UNIT – I – FUNDAMENTALS OF ARTIFICIAL NEURAL NETWORKS

UNIT I FUNDAMENTALS OF ANN

Fundamentals of ANN – Biological Neurons and Their Artificial Models – Types of ANN – Properties – Different Learning Rules – Types of Activation Functions – Training of ANN – Perceptron Model (Both Single &Multi-Layer) – Training Algorithm – Problems Solving Using Learning Rules and Algorithms – Linear Separability Limitation and Its Over Comings

1. FUNDAMENTALS OF ANN

Neural computing is an information processing paradigm, inspired by biological system, composed of a large number of highly interconnected processing elements(neurons) working in unison to solve specific problems.

Artificial neural networks (ANNs), like people, learn by example. An ANN is configured for a specific application, such as pattern recognition or data classification, through a learning process. Learning in biological systems involves adjustments to the synaptic connections that exist between the neurons. This is true of ANNs as well.

1.1 THE BIOLOGICAL NEURON

The human brain consists of a large number, more than a billion of neural cells that process information. Each cell works like a simple processor. The massive interaction between all cells and their parallel processing only makes the brain's abilities possible. Figure 1 represents a human biological nervous unit. Various parts of biological neural network(BNN) is marked in Figure 1.

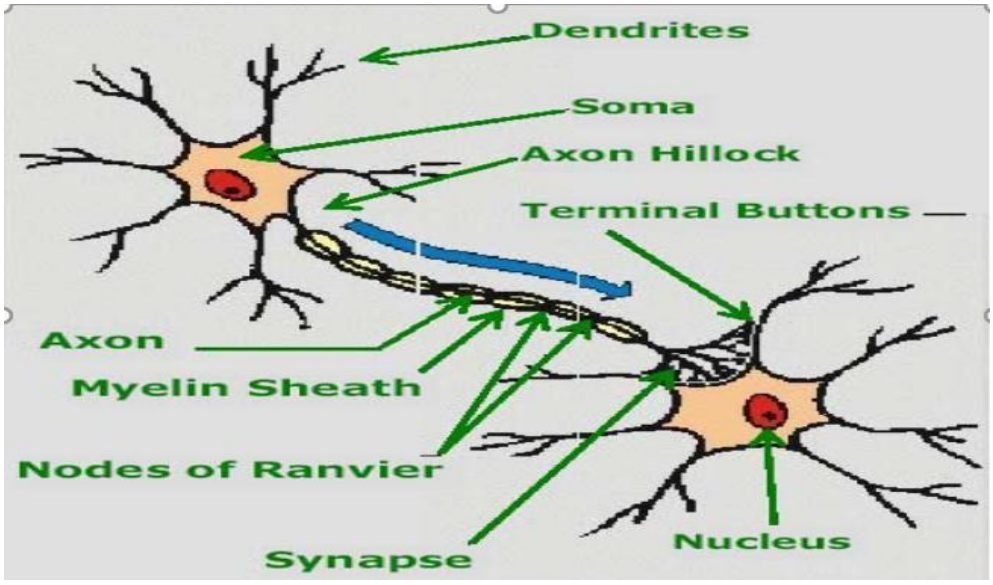


Figure 1: Biological Neural Network

Dendrites are branching fibres that extend from the cell body or soma.

Soma or cell body of a neuron contains the nucleus and other structures, support chemical processing and production of neurotransmitters.

Axon is a singular fiber carries information away from the soma to the synaptic sites of other neurons (dendrites ans somas), muscels, or glands.

Axon hillock is the site of summation for incoming information. At any moment, the collective influence of all neurons that conduct impulses to a given neuron will determine whether or n ot an action potential will be initiated at the axon hillock and propagated along the axon.

Myelin sheath consists of fat-containing cells that insulate the axon from electrical activity. This insulation acts to increase the rate of transmission of signals. A gap exists between each myelin sheath cell along the axon. Since fat inhibits the propagation of electricity, the signals jump from one gap to the next.

Nodes of Ranvier are the gaps (about 1 μm) between myelin sheath cells. Since fat serves as a good insulator, the myelin sheaths speed the rate of transmission of an electrical impulse along the axon.

Synapse is the point of connection between two neurons or a neuron and a muscle or a gland. Electrochemical communication between neurons take place at these junctions.

Terminal buttons of a neuron are the small knobs at the end of an axon that release chemicals called neurotransmitters.

Information flow in a neural cell

The input/output and the propagation of information are shown below.

1.2 ARTIFICIAL NEURON MODEL

An artificial neuron is a mathematical function conceived as a simple model of a real (biological) neuron.

- The McCulloch-Pitts Neuron
  This is a simplified model of real neurons, known as a Threshold Logic Unit.
- A set of input connections brings in activations from other neuron.

- A processing unit sums the inputs, and then applies a non-linear activation function (i.e. squashing/transfer/threshold function).
- An output line transmits the result to other neurons.

1.2.1 Basic Elements of ANN

Neuron consists of three basic components –weights, thresholds and a single activation function. An Artificial neural network(ANN) model based on the biological neural sytems is shown in Figure 2.

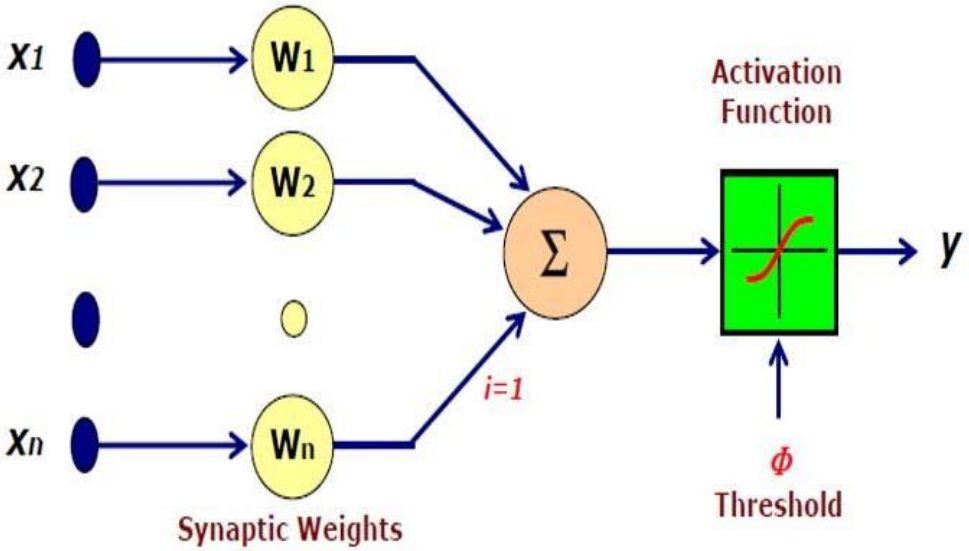


Figure 2: Basic Elements of Artificial Neural Network

1.3 DIFFERENT LEARNING RULES

A brief classification of Different Learning algorithms is depicted in figure 3.

- **Training: It is the process in which the network is taught to change its weight and bias.**
- **Learning: It is the internal process of training where the artificial neural system learns to update/adapt the weights and biases.**

Different Training /Learning procedure available in ANN are

- **Supervised learning**
- **Unsupervised learning**
- **Reinforced learning**
- **Hebbian learning**
- **Gradient descent learning**

- **Competitive learning**
- **Stochastic learning**

1.3.1 Requirements of Learning Laws

- **Learning Law should lead to convergence of weights**
- **Learning or training time should be less for capturing the information from the training pairs**
- **Learning should use the local information**
- **Learning process should able to capture the complex non linear mapping available between the input & output pairs**
- **Learning should able to capture as many as patterns as possible**
- **Storage of pattern information's gathered at the time of learning should be high for the given network**

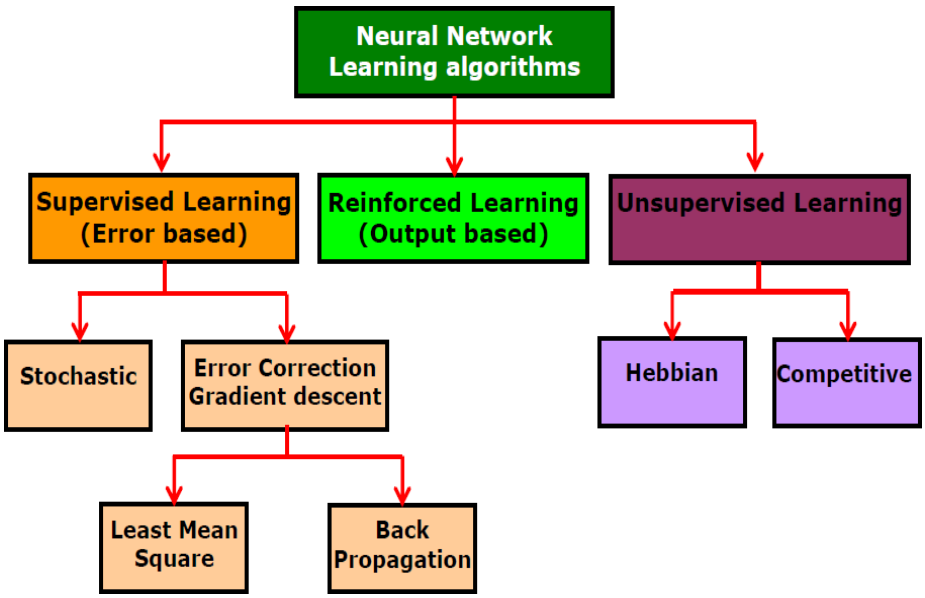


Figure 3: Different Training methods of Artificial Neural Network

1.3.1.1 Supervised learning

Every input pattern that is used to train the network is associated with an output pattern which is the target or the desired pattern.

A teacher is assumed to be present during the training process, when a comparison is made between the network's computed output and the correct expected output, to determine the error.The error can then be used to change network parameters, which result in an improvement in performance.

1.3.1.2 Unsupervised learning

In this learning method the target output is not presented to the network.It is as if there is no teacher to present the desired patterns and hence the system learns of its own by discovering and adapting to structural features in the input patterns.

1.3.1.3 Reinforced learning

In this method, a teacher though available, doesnot present the expected answer but only indicates if the computed output correct or incorrect.The information provided helps the network in the learning process**.**

1.3.1.4 Hebbian learning

This rule was proposed by Hebb and is based on correlative weight adjustment.This is the oldest learning mechanism inspired by biology.In this, the input-output pattern pairs ($\square_{\square}$, $\square_{\square}$) are associated by the weight matrix W, known as the correlation matrix.

It is computed as

$$W = \sum_{\square=1}^{\square} \square_{\square} \square_{\square}^{\square} \qquad \text{---------- eq (1)}$$

Here $\square_{\square}^{\square}$ is the transposeof the associated output vector $\square_{\square}$.Numerous variants of the rule have been proposed.

1.3.1.5 Radient descent learning

This is based on the minimization of error E defined in terms of weights and activation function of the network.Also it is required that the activation function employed by the network is differentiable, as the weight update is dependent on the gradient of the error E.

Thus if $\Delta\square_{\square\square}$ is the weight update of the link connecting the $\square^{\square h}$ and $\square^{\square h}$ neuron of the two neighbouring layers, then $\Delta\square_{\square\square}$ is defined as,

$$\Delta\square_{\square} = \eta \frac{\square\square}{\square\square_{\square\square}} \qquad \text{---------- eq (2)}$$

Where, η is the learning rate parameter and $\frac{\square\square}{\square\square_{\square\square}}$ is the error gradient with reference to the weight $\square_{\square\square}$.

1.3.1.5 Competitive learning

In this method, those neurons which respond strongly to input stimuli have their weights

updated.

When an input pattern is presented, all neurons in the layer compete and the winning neurons undergoes weight adjustment.Hence it is a winner-takes-all strategy.

1.3.1.6 Stochastic learning

In this method, weights are adjusted in a probablistic fashion.An example is evident in simulated annealing the learning mechanism employed by Boltzmann and Cauchy machines, which are a kind of NN systems.

1.3.2 Different Learning Rules

1. Hebb's Learning Law
2. Perceptron Learning Law
3. Delta Learning Law
4. Wldrow and Hoff LMS Learning Law
5. Correlation Learning Law
6. lnstar (Winner-take-all) Learning Law
7. Outstar Learning Law

The different learning laws or rules with their features is given in Table1 which is given below

Table 1: Different learning laws with their weight details and learning type

Learning law	Weight adjustment Δw_{ij}	Initial weights	Learning
Hebbian	$\Delta w_{ij} = \eta f(\mathbf{w}_i^T \mathrm{a}) a_j$ $= \eta s_i a_j$, for $j = 1, 2, \ldots, M$	Near zero	Unsupervised
Perceptron	$\Delta w_{ij} = \eta [b_i - \mathrm{sgn}(\mathbf{w}_i^T \mathbf{a})] a_j$ $= \eta (b_i - s_i) a_j$, for $j = 1, 2, \ldots, M$	Random	Supervised
Delta	$\Delta w_{ij} = \eta [b_i - f(\mathbf{w}_i^T \mathbf{a})] \dot{f}(\mathbf{w}_i^T \mathbf{a}) a_j$ $= \eta [b_i - s_i] f(x_i) a_j$, for $j = 1, 2, \ldots, M$	Random	Supervised
Widrow-Hoff	$\Delta w_{ij} = \eta [b_i - \mathbf{w}_i^T \mathbf{a}]$ a,, for $j = 1, 2, \ldots, M$	Random	Supervised
Correlation	$\Delta w_{ij} = \eta b_i a_j$, for $j = 1, 2, \ldots, M$	Near zero	Supervised
Winner-take-all	$\Delta w_{kj} = \eta (a_j - w_{kj})$, k is the winning unit, for $j = 1, 2, \ldots, M$	Random but **normalised**	Unsupervised
Outstar	$\Delta w_{jk} = \eta (b_j - w_{jk})$, for $j = 1, 2, \ldots, M$	Zero	Supervised

1.4 TYPES OF ACTIVATION FUNCTIONS

Common activation functions used in ANN are listed below

1.4.1 Identity Function

f(x) = x - for all x ---------- eq (3)

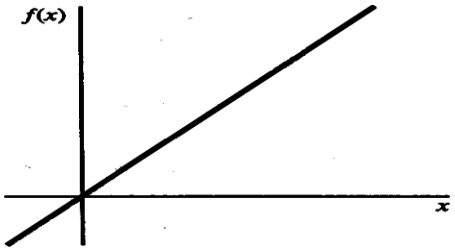


Figure 4: Identity function

Linear functions are simplest form of Activation function.Refer figure 4 . f(x) is just an identity function.Usually used in simple networks. It collects the input and produces an output which is proportionate to the given input. This is Better than step function because it gives multiple outputs, not just True or False

1.4.2. Binary Step Function (with threshold θ) (aka Heaviside Function or Threshold Function)

$$f(x) = \begin{cases} 1 & \text{if } x >= \theta \\ 0 & \text{if } x < \theta \end{cases}$$ ---------- eq (4)

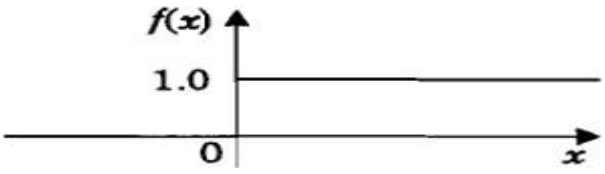


Figure 4: Binary step function

Binary step function is shown in figure 4. It is also called Heaviside function. Some literatures it is also known as Threshold function. Equation 4 gives the output for this function.

1.4.3. Binary Sigmoid

This is also known as Logistic function.The graphical representation is provided in figure5.Equation 5 gives the output values for this function.

$$F(x) = [\ 1/(1+ e^{-ax})] \text{----------eq (5)}$$

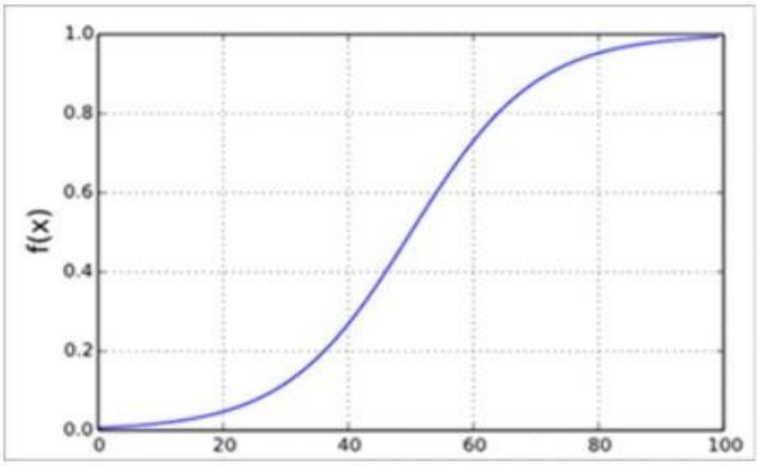


Figure 5: Binary sigmoidal function

1.4.4. Bipolar Sigmoid

Also known as Hyperbolic tangent or tanh function. It is a bounded function whose values lies in the range of (-1 to +1). This is a shifted version of binary Sigmoid Function. It is a Non Linear function.Equation 6 represents this type of function.The pictorial representation for this function is given in figure 6.

$$F(x) = [\ (1- e^{-ax})/(1+ e^{-ax})] \text{----------eq (6)}$$

It is better than Sigmoidal function and its output is zero centered

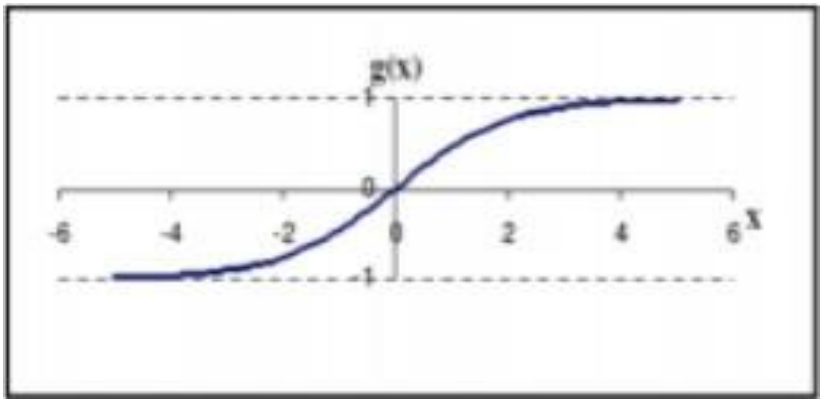


Figure 6: Bipolar sigmoidal function

1.5 PERCEPTRON MODEL

1.5.1 Simple Perceptron for Pattern Classification

Perceptron network is capable of performing pattern classification into two or more categories. The perceptron is trained using the perceptron learning rule. We will first consider classification into two categories and then the general multiclass classification later. For classification into only two categories, all we need is a single output neuron. Here we will use bipolar neurons. The simplest architecture that could do the job consists of a layer of N input neurons, an output layer with a single output neuron, and no hidden layers. This is the same architecture as we saw before for Hebb learning. However, we will use a different transfer function here for the output neurons as given below in eq (7). Figure 7 represents a single layer perceptron network.

$$y = \begin{cases} 1 & \text{if } y\_in > \theta \\ 0 & \text{if } -\theta \le y\_in \le \theta \\ -1 & \text{if } y\_in < -\theta \end{cases}$$

.....eq (7)

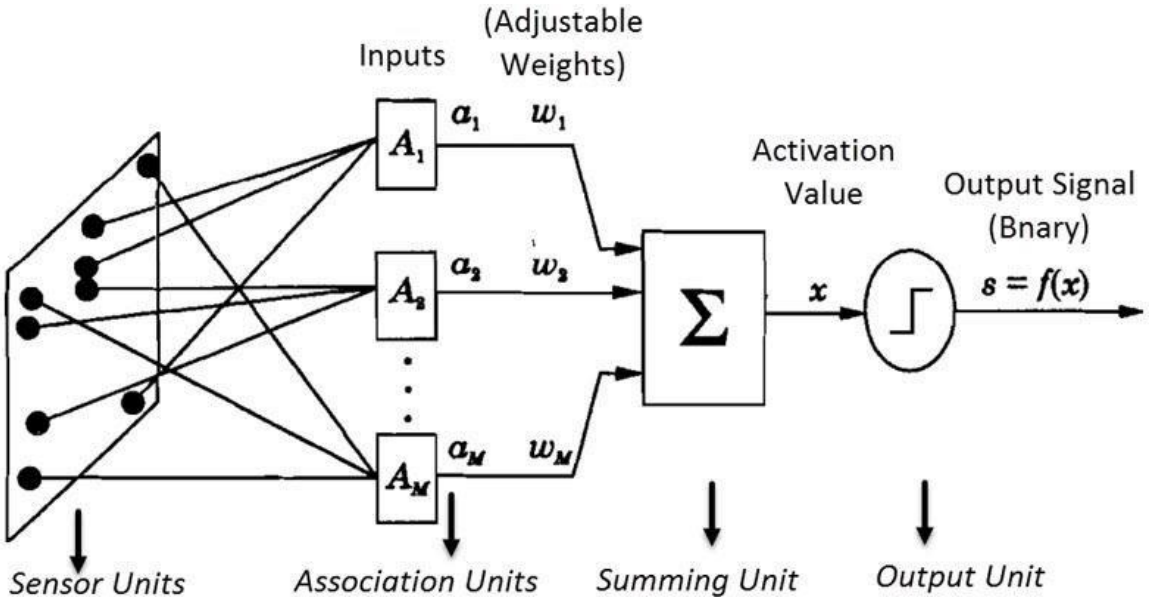


Figure 7: Single Layer Perceptron

Equation 7 gives the bipolar activation function which is the most common function used in the perceptron networks. Figure 7 represents a single layer perceptron network. The inputs arising from the problem space are collected by the sensors and they are fed to the aswociation units.Association units are the units which are responsible to associate the inputs based on their similarities. This unit groups the similar inputs hence the name association unit.

A single input from each group is given to the summing unit.Weights are randomnly fixed intially and assigned to this inputs. The net value is calculate by using the expression

$$x = \Sigma w_i a_i - \theta \text{----------} \quad \text{eq (8)}$$

This value is given to the activation function unit to get the final output response.The actual output is compared with the Target or desired .If they are same then we can stop training else the weights haqs to be updated .It means there is error .Error is given as **δ = b-s** , where b is the desired / Target output and S is the actual outcome of the machinehere the weights are updated based on the perceptron Learning law as given in equation 9.

Weight change is given as **Δw= η δ a_i.** So new weight is given as

$$W_{i\,(new)} = W_{i\,(old)} + \text{Change in weight vector } (\mathbf{\Delta w}) \text{----------} \quad \text{eq (9)}$$

1.5.2 Perceptron Algorithm

Step 1: Initialize weights and bias.For simplicity, set weights and bias to zero.Set learning rate in the range of zero to one.

- Step 2: While stopping condition is false do steps 2-6
- Step 3: For each training pair s:t do steps 3-5
- Step 4: Set activations of input units $x_i = ai$
- Step 5: Calculate the summing part value Net = Σ aiwi-θ
- Step 6: Compute the response of output unit based on the activation functions
- Step 7: Update weights and bias if an error occurred for this pattern (if *y is not equal to t)*
  Weight (new) = w_i(old) + atx_i , & bias (new) = b(old) + at
  Else *w_i(new) = w_i(old) & b(new) = b(old)*
- Step 8: Test Stopping Condition

1.5.3 Limitations of Single Layer Perceptrons

- Uses only Binary Activation function
- Can be used only for Linear Networks
- Since uses Supervised Learning ,Optimal Solution is provided
- Training Time is More
- Cannot solve Linear In-separable Problem

1.5.3 Multi-Layer Perceptron Model

Figure 8 is the general representation of Multi layer Perceptron network.Inbetween the input and output Layer there will be some more layers also known as Hidden layers.

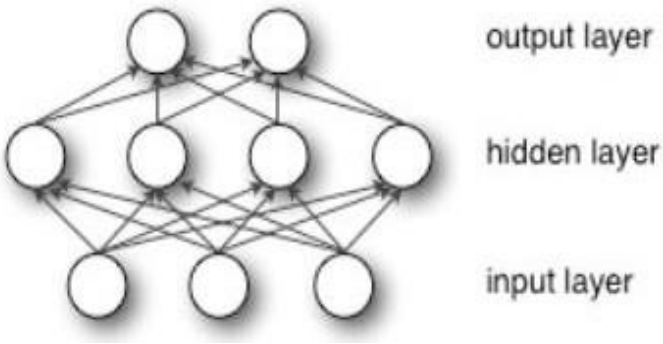


Figure 8: Multi-Layer Perceptron

1.5.4 Multi Layer Perceptron Algorithm

1. Initialize the weights (W_i) & Bias (B_0) to small random values near Zero
2. Set learning rate η or α in the range of "0" to "1"
3. Check for stop condition. If stop condition is false do steps 3 to 7
4. For each Training pairs do step 4 to 7
5. Set activations of Output units: $x_i = s_i$ for i=1 to N
6. Calculate the output Response

   $y_{in} = b_0 + \Sigma x_i w_i$
7. Activation function used is Bipolar sigmoidal or Bipolar Step functions

   For Multi Layer networks, based on the number of layers steps 6 & 7 are repeated
8. If the Targets is (not equal to) = to the actual output (Y), then update weights and bias based on Perceptron Learning Law

   $W_{i\,(new)} = W_{i\,(old)}$ + Change in weight vector

   Change in weight vector = $\eta t_i x_i$

   Where η = Learning Rate

   t_i = Target output of i^{th} unit

   x_i = i^{th} Input vector

   $b_{0(new)} = b_{0\,(old)}$ + Change in Bias

   Change in Bias = ηt_i

   Else $W_{i\,(new)} = W_{i\,(old)}$

   $b_{0(new)} = b_{0\,(old)}$
9. Test for Stop condition

1.6 LINEARLY SEPERABLE & LINEAR IN SEPARABLE TASKS

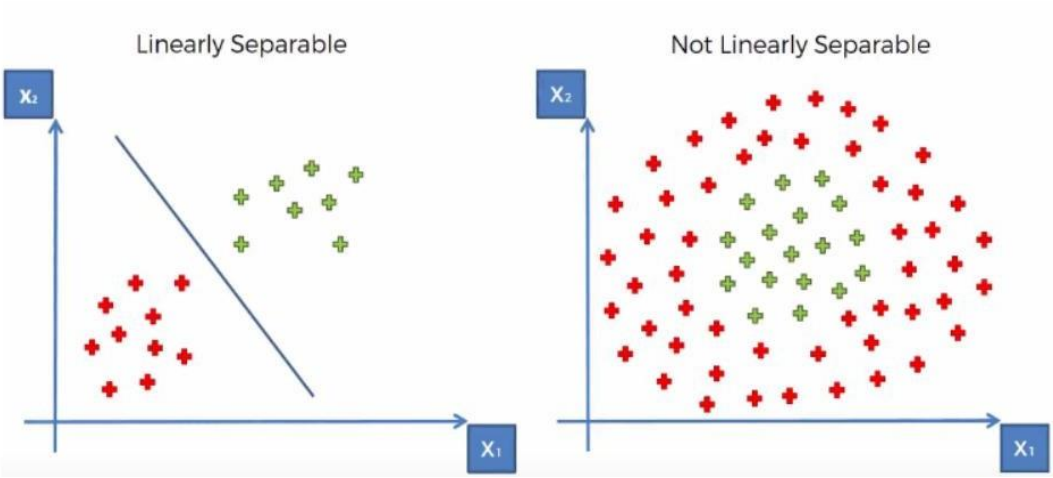


Figure 9: Representation of Linear seperable & Linear-in separable Tasks

Perceptron are successful only on problems with a linearly separable solution sapce. Figure 9 represents both linear separable as well as linear in seperable problem.Perceptron cannot handle, in particular, tasks which are not linearly separable.(Known as linear inseparable problem).Sets of points in two dimensional spaces are linearly separable if the sets can be seperated by a straight line.Generalizing, a set of points in n-dimentional space are that can be seperated by a straight line.is called Linear seperable as represented in Figure 9.

Single layer perceptron can be used for linear separation.Example AND gate.But it cant be used for non linear ,inseparable problems.(Example XOR Gate).Consider figure 10.

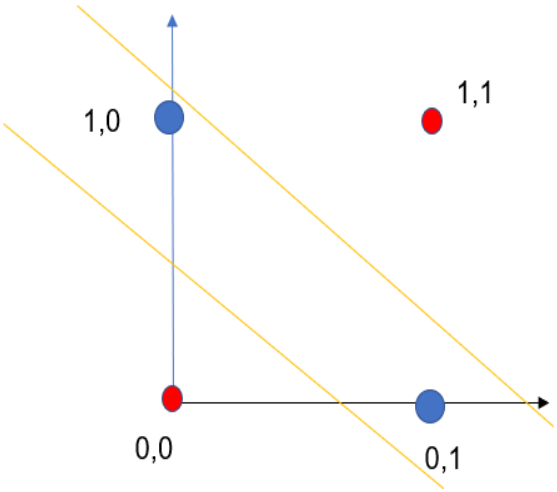


Figure 10: XOR representation (Linear-in separable Task)

Here a single decision line cannot separate the Zeros and Ones Linearly.At least Two lines are required to separate Zeros and Onesas shown in Figure 10. Hence single layer networks can not be used to solve inseparable problems. To over come this problem we go for creation of **convex regions**.

Convex regions can be created by multiple decision lines arising from multi layer networks.Single layer network cannot be used to solve inseparable problem.Hence we go for multilayer network there by creating convex regions which solves the inseparable problem.

1.6.1 Convex Region

Select any Two points in a region and draw a straight line between these two points. If the points selected and the lines joining them both lie inside the region then that region is **known as convex regions.**

1.6.2 Types of convex regions

(a) Open Convex region (b) Closed Convex region

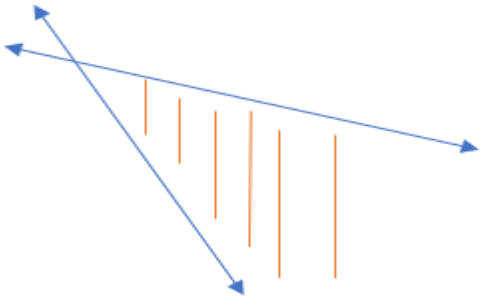


Figure 11: Open convex region

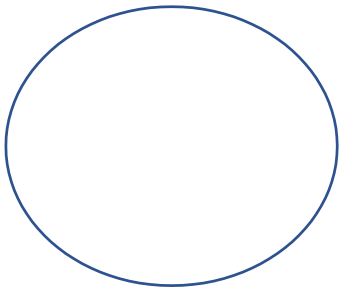


Figure 12 A: Circle - Closed convex region

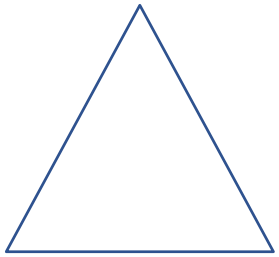


Figure 12 B: Triangle - Closed convex region

REFERENCE BOOKS

1. B. Yegnanarayana, “Artificial Neural Networks” Prentice Hall Publications.
2. Simon Haykin, “Artificial Neural Networks”, Second Edition, Pearson Education.
3. Laurene Fausett, “Fundamentals of Neural Networks, Architectures, Algorithms and Applications”, Prentice Hall publications.

Note: For further reference, kindly refer the class notes, PPTs, Video lectures available in the Learning Management System (Moodle)

********** **ALL THE BEST** **********

UNIT – II – MULTI LAYER NETWORKS

2. MULTI LAYER NETWORKS

UNIT II MULTI LAYER NETWORKS

Back Propagation Networks (BPN) - Training - Architecture-Algorithm, Counter Propagation Network (CPN) - Training - Architecture, Bi-Directional Associative Memory (BAM) - Training-stability analysis, Adaptive Resonance Theory – Adaptive Resonance Theory (ART) - ART1- ART2 – Architecture -Training, Hop Field Network - Energy Function - Discrete - Continuous - Algorithm - Application – Travelling Sales Man Problem TSP

2 BACK PROPAGATION NETWORKS (BPN)

2.1 NEED FOR MULTILAYER NETWORKS

- Single Layer networks cannot used to solve Linear Inseparable problems & can only be used to solve linear separable problems
- Single layer networks cannot solve complex problems
- Single layer networks cannot be used when large input-output data set is available
- Single layer networks cannot capture the complex information's available in the training pairs

Hence to overcome the above said Limitations we use Multi-Layer Networks.

2.2 MULTI-LAYER NETWORKS

- Any neural network which has at least one layer in between input and output layers is called Multi-Layer Networks
- Layers present in between the input and out layers are called Hidden Layers
- Input layer neural unit just collects the inputs and forwards them to the next higher layer
- Hidden layer and output layer neural units process the information's feed to them and produce an appropriate output
- Multi -layer networks provide optimal solution for arbitrary classification problems
- Multi -layer networks use linear discriminants, where the inputs are non linear

2.3 BACK PROPAGATION NETWORKS (BPN)

Introduced by Rumelhart, Hinton, & Williams in 1986. BPN is a Multi-layer Feedforward Network but error is back propagated, Hence the name Back Propagation Network (BPN). It uses Supervised Training process; it has a systematic procedure for training the network and is used in Error Detection and Correction. Generalized Delta Law /Continuous Perceptron Law/ Gradient Descent Law is used in this network. Generalized Delta rule minimizes the mean squared error of the output calculated from the output. Delta law has faster convergence rate when compared with Perceptron Law. It is the extended version of Perceptron Training Law. Limitations of this law is the Local minima problem. Due to this the convergence speed reduces, but it is better than perceptron's. Figure 1 represents a BPN network architecture. Even though Multi level perceptron's can be used they are flexible and efficient that BPN. In figure 1 the weights between input and the hidden portion is considered as Wij and the weight between first hidden to the next layer is considered as V_{jk}. This network is valid only for Differential Output functions. The Training process used in backpropagation involves three stages, which are listed as below

1. Feedforward of input training pair
2. Calculation and backpropagation of associated error
3. Adjustments of weights

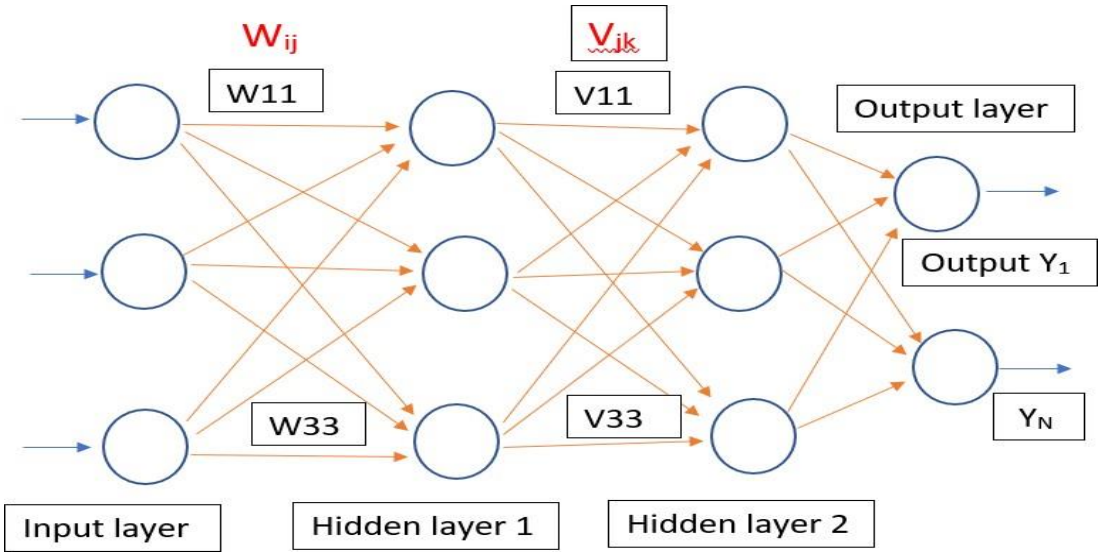


Figure 1: Back Propagation Network

2.3.1 BPN Algorithm

The algorithm for BPN is as classified int four major steps as follows:

1. Initialization of Bias, Weights
2. Feedforward process
3. Back Propagation of Errors
4. Updating of weights & biases

Algorithm

I. Initialization of weights

Step 1: Initialize the weights to small random values near zero

Step 2: While stop condition is false , Do steps 3 to 10

Step 3: For each training pair do steps 4 to 9

II. Feed forward of inputs

Step 4: Each input xi is received and forwarded to higher layers (next hidden)

Step 5: Hidden unit sums its weighted inputs as follows

$$Z_{inj} = W_{oj} + \Sigma x_i w_{ij}$$

Applying Activation function

$$Z_j = f(Z_{inj})$$

This value is passed to the output layer

Step 6: Output unit sums it's weighted inputs

$$y_{ink} = V_{oj} + \Sigma Z_j V_{jk}$$

Applying Activation function

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

III. Backpropagation of Errors

Step 7: $\delta_k = (t_k - Y_k) f(y_{ink})$

Step 8: $\delta_{inj} = \Sigma \delta_j V_{jk}$

IV. Updating of Weights & Biases

Step 8: Weight correction is $\Delta w_{ij} = \alpha \delta_k Z_j$

bias Correction is $\Delta w_{oj} = \alpha \delta_k$

V. Updating of Weights & Biases

Step 9: continued:

New Weight is

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

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

New bias is

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

$$V_{ok(new)} = V_{ok(old)} + \Delta V_{ok}$$

Step 10: Test for Stop Condition

2.3.2 Merits

- Has smooth effect on weight correction
- Computing time is less if weight's are small
- 100 times faster than perceptron model
- Has a systematic weight updating procedure

2.3.3 Demerits

- Learning phase requires intensive calculations
- Selection of number of Hidden layer neurons is an issue
- Selection of number of Hidden layers is also an issue
- Network gets trapped in Local Minima
- Temporal Instability
- Network Paralysis
- Training time is more for Complex problems

2.4 COUNTER PROPAGATION NETWORK [CPN]

This network was proposed by Hect & Nielsen in 1987.It implements both supervised & Unsupervised Learning. Actually it is a combination of two Neural architectures (a) Kohonan Layer - Unsupervised (b) Grossberg Layer – Supervised. It Provides good solution where long training is not tolerated. CPN functions like a Look-up Table Generalization. The training pairs may be Binary or Continuous. CPN produces a correct output even when input is partially incomplete or incorrect. Main types of CPN is (a) Full Counter Propagation (b) Forward only Counter Propagation. Figure 2 represents the architectural diagram of CPN network.

- Forward only nets are the simplified form of Full Counter Propagation networks
- Forward only nets are used for approximation problems

The first layer is Kohonan layer which uses competative learning law.The procedure used here is , when an input is provided the weighted net values is calculated for each node. Then the node with maximum output is selected and the signals from other neurons are inhibited. This output from the wining neuron only is provided to the next higher layer, which is the Supervised Grosssberg layer.Grossberg processing is similar to that of an normal supervised algorithm.

Training of Kohonan network

- Kohonan training implements self organising Unsupervised training procedure which takes time.
- Kohonan network involves winner takes tall weight updating rule
- Uses Euclidean distance measure or Dot product for clustering/ grouping of inputs

Training of Grossberg Network

- Uses Supervised training
- Similar to BPN forward pass
- Weights are updated based on Delta Law

2.4.1 Algorithm: Full CPN

Step 1: Initialize the weights & learning rate to a small random value near zero

Step 2: While stop condition is false , Do steps 3 to 9

Step 3: Set the X- input layer input Activations to vector X

Step 4: Each input x_i is received and forwarded to higher layers (Kohonan Layer)

Step 5: Kohonan unit sums its weighted inputs as follows

Inputs & Weights are normalised, then net value is calculated as follows

$$K_{inj} = W_{oj} + XW \text{ (in vectors)}$$

Applying Activation function

$$K_j = f(K_{inj})$$

Step 5A: Wining Cluster is identified.(The node with a maximum output is selected as Winner. Only this output is forwarded to the next Grossberg layer, All other units output are inhibited)

For clustering the inputs Xi's , Euclidian Distance Norm function is used

$$Dj = \sum(xi-vij) \wedge 2 + \sum(yk-wkj) \wedge 2$$

D_j should be minimum

Step 6: Update the weights over the calculated winner unit K_j

Step 7: Test for stop condition of phase -I

(Phase –I : Input X layer to Z Cluster layers)

Phase – II : Z Cluster layers to Y output layers

Step 8: Repeat steps 5,5A,6 for Phase –II layers

Step 9: Test for stop condition for phase II

2.4.2 Merits

- A combination of Unsupervised (Phase-I) & Supervised
- Network works as like a Look Up Table
- Fast and Coarse approximation
- 100 times faster than BPN model

2.4.3 Demerits

- Learning phase requires intensive calculations
- Selection of number of Hidden layer neurons is an issue
- Selection of number of Hidden layers is also an issue
- Network gets trapped in Local Minima

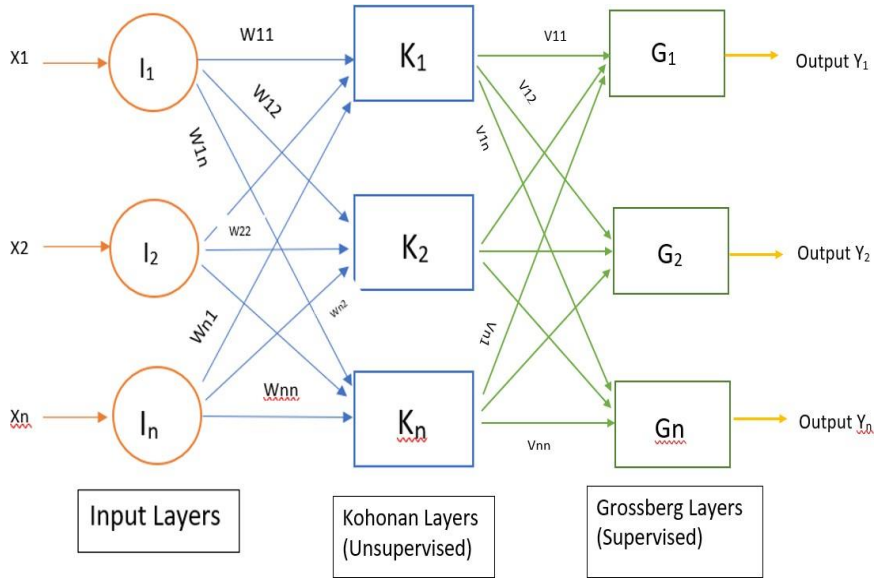


Figure 2: Counter Propagation Networks

2.5 BI-DIRECTIONAL ASSOCIATIVE MEMORIES

- Developed by Kosko in 1988
- Hetro Associative with Two layers (Input & Output) [Single layer]
- Transmits signals back and forth between these layers
- Stop condition is that the activations of all neurons remain constant for several iterations
- It is used to store and retrieve the patterns
- Utilizes directional weighted connection paths

2.5.1 Types of BAM

(a) Binary BAM

(b) Bipolar BAM

(c) Continuous BAM

- Continuous BAM: Uses Log-sigmoidal Function for activation function

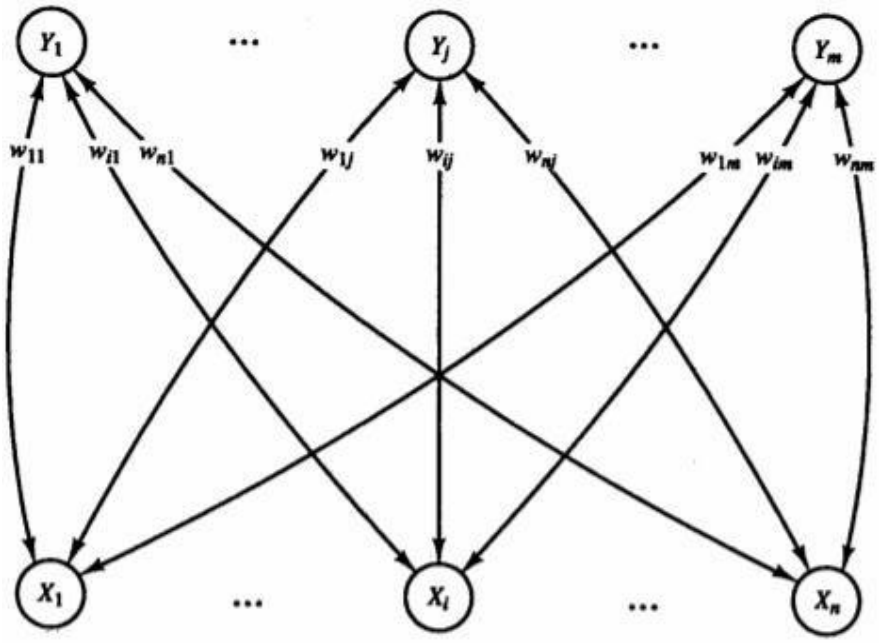


Figure 3: Bi-Directional Associative memory

Image adapted from Laurene Fausett, "Fundamentals of Neural Networks, Architectures, Algorithms and Applications", Prentice Hall publications.

❖ Figure 3 represents the BAM architecture. BAM contains ‘n’ neurons in X layer and ‘m’ neurons in Y layer. Both X & Y layers can act as input or Output layers. Weights for X layer is taken as W_{ij} . Weights for Y layer is taken as $W_{ij}{}^{T}$. If Binary or Bipolar Activations are used then it is known as Discrete BAM. If Continuous Activations are used then it comes under Continuous BAM type.

2.5.2 Algorithm

The following steps explain the procedural flow of BiDirectional Associative Memory

Step 0: Initialize the weights to store a set of input (P) Vectors.

Set all intial activations to Zero

Step 1: For each inputs do steps 2 to 6

Step 2A: Present the input pattern x to X layer

Step 2B: Present the input pattern y to Y layer

Step 3: While activations are not converged do steps 4 to 6

Step 4: Update the activations of units in Y layer.Compute Net value $y_{in} = \Sigma x_i w_{ij}$ and compute activatins $y_j = f(y_{in})$. Send signal to X layer

Step 5: Update the activations of units in X layer. Compute Net value $x_{in} = \Sigma y_j w_{ij}$ and compute activatins $x_i = f(x_{in})$. Send signal to Y layer

Step 6: Test for Convergence.If the activations of layers X and Y had reached equilibrium then stop else continue the above said process

2.5.3 Merits

- Unconditionally Stable network
- Best for Content type of address memory
- Special case of Hopfield network
- Best Recall

2.5.4 Demerits

- Incorrect Convergence
- Memory capacity is limited because storage of ‘m’ patterns should be lesser than ‘n’ neurons of smaller layer
- Sometimes the networks learns some patterns which are not provided to it

2.5.5 Applications of BAM

- Fault Detection
- Pattern Association
- Real Time Patient Monitoring
- Medical Diagnosis
- Pattern Mapping
- Pattern Recognition systems
- Optimization problems
- Constraint satisfaction problems

2.6 ADAPTIVE RESONANCE THEORY

- Invented by Grossberg in 1976 and based on unsupervised learning model.
- Resonance means a target vector matches close enough the input vector.
- ART matching leads to resonance and only in resonance state the ART network learns.
- Suitable for problems that uses online dynamic large databases.
- Types:
  (1) ART 1- classifies binary input vectors
  (2) ART 2 – clusters real valued input (continuous valued) vectors.
- Used to solve Plasticity – stability dilemma.

2.6.1 Plasticity - Stability

How to learn a new pattern without forgetting the old traces (patterns) and how to adapt to the changing environment (i/p). When there is change in the patterns (plasticity) how to remember previously learned vectors (stability problem) is a problem. ART uses competitive law (self-regulating control) to solve this PLACITICITY – STABILITY Dilemma. The simplified ART diagram is given below in Figure 5.

The Adaptive Resonance Theory (ART) consist of

(1) F1 Layer: I/P processing unit also called comparison layer.
(2) F2 Layer: clustering or competitive layer.
(3) Reset mechanism.

2.6.2 Comparison Layer: Take 1D i/p vector and transfers it to the best match in recognition field (best match - neuron in recognition unit whose weight closely matches with i/p vector).

2.6.3 Recognition Unit: produces an output proportional to the quality of match. In this way recognition field allows a neuron to represent a category to which the input vectors are classified.

Vigilance parameter: After the i/p vectors are classified a reset module compares the strength of match to vigilance parameter (defined by the user). Higher vigilance produces fine detailed memories and lower vigilance value gives more general memory.

The schematic representation of ART-1 is shown in Figure 4.Figure 6 represents the supplementary units present in ART-1 networks

2.6.4 Reset module: compares the strength of recognition phase. When vigilance threshold is met then training starts otherwise neurons are inhibited until a new i/p is provided.

There are two set of weights (1) Bottom up weight - from F1 layer to F2 Layer
(2) Top –Down weight – F2 to F1 Layer

2.6.5 Learning in ART

There are two types of Learning which are explained as below.

Fast learning: Happens in ART 1 – Weight changes are rapid and takes place during resonance. The network is stabilized when correct match at cluster unit is reached.

Slow Learning: Used in ART 2. weight change is slow and does not reach equilibrium in each learning iteration.so more memory to store more i/p patterns (to reach stability) is required.

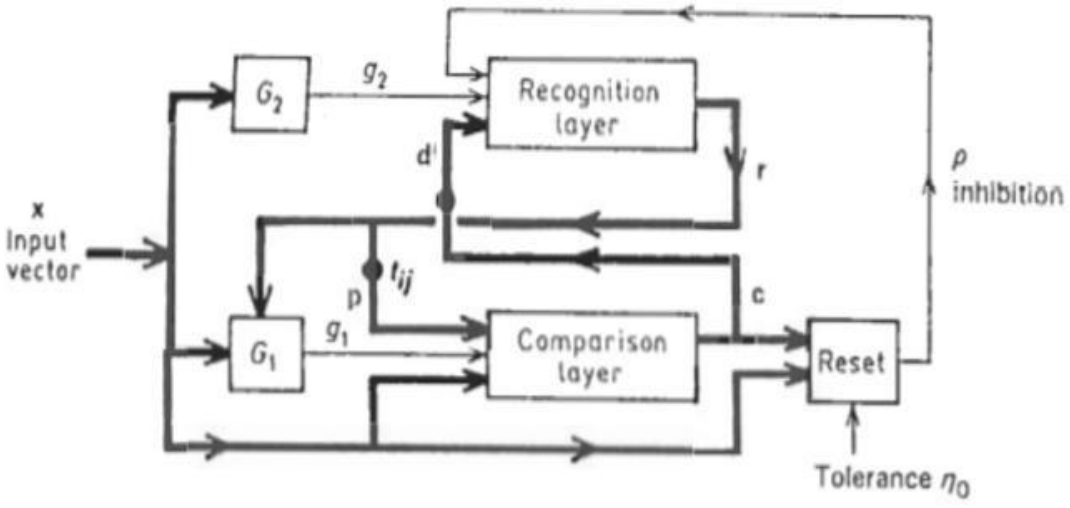


Figure 4: Schematic representation of ART-1

Images adapted from Laurene Fausett, "Fundamentals of Neural Networks, Architectures, Algorithms and

Applications", Prentice Hall publications

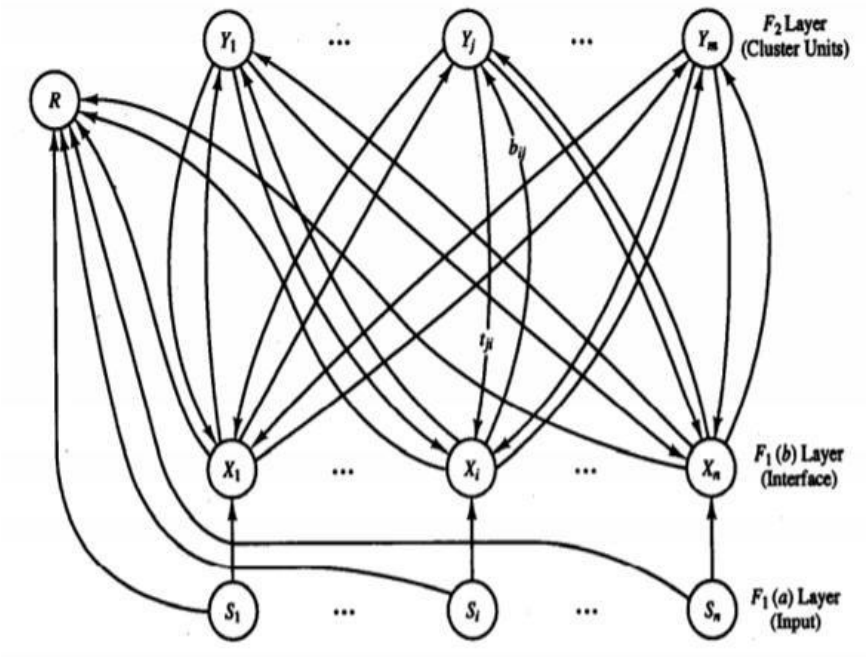


Figure 5: Basic structure of ART-1 Network

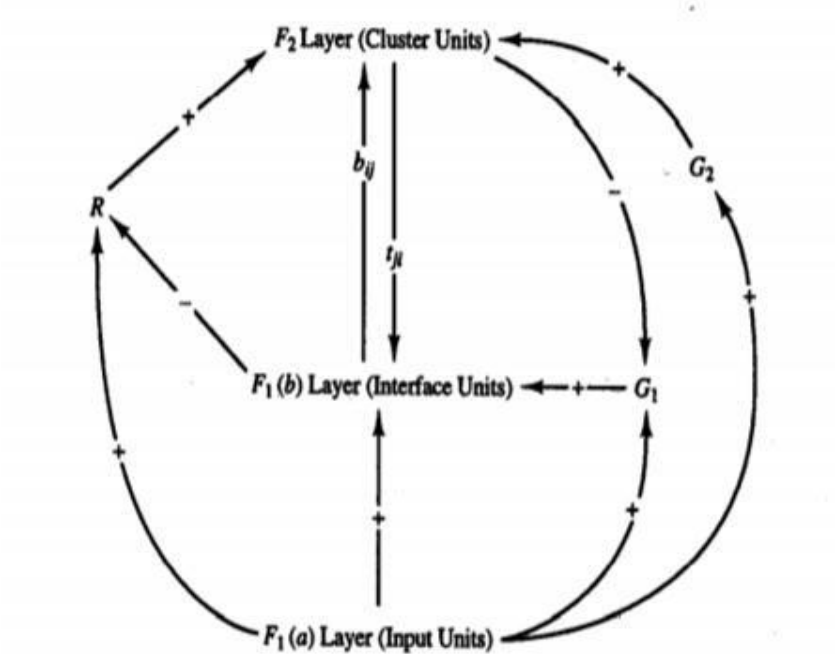


Figure 6: Supplementary Units for ART-1

Images adapted from Laurene Fausett, "Fundamentals of Neural Networks, Architectures, Algorithms and Applications", Prentice Hall publications

2.6.6 Basic ART Training Algorithm:

Step 1: Initialize the parameters

Step 2: For each input do steps 3 to 8

Step 3: Process F1 layer

Step 4: while Reset Condition is true do steps 5 -7

Step 5: Find a candidate to learn the input pattern. Select F2 unit with Maximum value

Step 6: F1(b) units combine their inputs with F1(a) and F2 layer

Step 7: Test reset condition. If Reset is true then the selected candidate is rejected else accepted.

Step 8: Learning weights are changed according to the differential equations

Step 9: Test for stop condition

2.6.7 Applications of ART

- Pattern Recognition
- Pattern Restoration
- Pattern Generalization
- Pattern Association
- Speech Recognition
- Image Enhancement
- Image Restoration
- Facial Recognition systems
- Optimization problems
- Used to solve Constraint satisfaction problems

2.7 HOPFIELD NETWORKS

The net is a fully interconnected neural net, in the sense that each unit is connected to every other unit. The net has symmetric weights with no self-connections, $W_{ij} = W_{ji}$ and $W_{ii} = 0$

Only one unit updates its activation at a time and each unit continues to receive an external signal in addition to the signal from the other units in the net. The asynchronous

updating of the units allows a function, known as an energy or Lyapunov function, to be found for the net.

2.7.1 Architecture of Hopfield Networks

The basic diagram for Hopfield Networks is given in Figure 7. Here no learning algorithm is used. No Hidden units/layers used. Patterns are simply stored by learning the energies. Similar to Human brain in storing and retrieving memory patterns. Some patterns / images are stored & when similar noisy input is provided the network recalls the related stored pattern. The neuron can be ONN(+1) or OFF(-1).The neurons can change state between +1 & -1 based on the inputs which they receives from other neurons. Hopfield Network is trained to store patterns(memories). It can recognize previously learned (stored) Pattern from partial (noisy) inputs.

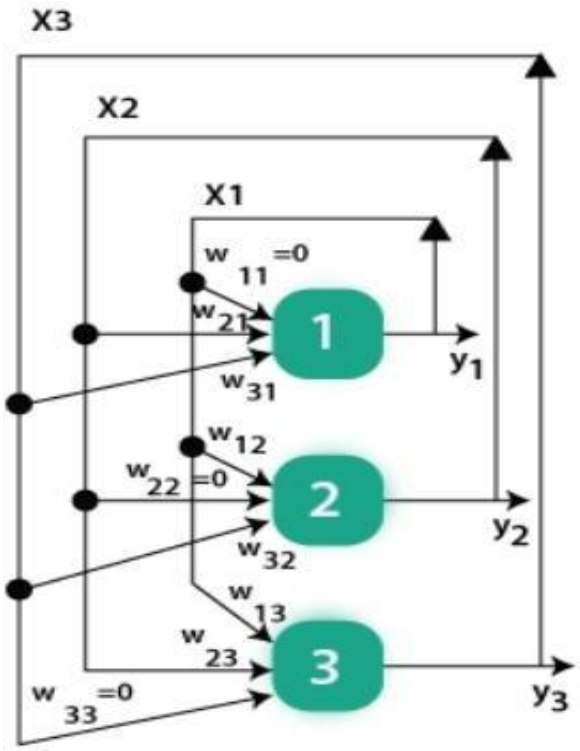


Figure 7: Architecture of Hopfield Networks

2.7.2 Types of Hopfield Network

Based on the activation functions used the Hopfield Network can Be classified into two types. They are

(a) Discrete Hopfield network (b) Continuous Hopfield Network

Discrete Hopfield Network – Uses Discrete Activation Function

Continuous Hopfield Network – Uses Continuous Activation Function

Hopfield Networks Uses Lyapunov Energy Function. Energy function guarantees the network to reach a stable minimum local energy state which resembles the stored patterns

2.7.3 Lyapunov Energy Function

The Lyapunov Energy function for discrete Hopfield is given as follows

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

Change in Energy is given as equal to $[-(net_i)\ \Delta Y_i]$

The Lyapunov Energy function for Continuous Hopfield is given as follows

$$E_f = \frac{1}{2}\sum_{i=1}^{n}\sum_{\substack{j=1 \\ j\neq i}}^{n} y_i y_j w_{ij} - \sum_{i=1}^{n} x_i y_i + \frac{1}{\lambda}\sum_{i=1}^{n}\sum_{\substack{j=1 \\ j\neq i}}^{n} w_{ij} g_{ri} \int_0^{y_i} a^{-1}(y)dy$$

2.7.4 Algorithm

Step 1: Initialize the weights To store the pattern

Step 2: For each i/p vector repeat steps 3 to 7

Step 3: Set the initial activations of the net equal to the external i/p vector X

$$Y_i = X_i (i=1,2,\ldots,n)$$

Step 4: Perform 5 to 7 for each unit y_i

Step 5: Compute

$$y_{ini} = X_i + \Sigma y_i W_{ij}$$

Step 6: Determine the activation response based on the activation function used

Step 7: Broadcast the value of Yi to all other units

Step 8: Test for Convergence

2.7.5 Merits and Demerits of Hopfield Networks

Merits

- Unconditionally Stable network
- Best for Content type of address memory
- Special case of Hopfield network
- Best Recall

Demerits

- Incorrect Convergence
- Memory capacity is limited because storage of ‘m’ patterns should be lesser than ‘n’ neurons of smaller layer
- Sometimes the networks learn some patterns which are not provided to it

REFERENCE BOOKS

1. B. Yegnanarayana, “Artificial Neural Networks” Prentice Hall Publications.
2. Simon Haykin, “Artificial Neural Networks”, Second Edition, Pearson Education.
3. Laurene Fausett, “Fundamentals of Neural Networks, Architectures, Algorithms and Applications”, Prentice Hall publications.
4. James A. Freeman & Skapura, “Neural Networks”, Pearson Education.

********** **ALL THE BEST** **********

UNIT – III – SOM & SPECIAL NETWORKS

2. MULTI LAYER NETWORKS

UNIT III- SOM & SPECIAL NETWORKS

SOM-Introduction - Kohonan SOM - Linear vector quantization, Probabilistic neural network, Cascade correlation, General Regression neural network, Cognitron - Application of ANN - Texture classification - Character recognition.

2.1 SELF-ORGANISING NETWORKS (SOM)

Developed by Finish Prof Teuvo Kohonan in 1980's**.** This network is also known as Topology Preserving maps. The name Topology preserving is provided since the location or position of the node varies in the stating time of training procedure and once the network learned the given input pattern the topology or the location of neural nodes are fixed.

SOM is a Feedforward network with Single computational layer. It utilizes Unsupervised training and is used for Dimensionality reduction. The main goal of SOM is to change the arbitrary dimension of the given input pattern into a one- or two-Dimensional space. Map is a 2-Dimensional space where the nodes are organized in Rectangular or Hexagonal Grid. Figure 1 shows the grids used in SOM

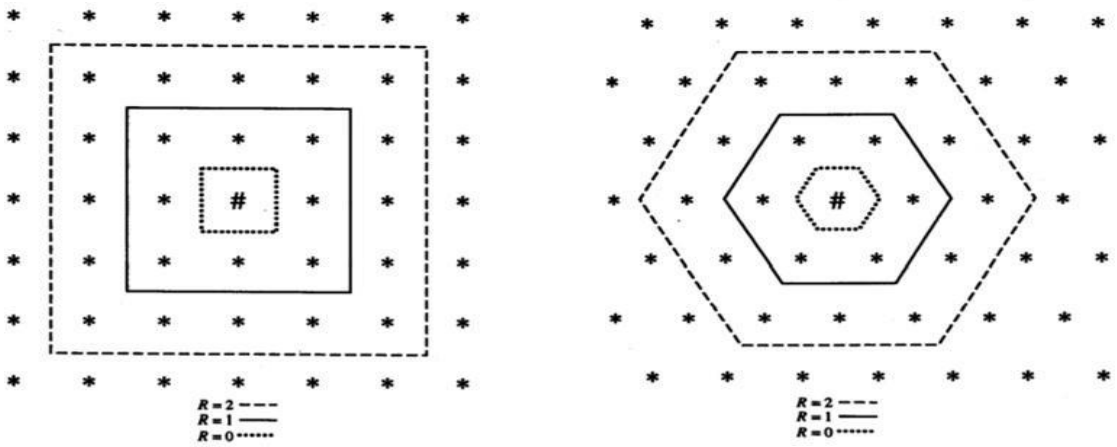


Figure 1(a). Rectangular Neural Grid **(b) Hexagonal Grid**

Each node is provided with a weight vector which is nothing but the position of that node in the input space or Map. Job of training is to adjust this weight vector so the distance in the map reduces. The weight moves towards the input. Thus, from a higher dimension the map reduces to a 2 Dimension. This is the Dimensionality Reduction Process. After training, SOM can classify the input by selecting a nearest node (small distance) with closest weight vector to the input space vector

This transformation is performed in an orderly manner. SOM uses only two-dimensional discretized input space known as Maps for its operation. Instead of error correction learning SOM uses Competitive / Winner Takes All learning is utilized in SOM.

SOM operates in Two modes

(1) Training

(2) Mapping

- Training Process: Develops the map using competitive procedure (Vector Quantization)
- Mapping Process: Classifies the new supplied input based on the training outcomes

2.1.1 SOM Algorithm

The general steps involved in SOM is given as follows

Step 1: Initialize the Weights W_{ij}. Initialize the learning rate and topological neighbourhood parameters

Step 2: While stop condition is false Do steps 3 to 9

Step 3: For each input Vector x, do steps 4 to 6

Step 4: For each j calculate $\mathbf{D(j) = \Sigma (W_{ij} - x_i)^2}$

Step 5: Find the index 'j' for which D(j) is Minimum

Step 6: For all units of j within a specific neighbourhood of j and for all i

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

Step 7: Update Learning Rate

Step 8: Reduce the radius of Topological Neighbourhood at specific time periods

Step 9: Test for stop condition

2.1.2 SOM Explanation

- **Initialization**

Weights are randomly initialized to a small value near zero (W_j)

- **Competition**

For the given each inputs patterns, the neurons calculate a discriminant function(Here we use Euclidean Distance function).This Discriminant function acts as a basis for the

competition among neurons. The neuron with smaller Distance value is selected as wining neuron(Winner Takes All law)

- **Cooperation**

The wining neuron determines the spatial locations of the excited neurons in that neighbourhood (Topological map). Thus, a cooperation between the neurons is established by the wining neuron in that rearranged neighbourhood

- **Adaptation**

The wining neurons by adjusting its weight values, tries to minimize the discriminant function (distance value) between them and the inputs. When similar inputs are provided the response of the wining neuron is enhance in a better way

2.1.3 SOM Merits and Demerits

Merits

- Easy to interpret
- Dimensionality Reduction
- Capable of handling different types of classification problems
- Can cluster large complex input set's
- SOM training Time is less
- Simple algorithm
- Easy to implement

Demerits

- It does not build a generative model for the data, i.e, the model does not understand how data is created.
- It does not behave so gently when using categorical data, even worse for mixed types data.
- The time for preparing model is slow, hard to train against slowly evolving data

2.1.4 Applications of SOM

Applications

- Character Recognition
- Speech Recognition

- Texture Recognition
- Image Clustering
- Data Clustering
- Classification problems
- Dimensionality reduction applications
- Seismic analysis
- Failure Analysis etc

2.2 LEARNING VECTOR QUANTIZATION [LVQ]

- Supports Single as well as Multi Class Classification
- CODE BOOK Vectors are Developed by Training process
- Prediction is done similar to that of K-Nearest Neighbor procedure
- Codebook vectors are created from the training dataset by moving them closer (when they are good match to the weight and input), and further away when they are a bad match.
- When a $K^{t\ h}$ new instance (i/p data) is given, the code book vectors are searched for a similar value. That node/codebook vector is selected and its associated output class is given as final output
- To select a similar input for the given K^{th} instance or training dataset, Euclidian Distance measure is used as shown below

Euclidean Distance (X, X_i) = sqrt (SUM ((X_j – X_{ij}) ^2)

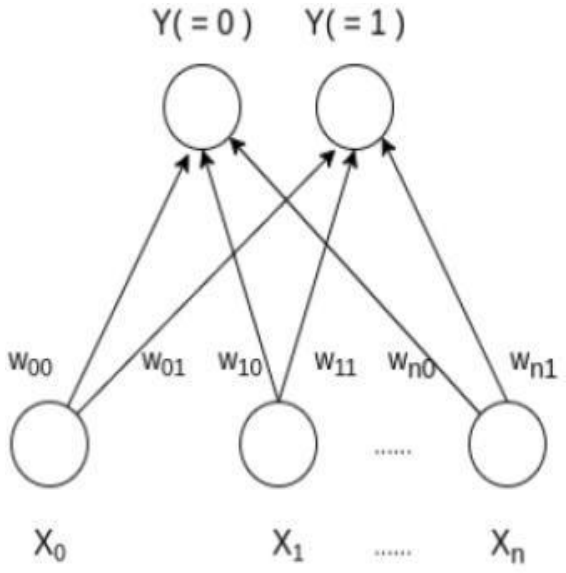


Figure 2: LVQ Schematic Diagram

2.2.1 Algorithm

Step 1: Initialize the weight vectors to the 'm' Training vectors, where 'm' is the number of different classification/Cluster. Start learning rate α near zero (small value)

Step 2: While Stop condition id false do steps 3 to 6

Step 3: For each input training vector X, do steps 4 to 5

Step 4: Find J such that D(J) is minimum

Step 5: Update the weights of J^{th} neural unit as given below

IF T = Cj then

$\mathbf{W_{j(new)} = W_{j(old)} + \alpha\ [x - w_{j(old)}]}$ **[** Move the weight vector W towards the input X**]**

If T is Not Equal to C then

$\mathbf{W_{j(new)} = W_{j(old)} - \alpha[x - w_{j(old)}]}$ **[** Move the weight vector W away from the input X]

Step 6: Reduce Learning Rate α

Step 7: Test for stop condition (Either a fixed number of Iteration reached or Learning rate αhas reached a very minimum value

2.2.2 Merits

- Due to Normalization (Data Preparation) Dominance of one unit is avoided
- Dimensionality Reduction
- Feature Engineering
- Multiple Best Matches
- Multiple Passes (Multiple runs)(higher learning rate for CODE BOOK Pool Generation & Lower Learning rate for tuning the vectors)
- Simple algorithm
- Easy to implement
- Precursor to K nearest Neighbour & SOM's

Demerits

1. Data has to be prepared before executing this algorithm.
2. Output class should be predefined

2.3. PROBABILISTIC NEURAL NETWORKS [PNN]

- PNN has three layers of Neural Node interconnections
- Input layer can take 'n' number of nodes
- Input nodes are connected with the feature vectors

- All input feature vectors are connected with the Middle-Hidden Layer
- Hidden nodes are connected into groups and Each group denotes a particular class 'K'
- Each node present in Hidden Layer resembles to a Gaussian Function centered on its Feature Vector for that K^{th} class
- All of these Gaussian function outputs of a group/class are fed to the K^{th} Output unit
- Hence, we have only 'K' Output units only
- PNN is closely related to PARZEN Window PDF Estimator or Mixed Gaussian Estimator
- For any output node 'K', all Gaussian values (of the previous Hidden layer) for that output class are summed up
- This summed up value is scaled to a Probability Density Function (PDF)
- If class 1 contains 'p' feature vectors and Class 2 contains 'Q' feature vectors, Then P nodes are present in Hidden layer for the class 1 & Q nodes for class 2 is present
- The equations for Gaussian functions for any input is given as

$$g_1(\mathbf{x}) = [1/\sqrt{(2\pi\sigma^2)^N}]\exp\{-\|\mathbf{x} - \mathbf{x}^{(p)}\|^2/(2\sigma^2)\}$$

$$g_2(\mathbf{y}) = [1/\sqrt{(2\pi\sigma^2)^N}]\exp\{-\|\mathbf{y} - \mathbf{y}^{(q)}\|^2/(2\sigma^2)\}$$

The schematic representation of Probabilistic Neural Network is shown in Figure 3.

2.3.1. Algorithm

To Set the PNN following steps are used:

Step 1: Read the exemplars vectors and class numbers

Step 2: Sort the above into 'K' sets, where each set contains one class of vector

Step 3: Define a Gaussian function centered on each exemplar vector and define the summed up gaussian output function

To Classify the following steps are used:

Step 4: Read the input and feed it to the Gaussian function in each set

Step 5: Calculate the Gaussian function values at the output

Step 6: Feed this gaussian out to single output of that group

Step 7: At each class output node, Sum all its input and multiply with a constant

Step 8: Find the maximum value of all summed function values at output

Step 9: Test for stop condition

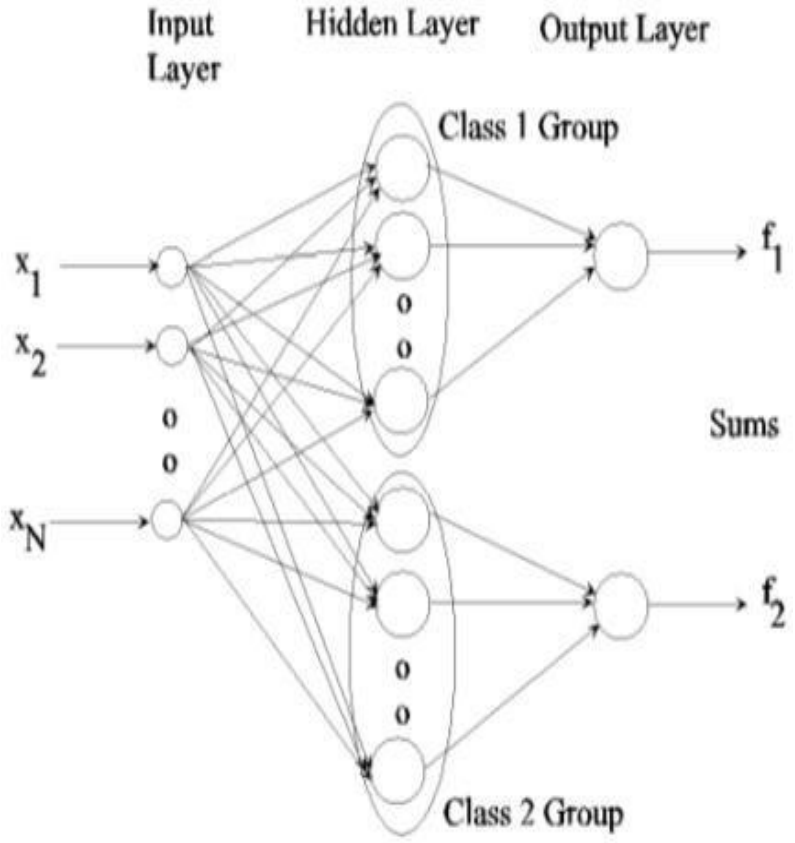


Figure 3: General Probabilistic Neural Network Architecture Diagram

2.4. CASCADE CORRELATION NEURAL NETWORKS [CCNN's]

- Cascade Co-relation network is a Supervised Feedforward type of Network
- Starts with a minimal network then automatically trains & adds more new Hidden nodes, thus creating a Multi-Layer Structure
- Cascade Correlation contains two parts: First Hidden units are added one at a time and after added they don't change
- Second is the Learning process where the new hidden units are created and installed
- For each new hidden unit CC tries to increase the magnitude of the correlation between the new unit's output and the residual error signal of the network
- We train only one-layer weights, the rest are maintained as constant so the results are cached
- Each unit sees the same inputs and error signals

The CCNN architecture is shown Figure 4.

Cascade correlation addresses both issues of slow rate of convergence and fixation of nodes while training by dynamically adding hidden units to the architecture-but only the minimum number necessary to achieve the specified error tolerance for the training set. Furthermore, a two-step weight-training process ensures that only one layer of weights is being trained at any time.

A cascade correlation net consists of input units, hidden units, and output units. Input units are connected directly to output units with adjustable weighted connections.

Connections from inputs to a hidden unit are trained when the hidden unit is added to the net and are then frozen. Connections from the hidden units to the output units are adjustable. Cascade correlation starts with a minimal network, consisting only of the required input and output units (and a bias input that is always equal to 1). This net is trained until no further improvement is obtained; the error for each output unit is then computed (summed over all training patterns).

Next, one hidden unit is added to the net in a two-step process. During the first step, a candidate unit is connected to each of the input units, but is not connected to the output units. The weights on the connections from the input units to the candidate unit are adjusted to maximize the correlation between the candidate's output and the residual error at the output units. The residual error is the difference between the target and the computed output, multiplied by the derivative of the output unit's activation function, i.e., the quantity that would be propagated back from the output units in the backpropagation algorithm. When this training is completed, the weights are frozen and the candidate unit becomes a hidden unit in the net.

The second step in which the new unit is added to the net now commences. The new hidden unit is connected to the output units, the weights on the connections being adjustable. Now all connections to the output units are trained. (The connections from the input units are trained again, and the new connections from the hidden unit are trained for the first time.)

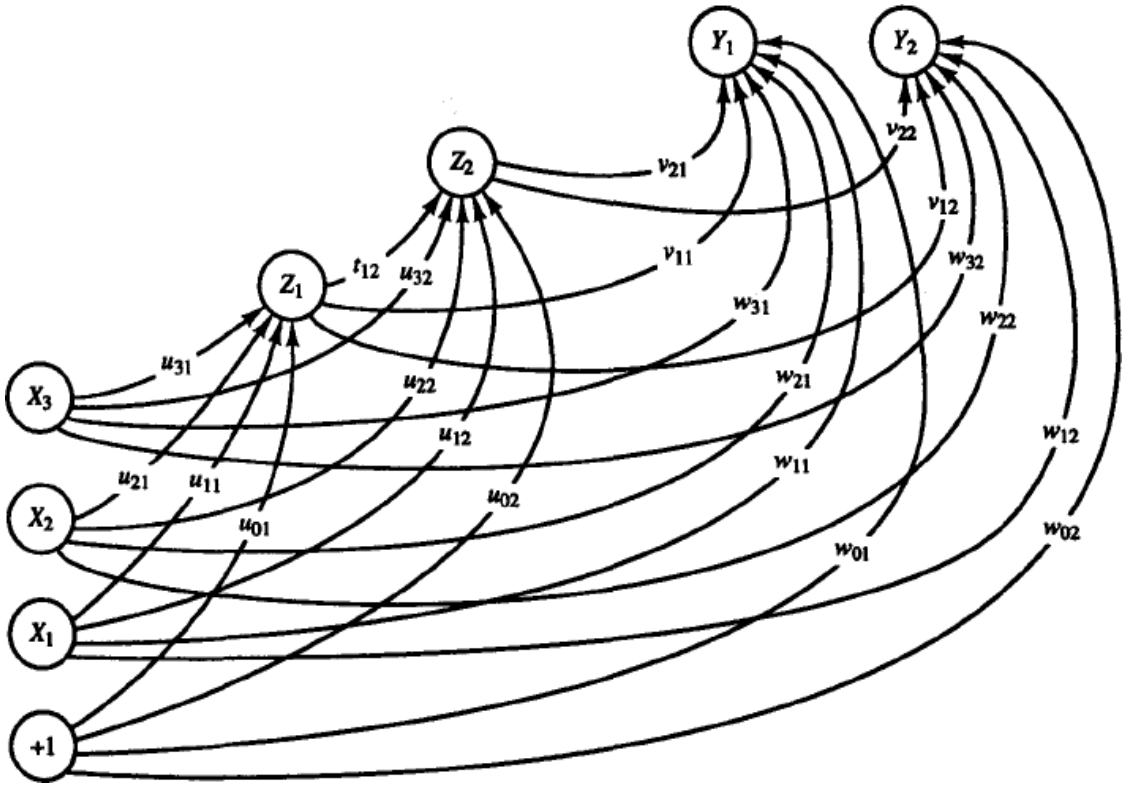


Figure 4. Schematic Representation of Cascade Corelation Network

2.4.1. Merits

- It learns at least 10 times faster than standard Back-propagation Algorithms.
- The network determines its own size and topologies.
- It is useful for incremental learning in which new information is added to the already trained network

2.4.2. Applications

- Recognition of different Geometric Shapes
- Vowel Recognition
- Cipher systems Identification
- Fault Classification problems

2.5. GENERAL REGRESSION NEURAL NETWORKS [GRNN's]

- General Regression Neural Networks [GRNN's] was proposed by D.F. Specht in 1991
- GRNN is a Single pass learning Network

- General Regression Neural Networks uses Gaussian Activation function for its Hidden Layers
- GRNN is based on Function Approximation or Function estimation procedures
- Output is estimated using weighted average of the outputs of training dataset, where the weight is calculated using the Euclidean distance between the training data and test data
- If the distance is large then the weight will be very less and if the distance is small more weight is given to the output
- Contains 4 layers: (1) Input layer (2) Hidden (pattern) Layer (3) Summation Layer (4) Output (division) Layer
- GRNN's Estimator is given by the equation

$$Y(x)=\sum Y_i e^{-(d_i^2/2\sigma^2)} / \sum e^{-(d_i^2/2\sigma^2)}$$

$$\text{Where, } d_i^2=(x-x_i)^T(x-x_i)$$

Where x = input

x_i = Training sample

$Y(x_i)$ = Output for sample I

d_i^2 = Euclidean Distance

$e^{-(d_i^2/2\sigma 2)}$ = Activation Function – This value is taken as weight value

σ = Spread constant (only Unknown parameter)

Select σ when MSE is Minimum

2.5.1. Training Procedure

Used to calculate optimum value of σ. First divide the samples into two parts. One part is used to train and the other is used to Test the network. Apply GRNN to Test data based on Training data & calculate MSE for different σ . Select the Minimum MSE and its Corresponding σ. The architecture Diagram of GRNN is given in Figure 5.

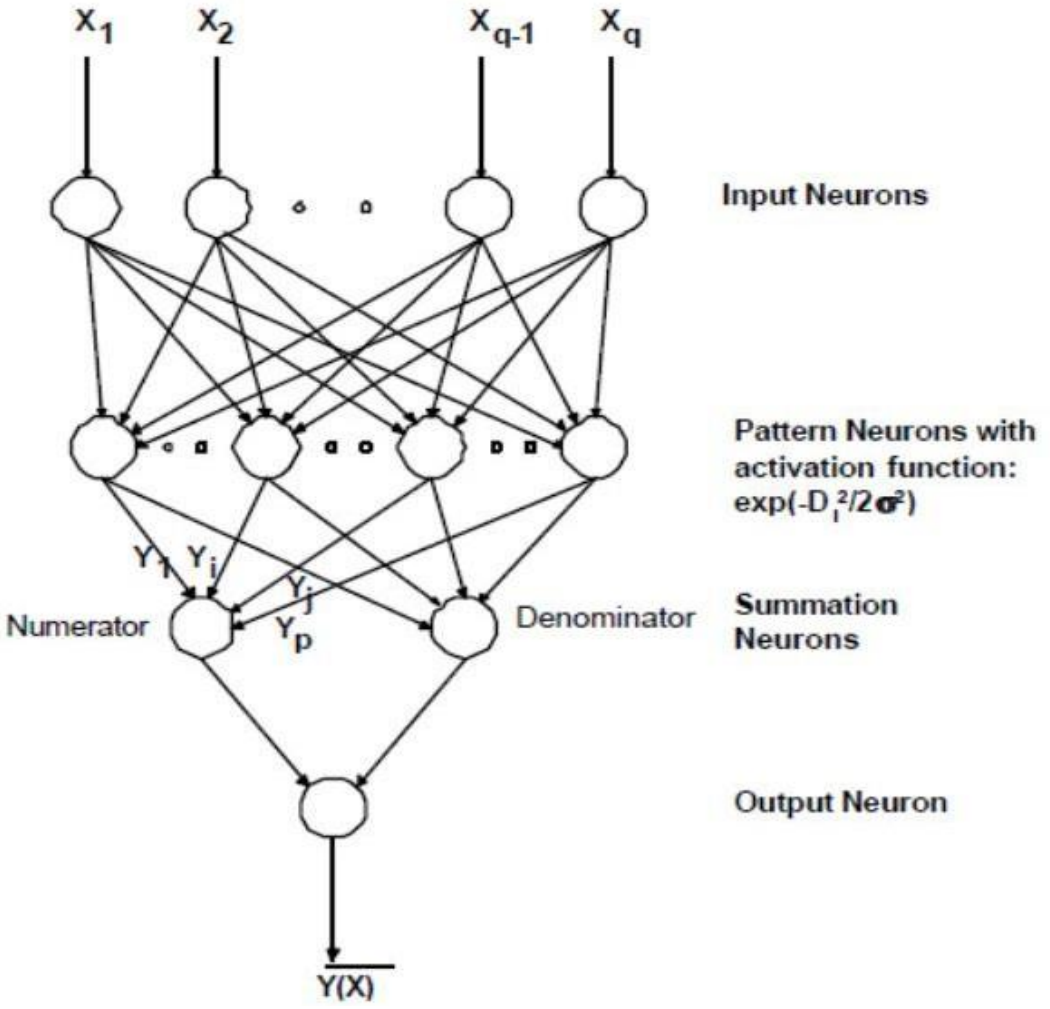


Figure 5. General Regression Neural Network

2.6. APPLICATIONS OF ANN

The major application are (1) Character Recognition system

(2) Texture Recognition System etc

2.6.1. Character Recognition System

Consider the characters given in figure 6. Now the objective is to recognise a particular alphabet, say 'A' in this example. Using Image analysis models the particular alphabet is segmented and converted into Intensity or Gray scale or Pixel values. The general work flow is shown in Figure 7. The first procedure is segmentation. Segmentation is the process of Subdividing the images into sub blocks. So alphabet "A" is isolated by using appropriate segmentation procedures like thresholding or region Growing or Edge detector based algorithms.

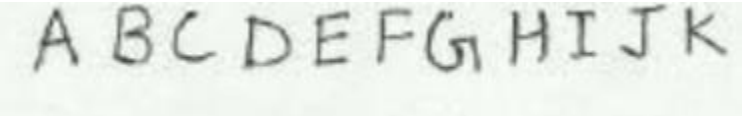


Figure 6. Input to Character recognition system

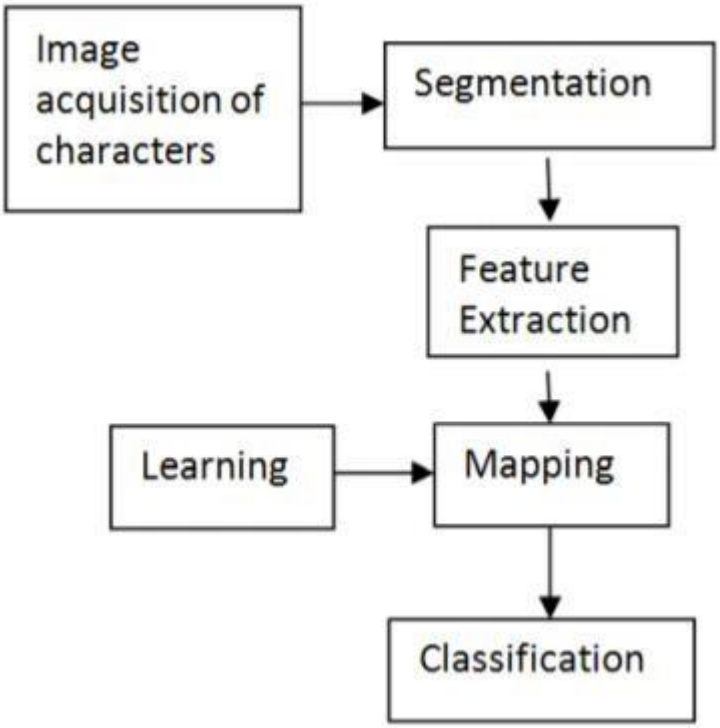


Figure 7. Work flow diagram for character recognition system

After implementing the segmentation procedure, we will obtain an output as shown in figure 8a. Now this image pattern has to be converted in terms of Binary values. The pattern is divided into different rows and columns as per the system resolution as shown in figure 8.b. Now for each square box values in the range of zero to 255 is provide if gray scale is used. These values represent whether the required object in present inside the square box or not. These Binary or Gray scale values are taken as input for further processing.

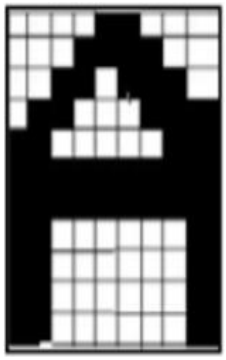


Figure 8a. Character Pattern values

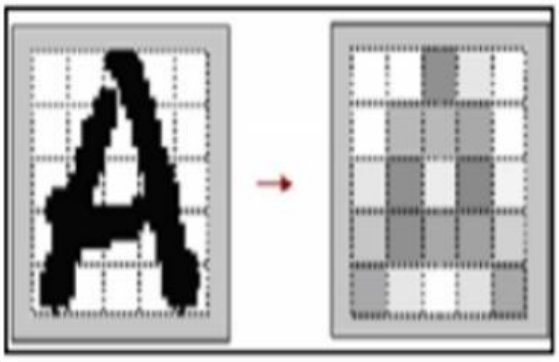
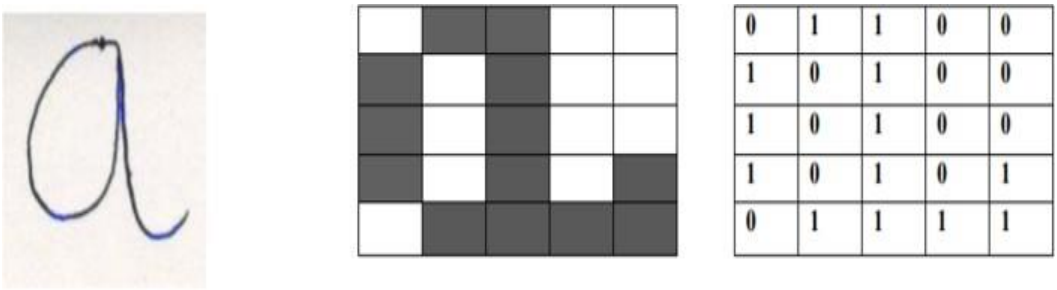


Figure 8b. Character Pattern conversion into intensity

Figures 6,7,8 are adapted from Praveen Kumar et al. (2012), "Character Recognition using Neural Network", vol3 ,issue 2., .Pp 978- 981, IJST

For figure 8.b Texture Features, Shape Features and or Boundary features etc can be extracted. This feature values are known as exemplars which is the actual input into the neural network. Consider any neural network. The input is the feature table created as explained in the above process, which is shown in Figure 9. This table is provided as in put to the neural system



0	1	1	0	0
1	0	1	0	0
1	0	1	0	0
1	0	1	0	1
0	1	1	1	1

Figure 9: Character 'a' is segmented and binary values extracted from it

Figure 9 Adopted from Yusuf Perwej et al. (2011), "Neural Networks for Handwritten English Alphabet Recognition", International Journal of Computer Applications (0975 – 8887) Volume 20– No.7, April 2011.

Figure 10 shows the full implementation using a multi-layer neural network

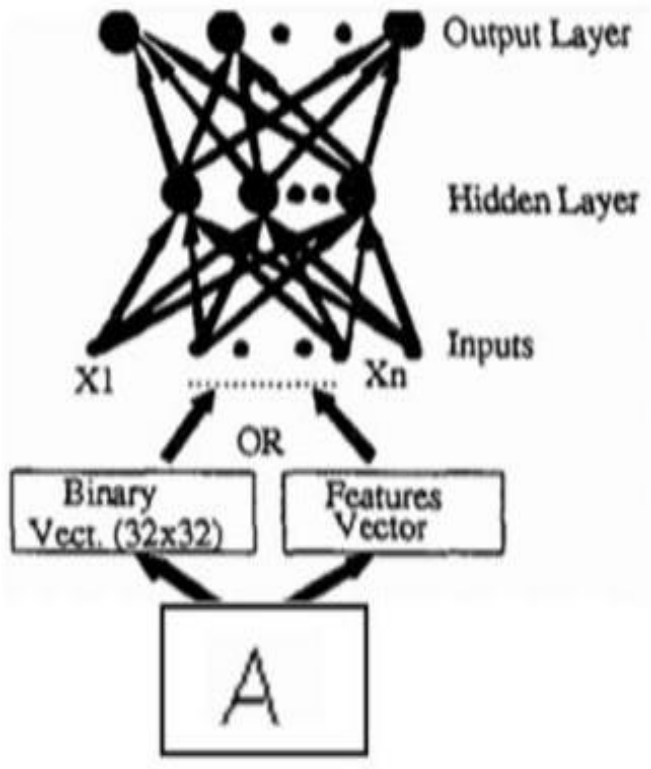


Figure 10. ANN implementation of character recognition system

Figure 10: Adopted from Anita pal et al. (2010), "Handwritten English Character Recognition Using Neural Network", International Journal of Computer Science & Communication, Vol. 1, No. 2, July-December 2010, pp. 141-144.

If the feature sets matches between the trained and current input features the output produces "1" , which denotes that the particular alphabet is trained else "0" not recognised.

Note: Similar procedure is used for texture classification Application

REFERENCE BOOKS

1. B. Yegnanarayana, "Artificial Neural Networks" Prentice Hall Publications.

2. Simon Haykin, "Artificial Neural Networks", Second Edition, Pearson Education.

3. Laurene Fausett, "Fundamentals of Neural Networks, Architectures, Algorithms and Applications", Prentice Hall publications.

4. James A. Freeman & Skapura, "Neural Networks", Pearson Education.

********** **ALL THE BEST** **********

UNIT – IV – INTRODUCTION TO FUZZY LOGIC

4. INTRODUCTION TO FUZZY LOGIC

UNIT IV- INTRODUCTION TO FUZZY LOGIC

Classical set - Operations and properties - Fuzzy Set - Operations and properties - Problems, Classical Relations - Operations and Properties, Fuzzy Relations - Operations and Properties - Compositions Membership function -FLCS - Need for FLC- Fuzzification - Defuzzification.

4.1 INTRODUCTION

The classical set theory is built on the fundamental concept of “set” of which an individual is either a member or not a member. A sharp, crisp, and unambiguous distinction exists between a member and a nonmember for any well-defined “set” of entities in this theory, and there is a very precise and clear boundary to indicate if an entity belongs to the set.

Namely, in the classical set theory, it is not allowed that an element is in a set and not in the set at the same time. Thus, many real-world application problems cannot be described and handled by the classical set theory, including all those involving elements with only partial membership of a set. On the contrary, fuzzy set theory accepts partial memberships, and, therefore, in a sense generalizes the classical set theory to some extent.

Fuzzy logic is an extension of Boolean logic by Lot Zadeh in 1965 based on the mathematical theory of fuzzy sets, which is a generalization of the classical set theory. By introducing the notion of degree in the verification of a condition, thus enabling a condition to be in a state other than true or false, fuzzy logic provides a very valuable flexibility for reasoning, which makes it possible to take into account inaccuracies and uncertainties. In order to introduce the concept of fuzzy sets, we first review the elementary set theory of classical mathematics. It will be seen that the fuzzy set theory is a very natural extension of the classical set theory, and is also a rigorous mathematical notion.

4.2 BASIC CONCEPTS OF FUZZY SETS

4.2.1 Fuzzy Logic

Fuzzy logic is defined as a Multivalued Logic with various degrees of values for its member elements. Fuzzy logic is based on "degrees of truth" than the (1 or 0) Boolean logic on which the modern computer is based.

4.2.2 Classical Sets

A classical set is defined by crisp boundaries; there is no uncertainty or vagueness in the prescription or location of the boundaries of the set.

4.2.2.1 Operations on Classical Sets

Let A and B be two subsets on the universe X. Operations are shown below

Union	$A \cup B = \{x \mid x \in A \text{ or } x \in B\}$	…. (1)
Intersection	$A \cap B = \{x \mid x \in A \text{ and } x \in B\}$	…. (2)
Complement	$A = \{x \mid x \notin A, x \in X\}$	…. (3)
Difference	$A \mid B = \{x \mid x \in A \text{ and } x \notin B\}$	…. (4)

4.2.2.2. Properties of Classical (Crisp) Sets

Commutativity $A \cup B = B \cup A$

$A \cap B = B \cap A.$ …….. (5)

Associativity $A \cup (B \cup C) = (A \cup B) \cup C$

$A \cap (B \cap C) = (A \cap B) \cap C$ …….. (6)

Distributivity $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$

$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$ …….. (7)

Idempotency $A \cup A = A$

$A \cap A = A$ …….. (8)

Identity $A \cup \emptyset = A$

$A \cap X = A$

$A \cap \emptyset = \emptyset.$ …….. (9)

$A \cup X = X.$

Transitivity If $A \subseteq B$ and $B \subseteq C$, then $A \subseteq C$ …… (10)

Involution $A'' = A$ …… (11)

4.2.3 Fuzzy Sets

A fuzzy set is a set with a smooth boundary. Fuzzy logic is based on the theory of fuzzy sets, which is a generalization of the classical set theory. Saying that the theory of fuzzy sets is a generalization of the classical set theory means that the latter is a special case of fuzzy sets theory. To make a metaphor in set theory speaking, the classical set theory is a subset of the theory of fuzzy sets,

A fuzzy set, is defined as a set containing elements that have varying degrees of membership values in the range of zero to one.

4.2.3.1 Fuzzy Set Operations

The following are commonly used fuzzy set operations

Union $$\mu_{\underset{\sim}{A}\cup\underset{\sim}{B}}(x) = \mu_{\underset{\sim}{A}}(x) \vee \mu_{\underset{\sim}{B}}(x). \quad \ldots.(15)$$

Intersection $$\mu_{\underset{\sim}{A}\cap\underset{\sim}{B}}(x) = \mu_{\underset{\sim}{A}}(x) \wedge \mu_{\underset{\sim}{B}}(x). \quad \ldots.(16)$$

Complement $$\mu_{\overline{\underset{\sim}{A}}}(x) = 1 - \mu_{\underset{\sim}{A}}(x). \quad \ldots.(17)$$

Fuzzy logic is based on fuzzy set theory, which is a generalization of the classical set theory. The classical sets are also called clear sets, classical logic is also known as Boolean logic or binary.

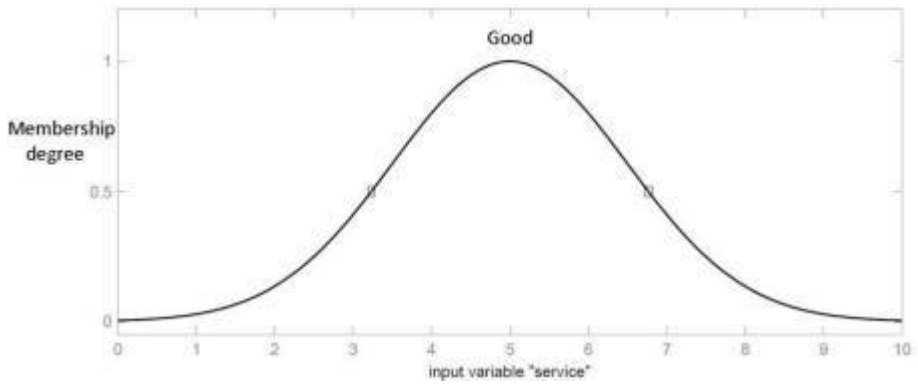


Figure 1: Membership curve

Figure 1 gives the Membership function characterizing the subset of 'good' quality of service. Let us consider X be a set. A fuzzy subset A of X is characterized by a membership function. Let Input 1is quality of service. Subsets: poor, good and excellent. Input 2 is quality of food. Subsets: awful and delicious. Consider the output tip amount. Subsets: low, medium

and high. Refer figure 3. For ‘Good’ a value of ‘1’ is taken and for ‘Poor’ a value of ‘0’ is taken. This is how membership functions converts a classical value into Fuzzy value.

The shape of the membership function is chosen arbitrarily by following the advice of the expert or by statistical studies: sigmoid, hyperbolic, tangent, exponential, Gaussian or any other form can be used.

A fuzzy set is defined by a function that maps objects in a domain of concern into their membership value in a set. Such a function is called the *membership function.*

A fuzzy set, then, is a set containing elements that have varying degrees of membership in the set. This idea is in contrast with classical, or crisp, sets because members of a crisp set would not be members unless their membership is full, or complete, in that set (i.e., their membership is assigned a value of 1). Elements in a fuzzy set, because their membership need not be complete, can also be members of other fuzzy sets on the same universe.

4.3 DIFFERENCE BETWEEN CLASSICAL SETS AND FUZZY SET

4.3.1 Classical set

A classical set is defined by crisp boundaries. there is no uncertainty in the prescription or location of the boundaries of the set. Refer figure 2.

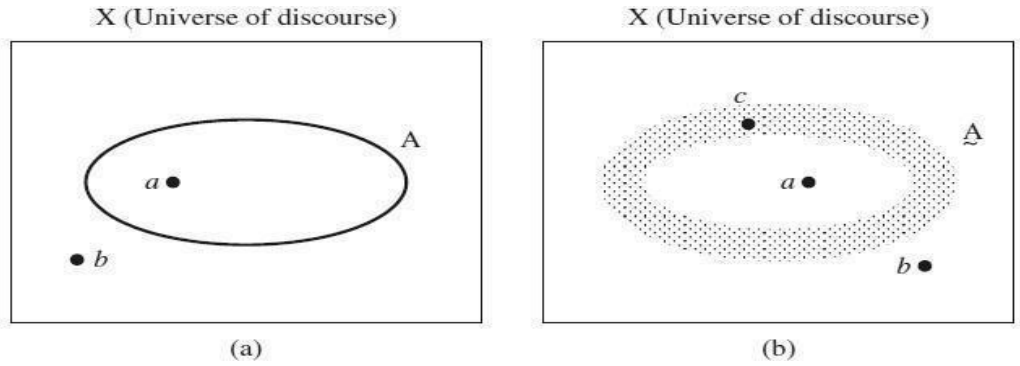


Figure 2: Diagrams for (a) crisp set boundary and (b) fuzzy set boundary

Examples

The clock speeds of computer CPUs

The operating currents of an electronic motor

The operating temperature of a heat pump (in degrees Celsius)

The total number of elements in a universe X is called its cardinal number, denoted nx, where x is a label for individual elements in the universe.

Collections of elements within a universe are called sets, and collections of elements within sets are called subsets. We define the null set, $\emptyset$, as the set containing no elements, and the whole set, X, as the set of all elements in the universe.

4.3.2 Fuzzy Set

A fuzzy set is prescribed by vague or ambiguous properties; hence its boundaries are ambiguously specified. Fuzzy set theory permits the gradual assessment of the membership of elements in a set, described with the aid of a membership function valued in the real unit [0,1].

Examples

Words like young, tall, good or high are fuzzy. There is no single quantitative value which defines the young term. For some people, age 25 is young, and for others, age 35 is young. The concept young have no boundary. Refer figure 3. The linguistic terms middle age, old age, etc represents the member element AGE.

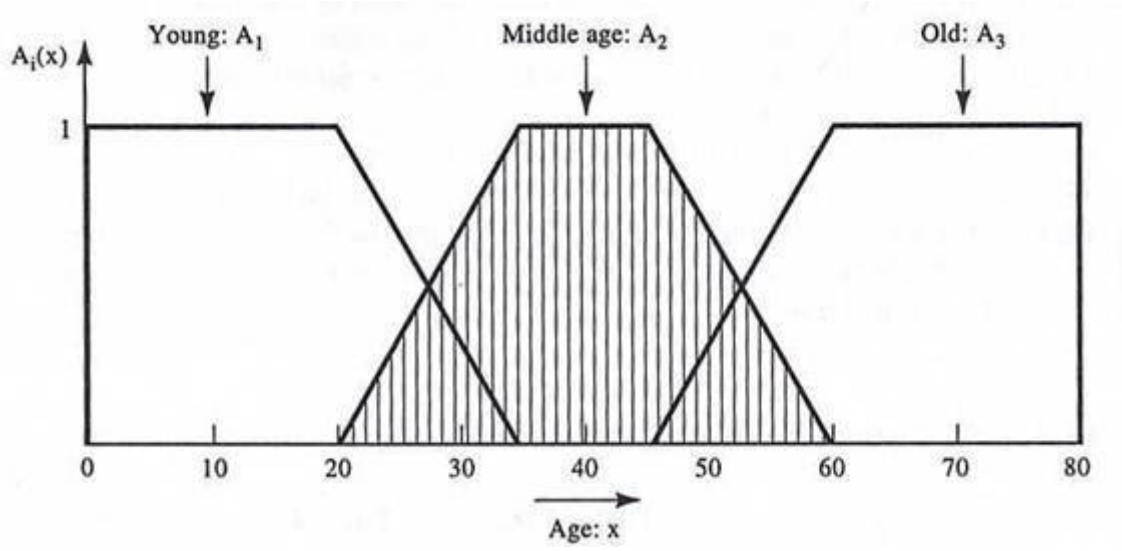


Figure 3: Fuzzy set represented in linguistic terms

4.3.2.1 Properties of Fuzzy set

Two special properties of set operations are known as the excluded middle axioms and De Morgan's principles. These properties are enumerated here for two sets A and B. The excluded middle axioms are very important because these are the only set operations described here that are not valid for both classical sets and fuzzy sets. There are two excluded middle

axioms. The first, called the axiom of the excluded middle, deals with the union of a set A and its complement; the second, called the axiom of contradiction, represents the intersection of a set A and its complement.

All Properties of classical sets also hold for fuzzy sets, except for the excluded middle and contradiction axioms. Fuzzy sets can overlap. A set and its complement can overlap. The excluded middle axioms, extended for fuzzy sets, are expressed as

$$\underset{\sim}{A} \cup \overline{\underset{\sim}{A}} \neq X.$$

$$\underset{\sim}{A} \cap \overline{\underset{\sim}{A}} \neq \emptyset.$$

Venn diagrams comparing the excluded middle axioms for classical (crisp) sets and fuzzy sets are shown in the below Figure 4.

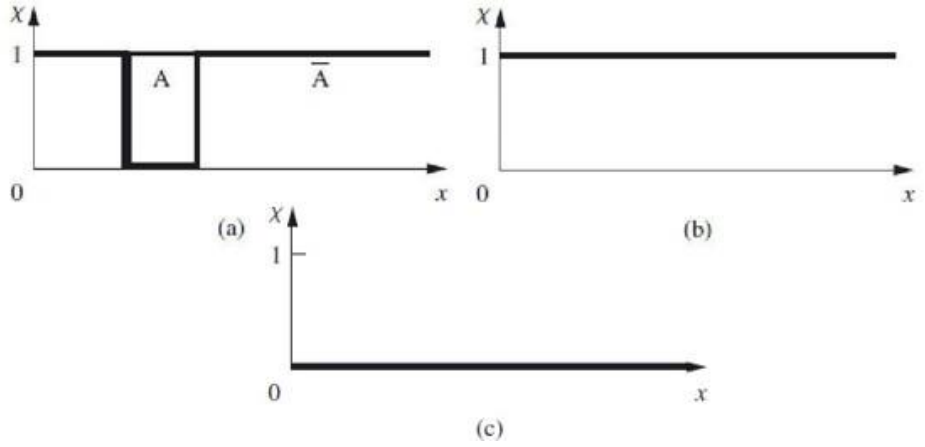


Figure 4(a) Crisp set A and its complement; (b) Fuzzy A∪A' ; and (c) crisp A ∩ A' = ϕ

All other operations on classical sets also hold for fuzzy sets, except for the excluded middle axioms. These two axioms do not hold for fuzzy sets since they do not form part of the basic axiomatic structure of fuzzy sets. Since fuzzy sets can overlap, a set and its complement can also overlap. The excluded middle axioms, extended for fuzzy sets, are expressed as:

$$\underset{\sim}{A} \cup \overline{\underset{\sim}{A}} \neq X.$$

$$\underset{\sim}{A} \cap \overline{\underset{\sim}{A}} \neq \emptyset.$$

4.4 FUZZY LOGIC CONTROL PRINCIPLES (FLCS)

Control systems abound in our everyday life; perhaps we do not see them as such, because some of them are larger than what a single individual can deal with, but they are

ubiquitous. For example, economic systems are large, global systems that can be controlled; ecosystems are large, amorphous, and long-term systems that can be controlled.

The general form of a closed-loop control system is illustrated in Figure 5.

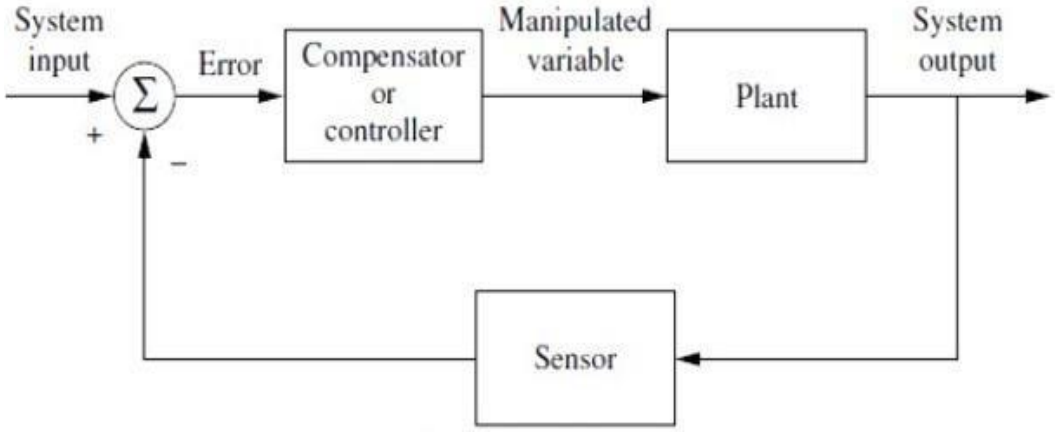


Figure 5: General Closed Loop Control System

Control systems are sometimes divided into two classes. If the objective of the control system is to maintain a physical variable at some constant value in the presence of disturbances, the system is called a regulatory type of control, or a regulator. The second class of control systems is set point tracking controllers. In this scheme of control, a physical variable is required to follow or track some desired time function. An example of this type of system is an automatic aircraft landing system, in which the aircraft follows a "ramp" to the desired touchdown point.

4.4.1 Assumptions involved in Fuzzy Control System

A number of assumptions are implicit in a fuzzy control system design. Six basic assumptions are commonly made whenever a fuzzy rule-based control policy is selected.

The plant is observable and controllable: state, input, and output variables are usually available for observation and measurement or computation. There exists a body of knowledge comprising a set of linguistic rules, engineering common sense, intuition, or a set of input–output measurements data from which rules can be extracted. A solution exists. The control engineer is looking for a "good enough" solution, not necessarily the optimum one. The controller will be designed within an acceptable range of precision. The problems of stability and optimality are not addressed explicitly; such issues are still open problems in fuzzy controller design.

Control system is a set of hardware component which regulates or alters or modifies the behavior of the system. Fuzzy control system uses approximation so that the nonlinearity, data or knowledge incompleteness is reduced. The General Block Diagram is shown in Figure 6.

To design a FLCS we must take into consideration the following points:

- ✓ The plant is observable and controllable
- ✓ There exists a set of knowledge about that process from which the rules can be framed
- ✓ There exists a solution
- ✓ The control engineer is looking for "Good enough solution" and not an optimum one
- ✓ Controller can be designed within an acceptable range of precision

4.4.2 Steps Involved in Designing a FLCS

- ✓ Identify the variables
- ✓ By assigning appropriate membership function convert these parameters in fuzzy sub sets.
- ✓ Assign fuzzy relationship to input states and fuzzy output states
- ✓ Use fuzzy approximate reasoning to infer the outcomes
- ✓ Aggregate the outcomes recommended by each rule
- ✓ Apply Defuzzification to form a crisp output

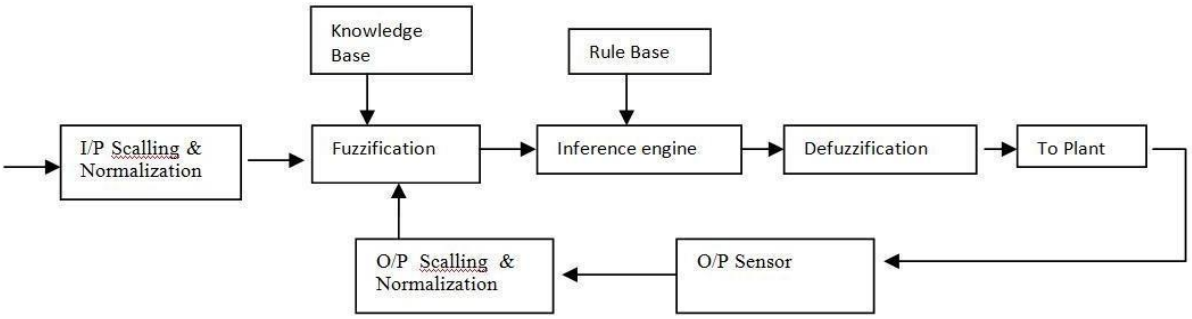


Figure 6: General Fuzzy logic based Control System

4.5 FUZZY RELATIONS

Fuzzy relations is used to map elements of one universe, say X, to those of another universe, say Y, with the help of Cartesian product. The "strength" of the relation measured

with a membership function having "degrees" of strength of the relation on the unit interval [0,1]. Hence, a fuzzy relation $\underset{\sim}{R}$ is a mapping from the Cartesian space X × Y to the interval [0,1], where the strength of the mapping is expressed by the membership function $\mu_{\underset{\sim}{R}}(x, y)$.

Cardinality of Fuzzy Relations

Cardinality of fuzzy sets is infinity; the cardinality of a fuzzy relation between two or more universes is also infinity.

4.5.1 Operations of Fuzzy Relations

Let $\underset{\sim}{R}$ and $\underset{\sim}{S}$ be fuzzy relations on the Cartesian space X × Y. Then the following operations apply for the membership values for various set operations

$$\underset{\sim}{R} \subset \underset{\sim}{S} \Rightarrow \mu_{\underset{\sim}{R}}(x, y) \leq \mu_{\underset{\sim}{S}}(x, y).$$

Let R and S be fuzzy relations on the Cartesian space X × Y. Then the following operations apply for the membership values for various set operations (these are similar to the same operations on crisp sets

Union $\mu_{\underset{\sim}{R} \cup \underset{\sim}{S}}(x, y) = \max(\mu_{\underset{\sim}{R}}(x, y), \mu_{\underset{\sim}{S}}(x, y))$.

Intersection $\mu_{\underset{\sim}{R} \cap \underset{\sim}{S}}(x, y) = \min(\mu_{\underset{\sim}{R}}(x, y), \mu_{\underset{\sim}{S}}(x, y))$.

Complement $\mu_{\overline{\underset{\sim}{R}}}(x, y) = 1 - \mu_{\underset{\sim}{R}}(x, y)$.

Containment $\underset{\sim}{R} \subset \underset{\sim}{S} \Rightarrow \mu_{\underset{\sim}{R}}(x, y) \leq \mu_{\underset{\sim}{S}}(x, y)$.

4.5.2 Properties of Fuzzy Relations

Just as for crisp relations, the properties of commutativity, associativity, distributivity, involution, and idempotency all hold for fuzzy relations. Moreover, De Morgan's principles hold for fuzzy relations just as they do for crisp (classical) relations, and the null relation, O, and the complete relation, E, are analogous to the null set and the whole set in set-theoretic form, respectively. Fuzzy relations are not constrained, as is the case for fuzzy sets in general, by the excluded middle axioms. Since a fuzzy relation R also a fuzzy set, there is overlap between a relation and its complement; hence,

The excluded middle axioms for fuzzy relations do not result, in general, in the null relation, O, or the complete relation, E.

$$\underset{\sim}{R} \cup \overline{\underset{\sim}{R}} \neq E. \qquad \text{.....(24)}$$

$$\underset{\sim}{R} \cap \overline{\underset{\sim}{R}} \neq \mathbf{O}. \qquad \text{.....(25)}$$

From the above equations, the excluded middle axioms for fuzzy relations do not result, in general, in the null relation, O, or the complete relation, E.

4.6 FUZZY CARTESIAN PRODUCT AND COMPOSITION

Let $\underset{\sim}{A}$ be a fuzzy set on universe X and $\underset{\sim}{B}$ be a fuzzy set on universe Y, then the Cartesian product between fuzzy sets $\underset{\sim}{A}$ and $\underset{\sim}{B}$ will result in fuzzy relation $\underset{\sim}{R}$, which is given as

$$\underset{\sim}{A} \times \underset{\sim}{B} = \underset{\sim}{R} \subset X \times Y, \qquad \text{.....(26)}$$

Where the fuzzy relation $\underset{\sim}{R}$ has membership function

$$\mu_{\underset{\sim}{R}}(x, y) = \mu_{\underset{\sim}{A} \times \underset{\sim}{B}}(x, y) = \min(\mu_{\underset{\sim}{A}}(x), \mu_{\underset{\sim}{B}}(y)). \qquad \text{.....(27)}$$

The Cartesian product defined by $\underset{\sim}{A} \times \underset{\sim}{B} = \underset{\sim}{R}$ is implemented in the same way as the cross product of two vectors. Cartesian product is not the same as the arithmetic product. Cartesian product employs the idea of pairing of elements among sets. For example, for a fuzzy set

A has four elements, for a fuzzy set B that has five elements, then the resulting fuzzy relation R will be represented by a matrix of size 4 × 5, that is, R will have four rows and five columns.

Fuzzy composition can be defined as of crisp relations. Suppose $\underset{\sim}{R}$ is a fuzzy relation on the Cartesian space X × Y, $\underset{\sim}{S}$ is a fuzzy relation on Y × Z, and $\underset{\sim}{T}$ is a fuzzy relation on X × Z, then fuzzy max–min composition is defined in the following manner:

$$\underset{\sim}{T} = \underset{\sim}{R} \circ \underset{\sim}{S}, \qquad \text{.....(28)}$$

$$\mu_{\underset{\sim}{T}}(x, z) = \bigvee_{y \in Y} (\mu_{\underset{\sim}{R}}(x, y) \wedge \mu_{\underset{\sim}{S}}(y, z)), \qquad \text{.....(29)}$$

and fuzzy max–product composition is defined in terms of the membership function theoretic notation as

$$\mu_{\underset{\sim}{T}}(x, z) = \bigvee_{y \in Y} (\mu_{\underset{\sim}{R}}(x, y) \bullet \mu_{\underset{\sim}{S}}(y, z)). \qquad \text{.....(30)}$$

It should be noted out that neither crisp nor fuzzy compositions are commutative in general so

$$\underset{\sim}{R} \circ \underset{\sim}{S} \neq \underset{\sim}{S} \circ \underset{\sim}{R}.$$

Different types of composition are (1) MAX-MIN (2) MAX –PRODUCT (3) MAX-MAX (4) MIN- MIN (5) MIN-MAX etc., Compositions provides more information which reduces the impreciseness present in the problem.

4.7 FUZZIFICATION

Process of converting a crisp value into fuzzy. Example if we have a variable TEMPERATURE = 35 ·C then this is converted into MAXTEMP, MINTEMP etc in the range of Zero to one by assigning membership functions.

4.7.1 Fuzzification

- ✓ Inference
- ✓ Intuition
- ✓ Rank ordering
- ✓ Using GA
- ✓ Using ANN
- ✓ Inductive reasoning
- ✓ Meta rules
- ✓ Fuzzy statistics

These are some methods used to generate membership values and there by used to convert a crisp value into fuzzy.

4.8 FUZZY RULES

In a FLS, a rule base is constructed to control the output variable. A fuzzy rule based system consists of simple IF-THE with a condition and a conclusion. Sample fuzzy rule for an air conditioner system is given below.

IF Temp = Too Cold THEN Command is HEAT

Table 1 shows the matrix representation of the fuzzy rules for the above said FLS. Row contains the values that current room temperature can take, column is the values for target

temperature, and each cell is the resulting command. For instance, if temperature is cold and target is warm then command is heat.

temperature/target	too-cold	cold	warm	hot	too-hot
too-cold	no-change	heat	heat	heat	heat
cold	cool	no-change	heat	heat	heat
warm	cool	cool	no-change	heat	heat
hot	cool	cool	cool	no-change	heat
too-hot	cool	cool	cool	cool	no-change

Table 1: Fuzzy matrix example

A fuzzy rule is defined as a conditional statement in the form:

IF *x is A* THEN *y is B*

where x and y are linguistic variables; A and B are linguistic values determined by fuzzy sets on the universe of discourse X and Y, respectively.

In the field of artificial intelligence (machine intelligence), there are various ways to represent knowledge. Perhaps the most common way to represent human knowledge is to form it into natural language expressions of the type IF premise (antecedent), THEN conclusion (consequent). The form is commonly referred to as the IF–THEN rule-based form; this form is generally referred to as the deductive form.

It typically expresses an inference such that if we know a fact (premise, hypothesis, antecedent), then we can infer, or derive, another fact called a conclusion (consequent).This form of knowledge representation, characterized as shallow knowledge, is quite appropriate in the context of linguistics because it expresses human empirical and heuristic knowledge in our own language of communication.

It does not, however, capture the deeper forms of knowledge usually associated with intuition, structure, function, and behavior of the objects around us simply because these latter forms of knowledge are not readily reduced to linguistic phrases or representations; this deeper form, is referred to as inductive.

The fuzzy rule-based system is most useful in modeling some complex systems that can be observed by humans because they make use of linguistic variables as their antecedents and consequents; as described here these linguistic variables can be naturally represented by fuzzy sets and logical connectives of these sets.

4.9 DEFUZZIFICATION

Defuzzification is the process of producing a quantifiable value. Fuzzy values can't be given to machines since they understand only two valued logic. Hence these linguistic values have to be converted into machine understandable two valued logic. Those techniques used to convert fuzzy into classical values are known as Defuzzification methods. It is the process of conversion of a fuzzy quantity into crisp quantity. Various Defuzzification methods are listed as:

- ✓ Centroid method
- ✓ Weighted average method
- ✓ Mean-max membership method
- ✓ Centre of sums method
- ✓ Centre of largest area method
- ✓ First (or last) of maxima method etc

Any of the above method can be used based on the level of intelligent control required.

REFERENCE BOOKS

1. Timothy Ross, "Fuzzy Logic with Engineering Application", McGraw Hill, Edition 1997.
2. Drainkov, H.Hallendoor and M.Reinfrank, "An Introduction to Fuzzy Control", Edition 2001..
3. Fuzzy Sets and Fuzzy Logic: Theory and Applications byGeorge J. Klir, Bo Yuan, Prentice Hall, 1995.
4. Soft Computing: Fundamentals and Applications by D.K.Pratihar, Narosa Publishing House, New-Delhi, 2014.

********** **ALL THE BEST** **********

UNIT – V– FLCS, CLASSIFICATION & APPLICATIONS

UNIT 5 FLCS, CLASSIFICATION & APPLICATIONS

Fuzzy decision making -Types, Fuzzy Rule Based System, Knowledge Based System, Non-linear Fuzzy Control system - Fuzzy Classification - Hard C Means - Fuzzy C Means. Applications of fuzzy - Water level controller, Fuzzy image Classification, Speed control of motor.

5.1 KNOWLEDGE BASE

Place in which different information related to the problem are stored here. It's the collection of expert knowledge about the problem. Advantages of fuzzy knowledge base.

- Comprehensibility
- Parsimony
- Modularity
- Uncertainty
- Parallelism
- Robust

5.2 RULE BASE SYSTEM

Collection of rules representing different environment is called Rule base system. Rules are framed by assigning relationship to fuzzy linguistic variables. In a FLS, a rule base is constructed to control the output variable. A fuzzy rule-based system consists of simple IF-THEN with a condition and a conclusion. Sample fuzzy rule for an air conditioner system is given below.

IF Temp = Too Cold THEN Command is HEAT

These IF-THEN rules are the base for fuzzy reasoning. If-THEN rules are of different types: They are as follows.

(1) Single rule with single antecedent

(2) Single rule with multiple antecedent

(3) Multiple with multiple antecedent

Table 1 shows the matrix representation of the fuzzy rules for the above said FLS. Rows contains the values that current room temperature can take, columns are the values for target temperature, and each cell is the resulting command. For instance, if temperature is cold and target is warm then command is heat.

temperature/target	too-cold	cold	warm	hot	too-hot
too-cold	no-change	heat	heat	heat	heat
cold	cool	no-change	heat	heat	heat
warm	cool	cool	no-change	heat	heat
hot	cool	cool	cool	no-change	heat
too-hot	cool	cool	cool	cool	no-change

Table 1: Fuzzy rule matrix example for Temperature control problem

5.3 DECISION MAKING SYSTEM

This block responsible for fuzzy control outcomes. Making decisions under uncertainty is tough since, we have to handle bulk information's. Different classifications in this category are,

- ✓ Single person single objective decision making
- ✓ Multi person single objective decision making
- ✓ Multi person single objective decision making
- ✓ Multi person Multi objective decision making

Decision making in FLCS is based on:

- ✓ **Synthetic Evaluation** - Individual evaluation based on the expert's knowledge.
- ✓ **Rank ordering** - Based on ranks of the rules. Rules are prioritized.
- ✓ **Preferences and consensus**- when multi persons are involved in DM, rules are selected based on individual preferences and degree of consensus.
- ✓ **Fuzzy multi objective based on BAYESIAN Method** – this method is based on Bayesian technique in which we calculate utility function for every option of selection.

5.4 APPLICATIONS OF FUZZY LOGIC CONTROL SYSTEM

Most control situations are more complex than we can deal with mathematically. In this situation, fuzzy control can be developed, provided a body of knowledge about the control process exists, and formed into a number of fuzzy rules. A simple FLCS is shown in figure 1.For example, suppose an industrial process output is given in terms of the pressure. We can calculate the difference between the desired pressure and the output pressure, called the pressure error (e), and we can calculate the difference between the desired rate of change of the pressure, dp/dt, and the actual pressure rate, called the pressure error rate, (˙e). Also, assume that knowledge can be expressed in the form of IF–THEN rules such as:

IF pressure error (e) is “positive big (PB)” or “positive medium (PM)” and

IF pressure error rate (˙e) is “negative small (NS),” THEN heat input change is “negative medium (NM).”

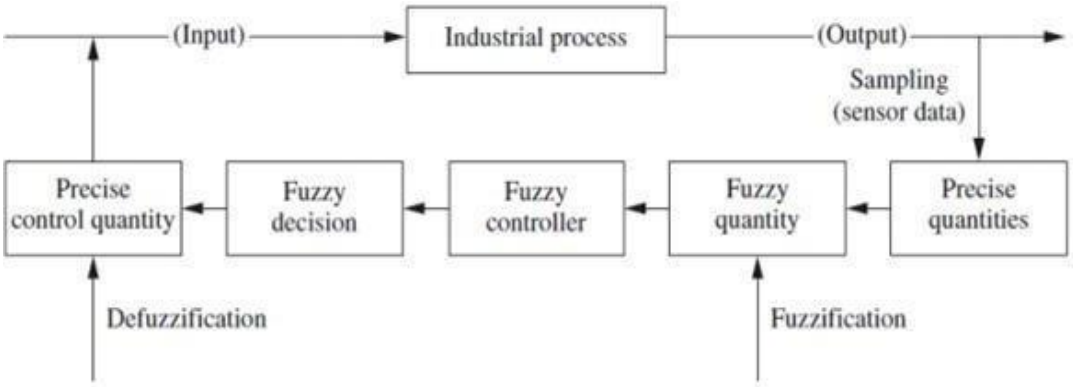


Figure (1) :Fuzzy Controller Block Diagram

The linguistic variables defining the pressure error, “PB” and “PM,” and the pressure error rate, “NS” and “NM,” are fuzzy, but the measurements of both the pressure and pressure rate as well as the control value for the heat (the control variable) ultimately applied to the system are precise (crisp). An input to the industrial process (physical system) comes from the controller. The physical system responds with an output, which is sampled and measured by some device. If the measured output is a crisp quantity, it can be fuzzified into a fuzzy set .This fuzzy output is then considered as the fuzzy input into a fuzzy controller, which consists of linguistic rules.

The output of the fuzzy controller is then another series of fuzzy sets. Since most physical systems cannot interpret fuzzy commands (fuzzy sets), the fuzzy controller output must be converted into crisp quantities using defuzzification methods. These crisp (de-fuzzified) control-output values then become the input values to the physical system and the entire closed-loop cycle is repeated device. If the measured output is a crisp quantity, it can be fuzzified into a fuzzy set.

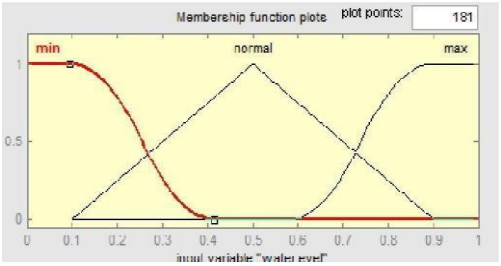
This fuzzy output is then considered as the fuzzy input into a fuzzy controller, which consists of linguistic rules. The output of the fuzzy controller is then another series of fuzzy sets. Since most physical systems cannot interpret fuzzy commands (fuzzy sets), the fuzzy controller output must be converted into crisp quantities using defuzzification methods These crisp (De- fuzzified) control-output values then become the input values to the physical system and the entire closed-loop cycle is repeated.

5.4.1 Fuzzy Logic Based Water Level Controller

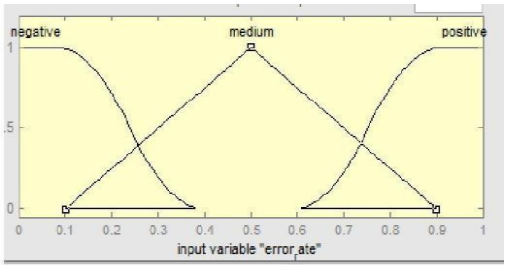
Step 1: Identify the i/p and o/p variables. Here in this case minimum and maximum level are inputs and valve position is output variables.

Step 2: Assign an appropriate membership function and perform Fuzzification process. i/p1 is water level, i/p2 is error rate. Valve position is the output being controlled.

The membership graphs for i/p and o/p variables are given in Figure (2) a, b and c. For water level three functions and for error rate again three membership functions are used.

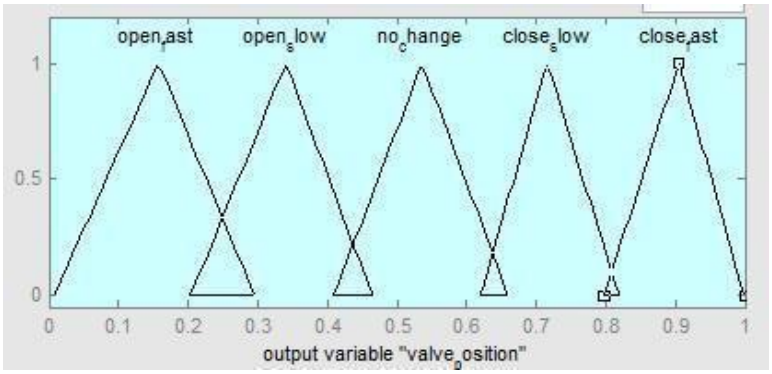


Fig(2a) I/p membership function for water level



Fig(2b)I/p membership function for error rate

For output variable Valve position, we consider open slow, open fast, close slow, close fast and no change as linguistic variables, which is shown in Figure c.



Fig(2c) I/p membership function for output variable valve position

* These responses are generated using MATLAB software.

Step 3: Framing of rules are done as follows

Rule 1: IF water level is min AND error rate is negative THEN valve position open fast
Rule 2: IF water level is max AND error rate is positive THEN valve position close fast
Similarly, rules are framed for remaining conditions and these rule outcomes are aggregated. Final outcome is the defuzzied value of this aggregated value. Based on this value only the value opens or closes.

5.4.2 Fuzzy Logic Based Image Classification

Fuzzy logic addresses the vagueness and ambiguity which is present in an image. Fuzzy Logic is a powerful tool that denotes and analyses human knowledge in form of fuzzy rules. The following Figure 3 represents a general Fuzzy based Image analysis procedure. In this block diagram the input image is first converted into a fuzzy image. Here the image intensities are fuzzified and converted into fuzzy values by assigning membership values for the intensities present in the image. Figure represent a general procedure of Fuzzy based image analysis procedure. Then based on an expert knowledge rules are created as given below.

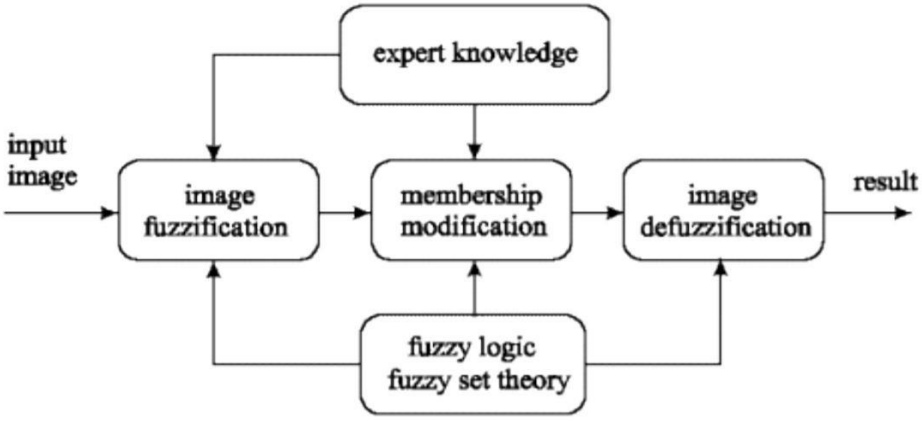


Figure 3: Fuzzy Image Processing System Block Diagram

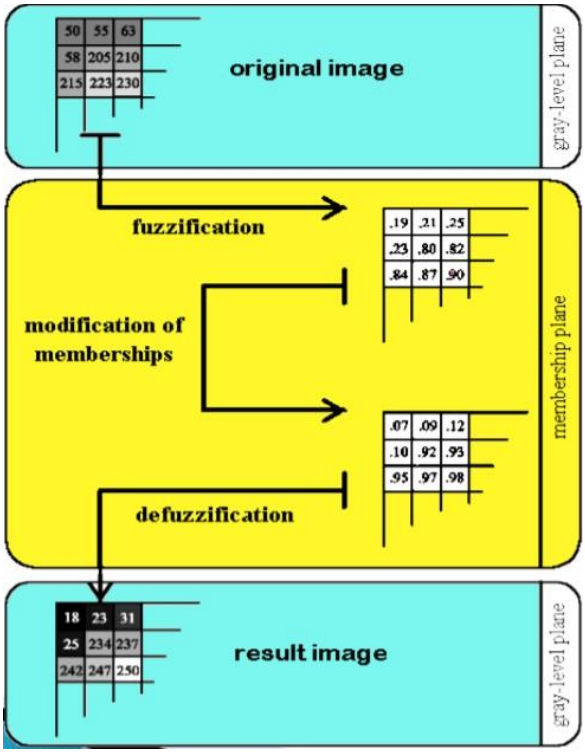


Figure 4: Image Fuzzification and Defuzzification procedures

Figure 3 &4 are adopted from: http://imageprocessingplace.com /downloads_V3/root downloads /tutorials/ fuzzy_image_processing.pdf

Figure 4 shows how the original image pixels are converted into fuzzy values. After obtaining a resultant image (fuzzified intensities) the other image processing procedure like Enhancement, Segmentation, Object recognition, Clustering or Classification process are implemented. The general image analysis procedures can be implemented to get the Region Of Interest (ROI).

5.4.3 Fuzzy Logic Based Motor Speed Control

As the basic rule for designing any fuzzy based application we have to identify the input and output variables. In this example let us consider Voltage and Error Rate as inputs and Speed as output variable. The various components present in a DC motor is shown in Figure 5. DC motors are used in various applications like Electric trains, Cranes, Trolly's, Rolling mills, Robotic Manipulators etc.

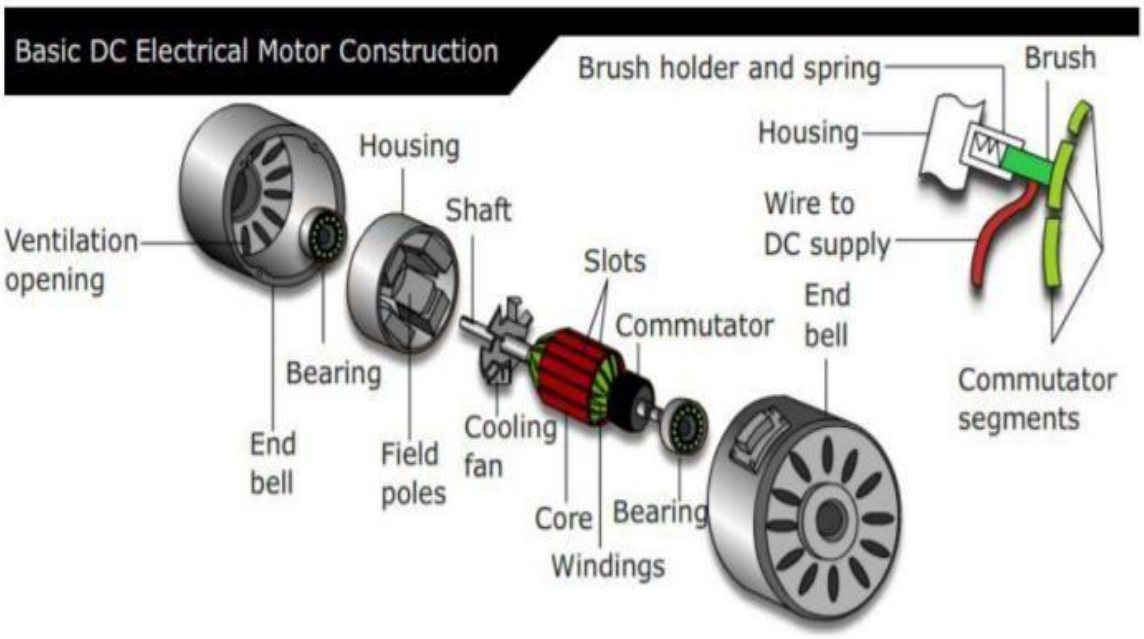


Figure : 5 Components of DC Motor

Consider the input variable Voltage. By applying Triangular Membership functions the crisp voltage values are converted into fuzzy as shown in Figure 6.Similarly Figure 7 shows the variable Error rate with three Triangular membership functions.For output variable Speed ,three membership functions namely Less, Average speed and Large speed is considered as shown in Figure 8. All these function values are normalised between the range 'o' to '1'. Figure

9 shows the Rule based system generated for this application. Figure 9 shows the rule based system (generated using MATLAB Software). Around 8 Rules have been formed as an example. Figure 10 shows the rule sfiring and deffuzified output.

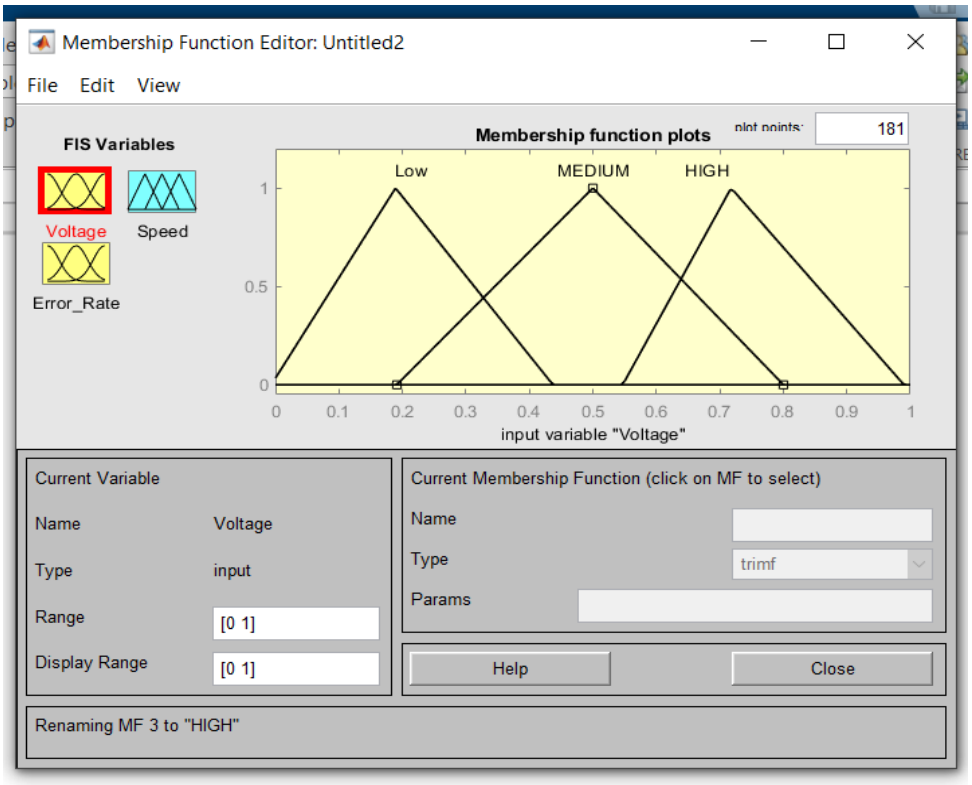


Figure 6 : Membership Graph for the input Voltage

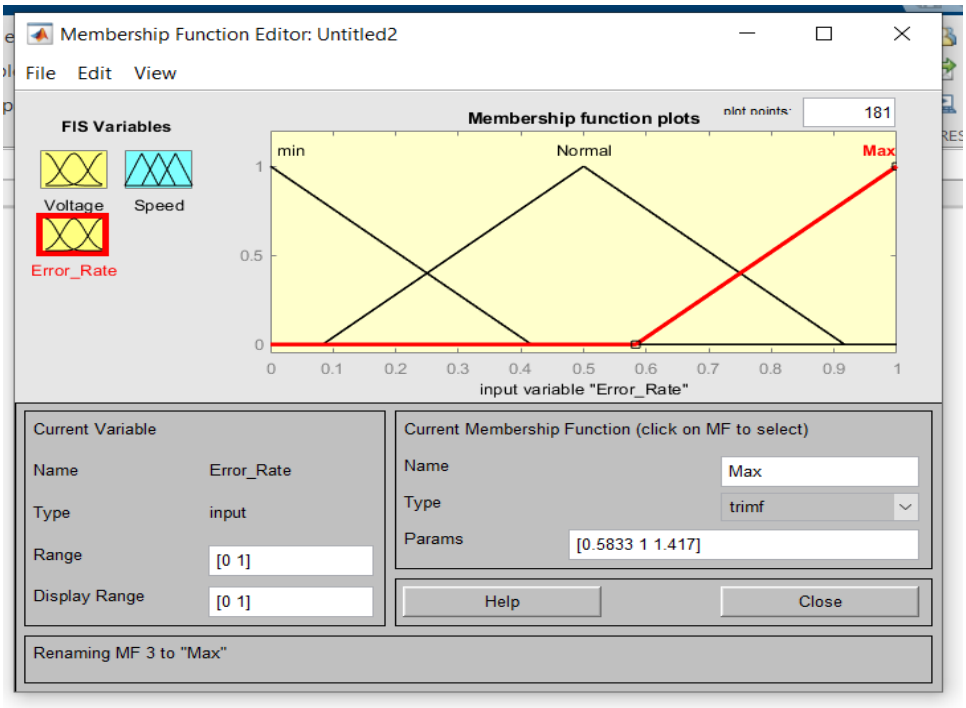


Figure : 7 Membership Graph for the input "Error Rate"

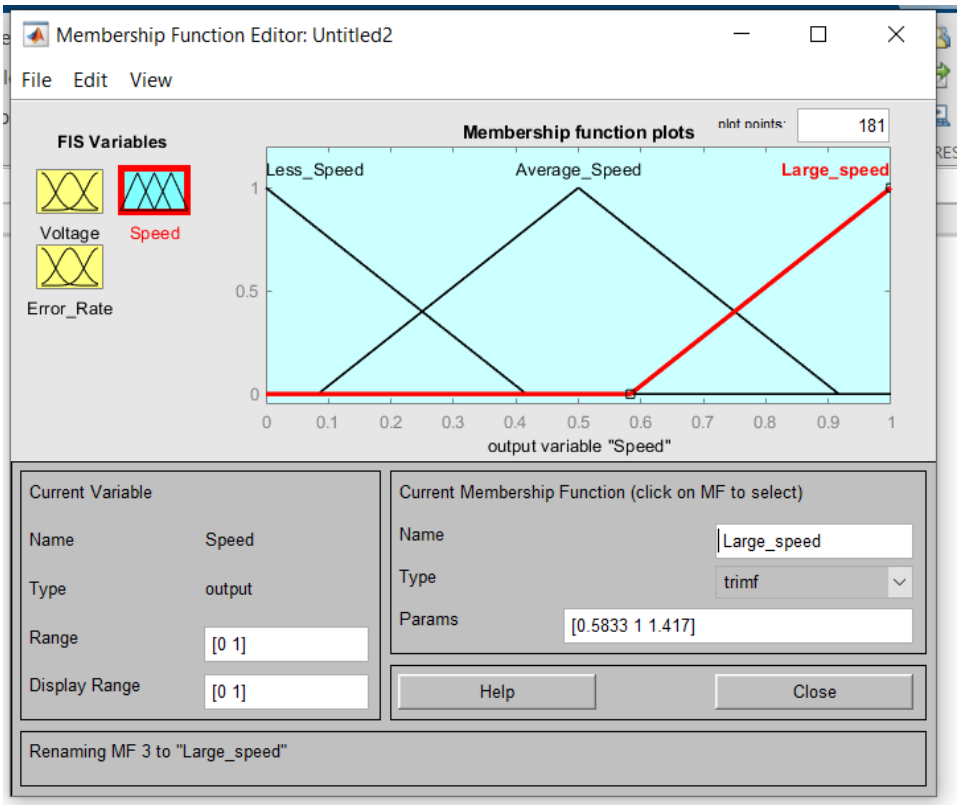


Figure : 8 Membership Graph for the Output "Speed"

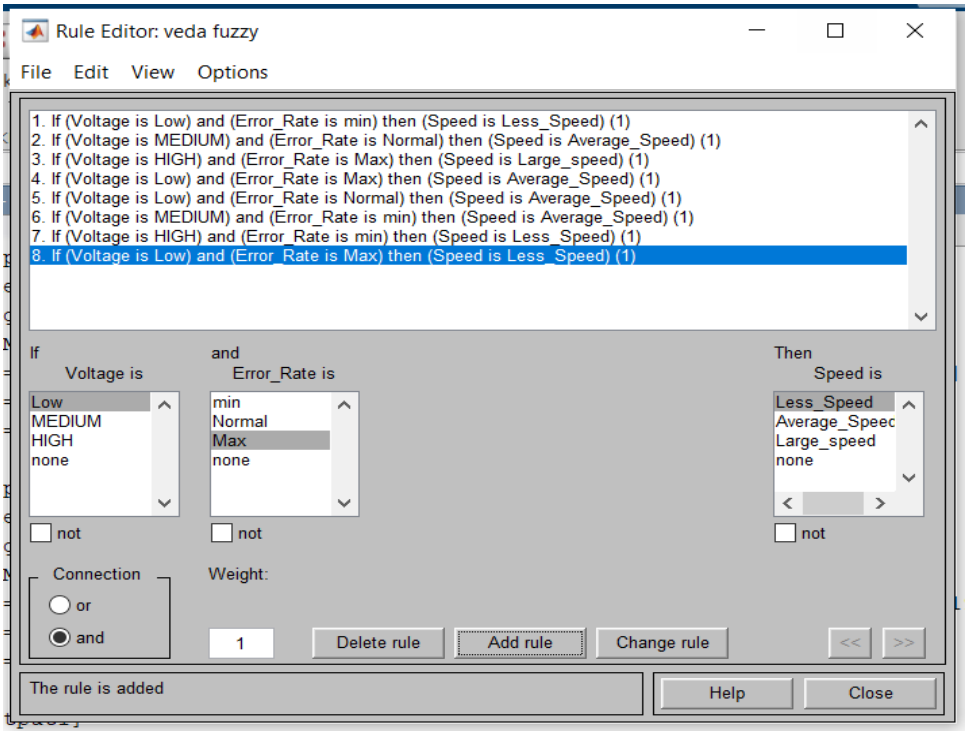


Figure : 9 Sample Rules for the Speed control application

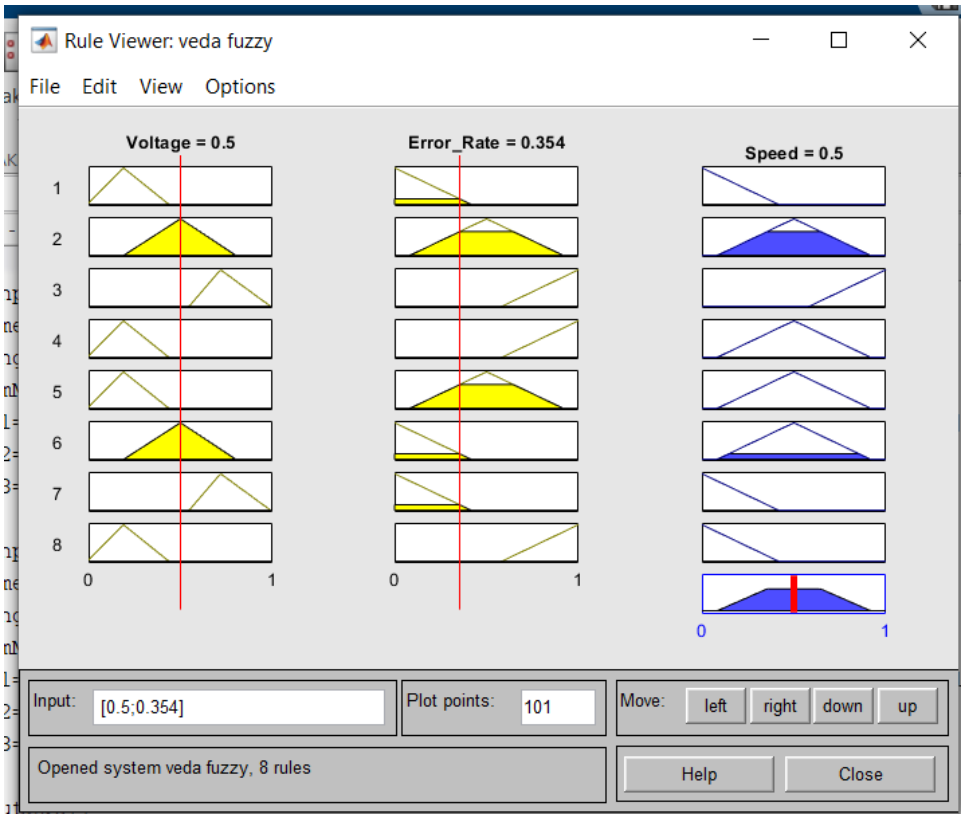


Figure : 10 Rule Viewer

The last graph on the right side is the defuzzified output.This is the value which is given to the final device.

5.5 CLUSTERING

Clustering is a process in which large sets of data is grouped into clusters / groups of smaller sets of similar data. Fuzzy c-means (FCM) is a method of clustering which allows one piece of data to be associated with two or more clusters. This method was developed by Dunn in 1973 and improved by Bezdek in 1981 and it is frequently used in pattern recognition applications. Fuzzy C means algorithm is an extended version of Hard C means technique. Fuzzy C means is more adaptable procedure where Hard c means is not a flexible process. FCM assigns membership values to each input data point with respect to the cluster centre or cluster head. The distance between the cluster head and the current input data point position which are present in each cluster is measured. For this purpose, Euclidean Distance measure is used. FCM is similar to that of K Means algorithm.

5.5.1 Fuzzy C-Means Algorithm

(1) Initialize U=[U_{ij}] matrix, $U^{(0)}$

(2) At k-step: calculate the C^k=[c_j] with $U^{(k)}$

$$c_j = \sum_{i=1}^{N} (u_{i,j}^m . x_i) / \sum_{i=1}^{N} u_{i,j}^m$$

(3) Update

$$u_{i,j} = 1 / \sum_{k=1}^{C} (\|x_i - c_j\| / \|x_i - c_k\|)^{2/(m-1)}$$

(4) if $\| U^{(k+1)} - U^{(k)} \|$ < threshold, then stop iterating; otherwise return to step 2

5.6 HARD C-MEANS

Hard C-Means clustering is also known as K-Means. _ The k-means algorithm partitions a collection of N vector into c groups. The aim of the algorithm is to find the cluster canter (centroids) for each group. The algorithm minimizes a dissimilarity function.

5.6.1 Hard C-Means Algorithm:

Step 1. Initialize the centroids ci,i=1,..c. This is typically achieved by randomly selecting c points from among all of the data points.

Step 2 Determine the membership matrix U by Equation

Step 3. Compute the dissimilarity function by using Equation below. Stop if its improvement over previous iteration is below a threshold.

Step 4. Compute new centroids using ci=1\Gi(Xi)

The performance of the algorithm depends on the initial positions of centroids. So the algorithm gives no guarantee for an optimum solution. _ Hard k-means algorithm executes a sharp classification, in which each object is either assigned to a class or not.

5.6.2 Advantages

1) Achieves best result for overlapped data

2) Best when compared with K means Algorithm.

2) Unlike k-means algorithm, in FCM a data point can be present in more than one cluster.

5.6.3 Disadvantages

1) Number of clusters should be known before the starting of iteration
2) If β parameter is less than good results is achieved, but more number of iterations are required.

REFERENCE BOOKS

1. Timothy Ross, “Fuzzy Logic with Engineering Application”, McGraw Hill, Edition 1997.
2. Drainkov, H.Hallendoor and M.Reinfrank, “An Introduction to Fuzzy Control”, Edition 2001..
3. Fuzzy Sets and Fuzzy Logic: Theory and Applications byGeorge J. Klir, Bo Yuan, Prentice Hall, 1995.
4. Soft Computing: Fundamentals and Applications by D.K.Pratihar, Narosa Publishing House, New-Delhi, 2014.