 0.8} = 4.636 \quad z^* \approx 4.64$$

Tsukamoto Method

MONOTONIC MEMBERSHIP FUNCTIONS



Tsukamoto Method

$$\omega_1 = 0.3 \quad z_1 = 3.5 \quad \omega_2 = 0.8 \quad z_2 = 3$$

$$z^* = \frac{\omega_1 z_1 + \omega_2 z_2}{\omega_1 + \omega_2}$$

$$= \frac{(0.3 \times 3.5) + (0.8 \times 3)}{0.3 + 0.8} = 3.1$$

$$z^* \approx 3.14$$

Fuzzy Implications

- A fuzzy implication (also known as fuzzy If-Then rule, fuzzy rule, or fuzzy conditional statement) assumes the form :

If x is A then y is B

where, A and B are two linguistic variables defined by fuzzy sets A and B on the universe of discourses X and Y , respectively.

- If pressure is High then temperature is Low
- If mango is Yellow then mango is Sweet else mango is Sour
- If road is Good then driving is Smooth else traffic is High
- The fuzzy implication is denoted as $R : A \rightarrow B$
- In essence, it represents a binary fuzzy relation R on the (Cartesian) product of $A \times B$

Fuzzy implication : Example 2

- Suppose, P and T are two universes of discourses representing pressure and temperature, respectively as follows.
- $P = \{ 1,2,3,4\}$ and $T = \{ 10, 15, 20, 25, 30, 35, 40, 45, 50 \}$
- Let the linguistic variable **High temperature** and **Low pressure** are given as
- $T_{HIGH} = \{(20, 0.2), (25, 0.4), (30, 0.6), (35, 0.6), (40, 0.7), (45, 0.8), (50, 0.8)\}$
- $P_{LOW} = (1, 0.8), (2, 0.8), (3, 0.6), (4, 0.4)$

- Then the fuzzy implication **If temperature is High then pressure is Low** can be defined as

$$R : T_{HIGH} \rightarrow P_{LOW}$$

where, $R =$

	1	2	3	4
20	0.2	0.2	0.2	0.2
25	0.4	0.4	0.4	0.4
30	0.6	0.6	0.6	0.4
35	0.6	0.6	0.6	0.4
40	0.7	0.7	0.6	0.4
45	0.8	0.8	0.6	0.4
50	0.8	0.8	0.6	0.4

Note : If temperature is 40 then what about low pressure?

Example 3: Zadeh's Max-Min rule

If x is A then y is B with the implication of Zadeh's max-min rule can be written equivalently as :

$$R_{mm} = (A \times B) \cup (\bar{A} \times Y)$$

Here, Y is the universe of discourse with membership values for all $y \in Y$ is 1, that is , $\mu_Y(y) = 1 \forall y \in Y$.

Suppose $X = \{a, b, c, d\}$ and $Y = \{1, 2, 3, 4\}$

and $A = \{(a, 0.0), (b, 0.8), (c, 0.6), (d, 1.0)\}$

$B = \{(1, 0.2), (2, 1.0), (3, 0.8), (4, 0.0)\}$ are two fuzzy sets.

We are to determine $R_{mm} = (A \times B) \cup (\bar{A} \times Y)$

The computation of $R_{mm} = (A \times B) \cup (\bar{A} \times Y)$ is as follows:

$$A \times B = \begin{array}{c} \begin{matrix} & 1 & 2 & 3 & 4 \\ \begin{matrix} a \\ b \\ c \\ d \end{matrix} & \left[\begin{matrix} 0 & 0 & 0 & 0 \\ 0.2 & 0.8 & 0.8 & 0 \\ 0.2 & 0.6 & 0.6 & 0 \\ 0.2 & 1.0 & 0.8 & 0 \end{matrix} \right] \end{matrix} \end{array} \text{ and}$$

$$\bar{A} \times Y = \begin{array}{c} \begin{matrix} & 1 & 2 & 3 & 4 \\ \begin{matrix} a \\ b \\ c \\ d \end{matrix} & \left[\begin{matrix} 1 & 1 & 1 & 1 \\ 0.2 & 0.2 & 0.2 & 0.2 \\ 0.4 & 0.4 & 0.4 & 0.4 \\ 0 & 0 & 0 & 0 \end{matrix} \right] \end{matrix} \end{array}$$

Therefore,

$$R_{mm} = (A \times B) \cup (\bar{A} \times Y) = \begin{array}{c} \begin{matrix} & 1 & 2 & 3 & 4 \\ \begin{matrix} a \\ b \\ c \\ d \end{matrix} & \left[\begin{matrix} 1 & 1 & 1 & 1 \\ 0.2 & 0.8 & 0.8 & 0.2 \\ 0.4 & 0.6 & 0.6 & 0.4 \\ 0.2 & 1.0 & 0.8 & 0 \end{matrix} \right] \end{matrix} \end{array}$$

$$X = \{a, b, c, d\}$$

$$Y = \{1, 2, 3, 4\}$$

Let, $A = \{(a, 0.0), (b, 0.8), (c, 0.6), (d, 1.0)\}$

$B = \{(1, 0.2), (2, 1.0), (3, 0.8), (4, 0.0)\}$

Determine the implication relation :

If x is A then y is B

Here, $A \times B =$

	1	2	3	4
a	0	0	0	0
b	0.2	0.8	0.8	0
c	0.2	0.6	0.6	0
d	0.2	1.0	0.8	0

and $\bar{A} \times Y =$

$$\begin{array}{cccc} & 1 & 2 & 3 & 4 \\ a & 1 & 1 & 1 & 1 \\ b & 0.2 & 0.2 & 0.2 & 0.2 \\ c & 0.4 & 0.4 & 0.4 & 0.4 \\ d & 0 & 0 & 0 & 0 \end{array}$$

$R_{mm} = (A \times B) \cup (\bar{A} \times Y) =$

$$\begin{array}{cccc} & 1 & 2 & 3 & 4 \\ a & 1 & 1 & 1 & 1 \\ b & 0.2 & 0.8 & 0.8 & 0.2 \\ c & 0.4 & 0.6 & 0.6 & 0.4 \\ d & 0.2 & 1.0 & 0.8 & 0 \end{array}$$

This R represents **If x is A then y is B**

IF x is A THEN y is B ELSE y is C.

The relation R is equivalent to

$$R = (A \times B) \cup (\bar{A} \times C)$$

The membership function of R is given by

$$\mu_R(x, y) = \max[\min\{\mu_A(x), \mu_B(y)\}, \min\{\mu_{\bar{A}}(x), \mu_C(y)\}]$$

Example 4:

$$X = \{a, b, c, d\}$$

$$Y = \{1, 2, 3, 4\}$$

$$A = \{(a, 0.0), (b, 0.8), (c, 0.6), (d, 1.0)\}$$

$$B = \{(1, 0.2), (2, 1.0), (3, 0.8), (4, 0.0)\}$$

$$C = \{(1, 0), (2, 0.4), (3, 1.0), (4, 0.8)\}$$

Determine the implication relation :

If x is A then y is B else y is C

$$\text{Here, } A \times B = \begin{array}{cccc} & 1 & 2 & 3 & 4 \\ \begin{matrix} a \\ b \\ c \\ d \end{matrix} & \left[\begin{matrix} 0 & 0 & 0 & 0 \\ 0.2 & 0.8 & 0.8 & 0 \\ 0.2 & 0.6 & 0.6 & 0 \\ 0.2 & 1.0 & 0.8 & 0 \end{matrix} \right] \end{array}$$

$$\text{and } \bar{A} \times C =$$

$$R =$$

$$\begin{array}{ccccc} & 1 & 2 & 3 & 4 \\ \begin{matrix} a \\ b \\ c \\ d \end{matrix} & \left[\begin{matrix} 0 & 0.4 & 1.0 & 0.8 \\ 0 & 0.2 & 0.2 & 0.2 \\ 0 & 0.4 & 0.4 & 0.4 \\ 0 & 0 & 0 & 0 \end{matrix} \right] \end{array}$$

$$\begin{array}{ccccc} & 1 & 2 & 3 & 4 \\ \begin{matrix} a \\ b \\ c \\ d \end{matrix} & \left[\begin{matrix} 0 & 0.4 & 1.0 & 0.8 \\ 0.2 & 0.8 & 0.8 & 0.2 \\ 0.2 & 0.6 & 0.6 & 0.4 \\ 0.2 & 1.0 & 0.8 & 0 \end{matrix} \right] \end{array}$$

Fuzzy c-Means Clustering Algorithm

**Dr. Shruti Mishra
School of Computer Sc. & Engg. (SCOPE)**

Fuzzy C-Means

- An extension of k-means
- Hierarchical, k-means generates partitions
 - each data point can only be assigned in one cluster
- Fuzzy c-means allows data points to be assigned into more than one cluster
 - each data point has a degree of membership (or probability) of belonging to each cluster

Note:

Fuzzy c-means (FCM) is a method of clustering which allows one piece of data to belong to two or more clusters. This method is frequently used in pattern recognition. It is based on minimization of the objective function !

Algorithm

- The algorithm relies on the user to specify the number of clusters present in the set of data to be clustered.
- Given a number of clusters c , FCMC partitions the data $X = \{x_1, x_2, \dots, x_n\}$ into c fuzzy clusters by minimizing the within group sum of squared error objective function as follows:

$$J_m(U, V) = \sum_{k=1}^n \sum_{i=1}^c (U_{ik})^m \|x_k - v_i\|^2, \quad (1)$$
$$1 \leq m \leq \infty$$

- where $J_m(U,V)$ is the sum of squared error for the set of fuzzy clusters represented by the membership matrix U , and the associated set of cluster centres V .
- $\| \cdot \|$ is some inner product induced norm.
- In the formula, $\|x_k - v_i\|^2$ represents the distance between the data x_k and the cluster centre v_i .
- The squared error is used as a performance index that measures the weighted sum of distances between cluster centers and elements in the corresponding fuzzy clusters.
- The number m governs the influence of membership grades in the performance index.
- The partition becomes fuzzier with increasing m and it has been shown that the FCMC algorithm converges for any $m \in (1, \infty)$.

- The necessary conditions for Eq. (1) to reach its minimum are:

$$U_{ik} = \left(\sum_{j=1}^c \left(\frac{\|x_k - v_i\|}{\|x_k - v_j\|} \right)^{2/(m-1)} \right)^{-1} \quad \forall i, \forall k \quad (2)$$

and

$$v_i = \frac{\sum_{k=1}^n (U_{ik})^m x_k}{\sum_{k=1}^n (U_{ik})^m} \quad (3)$$

- In each iteration of the FCMC algorithm, matrix U is computed using Eq. (2) and the associated cluster centers are computed as Eq. (3).
- This is followed by computing the square error in Eq. (1).
- The algorithm stops when either the error is below a certain tolerance value or its improvement over the previous iteration is below a certain threshold

Step-1: Randomly initialize the membership matrix using this equation,

$$\sum_{j=1}^c \mu_j(x_i) = 1 \quad i = 1, 2, \dots, k$$

Step-2: Calculate the Centroid using equation,

$$C_j = \frac{\sum_i [\mu_j(x_i)]^m x_i}{\sum_i [\mu_j(x_i)]^m}$$

Step-3: Calculate dissimilarity between the data points and Centroid using the Euclidean distance.

$$D_i = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Step-4: Update the New membership matrix using the equation,

$$\mu_j(x_i) = \frac{[\frac{1}{d_{ji}}]^{1/m-1}}{\sum_{k=1}^c [\frac{1}{d_{ki}}]^{1/m-1}}$$

Here **m** is a fuzzification parameter.

The range **m** is always [1.25, 2]

Step -5: Go back to Step 2, unless the centroids are not changing.

Advantages

- 1) Gives best result for overlapped data set and comparatively better than k-means algorithm.
- 2) Unlike k-means where data point must exclusively belong to one cluster center here data point is assigned membership to each cluster center as a result of which data point may belong to more than one cluster center.

Disadvantages

- 1) Apriori specification of the number of clusters.
- 2) With lower value of β we get the better result but at the expense of more number of iteration.
- 3) Euclidean distance measures can unequally weight underlying factors.

Example

Input: Number of Objects = 6 Number of clusters = 2

X	Y	C1	C2
1	6	0.8	0.2
2	5	0.9	0.1
3	8	0.7	0.3
4	4	0.3	0.7
5	7	0.5	0.5
6	9	0.2	0.8

Step-1: Initialize the membership matrix.

Step-2: Find the constraint using the equation

$$C_j = \left[\frac{\sum_i [\mu_j(x_i)]^m x_i}{\sum_i [\mu_j(x_i)]^m}, \frac{\sum_i [\mu_j(y_i)]^m y_i}{\sum_i [\mu_j(y_i)]^m} \right]$$

$$C_1 = \left[\frac{1*0.8^2 + 2*0.9^2 + 3*0.7^2 + 4*0.3^2 + 5*0.5^2 + 6*0.2^2}{0.8^2 + 0.9^2 + 0.7^2 + 0.3^2 + 0.5^2 + 0.2^2}, \frac{6*0.8^2 + 5*0.9^2 + 8*0.7^2 + 4*0.3^2 + 7*0.5^2 + 9*0.2^2}{0.8^2 + 0.9^2 + 0.7^2 + 0.3^2 + 0.5^2 + 0.2^2} \right]$$

$$C_1 = \frac{5.58}{2.32}, \frac{14.28}{2.32}$$

$$C_1 = (2.4, 6.1)$$

$$C_2 = \left[\frac{1*0.2^2 + 2*0.1^2 + 3*0.3^2 + 4*0.7^2 + 5*0.5^2 + 6*0.8^2}{0.2^2 + 0.1^2 + 0.3^2 + 0.7^2 + 0.5^2 + 0.8^2}, \right. \\ \left. \frac{6*0.2^2 + 5*0.1^2 + 8*0.3^2 + 4*0.7^2 + 7*0.5^2 + 9*0.8^2}{0.2^2 + 0.1^2 + 0.3^2 + 0.7^2 + 0.5^2 + 0.8^2} \right]$$

$$C_2 = \frac{7.38}{1.52}, \frac{10.48}{1.52}$$

$$C_2 = (4.8, 6.8)$$

Step- 3 : Find Distance

$$D_i = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Centroid 1:

$$(1,6)(2.4,6.1) = \sqrt{(1.4)^2 + (0.1)^2} = \sqrt{1.96 + 0.01} = \sqrt{1.97} = 1.40$$

$$(2,5)(2.4,6.1) = \sqrt{0.16 + 1.21} = \sqrt{1.37} = 1.17$$

$$(3,8)(2.4,6.1) = \sqrt{0.36 + 3.61} = \sqrt{3.97} = 1.99$$

$$(4,4)(2.4,6.1) = \sqrt{2.56 + 4.41} = \sqrt{6.97} = 2.64$$

$$(5,7)(2.4,6.1) = \sqrt{6.76 + 0.81} = \sqrt{7.57} = 2.75$$

$$(6,9)(2.4,6.1) = \sqrt{12.96 + 8.41} = \sqrt{21.37} = 4.62$$

Centroid 2:

$$(1,6)(4.8,6.8) = \sqrt{14.44 + 0.64} = \sqrt{15.08} = 3.88$$

$$(2,5)(4.8,6.8) = \sqrt{7.84 + 3.24} = \sqrt{11.08} = 3.32$$

$$(3,8)(4.8,6.8) = \sqrt{3.24 + 1.44} = \sqrt{4.68} = 2.16$$

$$(4,4)(4.8,6.8) = \sqrt{0.64 + 7.84} = \sqrt{8.48} = 2.91$$

$$(5,7)(4.8,6.8) = \sqrt{0.04 + 0.04} = \sqrt{0.08} = 0.28$$

$$(6,9)(4.8,6.8) = \sqrt{1.44 + 4.84} = \sqrt{6.28} = 2.50$$

Cluster 1		Cluster 2	
Datapoint	Distance	Datapoint	Distance
(1,6)	1.40	(1,6)	3.88
(2,5)	1.17	(2,5)	3.32
(3,8)	1.99	(3,8)	2.16
(4,4)	2.64	(4,4)	2.91
(5,7)	2.75	(5,7)	0.28
(6,9)	4.62	(6,9)	2.50

Step-4 : Update the membership value

$$\mu_j(x_i) = \frac{[\frac{1}{d_{ji}}]^{1/m-1}}{\sum_{k=1}^c [\frac{1}{d_{ki}}]^{1/m-1}}$$

here m = 2, i - first data point, j - first cluster

Cluster 1

$$\begin{aligned}\mu_{11} &= (1/d_{11})^{1/2-1} / (1/d_{11})^{1/2-1} + (1/d_{21})^{1/2-1} \\ &= (1/1.40)^1 / (1/1.40)^1 + (1/3.88)^1 = 0.71 / 0.71 + 0.25 \\ &= 0.71 / 0.96 = 0.7\end{aligned}$$

$$\begin{aligned}\mu_{12} &= \frac{1}{d_{12}} / \left(\frac{1}{d_{12}} + \frac{1}{d_{22}} \right) \\ &\Rightarrow \frac{1}{1.17} / \left(\frac{1}{1.17} + \frac{1}{3.32} \right) = 0.56 / 0.56 + 0.30 \\ &= 0.56 / 0.86 = 0.6\end{aligned}$$

$$\begin{aligned}\mu_{13} &= \frac{1}{d_{13}} / \left(\frac{1}{d_{13}} + \frac{1}{d_{23}} \right) \\ &\Rightarrow \frac{1}{1.99} / \left(\frac{1}{1.99} + \frac{1}{2.16} \right) = 0.50 / 0.50 + 0.46 \\ &= 0.50 / 0.96 = 0.5\end{aligned}$$

$$\begin{aligned}\mu_{14} &= \frac{1}{d_{14}} / \left(\frac{1}{d_{14}} + \frac{1}{d_{24}} \right) \\ &\Rightarrow \frac{1}{2.64} / \left(\frac{1}{2.64} + \frac{1}{2.91} \right) = 0.37 / 0.37 + 0.34 \\ &= 0.37 / 0.71 = 0.5\end{aligned}$$

$$\begin{aligned}\mu_{15} &= \frac{1}{d_{15}} / \left(\frac{1}{d_{15}} + \frac{1}{d_{25}} \right) \\ &\Rightarrow \frac{1}{2.75} / \left(\frac{1}{2.75} + \frac{1}{0.28} \right) = 0.36 / 0.36 + 3.57 \\ &= 0.36 / 3.93 = 0.1\end{aligned}$$

$$\begin{aligned}\mu_{16} &= \frac{1}{d_{16}} / \left(\frac{1}{d_{16}} + \frac{1}{d_{26}} \right) \\ &\Rightarrow \frac{1}{4.62} / \left(\frac{1}{4.62} + \frac{1}{2.50} \right) = 0.21 / 0.21 + 0.4 \\ &= 0.21 / 0.61 = 0.3\end{aligned}$$

Cluster 2

$$\mu_{21} = \frac{1/d_{21}}{(1/d_{12}) + (1/d_{21})}$$

$$\Rightarrow \frac{1/3.88}{1/1.40} + \frac{1/3.88}{1/3.88} = \frac{0.25}{0.71+0.25} \\ = 0.25 / 0.96 = 0.3$$

$$\mu_{22} = \frac{1/d_{22}}{(1/d_{12}) + (1/d_{22})}$$

$$\Rightarrow \frac{1/3.32}{1/1.17} + \frac{1/3.32}{1/3.32} = \frac{0.30}{0.56+0.30} \\ = 0.30 / 0.86 = 0.4$$

$$\mu_{23} = \frac{1/d_{23}}{(1/d_{13}) + (1/d_{23})}$$

$$\Rightarrow \frac{1/2.16}{1/1.99} + \frac{1/2.16}{1/2.16} = \frac{0.46}{0.50+0.46} \\ = 0.46 / 0.96 = 0.5$$

$$\mu_{24} = \frac{1}{d_{24}} / \left(\frac{1}{d_{14}} + \frac{1}{d_{24}} \right)$$

$$\Rightarrow \frac{1}{2.19} / \frac{1}{2.64} + \frac{1}{2.19} = 0.34 / 0.37 + 0.34 \\ = 0.34 / 0.71 = 0.5$$

$$\mu_{25} = \frac{1}{d_{25}} / \left(\frac{1}{d_{15}} + \frac{1}{d_{25}} \right)$$

$$\Rightarrow \frac{1}{0.28} / \frac{1}{2.75} + \frac{1}{0.28} = 3.57 / 0.36 + 3.57 \\ = 3.57 / 3.93 = 0.9$$

$$\mu_{26} = \frac{1}{d_{26}} / \left(\frac{1}{d_{16}} + \frac{1}{d_{26}} \right)$$

$$= 0.4 / 0.21 + 0.4 = 0.4 / 0.61 \\ = 0.7$$

Now the New Membership value is

X	Y	C1	C2
1	6	0.7	0.3
2	5	0.6	0.4
3	8	0.5	0.5
4	4	0.5	0.5
5	7	0.1	0.9
6	9	0.3	0.7

Step 5 : Now continue this process until get the same centroids.

Genetic Algorithm

Dr. Shruti Mishra
School of Computer Sc. & Engg. (SCOPE)

Optimization

Optimization is a **process** that finds a best, or optimal, solution for a problem. The Optimization problems are centered around three factors :

1. **An objective function** which is to be minimized or maximized;

Examples

- ‡ In manufacturing, we want to maximize the profit or minimize the cost .

2. **A set of unknowns or variables** that affect the objective function,

Examples

- ‡ In manufacturing, the variables are amount of resources used or the time spent.

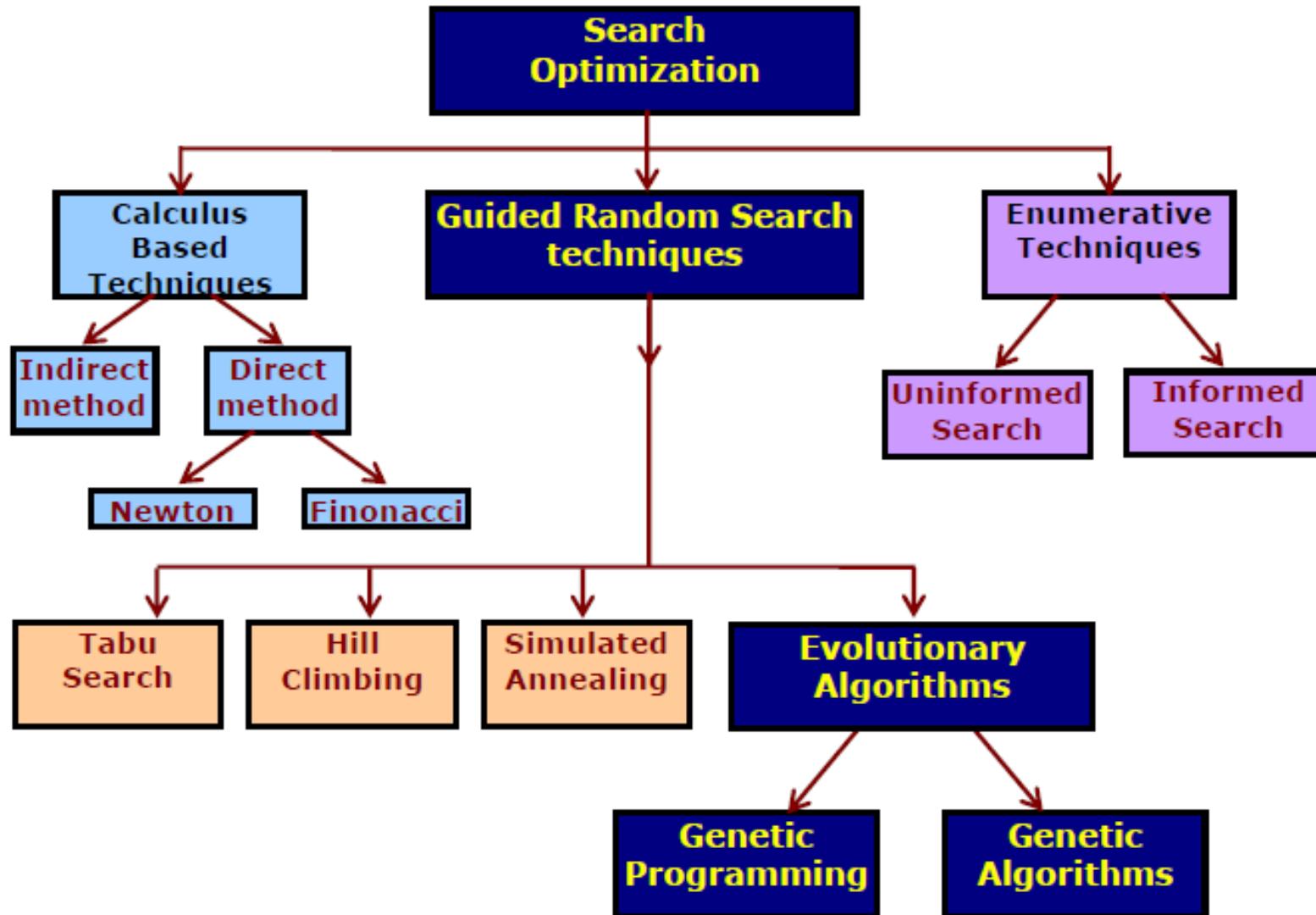
3. A set of constraints that allow the unknowns to take on certain values but exclude others;

Examples

- + In manufacturing, one constraint is, that all "time" variables to be non-negative.

An optimization problem is defined as : Finding values of the variables that minimize or maximize the objective function while satisfying the constraints.

Types of Search Optimization Techniques



We are interested in evolutionary search algorithms.

Our main concern is to understand the evolutionary algorithms :

- how to describe the process of search,
- how to implement and carry out search,
- what are the elements required to carry out search, and
- the different search strategies

The Evolutionary Algorithms include :

- Genetic Algorithms and
- Genetic Programming

Evolutionary Algorithm (EAs)

Evolutionary Algorithm (EA) is a subset of Evolutionary Computation (EC) which is a subfield of Artificial Intelligence (AI).

Evolutionary Computation (EC) is a general term for several computational techniques. Evolutionary Computation represents powerful search and optimization paradigm influenced by biological mechanisms of evolution : that of natural selection and genetic.

Evolutionary Algorithms (EAs) refers to Evolutionary Computational models using randomness and genetic inspired operations. EAs involve selection, recombination, random variation and competition of the individuals in a population of adequately represented potential solutions. The candidate solutions are referred as chromosomes or individuals.

Introduction to GAs

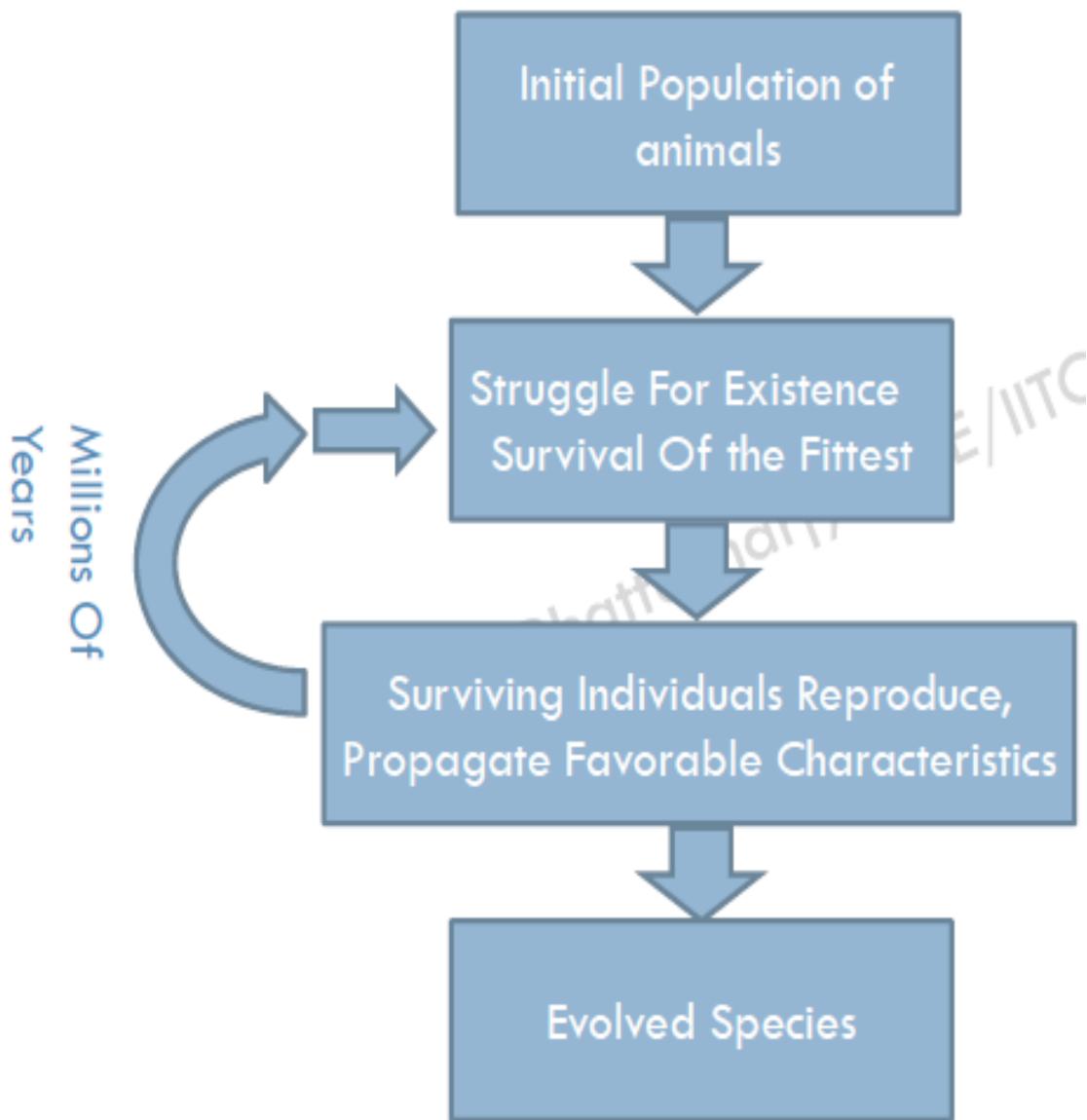
- The word “**genetics**” is derived from a Greek word “**genesis**” that indicate “to grow” or “**to become**”.
- Developed by J. Holland, Genetic Algorithms (GAs) are the heuristic search and optimization techniques that mimic the process of natural evolution.
- One of the useful tool for search and optimization problems.
- It seems to be randomized but by no means are random. They seeks historical information to direct search space into region to better performance.
- It is always “Survival of the fittest” from the wide search space.

“Select The Best, Discard The Rest”

Why GA?

- Better than conventional algorithms as GA is quite robust.
- Unlike traditional AI, they do not break with the change in the input or due to presence of noise.
- Searching in larger space, GAs provide significant benefits over more typical optimization techniques.

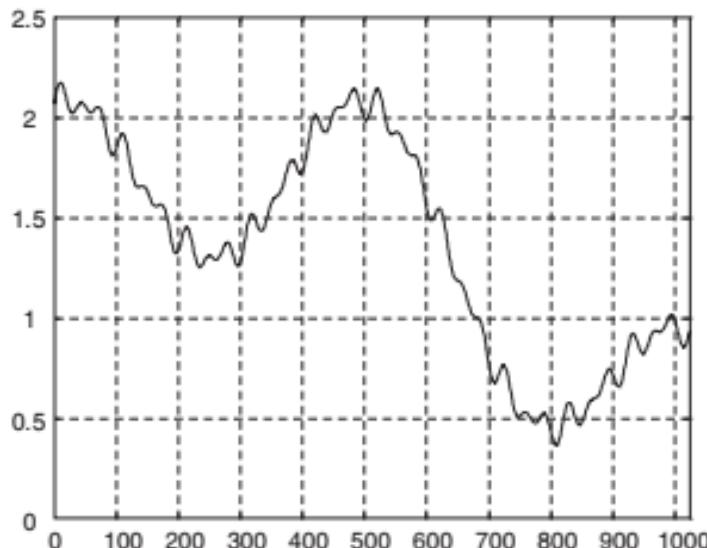
Evolution of species



Note: Thus genetic algorithms implement the optimization strategies by simulating evolution of species through natural selection.

Search Space

- The space of all feasible solutions (the set of solutions among which the desired solution resides) is called search space.
- Each and every point in the search space represents one possible solution.
- Each possible solution is marked by a *fitness value*.

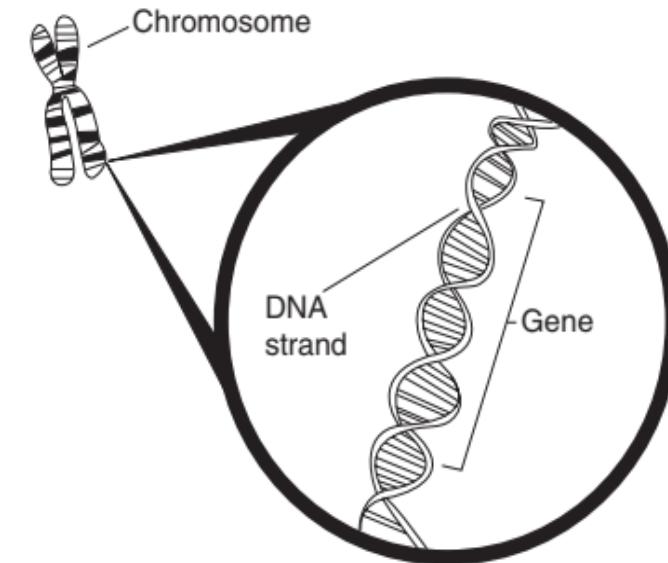


An example of search space.

Terminologies in GA

1) Chromosomes

- All genetic information gets stored in chromosomes. In human, it exists in pairs (23 pairs).
- Chromosomes are divided into several parts called *genes*.
- Gene code the properties of species i.e. the characteristics of an individual.
- The possibilities of combination of the genes for one property are called *alleles*, and a gene can take different alleles.
- The set of all possible alleles present in a particular pool forms a *gene pool*.
- The set of all the genes of a specific species is called *genome*.



2) Genetics

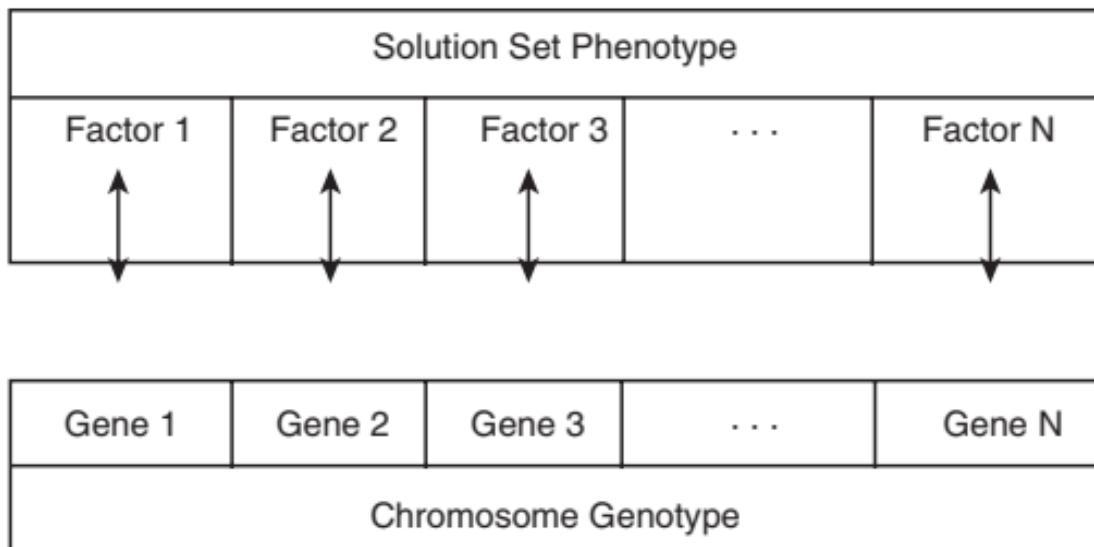
- For a particular individual, the entire combination of genes is called **genotypes**.
- The phenotypes describes the physical aspect of decoding a genotype to produce the **phenotype**.

Comparison of Natural Evolution and GA terminology

Natural evolution	Genetic algorithm
Chromosome	String
Gene	Feature or character
Allele	Feature value
Locus	String position
Genotype	Structure or coded string
Phenotype	Parameter set, a decoded structure

3) Individuals

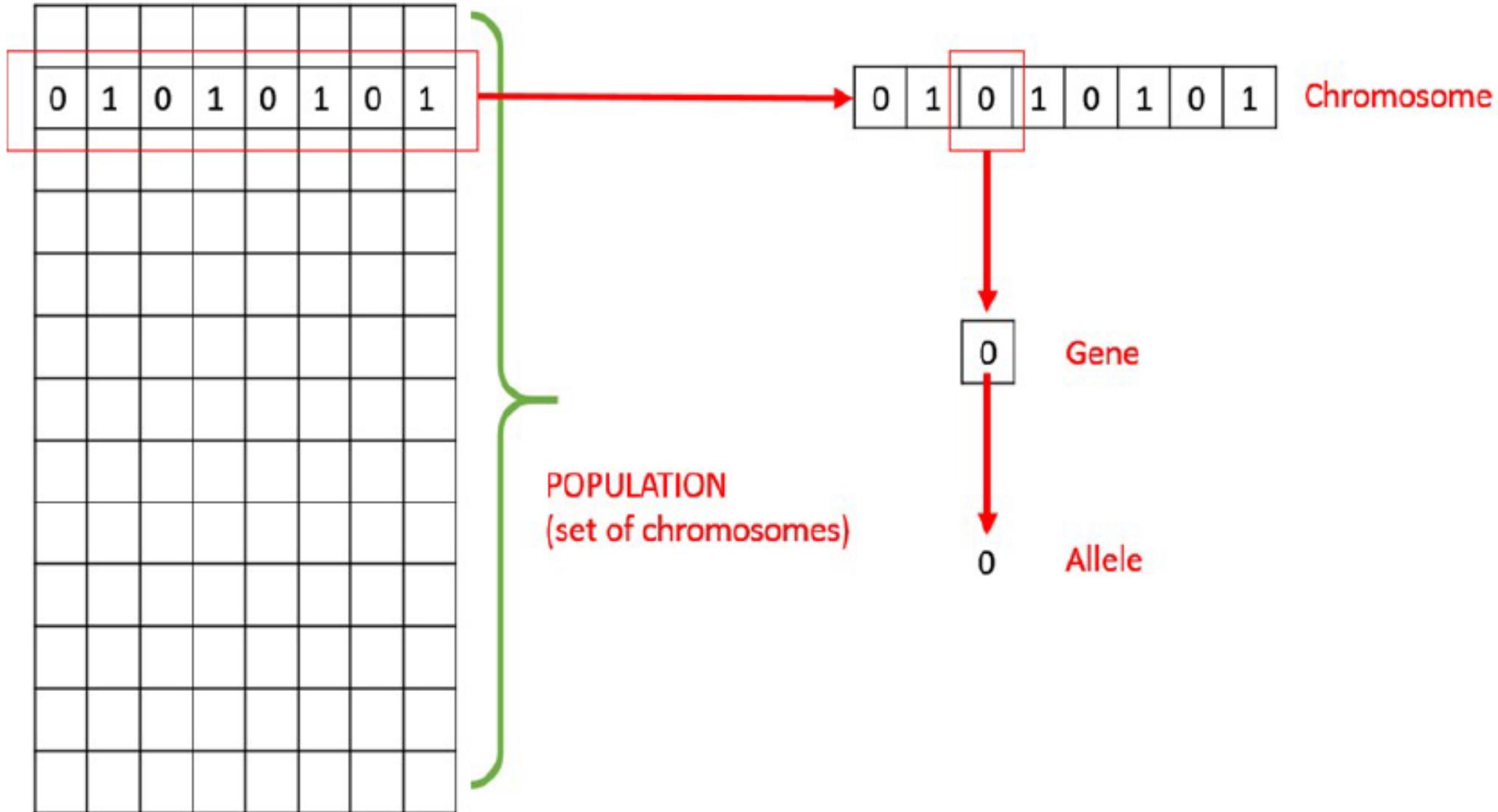
- It's usually a single solution.
- An individual groups together two forms of solutions as given below:
 - The chromosomes which is a raw genetic information (genotype) that GA deals.
 - The phenotype which is expressive of the chromosomes in the terms of the models.
- A chromosome is encoded by a bit string.



Representation of genotype and phenotype.

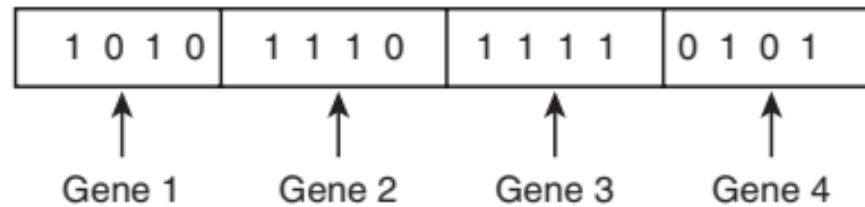


Representation of a chromosome.



4) Gene

- A chromosome is a sequence of “genes”.
- A gene is a bit string of arbitrary lengths.
- Bit string is a binary representation of number of intervals from a lower bound.
- A bit string of length “n” can represent $(2^n - 1)$ intervals.
- The size of the interval would be $(\text{range})/(2^n - 1)$.



Representation of a gene.

5) Fitness

- The fitness of an individual in a GA is the value of an objective function for its phenotype.
- For calculating fitness, the chromosome has to be first decoded and the objective function has to be evaluated.
- It is an indication towards how good the solution is and how close the chromosome is to the optimal one.

6) Populations

- It is a collection of individuals.
- Two aspects of populations:
 - The initial population generation.
 - The population size- It usually depends on the complexity of the problem.
Random initialization of population. Each bit is initialized to 0 or 1.

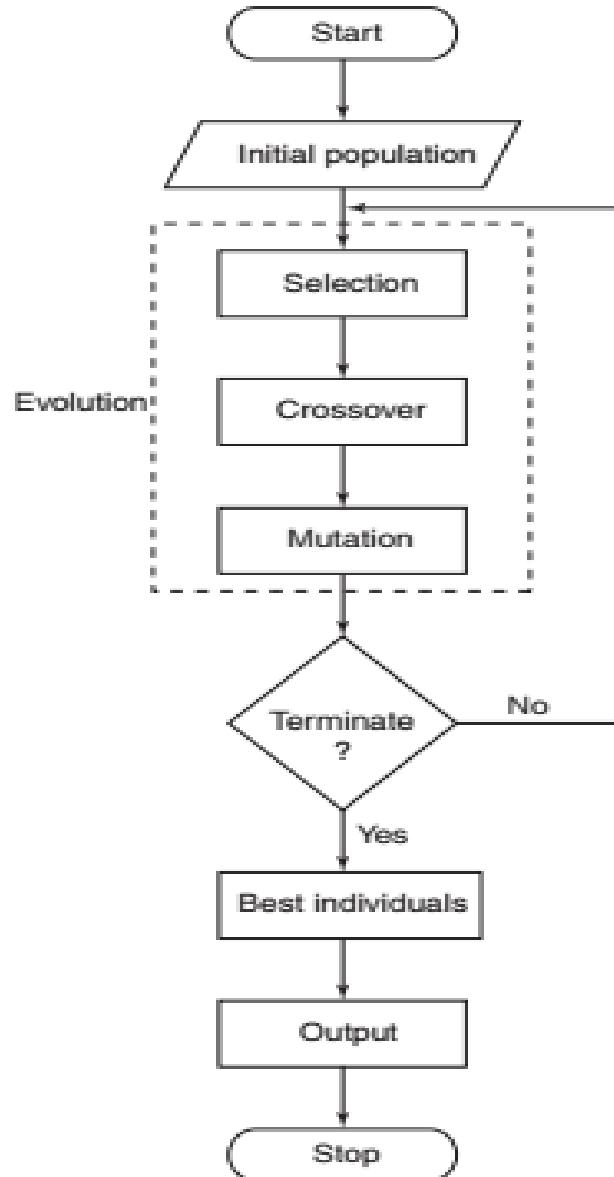
Population	Chromosome 1	1 1 1 0 0 0 1 0
	Chromosome 2	0 1 1 1 1 0 1 1
	Chromosome 3	1 0 1 0 1 0 1 0
	Chromosome 4	1 1 0 0 1 1 0 0

Population.

GA Operators and Parameters

- Selection
- Crossover
- Mutation

Simple Genetic Algorithms



The simple form of GA is given by the following.

1. Start with a randomly generated population.
2. Calculate the fitness of each chromosome in the population.
3. Repeat the following steps until n offsprings have been created:
 - Select a pair of parent chromosomes from the current population.
 - With probability p_c crossover the pair at a randomly chosen point to form two offsprings.
 - Mutate the two offsprings at each locus with probability p_m .
4. Replace the current population with the new population.
5. Go to step 2.

1. *Selection:* The first step consists in selecting individuals for reproduction. This selection is done randomly with a probability depending on the relative fitness of the individuals so that best ones are often chosen for reproduction rather than the poor ones.
2. *Reproduction:* In the second step, offspring are bred by selected individuals. For generating new chromosomes, the algorithm can use both recombination and mutation.
3. *Evaluation:* Then the fitness of the new chromosomes is evaluated.
4. *Replacement:* During the last step, individuals from the old population are killed and replaced by the new ones.

The algorithm is stopped when the population converges toward the optimal solution.

Operators in Genetic Algorithm

1) **Encoding:** Process of representing individual genes. The process can be performed using bits, numbers, trees, arrays, lists or any other objects.

a) **Binary Encoding:** Use a binary string. Each bit represents a characteristics.

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

Binary encoding.

Binary coded strings with 1s and 0s are mostly used. The length of the string depends on the accuracy. In such coding

1. Integers are represented exactly.
2. Finite number of real numbers can be represented.
3. Number of real numbers represented increases with string length.

b) Octal Encoding

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

Chromosome 1	03467216
Chromosome 2	15723314

Octal encoding.

c) Hexadecimal Encoding

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

Chromosome 1	9CE7
Chromosome 2	3DBA

Hexadecimal encoding.

d) Permutation Encoding (Real Number Encoding)

- Every chromosome is a string of integers/real values, which represents number in sequence.
- Used for ordering problems.

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

Permutation encoding.

e) Value Encoding

- Used where some complicated values such as real numbers are used.

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

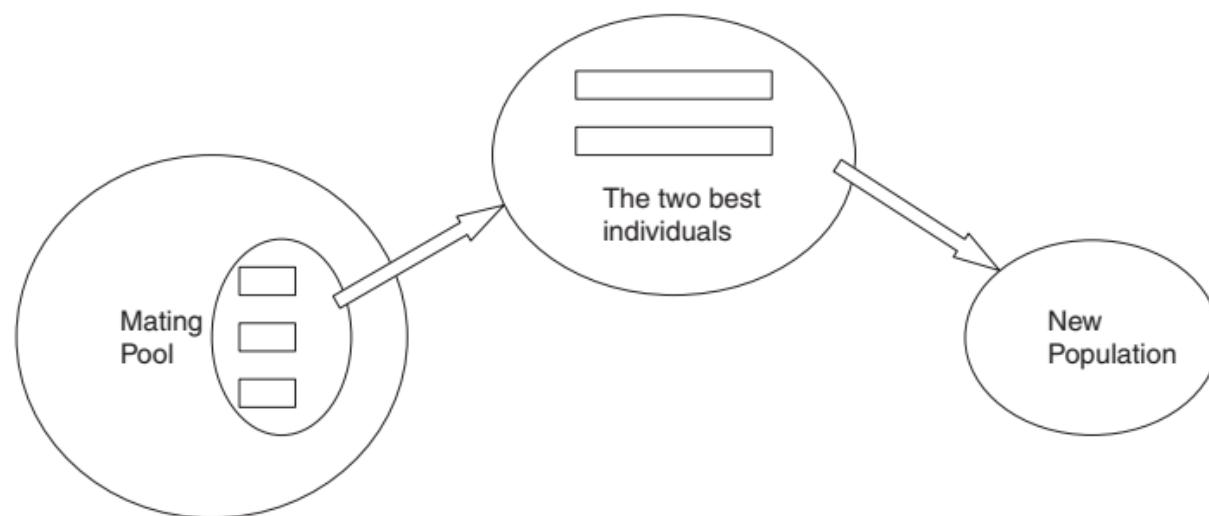
Value encoding.

f) Tree Encoding

- Used for evolving program expressions for genetic programming.

2) Selection:

- It is the process of choosing two parents from the population for crossing to create offspring for the next generation.
- Chromosomes are selected from the initial population to be parents for offspring generation.
- It randomly pick chromosomes out of the population with higher fitness function.

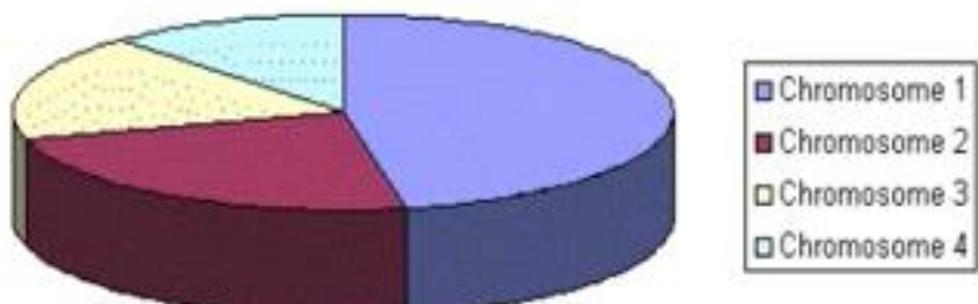


- Types of selection

- Roulette Wheel Selection
- Random Selection
- Rank ordering Selection
- Tournament Selection
- Boltzmann Selection
- Stochastic Universal Sampling

a) Roulette Wheel Selection

- Parents are selected according to their fitness
- The better the chromosomes are, the more chances to be selected they have
- Imagine a roulette wheel where all chromosomes in the population are placed and the area each chromosome has corresponds to its fitness function, like on the following picture
- Then a marble is thrown there and it selects the chromosome
- Chromosomes with **bigger fitness will be selected more times**



- Probability for an individual being selected is considered to be proportional to its fitness.

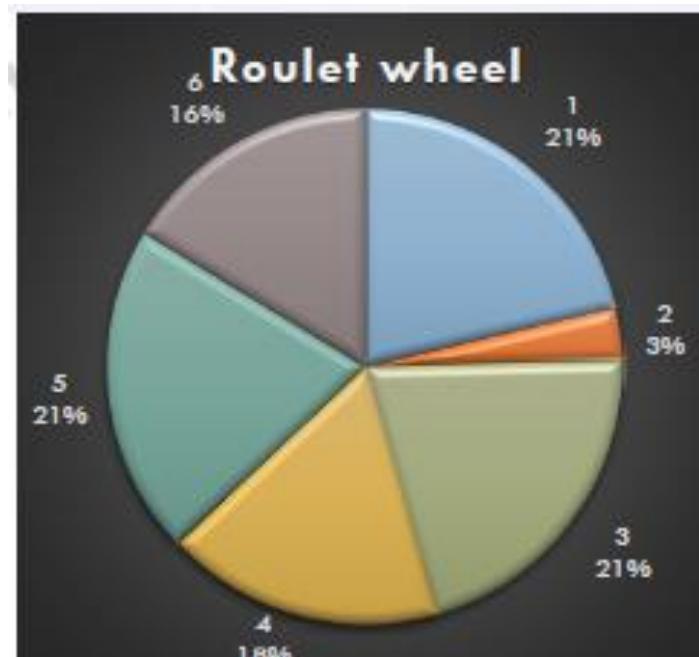
The probability that i-th individual will be pointed is

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

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

- Parents are selected according to their fitness values
- The better chromosomes have more chances to be selected

Chromosome	Fitness	Probability	Chance (N*Pi)	Actual count
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	6



b) 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.

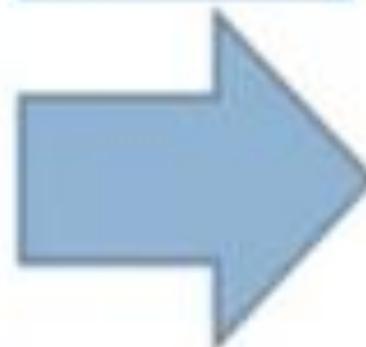
The % area to be occupied by a particular individual i , is given by

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

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

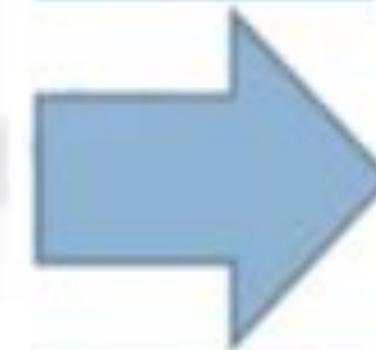
Chrom #	Fitness
1	37
2	6
3	36
4	30
5	36
6	28

Sort according to fitness



Chrom #	Fitness
1	37
3	36
5	36
4	30
6	28
2	6

Assign ranking



Chrom #	Rank
1	6
3	5
5	4
4	3
6	2
2	1

Chrom #	% of RW
1	$6/21=29$
3	$5/21=24$
5	$4/21=19$
4	$3/21=14$
6	$2/21=10$
2	$1/21=5$

Roulette wheel



Chrom #	EC	AC
1	$.29 * 6 = 1.714$	2
3	1.429	1
5	1.143	1
4	0.857	1
6	0.571	1
2	0.286	0

■ 6 ■ 5 ■ 4 ■ 3 ■ 2 ■ 1

c) 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

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

Input:

Individual
Fitness

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

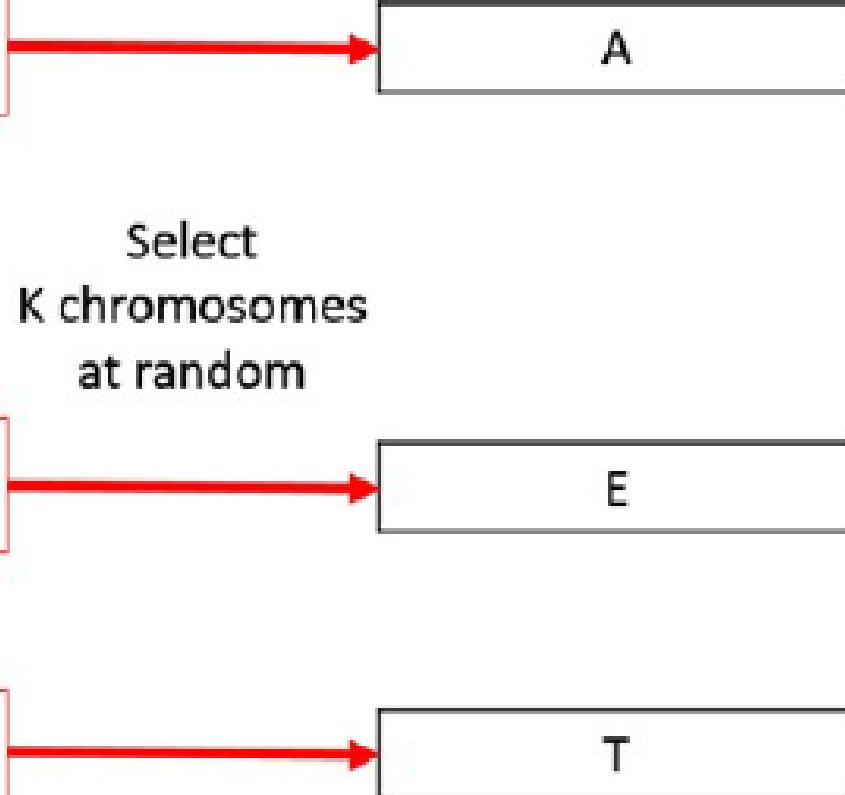
Output:

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

Fitness
Value

	Chromosome
1	Q
5	A
9	Z
8	W
7	S
4	X
2	E
3	F
6	R
2	T
2	Y
1	U
0	I

Select
K chromosomes
at random



Pick the best
as parent

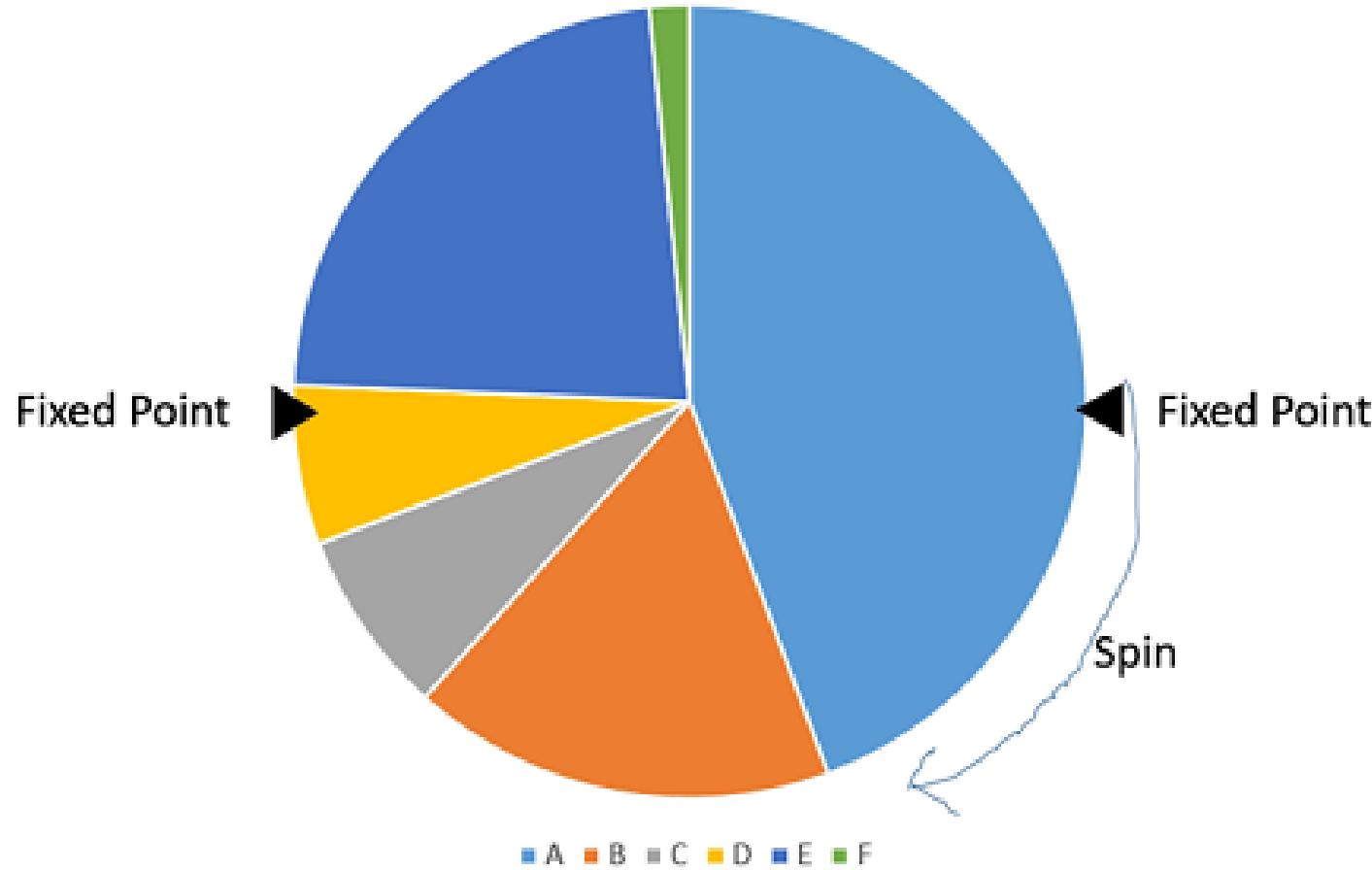


d) Random Selection

- In this strategy we randomly select parents from the existing population.
- There is no selection pressure towards fitter individuals and therefore this strategy is usually avoided.

e) Stochastic Universal Sampling (SUS)

- Stochastic Universal Sampling is quite similar to Roulette wheel selection, however instead of having just one fixed point, we have multiple fixed points as shown in the following image.
- Therefore, all the parents are chosen in just one spin of the wheel.
- Also, such a setup encourages the highly fit individuals to be chosen at least once.



It is to be noted that fitness proportionate selection methods don't work for cases where the fitness can take a negative value.

f) Boltzmann Selection

- In Boltzmann selection, a continuously varying temperature controls the rate of selection according to a preset schedule.
- The temperature starts out 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

3) Crossover (Reproduction)

- In this more than one parent is selected and one or more off-springs are produced using the genetic material of the parents.
- Crossover is usually applied in a GA with a high probability – p_c

Crossover is a recombination operator that proceeds in three steps:

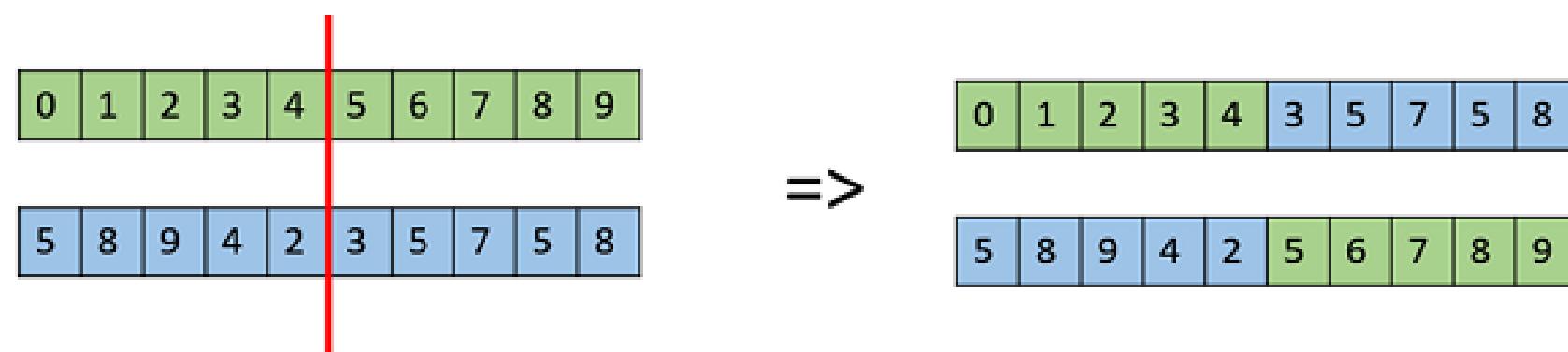
1. The reproduction operator selects at random a pair of two individual strings for the mating.
2. A cross site is selected at random along the string length.
3. Finally, the position values are swapped between the two strings following the cross site.

Types of Crossover Operators

- Single-Point Crossover
- Two-Point Crossover
- Multi-Point (N-Point) Crossover
- Uniform Crossover
- Three-Parent Crossover
- Crossover with Reduced Surrogate
- Shuffle Crossover
- Precedence Preservative Crossover
- Ordered Crossover
- Partially Matched Crossover

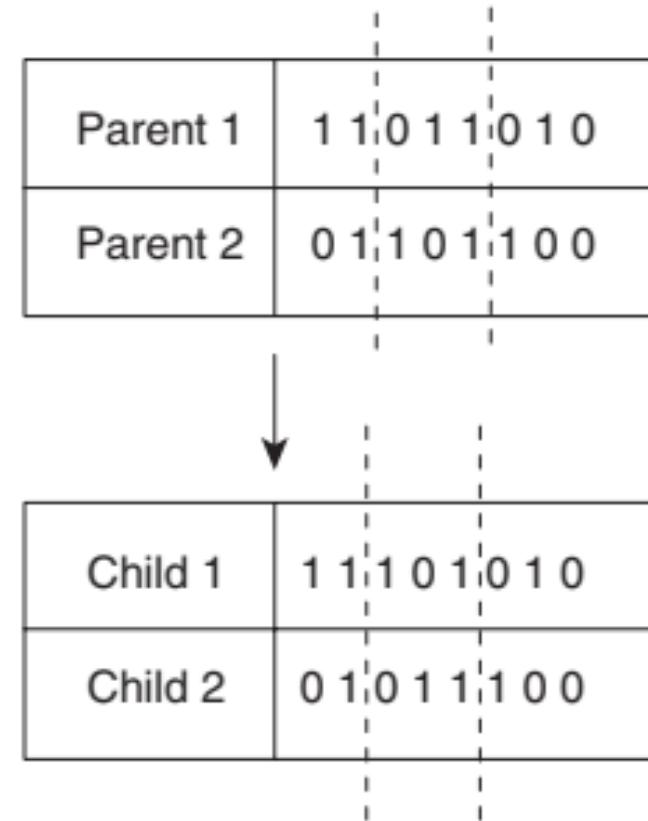
Single Point Crossover

- In this one-point crossover, a random crossover point is selected and the tails of its two parents are swapped to get new off-springs.
- Cross site is selected randomly.



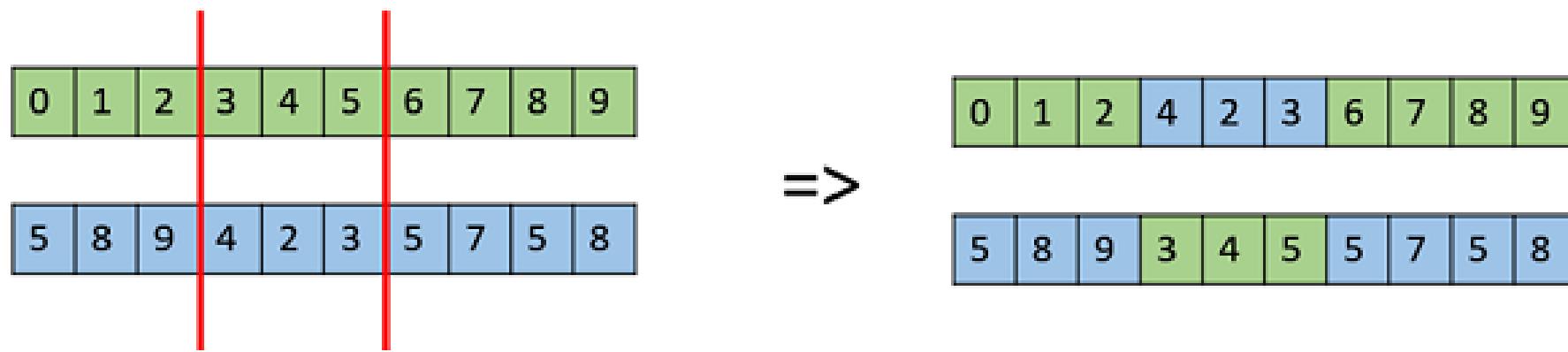
Two Point Crossover

- Two crossover points are picked randomly from the parent chromosomes. The bits in between the two points are swapped between the parent organisms.



Multi- Point (N-Point) Crossover

- Multi point crossover is a generalization of the one-point crossover wherein alternating segments are swapped to get new off-springs.



Uniform Crossover

Uniform crossover is quite different from the N-point 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. Offspring, therefore, contain a mixture of genes from each parent. The number of effective crossing point is not fixed, but will average $L/2$ (where L is the chromosome length).

In Figure 21-24, new children are produced using uniform crossover approach. It can be noticed that 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.

Parent 1	1 0 1 1 0 0 1 1
Parent 2	0 0 0 1 1 0 1 0
Mask	1 1 0 1 0 1 1 0
Child 1	1 0 0 1 1 0 1 0
Child 2	0 0 1 1 0 0 1 1

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. This concept is illustrated in Figure 21-25.

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

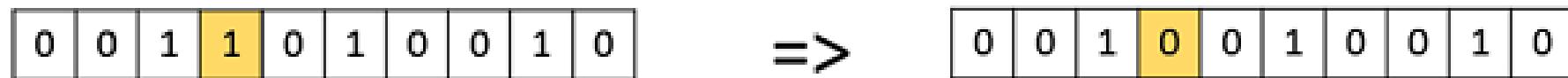
4) Mutation

- In simple terms, mutation may be defined as a small random tweak in the chromosome, to get a new solution.
- It is used to maintain and introduce diversity in the genetic population and is usually applied with a low probability – p_m .
- If the probability is very high, the GA gets reduced to a random search.
- Mutation is the part of the GA which is related to the “exploration” of the search space.
- It has been observed that mutation is essential to the convergence of the GA while crossover is not.

Types of Mutation Operators

- Flipping
- Interchanging
- Reversing

- **Flipping:** In this bit flip mutation, we select one or more random bits and flip them. This is used for binary encoded GAs.

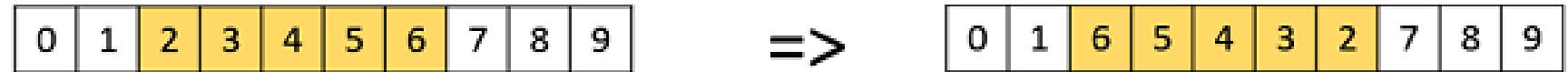


- **Interchanging (Swap Mutation):** We select two positions on the chromosome at random, and interchange the values. This is common in permutation based encodings.



Reversing:

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



Stopping Condition for GA

The various stopping conditions to terminate the flow of genetic algorithm are listed as follows:

1. *Maximum generations*: The GA stops when the specified number of generations has evolved.
2. *Elapsed time*: The genetic process will end when a specified time has elapsed.

Note: If the maximum number of generation has been reached before the specified time has elapsed, the process will end.

3. *No change in fitness*: The genetic process will end if there is no change to the population's best fitness for a specified number of generations.

Note: If the maximum number of generation has been reached before the specified number of generation with no changes has been reached, the process will end.

4. *Stall generations*: The algorithm stops if there is no improvement in the objective function for a sequence of consecutive generations of length "Stall generations".
5. *Stall time limit*: The algorithm stops if there is no improvement in the objective function during an interval of time in seconds equal to "Stall time limit".

The termination or convergence criterion finally brings the search process to a halt. The following are specific termination techniques used for real-time problems.

Example

Step 1: Choose a encoding technique

Maximize the function $f(x)=x^2$, where x value range from 0-31



Step 1: For using GA approach, one must first code the decision variable "x" into a finite length string. Using a five bit (binary integer) unsigned integer, numbers between 0(00000) and 31(11111) can be obtained.

The objective function here is $f(x)=x^2$ which is to be maximized. A single generation of a GA is performed here with encoding, selection, crossover and mutation. To start with, select initial population at random. Here initial population of size 4 is chosen, but any number of populations can be selected based on the requirement and application. Table 21-4 shows an initial population randomly selected.

Step 2: Obtain the decoded x values for the initial population generated. Consider string 1,

$$\begin{aligned}01100 &= 0 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0 \\&= 0 + 8 + 4 + 0 + 0 \\&= 12\end{aligned}$$

Thus for all the four strings the decoded values are obtained.

TABLE 21-4 SELECTION

String no.	Initial population (randomly selected)	x value	Fitness $f(x) = x^2$	Prob. _i	Percentage probability (%)	Expected count	Actual count
1	0 1 1 0 0	12	144	0.1247	12.47	0.4987	1
2	1 1 0 0 1	25	625	0.5411	54.11	2.1645	2
3	0 0 1 0 1	5	25	0.0216	2.16	0.0866	0
4	1 0 0 1 1	19	361	0.3126	31.26	1.2502	1
Sum		1155	1.0000	100	4.0000	4	
Average		288.75	0.2500	25	1.0000	1	
Maximum		625	0.5411	54.11	2.1645	2	

Step 3: Calculate the fitness or objective function. This is obtained by simply squaring the "x" value, since the given function is $f(x) = x^2$. When $x = 12$, the fitness value is

$$f(x) = x^2 = (12)^2 = 144$$

$$\text{For } x = 25, \quad f(x) = x^2 = (25)^2 = 625$$

and so on, until the entire population is computed.

Step 4: Compute the probability of selection,

$$\text{Prob}_i = \frac{f(x)_i}{\sum_{i=1}^n f(x)_i} \quad (21.15)$$

where n is the number of populations; $f(x)$ is the fitness value corresponding to a particular individual in the population;

$\Sigma f(x)$ is the summation of all the fitness value of the entire population.

Considering string 1,

$$\text{Fitness } f(x) = 144$$

$$\sum f(x) = 1155$$

The probability that string 1 occurs is given by

$$P_1 = 144 / 1155 = 0.1247$$

The percentage probability is obtained as

$$0.1247 * 100 = 12.47\%$$

The same operation is done for all the strings. It should be noted that summation of probability select is 1.

Step 5: The next step is to calculate the expected count, which is calculated as

$$\text{Expected count} = \frac{f(x)_i}{[\text{Avg } f(x)]_i} \quad (21.16)$$

where

$$(\text{Avg } f(x))_i = \left[\frac{\sum_{i=1}^n f(x)_i}{n} \right]$$

For string 1,

$$\text{Expected count} = \text{Fitness/Average} = 144 / 288.75 = 0.4987$$

We then compute the expected count for the entire population. The expected count gives an idea of which population can be selected for further processing in the mating pool.

Step 6: Now the actual count is to be obtained to select the individuals who would participate in the crossover cycle using Roulette wheel selection. The Roulette wheel is formed as shown in Figure 21-33.

The entire Roulette wheel covers 100% and the probabilities of selection as calculated in step 4 for the entire populations are used as indicators to fit into the Roulette wheel. Now the wheel may be spun and the number of occurrences of population is noted to get actual count.

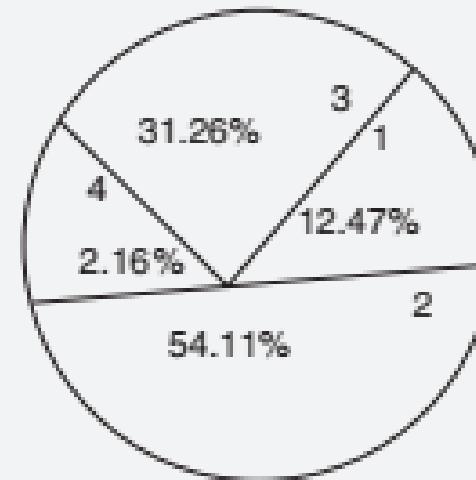


Figure 21-33 Selection using Roulette wheel.

- String 1 occupies 12.47%, so there is a chance for it to occur at least once. Hence its actual count may be 1.
- With string 2 occupying 54.11% of the Roulette wheel, it has a fair chance of being selected twice. Thus its actual count can be considered as 2.
- On the other hand, string 3 has the least probability percentage of 2.16%, so their occurrence for next cycle is very poor. As a result, its actual count is 0.
- String 4 with 31.26% has at least one chance for occurring while Roulette wheel is spun, thus its actual count is 1.

The above values of actual count are tabulated as shown in Table 21-5.

Step 7: Now, write the mating pool based upon the actual count as shown in Table 21-5.

TABLE 21-5 CROSSOVER

String no.	Mating Pool	Crossover point	Offspring after crossover	x value	Fitness value $f(x) = x^2$
1	0 1 1 0 0	4	0 1 1 0 1	13	169
2	1 1 0 0 1	4	1 1 0 0 0	24	576
3	1 1 0 0 1	2	1 1 0 1 1	27	729
4	1 0 0 1 1	2	1 0 0 0 1	17	289
Sum					1763
Average					440.75
Maximum					729

The actual count of string no. 1 is 1, hence it occurs once in the mating pool. The actual count of string no. 2 is 2, hence it occurs twice in the mating pool. Since the actual count of string no. 3 is 0, it does not occur in the mating pool. Similarly, the actual count of string no. 4 being 1, it occurs once in the mating pool. Based on this, the mating pool is formed.

Step 8: Crossover operation is performed to produce new offspring (children). The crossover point is specified and based on the crossover point, single-point crossover is performed and new offspring is produced. The parents are

Parent 1	0 1 1 0 0
Parent 2	1 1 0 0 1

The offspring is produced as

Offspring 1	0 1 1 0 1
Offspring 2	1 1 0 0 0

In a similar manner, crossover is performed for the next strings.

Step 9: After crossover operations, new offspring are produced and “x” values are decoded and fitness is calculated.

Step 10: In this step, mutation operation is performed to produce new offspring after crossover operation. As discussed in Section 21.9.4.1 mutation-flipping operation is performed and new offspring are produced. Table 21-6 shows the new offspring after mutation. Once the offspring are obtained after mutation, they are decoded to x value and the fitness values are computed.

TABLE 21-6 MUTATION

String no.	Offspring after crossover	Mutation chromosomes for flipping	Offspring after mutation	x value	Fitness $f(x) = x^2$
1	0 1 1 0 1	1 0 0 0 0	1 1 1 0 1	29	841
2	1 1 0 0 0	0 0 0 0 0	1 1 0 0 0	24	576
3	1 1 0 1 1	0 0 0 0 0	1 1 0 1 1	27	729
4	1 0 0 0 1	0 0 1 0 0	1 0 1 0 0	20	400
Sum					2546
Average					636.5
Maximum					841

Note: This completes one generation.

Swarm Intelligence: PSO & ACO

Dr. Shruti Mishra
School of Computer Sc. & Engg. (SCOPE)

Swarm Intelligence

- Origins in Artificial Life (Alife) Research
 1. ALife studies how computational techniques can help when studying biological phenomena
 2. ALife studies how biological techniques can help out with computational problems
- Two main Swarm Intelligence based methods
 - Particle Swarm Optimization (PSO)
 - Ant Colony Optimization (ACO)

Swarm Intelligence

- Swarm Intelligence (SI) is the property of a system whereby
 - the collective behaviors of (unsophisticated) agents interacting locally with their environment cause coherent functional global patterns to emerge.
- SI provides a basis with which it is possible to explore collective (or distributed) problem solving without centralized control or the provision of a global model.
- Leverage the power of complex adaptive systems to solve difficult non-linear stochastic problems

Swarm Intelligence

- Characteristics of a swarm:
 - Distributed, no central control or data source;
 - Limited communication
 - No (explicit) model of the environment;
 - Perception of environment (sensing)
 - Ability to react to environment changes.

Particle Swarm Optimization



Particle Swarm Optimization (PSO)

- PSO is stochastic optimization technique proposed by Kennedy and Eberhart (1995) [2].
- A population based search method with position of particle is representing solution and Swarm of particles as searching agent.
- PSO is a robust evolutionary optimization technique based on the movement and intelligence of swarms.
- PSO find the minimum value for the function.

Particle Swarm Optimization (PSO)

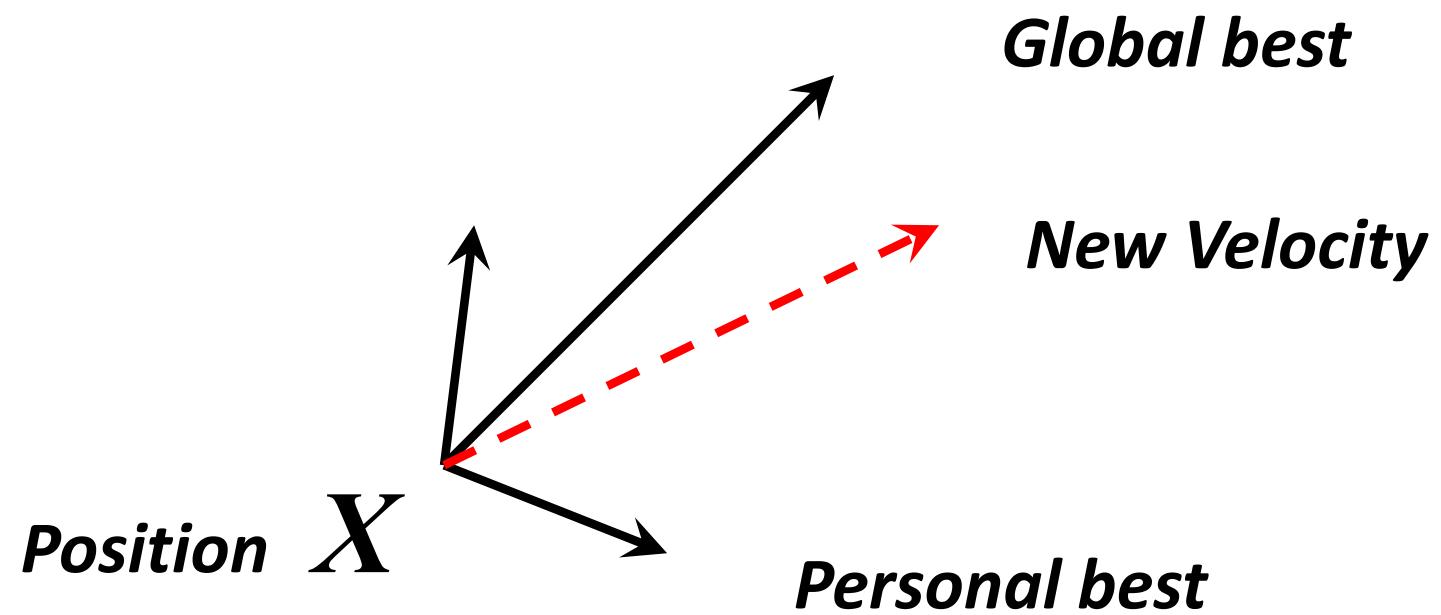
- The idea is similar to bird flocks searching for food.
 - Bird = a particle, Food = a solution
 - $pbest$ = the best solution (fitness) a particle has achieved so far.
 - $gbest$ = the global best solution of all particles within the swarm

PSO Search Scheme

- **pbest** : the best solution achieved so far by **that particle**.
- **gbest** : the best value obtained so far by **any particle** in the neighborhood of that particle.
- The basic concept of PSO lies in accelerating each particle toward its **pbest** and the **gbest** locations, with a **random weighted acceleration** at each time.

PSO Search Scheme

- PSO uses a number of agents, i.e., particles, that constitute a swarm **flying** in the search space **looking for the best solution**.
- Each particle is treated as a point (candidate solution) in a **N-dimensional** space which adjusts its “**flying**” according to its **own flying experience** as well as the **flying experience** of other particles.



Particle Swarm Optimization (PSO)

Each particle tries to modify its position X using the following formula:

$$X(t+1) = X(t) + V(t+1) \quad (1)$$

$$V(t+1) = wV(t) + c_1 \times rand() \times (X_{pbest} - X(t)) + c_2 \times rand() \times (X_{gbest} - X(t)) \quad (2)$$

$V(t)$	velocity of the particle at time t
$X(t)$	Particle position at time t
w	Inertia weight
c_1, c_2	learning factor or accelerating factor
rand	uniformly distributed random number between 0 and 1
X_{pbest}	particle's best position
X_{gbest}	global best position

PSO Algorithm

The PSO algorithm pseudocode [2] as following:

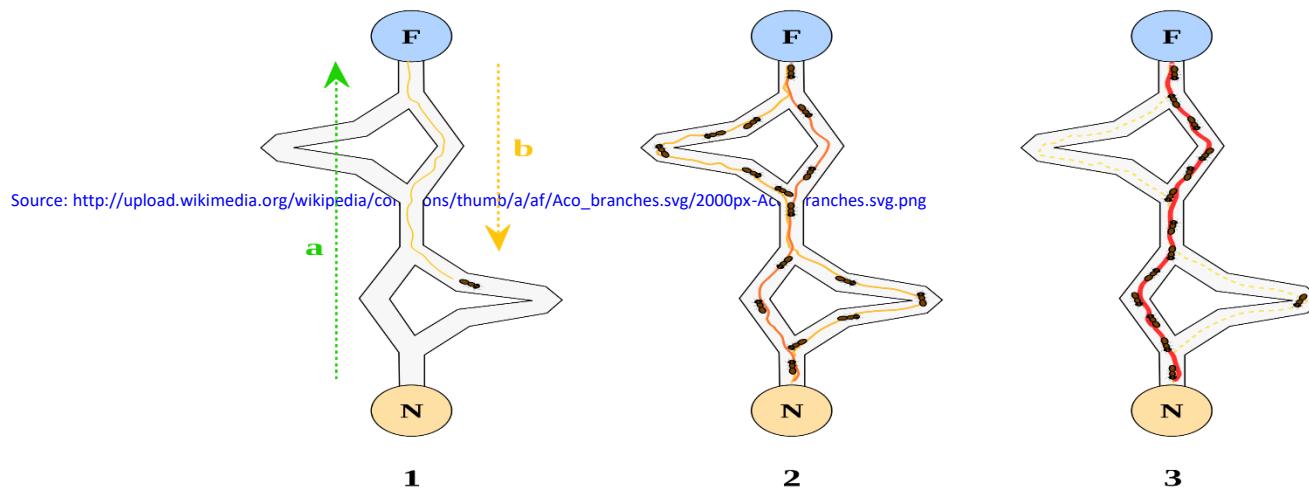
Input: Randomly initialized position and velocity of Particles:
 $X_i(0)$ and $V_i(0)$

Output: Position of the approximate global minimum X^*

```
1: while terminating condition is not reached do
2:   for  $i = 1$  to number of particles do
3:     Calculate the fitness function  $f$ 
4:     Update personal best and global best of each particle
5:     Update velocity of the particle using Equation 2
6:     Update the position of the particle using equation 1
7:   end for
8: end while
```

Ant Colony Optimization

“Ant Colony Optimization (ACO) studies artificial systems that take inspiration from the *behavior of real ant colonies* and which are used to solve discrete optimization problems.” ACO Website [1]



Ant Colony Optimization

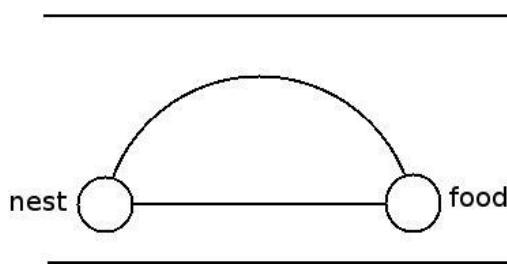
- Probabilistic Techniques to solve optimization Problem
- It is a population based metaheuristic used to find approximate solution to an optimization problem.
- The Optimization Problem must be written in the form of path finding with a weighted graph

Application of ACO

- Shortest paths and routing
- Assignment problem
- Set Problem

Idea

- The way ants find their food in shortest path is interesting.
- Ants hide pheromones to remember their path.
- These pheromones evaporate with time.
- Whenever an ant finds food , it marks its return journey with pheromones.
- Pheromones evaporate faster on longer paths.



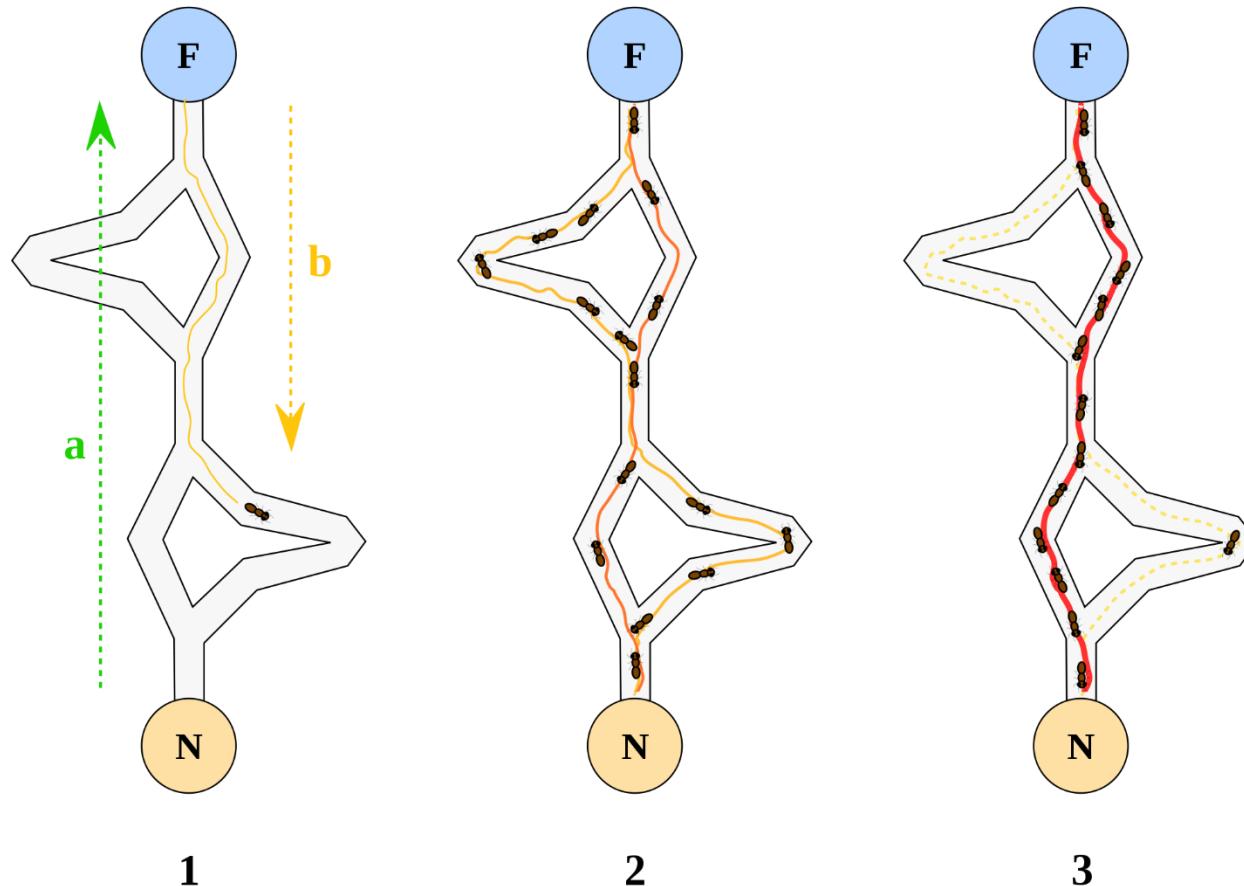
Idea (cont.)

- Shorter paths serve as the way to food for most of the other ants.
- The shorter path will be reinforced by the pheromones further.
- Finally , the ants arrive at the shortest path.

ACO Concept

- Ants navigate from nest to food source. Ants are blind!
- Shortest path is discovered via pheromone trails. Each ant moves at random
- Pheromone is deposited on path
- More pheromone on path increases probability of path being followed

Ant Colony Optimization



Source: http://upload.wikimedia.org/wikipedia/commons/thumb/a/af/Aco_branches.svg/2000px-Aco_branches.svg.png

Ant Colony Algorithm

Algorithm 1 The Ant Colony Optimization Metaheuristic

Set parameters, initialize pheromone trails

while termination condition not met **do**

ConstructAntSolutions

ApplyLocalSearch (optional)

UpdatePheromones

end while

- *ConstructAntSolutions*: Partial solution extended by adding an edge based on stochastic and pheromone considerations.
- *ApplyLocalSearch*: problem-specific, used in state-of-art ACO algorithms.
- *UpdatePheromones*: increase pheromone of good solutions, decrease that of bad solutions (pheromone evaporation).