SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

SCHOOL OF ELECTRICAL AND ELECTRONICS
DEPARTMENT OF ELECTRONICS AND INSTRUMENTATION ENGINEERING

UNIT – I -INTRODUCTION TO SOFT COMPUTING AND ARTIFICIAL NEURAL NETWORKS – SIC1614

Evolution of Computing - Soft Computing Constituents - From Conventional AI to Computational Intelligence - MachineLearning Basics, Fundamentals of ANN - Biological Neurons and Their Artificial Models - Types of ANN - Properties -Different Learning Rules - Types of Activation Functions - Training of ANN - Hebb learning - Perceptron Model (Both Single& Multi Layer) - Training Algorithm - Problems Solving Using Learning Rules and Algorithms - Linear Separability –Limitation.

Evolution of Computing

Soft computing is a collection of artificial intelligence-based computational techniques[1] including the fundamentals of neural network, fuzzy logic, and genetic algorithm which, in turn, offers the superiority of humanlike problem solving capabilities.

Introduction Basics of Soft Computing What is Soft Computing

• The idea of soft computing was initiated in 1981 when Lotfi A. Zadeh published his first paper on soft data analysis "What is Soft Computing", Soft Computing. Springer-Verlag Germany/USA 1997.]

• Zadeh, defined Soft Computing into one multidisciplinary system as the fusion of the fields of Fuzzy Logic, Neuro-Computing, Evolutionary and Genetic Computing, and Probabilistic Computing.

• Soft Computing is the fusion of methodologies designed to model and enable solutions to real world problems, which are not modeled or too difficult to model mathematically.

• The aim of Soft Computing is to exploit the tolerance for imprecision, uncertainty, approximate reasoning, and partial truth in order to achieve close resemblance with human like decision making.

• The Soft Computing – development history SC = EC + NN + FL

Soft Evolutionary Neural Fuzzy Computing Computing Network Logic Zadeh Rechenberg McCulloch Zadeh 1981 1960 1943 1965

EC = GP + ES + EP + GA Evolutionary Genetic Evolution Evolutionary Genetic Computing Programming Strategies Programming Algorithms Rechenberg Koza Rechenberg Fogel Holland 1960 1992 1965 1962 1970 3 Definitions of Soft Computing (SC) Lotfi A. Zadeh, 1992 :

"Soft Computing is an emerging approach to computing which parallel the remarkable ability of the human mind to reason and learn in a environment of uncertainty and imprecision". The Soft Computing consists of several computing paradigms mainly : Fuzzy Systems, Neural Networks, and Genetic Algorithms.

• **Fuzzy set** : for knowledge representation via fuzzy If – Then rules.

• **Neural Networks :** for learning and adaptation

• **Genetic Algorithms** : for evolutionary computation These methodologies form the core of SC. Hybridization of these three creates a successful synergic effect; that is, hybridization creates a situation where different entities cooperate advantageously for a final outcome. Soft Computing is still growing and developing. Hence, a clear definite agreement on what comprises Soft Computing has not yet been reached. More new sciences are still merging into Soft Computing. Goals of Soft Computing Soft Computing is a new multidisciplinary field, to construct new generation of Artificial Intelligence, known as Computational Intelligence.

• The main goal of Soft Computing is to develop intelligent machines to provide solutions to real world problems, which are not modeled, or too difficult to model mathematically.

• Its aim is to exploit the tolerance for Approximation, Uncertainty, Imprecision, and Partial Truth in order to achieve close resemblance with human like decision making.

Approximation : here the model features are similar to the real ones, but not the same.

Uncertainty : here we are not sure that the features of the model are the same as that of the entity .

Imprecision : here the model features (quantities) are not the same as that of the real ones, but close to them. Importance of Soft Computing Soft computing differs from hard (conventional) computing. Unlike hard computing, the soft computing is tolerant of imprecision, uncertainty, partial truth, and approximation. The guiding principle of soft computing is to exploit these tolerance to achieve tractability, robustness and low solution cost. In effect, the role model for soft computing is the human mind. The four fields that constitute Soft Computing (SC) are : Fuzzy Computing (FC), Evolutionary Computing (EC), Neural computing (NC), and Probabilistic Computing (PC), with the latter subsuming belief networks, chaos theory and parts of learning theory. Soft computing is not a concoction, mixture, or combination, rather, Soft computing is a partnership in which each of the partners contributes a distinct methodology for addressing problems in its domain. In principal the constituent methodologies in Soft computing are complementary rather than competitive. Soft computing may be viewed as a foundation component for the emerging field of Conceptual Intelligence. 5 Fuzzy Computing In the real world there exists mu

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.

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.

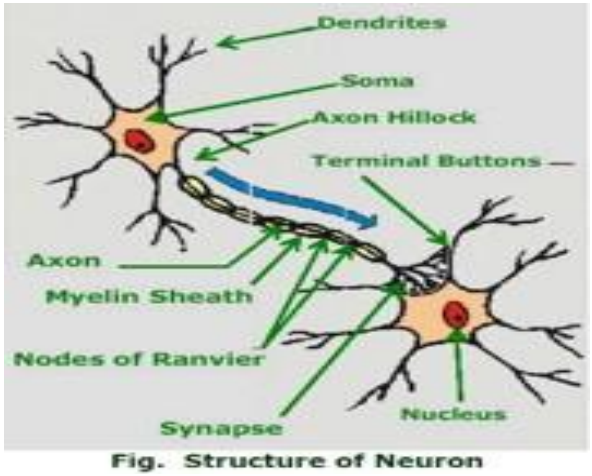


Fig. Structure of Neuron

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.

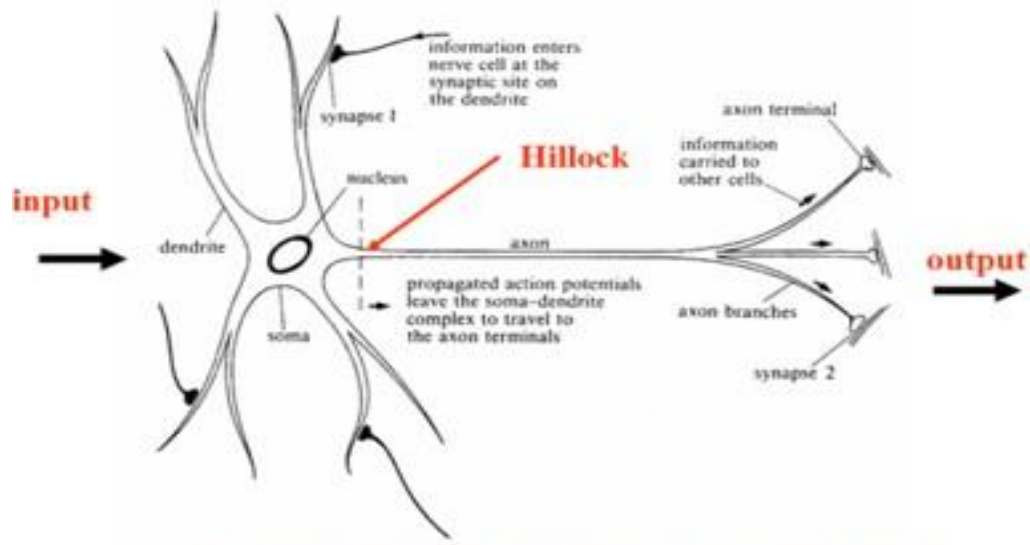


Fig. Structure of a neural cell in the human brain

- Dendrites receive activation from other neurons.
- Soma processes the incoming activations and converts them into output activations.
- Axons act as transmission lines to send activation to other neurons.
- Synapses the junctions allow signal transmission between the axons and dendrites.
- The process of transmission is by diffusion of chemicals called neuro-

transmitters.

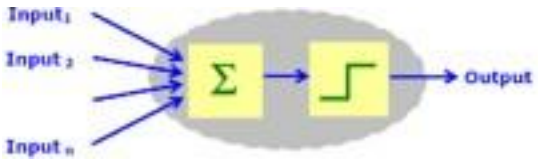
McCulloch-Pits introduced a simplified model of this real neurons.

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.
- In other words,
  - The input to a neuron arrives in the form of signals.
  - The signals build up in the cell.
  - Finally the cell discharges (cell fires) through the output.
  - The cell can start building up signals again.

The equation for the output of a McCulloch-Pitts neuron as a function of 1 to n inputs is written as

Output = sgn (Input i – φ)

where φ is the neuron's activation threshold.

If Input i ≥ φ then Output = 1

If Input i < φ then Output = 0

In this McCulloch-Pitts neuron model, the missing features are:

- Non-binary input and output
- Non-linear summation
- Smooth thresholding
- Stochastic

- Temporal information processing.

Basic Elements:

Neuron consists of three basic components – weights, thresholds and a single activation function.

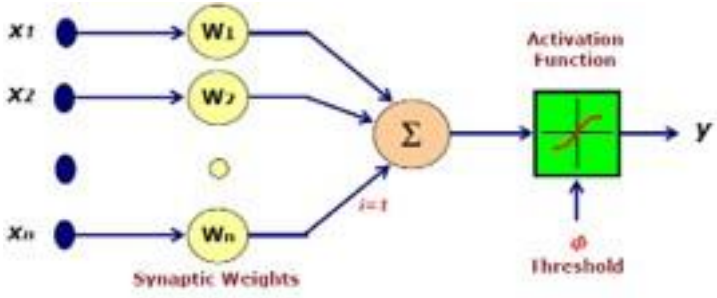
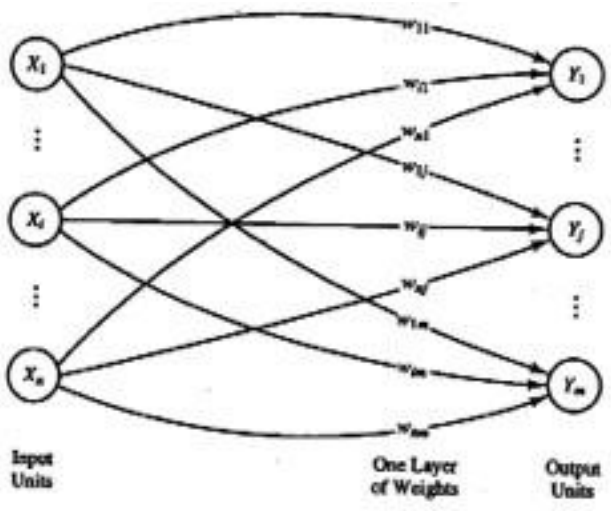


Fig Basic Elements of an Artificial Linear Neuron

Types of ANN

Single Layer Feedforward NN

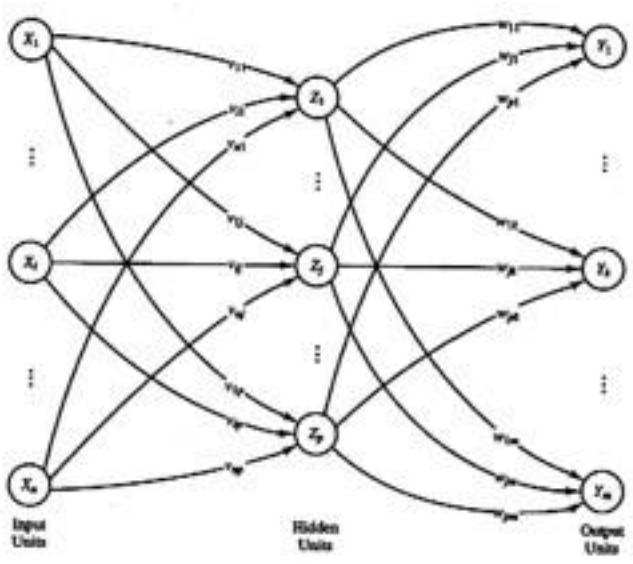
- A single layer network has one layer of connection weights.
- Often the units can be distinguished as input units which receive signals from the outside world and output units from which the response of the net can be read
- In a typical single layer net the input units are fully connected to output units but are not connected to other units and the output are not connected to other output units.



Multilayer feedforward NN

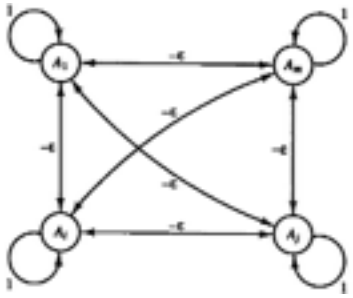
- A multilayer net is a net with one or more layers of nodes between the input units and the output units.
- Typically there is a layer of weights between two adjacent levels of units.

- Multilayer nets can solve more complicated problems that can single layer nets, but training may be more difficult.



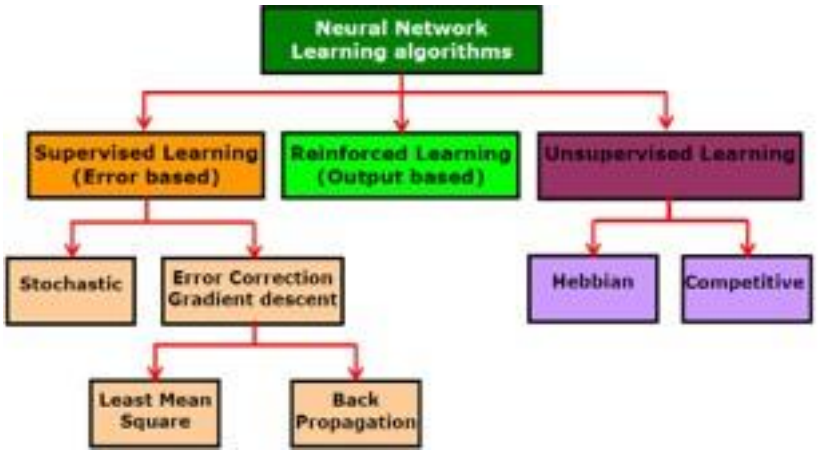
Competitive layer:

- A competitive layer form a part of a large number of neural networks.
- The competitive interconnections have weights of -ε



Different Learning Rules

- Supervised learning
- Unsupervised learning
- Reinforced learning
- Hebbian learning
- Gradient descent learning
- Competitive learning
- Stochastic learning



- 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.

- 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.

- 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.

- 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 () are associated by the weight matrix W, known as the correlation matrix.

It is computed as

W =

Here is the transpose of the associated output vector Numerous

variants of the rule have been proposed.

-
- Gradient 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 is the weight update of the link connecting the and neuron of the two neighbouring layers, then is defined as,

$$= \eta$$

Where, η is the learning rate parameter and is the error gradient with reference to the weight .

- 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.

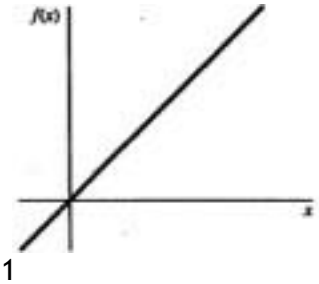
- Stochastic learning:

In this method, weights are adjusted in a probablistic fashion.

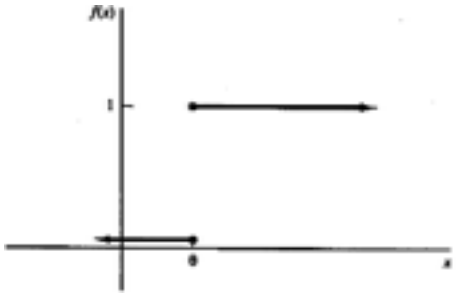
An example is evident in simulated annealing the learning mechanism employed by Boltzmann and Cauchy machines, which are a kind of NN systems.

Types of Activation Functions

- Common activation functions
  - Identity function
    - f(x) = x for all x



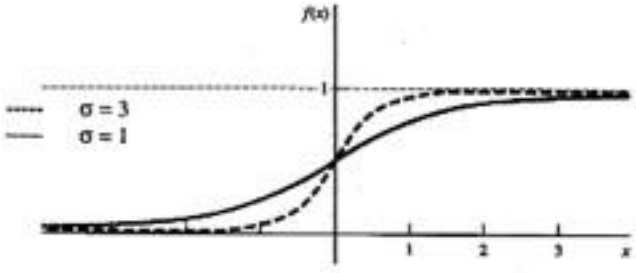
– Binary step function (with threshold θ) (aka Heaviside function or threshold function)



S

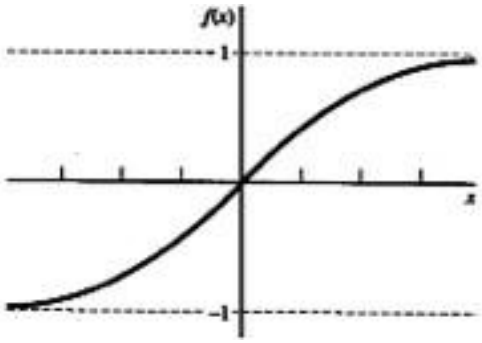
-- Binary sigmoid

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



-

Bipolar sigmoid



Training a Neural Network

- Whether our neural network is a simple Perceptron, or a much more complicated multilayer network with special activation functions, we need to develop a systematic procedure for determining appropriate connection weights.
- The general procedure is to have the network ***learn*** the appropriate weights from a representative set of training data
- In all but the simplest cases, however, direct computation of the weights is intractable

Instead, we usually start off with ***random initial weights*** and adjust them in small steps until the required outputs are produced

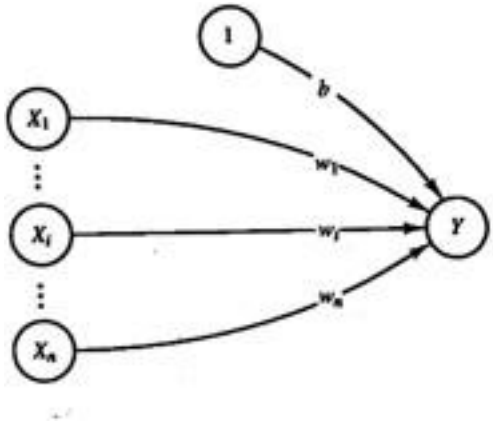
- We shall now look at a brute force derivation of such an ***iterative learning algorithm*** for simple Perceptrons.

Perceptron Model (Both Single & Multi Layer)

Simple Perceptron for Pattern Classification

We consider here a NN, known as the Perceptron, which 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 neuron:

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



Perceptron Algorithm

- Step 0: Initialize weights and bias
  - For simplicity, set weights and bias to zero
  - Set learning rate a (0 <= a <= 1) (h)
- Step 1: While stopping condition is false do steps 2-6
- Step 2: For each training pair s:t do steps 3-5
- Step 3: Set activations of input units

  x_i =
- Step 4: Compute response of output unit:

y_in □ b □ □ x_i □ w_i

- Step 5: Update weights and bias if an error occurred for this pattern

  if y != t

  $w_i(new) = w_i(old) + atx_i$

  $b(new) = b(old) + at$

  else

  $w_i(new) = w_i(old)$

  $b(new) = b(old)$

- Step 6: Test Stopping Condition
  - If no weights changed in Step 2, stop, else, continue

Note that instead of one separating line, we have a line separating the region of positive reponse from the region of zero response, namely, the bounding the inequality.

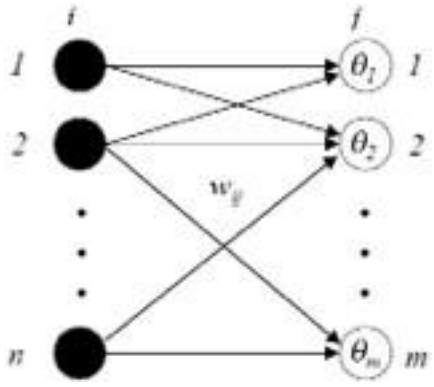
$W_1 x_1 + w_2 x_2 + b > \theta$

And a line separating the region of zero reponse from the region of negative response, namely, the bounding the inequality.

$W_1 x_1 + w_2 x_2 + b < -\theta$

PROBLEM----------

Multi Layer Perceptron Model:



$$out_j = \mathrm{sgn}(\sum_{i=1}^{n} out_i w_{ij} - \theta_j)$$

Perceptron and linearly seperable tasks:

Perceptron are successful only on problems with a linearly separable solution sapce and cited the XOR problem a an illustration.

Perceptron cannot handle, in particular, tasks which are not linearly separable.

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 linearly seperable if the sets can be seperated by a straight line.

Generalizing, a set of points in n-dimentional space are linearly seperable if there is a hyperplane of (n-1) dimensions that separates the sets.

Hebbian net:

Earliest and simplest learning rule for a neural net. Hebb proposed that learning occurs by modificatio of the synapse strenghts in a manner such that if two interconnected neurons are both "ON" at the sametime, there the weight between those neurons should be increased.

- Step 0: Initialize all weights

– For simplicity, set weights and bias to zero

- Step 1: For each input training vector do steps 2-4
  - Step 2: Set activations of input units

$$x_i = s_i$$

  - Step 3: Set the activation for the output unit

$$y = t$$

  - Step 4: Adjust weights and bias

$$w_i(new) = w_i(old) + yx_i \quad b(new) = b(old) + y$$

TEXT / REFERENCE BOOKS

1. Laurene Fausett, " Fundamentals of Neural Networks: Architectures, Algorithms and Applications", 2008.
2. Timothy J. Ross , “Fuzzy Logic with Engineering Applications”, McGraw - Hill International Editions, 2004.
3. Jang J.S.R., Sun C.T. and Mizutani E, "Neuro-Fuzzy and soft computing", Pearson Education, 2003.
4. Rajasekaran. S, Pai. G.A.V. “Neural Networks, Fuzzy Logic and Genetic Algorithms”, Prentice Hall of India, 2003.

Part- A

1. What is meant by evolutionary computation
2. What is meant by ANN
3. Define Architecture in NN
4. Define Training algorithm
5. Give some applications of ANN
6. Give the properties of biological neurons
7. Give the different types of ANN
8. Write the types of learning rules
9. What is meant by supervised learning
10. What is meant by competitive learning
11. Give the properties of ANN
12. What is meant by activation function
13. What is meant by Hebb Net.
14. What is meant by linear separability
15. Give the limitations of linear separability

Part-B

1. Explain the architecture, Learning rule and algorithm of Perceptron Model
2. Explain in detail about Biological Neural Network and compare with ANN.
3. Explain in detail about the different types of architecture
4. Elaborate on different learning rules
5. Explain in detail about Hebb Net .
6. Write notes on different types of activation function
7. Train logical AND gate with binary input & bipolar target using perceptron algorithm with initial weights and bias as

$w_1=2$, $w_2=2$ and $b=-4$

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

SCHOOL OF ELECTRICAL AND ELECTRONICS
DEPARTMENT OF ELECTRONICS AND INSTRUMENTATION ENGINEERING

UNIT – II – DETERMINISTIC AND STATISTICAL NETWORKS – SIC1614

DETERMINISTIC AND STATISTICAL NETWORKS

Back Propagation Training Algorithm - Practical Difficulties - Counter Propagation Network - Structure & Operation - Training of Kohonen and Grossberg Layer - Applications of BPN & CPN - Statistical Method – Training Application - Boltzman Training - Cauchy Training - Hop Field Network and Boltzman Machine - Speed Energy Function - Network Capacity - RBF Network, BAM, Architecture of SOM, ANN based water level controller.

Introduction

Back propagation is a way of learning the internal representation of a multilayered network. How back propagation does this is based on a simple idea. We input a vector of values pi into the network and we get out a corresponding output a3 i (see fig on preceding slide). We compare this output to our target (desired) output ti to determine the cumulative error among the different outputs units. But the output units are themselves connected with the hidden units in the network. We don't know what the hidden units ought to do, but we can compute how fast the error changes as we change a hidden activity. But instead of using the desired activities to train the hidden units, we use error derivatives with respect to the hidden activities. Each hidden activity can affect many output units and can therefore have many separate effects on the error. These effects must be combined. The point of the back propagation algorithm is to show how we can compute the error derivatives for all the hidden units efficiently. Once we have the error derivatives for the hidden activities, its easy to get the error derivatives for the weights going into a hidden unit.

Algorithm:

The training involves three stages

The feedforward of the input training pattern

The back propagation of the associated error

The adjustment of the weights

Activation function:

Backpropagation net should have several important characteristics

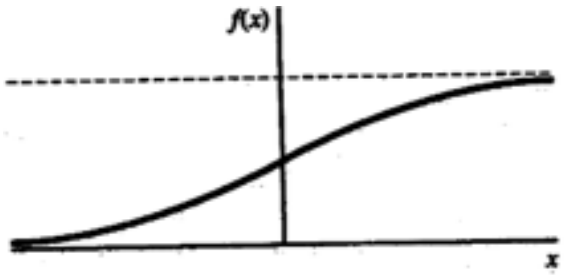
- Continuous
- Differentiable
- Monotonically non decreasing

One of the most typical activation functions is the binary sigmoid function, which has range of (0,1) and is defined as

$$f_1(x) = \frac{1}{1 + \exp(-x)},$$

with

$$f_1'(x) = f_1(x)[1 - f_1(x)].$$



Another common activation function is bipolar sigmoid, which has range of (-1,1) and is defined as,

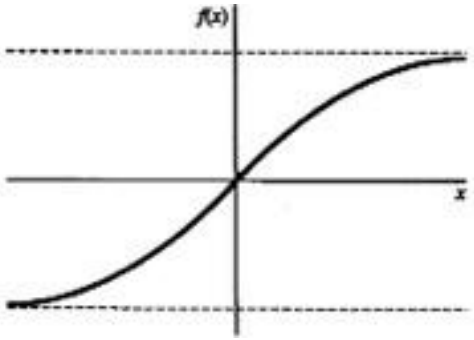
$$f_2(x) = \frac{2}{1 + \exp(-x)} - 1,$$

with

$$f_2'(x) = \frac{1}{2}[1 + f_2(x)][1 - f_2(x)].$$

The bipolar sigmoid function is closely related to the function,

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



Slop O. Initialize weights.

(I ‹a smzlt random values).

Sizp 1. **While StO{3}jjng cond tion is false, ds Steps z—9.**

Siep 2. For cach training pair, do Steps 3-B.

T'ee@o •ard• n) receives

Step 3. Each input unit (A/, ¿ = I. . „ inp£Jt g gFI,Bl x, and braadcasts this sigzud to all uni‹s in the Iay<r abov« ‹tkc rida,n ul«itsj.

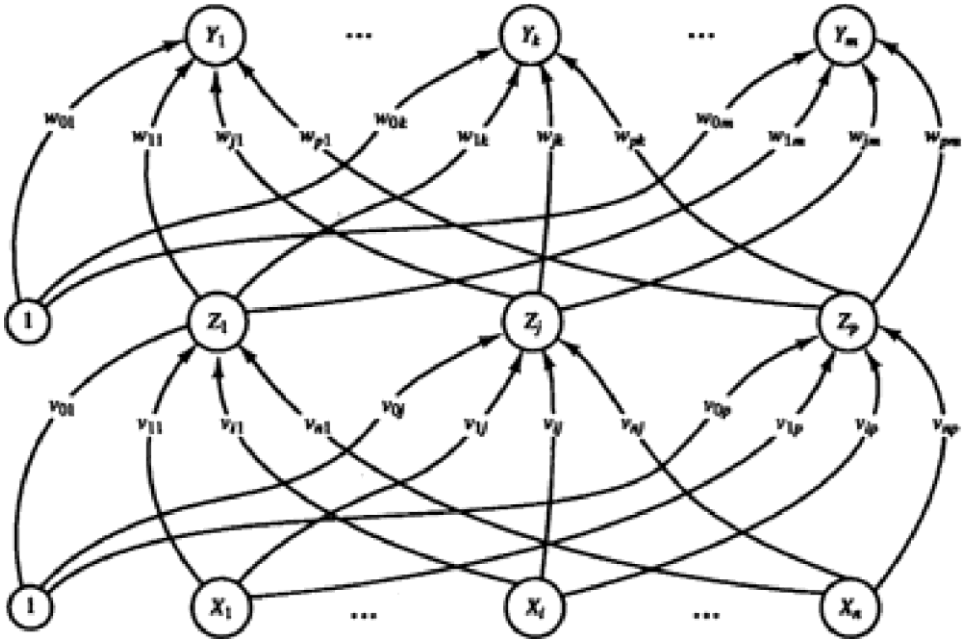
S/ @ **Each hiddcn unll (Z„/ = f. . .,. p) suatsits wcightc‹1 input signals,**

$$z\_in_j = v_{0j} + \sum x_i v_{ij},$$

applies its activation f••c‹iOfl tO compute i\g output signal,

$$z_j = f(z\_in_j),$$

and sends this signal to al) units in the layer above foutput units).



!+6 • b 0tJt§Ut «*it (Y# , ñ , I , . rig) SlZf¥lS it.• weighted input signals,

and applies i‹s w‹ivation functio»

i‹s output signal. to compute

Bac ropn8aIiuo of rrror.'

5'fep 6. Each oaipai anii I Yz,*t‹* , *I* . . . , m l receives B t8f@gE p€tttc <e pending to the input twining pattern. computes its error informa- tion term.

$$\delta_k = (t_k - y_k)f'(y\_in_k),$$

calculates its weight correction term (used to

calculates its bias correction term (used to update w_{0k} later),

B•d aends 8‹ to units in the layer betw. *Step 7.*

&ch ~~hirfrtr•n~~ uni(I-Jt,/ —- I, . . . , p) en its d+ tR input» {from units in the layer a@),

multiplies by the denvativc oP its activation ftC(IOU \O calculate US error information term,

vaIr›ik•trs its wetght correction term (ugpd to

and ~~•nI•tiIn •1~~ its bias correction term fuasd to update ; lafer).

Update weights and biases:

Step 8. Each output unit ($Y_k, k = 1, \ldots, m$) updates its bias and weights ($j = 0, \ldots, p$):

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

Each hidden unit ($Z_j, j = 1, \ldots, p$) updates its bias and weights ($i = 0, \ldots, n$):

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

Step 9. Test stopping condition.

Counter propagation network

They are multilayer networks based on a combination of input, clustering and output layers.

Counter propagation networks are trained in two stages:

First stage – the input vectors are clustered

- No topology was assumed for the cluster units.

Second stage – the weights from the cluster units to the output units are adapted to produce the desired response

There are two types of counter propagation nets:

- Full counter propagation
- Forward only counter propagation

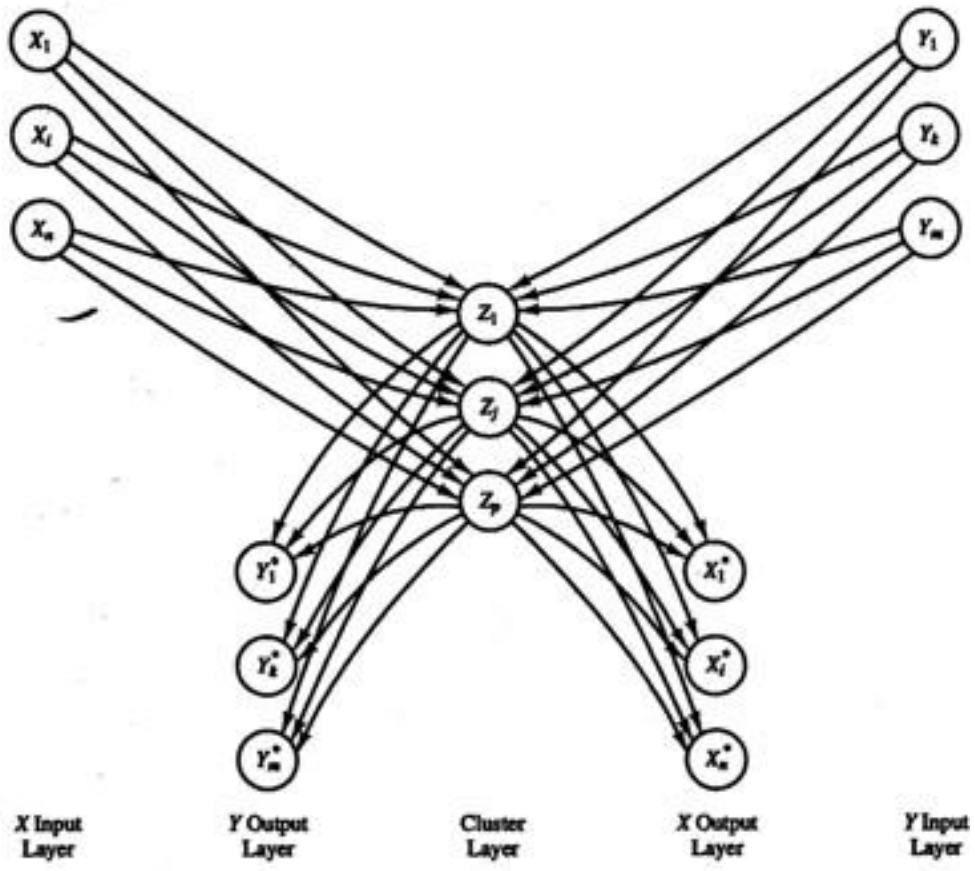
Full counter propagation

This was developed to provide an efficient method of representing a large number of vector pairs, x:y by adaptively constructing a look-up table.

It produces an approximation: based on input of an x vector or input of a y vector only, or input of an x:y pair, possibly with some distorted or missing elements in either or both vectors.

Architecture:

Full counter propagation network:



First phase:

The units in the X input, cluster and Y input layers are active.

The learning rule for weight updates on the winning cluster unit is.

$$v_{iJ}(\text{new}) = (1 - \alpha)v_{iJ}(\text{old}) + \alpha x_i, \quad i = 1, \ldots, n;$$

$$w_{kJ}(\text{new}) = (1 - \beta)w_{kJ}(\text{old}) + \beta y_k, \quad k = 1, \ldots, m.$$

Second phase:

Only j unit remains active in the cluster region.

The weight updates for the units in the Y output and X outout layers are,

$$u_{Jk}(\text{new}) = (1 - a)u_{Jk}(\text{old}) + ay_k, \quad k = 1, \ldots, m;$$

$$t_{Ji}(\text{new}) = (1 - b)t_{Ji}(\text{old}) + bx_i, \quad i = 1, \ldots, n.$$

Algorithm:

Step 0. Initialize weights, learning rates, etc.

Step 1. While stopping condition for phase 1 training is false, do Steps 2–7.

Step 2. For each training input pair **x**:**y**, do Steps 3–5.

Step 3. Set X input layer activations to vector **x**; set Y input layer activations to vector **y**.

Step 4. Find winning cluster unit; call its index J;

Step 5. Update weights for unit Z_J:

$$v_{iJ}(\text{new}) = (1 - \alpha)v_{iJ}(\text{old}) + \alpha x_i, \quad i = 1, \ldots, n;$$

$$w_{kJ}(\text{new}) = (1 - \beta)w_{kJ}(\text{old}) + \beta y_k, \quad k = 1, \ldots, m.$$

Step 6. Reduce learning rates α and β.

Step 7. Test stopping condition for phase 1 training.

Step 8. While stopping condition for phase 2 training is false, do Steps 9–15.
(*Note:* α and β are small, constant values during phase 2.)

Step 9. For each training input pair **x**:**y**, do Steps 10–13.

Step 10. Set X input layer activations to vector **x**; set Y input layer activations to vector **y**.

Step 11. Find winning cluster unit; call its index J.

Step 12. Update weights into unit Z_J:

$$v_{iJ}(\text{new}) = (1 - \alpha)v_{iJ}(\text{old}) + \alpha x_i, \quad i = 1, \ldots, n;$$

$$w_{kJ}(\text{new}) = (1 - \beta)w_{kJ}(\text{old}) + \beta y_k, \quad k \,) 1, \ldots, m.$$

Step 13. Update weights from unit Z_J to the output layers:

$$u_{Jk}(\text{new}) = (1 - a)u_{Jk}(\text{old}) + ay_k, \qquad k = 1, \ldots, m.$$

$$t_{Ji}(\text{new}) = (1 - b)t_{Ji}(\text{old}) + bx_i, \qquad i = 1, \ldots, n.$$

Step 14. Reduce learning rate.

Step 15. Test stopping condition for phase 2 training.

In steps 4 and 11

- In case of a tie, take the unit with the smallest index.
- To use the dot product metric, find the cluster unit with the largest net input:

$$z\_in_j = \sum_i x_i v_{ij} + \sum_k y_k w_{kj}$$

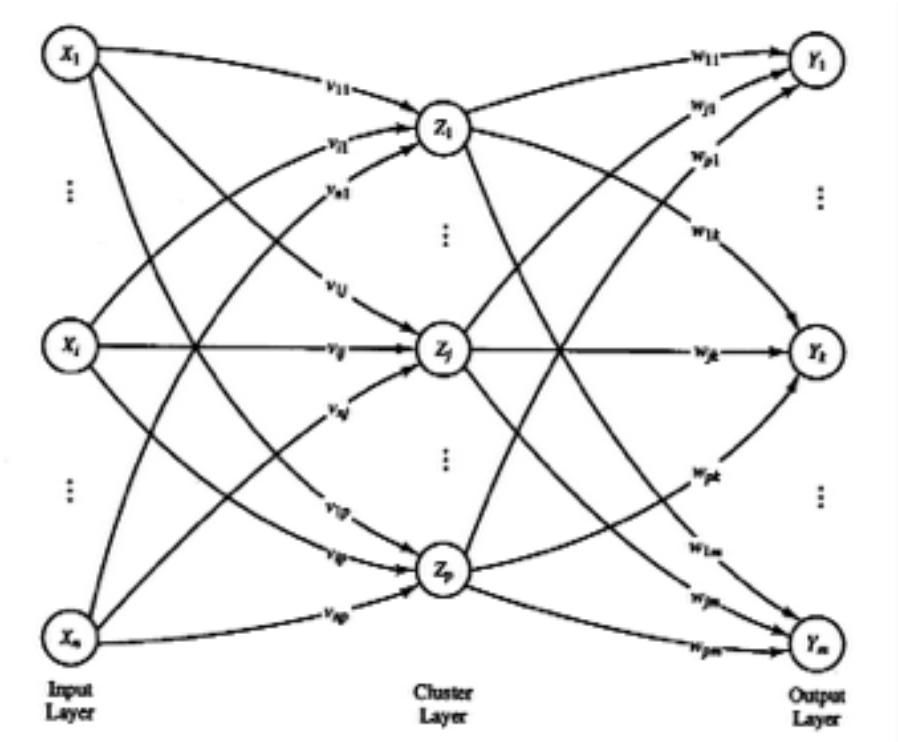
The weight vectors and input vectors should be normalized to use the dot product metric.

To use the Euclidean distance metric, find the cluster unit , the square of whose distance from the input vectors is smallest.

$$D_j = \sum_i (x_i - v_{ij})^2 + \sum_k (y_k - w_{kj})^2.$$

Forward only counter propagation:

- Simplified version
- Differs from the other net in using only the x vectors to form the clusters.



Algorithm:

Learning rule for weights from input units to cluster units

$$v_{iJ}(\text{new}) = v_{iJ} + \alpha(x_i - v_{iJ})$$
$$= (1 - \alpha)v_{iJ}(\text{old}) + \alpha x_i.$$

Learning rule for weights from cluster units to output units

$$w_{Jk}(\text{new}) = w_{Jk} + a(y_k - w_{Jk})$$
$$= (1 - a)w_{Jk}(\text{old}) + ay_k$$

Step 0. Initialize weights, learning rates, etc.
Step 1. While stopping condition for phase 1 is false, do Steps 2–7.
Step 2. For each training input **x**, do Steps 3–5.
Step 3. Set X input layer activations to vector **x**.
Step 4. Find winning cluster unit; call its index J.
Step 5. Update weights for unit Z_J:

$$v_{iJ}(\text{new}) = (1 - \alpha)v_{iJ}(\text{old}) + \alpha x_i,$$
$$i = 1, \ldots, n.$$

Step 6. Reduce learning rate α.
Step 7. Test stopping condition for phase 1 training.
Step 8. While stopping condition for phase 2 training is false, do Steps 9–15.
(*Note:* α is a small, constant value during phase 2.)
Step 9. For each training input pair **x**:**y**, do Steps 10–13.
Step 10. Set X input layer activations to vector **x**; set Y output layer activations to vector **y**.
Step 11. Find the winning cluster unit; call its index J.
Step 12. Update weights into unit Z_J (α is small):

$$v_{iJ}(\text{new}) = (1 - \alpha)v_{iJ}(\text{old}) + \alpha x_i,$$
$$i = 1, \ldots, n.$$

Step 13. Update weights from unit Z_J to the output units:

$$w_{Jk}(\text{new}) = (1 - a)w_{Jk}(\text{old}) + ay_k,$$
$$k = 1, \ldots, m.$$

Step 14. Reduce learning rate a.
Step 15. Test stopping condition for phase 2 training.

In steps 4 and 11

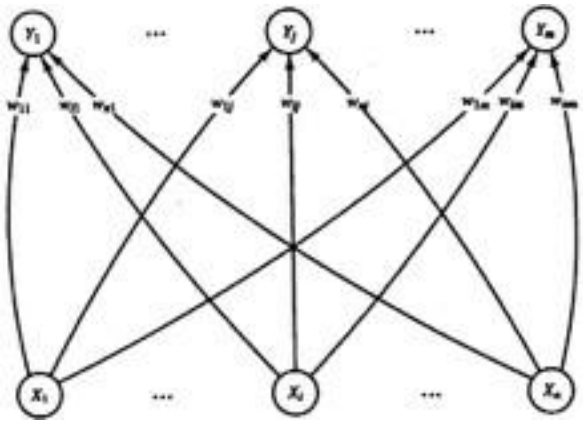
- In case of a tie, take the unit with the smallest index.
- To use the dot product metric, find the cluster unit with the largest net input:

$$z\_in_j = \sum_i x_i v_{ij}.$$

To use the Euclidean distance metric, find the cluster unit , the square of whose distance from the input vectors is smallest.

$$D_j = \sum_i (x_i - v_{ij})^2,$$

Training of Kohonen:



Algorithm:

Step 0. Initialize weights w_{ij}. (Possible choices are discussed below.)
Set topological neighborhood parameters.
Set learning rate parameters.

Step 1. While stopping condition is false, do Steps 2–8.

Step 2. For each input vector **x**, do Steps 3–5.

Step 3. For each j, compute:

$$D(j) = \sum_i (w_{ij} - x_i)^2.$$

Step 4. Find index J such that $D(J)$ is a minimum.

Step 5. For all units j within a specified neighborhood of J, and for all i:

Step 7. Reduce radius of topological neighborhood at specified times.

Step 8. Test stopping condition.

CLUSTER UNIT	v	w
Z_1	0.11	9.0
Z_2	0.14	7.0
Z_3	0.20	5.0
Z_4	0.30	3.3
Z_5	0.6	1.6
Z_6	1.6	0.6
Z_7	3.3	0.30
Z_8	5.0	0.20
Z_9	7.0	0.14
Z_{10}	9.0	0.11

To obtain approximate value of y for x = 0.12.

Step 0: Initialize weights.

Step 1: For the input x = 0.12, y = 0.0, do steps 2-4.

Step 2: Set X input layer activations to vector x.

Set Y input layer activations to vector y.

Step 3: Find the index J of the winning cluster unit, the squares of the distances from the input to each of the cluster units are,

$$D_1 = (0.12 - 0.11)^2 + (0.00 - 9.00)^2 = 81$$
$$D_2 = (0.12 - 0.14)^2 + (0.00 - 7.00)^2 = 49$$
$$D_3 = (0.12 - 0.20)^2 + (0.00 - 5.00)^2 = 25$$
$$D_4 = (0.12 - 0.30)^2 + (0.00 - 3.30)^2 = 11$$
$$D_5 = (0.12 - 0.60)^2 + (0.00 - 1.60)^2 = 2.8$$
$$D_6 = (0.12 - 1.60)^2 + (0.00 - 0.60)^2 = 2.6$$
$$D_7 = (0.12 - 3.30)^2 + (0.00 - 0.30)^2 = 10.2$$
$$D_8 = (0.12 - 5.00)^2 + (0.00 - 0.20)^2 = 23.9$$
$$D_9 = (0.12 - 7.00)^2 + (0.00 - 0.14)^2 = 47.4$$
$$D_{10} = (0.12 - 9.00)^2 + (0.00 - 0.11)^2 = 78.9$$

Thus, based on the total input, the closest cluster unit is J = 6.

Step 4: Compute approximations to x and y:

$$\mathbf{x^*} = t_J = 1.6$$
$$\mathbf{y^*} = u_J = 0.6.$$

Step 5: Find the index J of the winning cluster unit, the squares of the distances from the input to each of the cluster units are,

$$D_1 = (0.12 - 0.11)^2 = 0.0001$$
$$D_2 = (0.12 - 0.14)^2 = 0.0004$$
$$D_3 = (0.12 - 0.20)^2 = 0.064$$
$$D_4 = (0.12 - 0.30)^2 = 0.032$$
$$D_5 = (0.12 - 0.60)^2 = 0.23$$
$$D_6 = (0.12 - 1.60)^2 = 2.2$$
$$D_7 = (0.12 - 3.30)^2 = 10.1$$
$$D_8 = (0.12 - 5.00)^2 = 23.8$$
$$D_9 = (0.12 - 7.00)^2 = 47.3$$
$$D_{10} = (0.12 - 9.00)^2 = 81$$

Thus based on the input from x only, the closest cluster unit is J = 1. Step

$$\mathbf{x}^* = t_J = 0.11,$$
$$\mathbf{y}^* = u_J = 9.00.$$

6: Compute approximations to x and y.

Boltzmann machine:

The states of the units are binary valued, with probabilistic state transitions. This machine described in this section has fixed weights, which express the degree of desirability that units and both be 'ON'. In applying Boltzmann machines to constrained optimization problems, the weights represent the constraints of the problem and the quantity to be optimized.

The objective of the neural net is to maximized the consensus function,

$$C = \sum_i [\sum_{j \leq i} w_{ij}\, x_i x_j].$$

The sum runs over all units of the net. In sequential Boltzmann machine, the change in consensus if unit, were to change its state is,

$$\Delta C(i) = [1 - 2x_i][w_{ii} + \sum_{j \neq i} w_{ij} x_j],$$

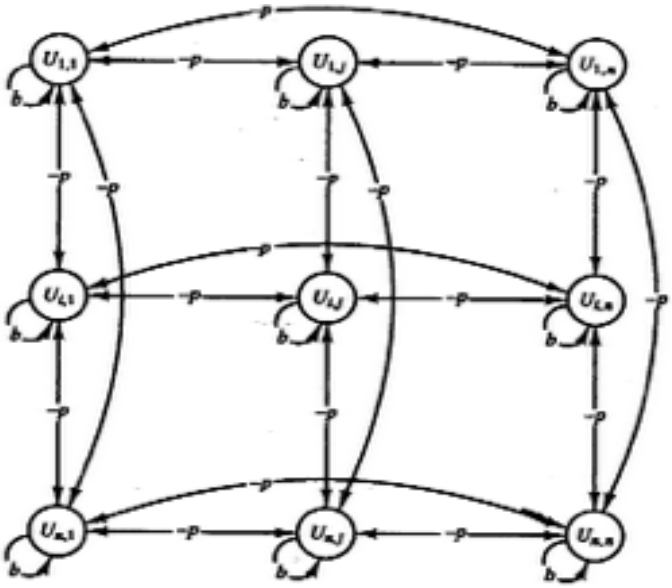
The probability of the net accepting a change in state for unit is,

$$A(i, T) = \frac{1}{1 + \exp\left(-\frac{\Delta C(i)}{T}\right)}.$$

The control parameters T called the temperature is gradually reduced as the net searches for a maximal consensus.

Architecture:

The architecture of a Boltzmann machine for units arranged in a two dimensional array.



Step 0. Initialize weights to represent the constraints of the problem.
Initialize the control parameter (temperature) T.
Initialize activations of units (random binary values).

Step 1. While stopping condition is false, do Steps 2–8.

Step 2. Do Steps 3–6 n^2 times. (This constitutes an epoch.)

Step 3. Choose integers I and J at random between 1 and n.
(Unit U_{IJ} is the current candidate to change its state.)

Step 4. Compute the change in consensus that would result:

$$\Delta C = [1 - 2u_{I,J}][w(I, J; I, J) + \sum_{i,j \neq I,J} \sum w(i, j; I, J)u_{i,j}].$$

Step 5. Compute the probability of acceptance of the change:

$$A(T) = \frac{1}{1 + \exp\left(-\frac{\Delta C}{T}\right)}.$$

Step 6. Determine whether or not to accept the change.
Let R be a random number between 0 and 1.
If $R < A$, accept the change:

$u_{IJ} = 1 - u_{IJ}$. (This changes the state of unit U_{IJ}.)

If $R \geq A$, reject the proposed change.

Step 7. Reduce the control parameter:

$T(\text{new}) = 0.95T(\text{old})$.

Step 8. Test stopping condition:

Hopfield Network:

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,

$$w_{ij} = w_{ji}$$

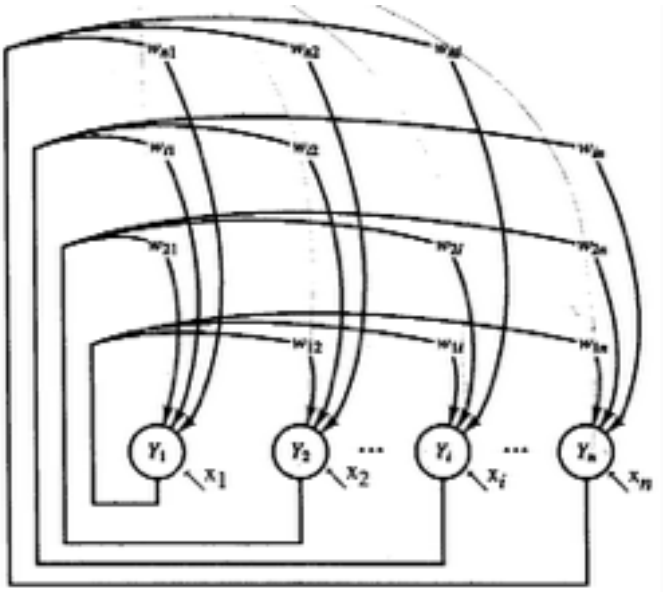
And

$$w_{ii} = 0.$$

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.

Architecture:

An expanded form of a common representation of the Hopfield net,



Algorithm:

To store a set of binary patterns $\mathbf{s}(p), p = 1, \ldots, P$, where,

$$\mathbf{s}(p) = (s_1(p), \ldots, s_i(p), \ldots, s_n(p))$$

The weight matrix $\mathbf{W} = \{w_{ij}\}$ is given by,

$$w_{ij} = \sum_p [2s_i(p) - 1][2s_j(p) - 1] \qquad \text{for } i \neq j$$

And = 0

To store a set of bipolar patterns $\mathbf{s}(p), p = 1, \ldots, P$, where,

$$\mathbf{s}(p) = (s_1(p), \ldots, s_i(p), \ldots, s_n(p))$$

The weight matrix $\mathbf{W} = \{w_{ij}\}$ is given by,

$$w_{ij} = \sum_p s_i(p)s_j(p) \qquad \text{for } i \neq j$$

And = 0.

Step 0. Initialize weights to store patterns.
(Use Hebb rule.)

While activations of the net are not converged, do Steps 1–7.

Step 1. For each input vector **x**, do Steps 2–6.

Step 2. Set initial activations of net equal to the external input vector **x**:

$$y_i = x_i, (i = 1, \ldots n)$$

Step 3. Do Steps 4–6 for each unit Y_i.
(Units should be updated in random order.)

Step 4. Compute net input:

$$y\_in_i = x_i + \sum_j y_j w_{ji}.$$

Step 5. Determine activation (output signal):

$$y_i = \begin{cases} 1 & \text{if } y\_in_i > \theta_i \\ y_i & \text{if } y\_in_i = \theta_i \\ 0 & \text{if } y\_in_i < \theta_i. \end{cases}$$

Step 6. Broadcast the value of y_i to all other units.
(This updates the activation vector.)

Step 7. Test for convergence.

Example:

Testing a discrete Hopfield net: mistakes in the first and second components of the stored vector.

The input vector is (1, 1, 1, 0) or (1, 1, 1,-1)

Mistake input (0,0,1,0)

Update order (, , ,)

Step 0. Initialize weights to store patterns:

$$\mathbf{W} = \begin{bmatrix} 0 & 1 & 1 & -1 \\ 1 & 0 & 1 & -1 \\ 1 & 1 & 0 & -1 \\ -1 & -1 & -1 & 0 \end{bmatrix}.$$

Step 1. The input vector is $\mathbf{x} = (0, 0, 1, 0)$. For this vector,

Step 2. $\mathbf{y} = (0, 0, 1, 0)$.

Step 3. Choose unit Y_1 to update its activation:

Step 4. $y\_in_1 = x_1 + \sum_j y_j w_{j1} = 0 + 1.$

Step 5. $y\_in_1 > 0 \rightarrow y_1 = 1.$

Step 6. $\mathbf{y} = (1, 0, 1, 0)$.

Step 3. Choose unit Y_4 to update its activation:

Step 4. $y\_in_4 = x_4 + \sum_j y_j w_{j4} = 0 + (-2).$

Step 5. $y\_in_4 < 0 \rightarrow y_4 = 0.$

Step 6. $\mathbf{y} = (1, 0, 1, 0)$.

Step 3. Choose unit Y_3 to update its activation:

Step 4. $y\_in_3 = x_3 + \sum_j y_j w_{j3} = 1 + 1.$

Step 5. $y\_in_3 > 0 \rightarrow y_3 = 1.$

Step 6. $\mathbf{y} = (1, 0, 1, 0)$.

Step 3. Choose unit Y_2 to update its activation:

Step 4. $y\_in_2 = x_2 + \sum_j y_j w_{j2} = 0 + 2.$

Step 5. $y\_in_2 > 0 = y_2 = 1.$

Step 6. $y = (1, 1, 1, 0)$.

Step 7. Test for convergence.

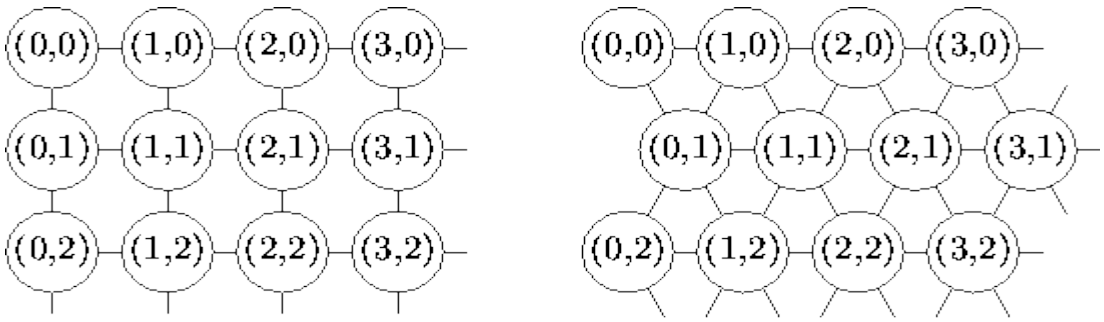
Self-Organizing Map (SOM)

The Self-Organizing Map is one of the most popular neural network models. It belongs to the category of competitive learning networks. The Self-Organizing Map is based on unsupervised learning, which means that no human intervention is needed during the learning and that little needs to be known about the characteristics of the input data. We could, for example, use the SOM for clustering data without knowing the class memberships of the input data. The SOM can be used to detect features inherent to the problem and thus has also been called SOFM, the Self-Organizing Feature Map.

The Self-Organizing Map was developed by professor Kohonen . The SOM has been proven useful in many applications . For closer review of the applications published in the open literature, see section .

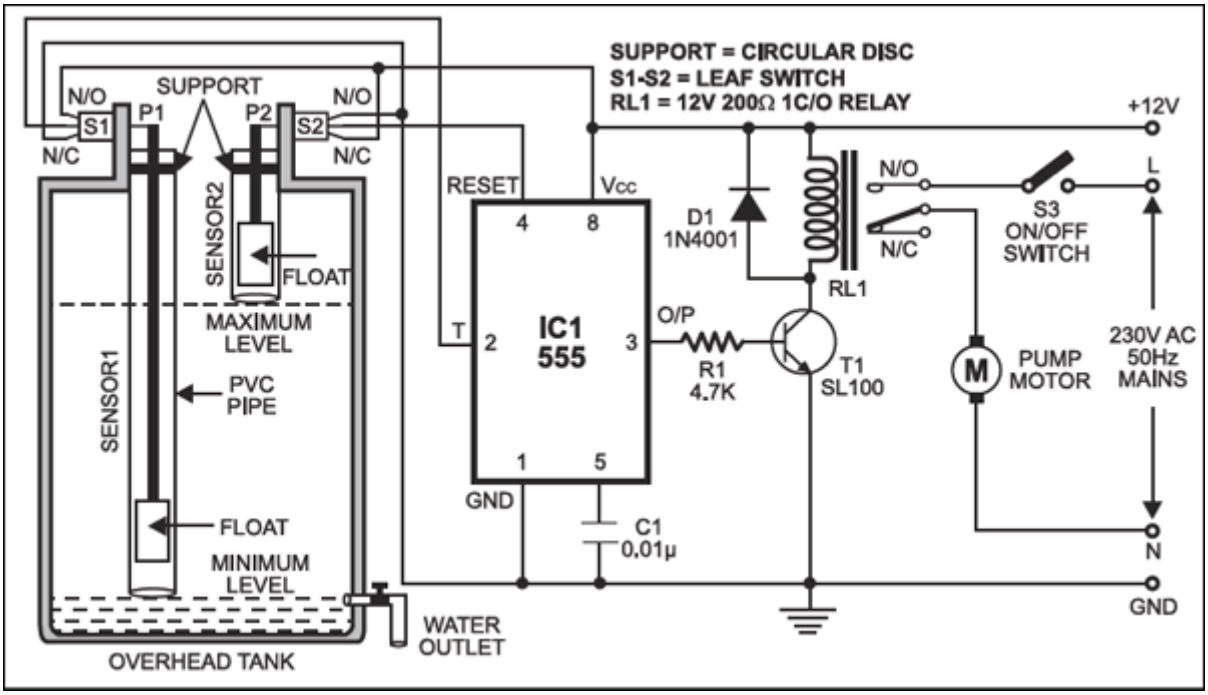
The SOM algorithm is based on unsupervised, competitive learning. It provides a topology preserving mapping from the high dimensional space to map units. Map units, or neurons, usually form a two-dimensional lattice and thus the mapping is a mapping from high dimensional space onto a plane. The property of topology preserving means that the mapping preserves the relative distance between the points. Points that are near each other in the input space are mapped to nearby map units in the SOM. The SOM can thus serve as a cluster analyzing tool of high-dimensional data. Also, the SOM has the capability to generalize. Generalization capability means that the network can recognize or characterize inputs it has never encountered before. A new input is assimilated with the map unit it is mapped to.

The Self-Organizing Map is a two-dimensional array of neurons:



This is illustrated in Figure1 . One neuron is a vector called the codebook vector.

ANN based water level controller.



TEXT / REFERENCE BOOKS

1. Laurene Fausett, " Fundamentals of Neural Networks: Architectures, Algorithms and Applications", 2008.
2. Timothy J. Ross , “Fuzzy Logic with Engineering Applications”, McGraw - Hill International Editions, 2004.
3. Jang J.S.R., Sun C.T. and Mizutani E, "Neuro-Fuzzy and soft computing", Pearson Education, 2003.
4. Rajasekaran. S, Pai. G.A.V. “Neural Networks, Fuzzy Logic and Genetic Algorithms”, Prentice Hall of India, 2003.

Part- A

1. Write the three stages of operation in BPN
2. State the important characteristics of BPN
3. Give the practical consideration of BPN
4. What is meant by CPN
5. Give the application of CPN
6. What is meant by speed energy function
7. What is meant by RBF network
8. List out statistical methods of training
9. What is Boltzmann training
10. What is BAM
11. Write about network capacity
12. wite about grossberg layer
13. What is meant by Cauchy training

Part-B

1. Explain about back propagation algorithm in detail and its limitations
2. Explain the architecture and algorithm of Full CPN
3. Explain the architecture and algorithm of Forward CPN
4. Elaborate on the training of Hopfield Network
5. Explain in detail about Discrete BAM
6. With neat diagram explain the ANN based water level controller
7. Explain the Architecture of Self Organizing Map algorithm.

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

SCHOOL OF ELECTRICAL AND ELECTRONICS

DEPARTMENT OF ELECTRONICS AND INSTRUMENTATION ENGINEERING

UNIT – III – FUZZY LOGIC – SIC 1614

Introduction to Fuzzy Set Theory - Basic Concepts of Fuzzy Sets - Classical Set Vs Fuzzy Set - Properties of Fuzzy Set - Fuzzy Logic Operation on Fuzzy Sets - Fuzzy Logic Control Principles - Fuzzy Relations - Fuzzy Rules - Defuzzification - Fuzzy Inference Systems - Fuzzy Expert Systems - Fuzzy Decision Making

Introduction to fuzzy set theory:

- In fuzzy logic, exact reasoning is viewed as a limiting case of approximate reasoning.
- In fuzzy logic, everything is a matter of degree.
- In fuzzy logic, knowledge is interpreted a collection of elastic or, equivalently, fuzzy constraint on a collection of variables.
- Inference is viewed as a process of propagation of elastic constraints.
- Any logical system can be fuzzified.

There are two main characteristics of fuzzy systems that give them better performance for specific applications.

• Fuzzy systems are suitable for uncertain or approximate reasoning, especially for the system with a mathematical model that is difficult to derive.

• Fuzzy logic allows decision making with estimated values under incomplete or uncertain information.

CLASSICAL SETS

Define a universe of discourse, X, as a collection of objects all having the same characteristics. The individual elements in the universe X will be denoted as x. The features of the elements in X can be discrete, countable integers, or continuous valued quantities on the real line.

Examples of elements of various universes might be as follows:

The clock speeds of computer CPUs;

The operating currents of an electronic motor;

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

The Richter magnitudes of an earthquake;

The integers 1 to 10.

For crisp sets A and B consisting of collections of some elements in X, the following notation is defined:

x □ X → x belongs to X

x □ A → x belongs to A

x /□ A → x does not belong to A

For sets A and B on X,

we also have

A □ B → A is fully contained in B (if x □ A, then x □ B)

A □ B → A is contained in or is equivalent to B

(A ↔ B) → A □ B and B □ A (A is equivalent to

B)

We define the null set, Ø, as the set containing no elements, and the whole set, X, as the set of all elements in the universe. The null set is analogous to an impossible event, and the whole set is analogous to a certain event. All possible sets of X constitute a special set called the power set, P(X).

For a specific universe X, the power set P(X) is enumerated in the following example.

Example

We have a universe composed of three elements, X = {a,b,c}, so the cardinal number is $n^X = 3$.

The power set is P(X) = {Ø,{a},{b},{c},{a,b},{a,c},{b,c},{a,b,c}}.

The cardinality of the power set, denoted nP(X), is found as nP(X) = 2nX = 23 = 8.

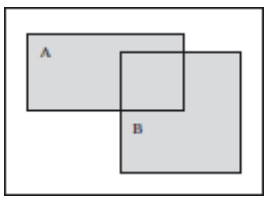
Note that if the cardinality of the universe is infinite, then the cardinality of the power set is also infinity, that is, nX =∞□ nP(X) = ∞.

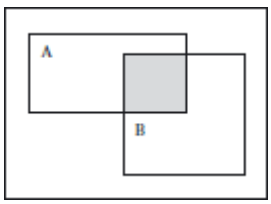
Operations on Classical Sets:

Union	$A \cup B = [x \mid x \in A \text{ or } x \in B].$
Intersection	$A \cap B = [x \mid x \in A \text{ and } x \in B].$
Complement	$\overline{A} = [x \mid x \notin A, x \in X].$
Difference	$A \mid B = [x \mid x \in A \text{ and } x \notin B].$

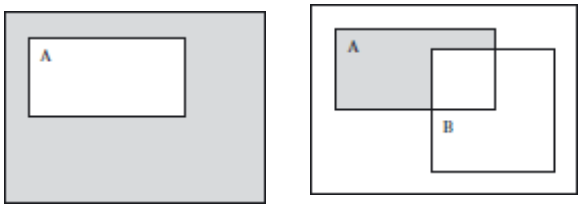
The four terms are expressed using Venn Diagrams,

Union Intersection





Complement Difference



Properties of Classical (Crisp) Sets

Certain properties of sets are important because of their influence on the mathematical manipulation of sets. The most appropriate properties for defining classical sets and showing their similarity to fuzzy sets are as follows:

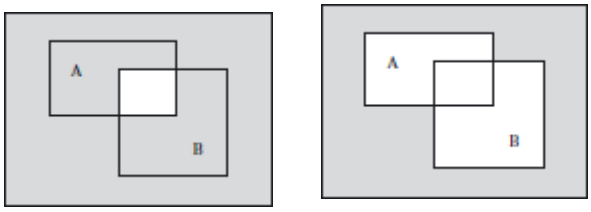
Commutativity	$A \cup B = B \cup A$
	$A \cap B = B \cap A.$
Associativity	$A \cup (B \cup C) = (A \cup B) \cup C$
	$A \cap (B \cap C) = (A \cap B) \cap C.$
Distributivity	$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$
	$A \cap (B \cup C) = (A \cap B) \cup (A \cap C).$
Idempotency	$A \cup A = A$
	$A \cap A = A.$
Identity	$A \cup \emptyset = A$
	$A \cap X = A$
	$A \cap \emptyset = \emptyset.$
	$A \cup X = X.$
Transitivity	If $A \subseteq B$ and $B \subseteq C$, then $A \subseteq C.$
Involution	$\overline{\overline{A}} = A.$

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.

Axiom of the excluded middle	$A \cup \overline{A} = X.$
Axiom of the contradiction	$A \cap \overline{A} = \emptyset.$

De Morgan's principles are important because of their usefulness in proving tautologies and contradictions in logic, as well as in a host of other set operations and proofs. De Morgan's principles are displayed in the shaded areas of the Venn diagrams

$\overline{A \cap B} = \overline{A} \cup \overline{B}.$ $\overline{A \cup B} = \overline{A} \cap \overline{B}.$



FUZZY SETS

In classical, or crisp, sets the transition for an element in the universe between membership and nonmembership in a given set is abrupt and well defined (said to be "crisp"). For an element in a universe that contains fuzzy sets, this transition can be gradual. This transition among various degrees of membership can be thought of as conforming to the fact that the boundaries of the fuzzy sets are vague and ambiguous. Hence, membership of an element from the universe in this set is measured by a function that attempts to describe vagueness and ambiguity.

A notation convention for fuzzy sets when the universe of discourse, X, is discrete and finite, is as follows for a fuzzy set A:

$$\underset{\sim}{A} = \left\{ \frac{\mu_{\underset{\sim}{A}}(x_1)}{x_1} + \frac{\mu_{\underset{\sim}{A}}(x_2)}{x_2} + \cdots \right\} = \left\{ \sum_i \frac{\mu_{\underset{\sim}{A}}(x_i)}{x_i} \right\}.$$

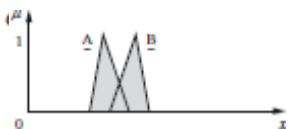
$$\underset{\sim}{A} = \left\{ \int \frac{\mu_{\underset{\sim}{A}}(x)}{x} \right\}.$$

In both notations, the horizontal bar is not a quotient but rather a delimiter. The numerator in each term is the membership value in set A□ associated with the element of the universe indicated in the denominator. In the first notation, the summation symbol is not for algebraic summation, but rather denotes the collection or aggregation of each element; hence, the "+" signs in the first notation are not the algebraic "add" but are an aggregation or collection operator. In the second notation, the integral sign is not an algebraic integral but a continuous function-theoretic aggregation operator for continuous variables.

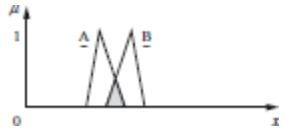
Fuzzy Set Operations

Define three fuzzy sets A□ , B □ , and C□ on the universe X. For a given element x of the universe, the following function-theoretic operations for the set-theoretic operations of union, intersection, and complement are defined for A□ , B □ , and C□ on X:

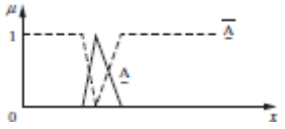
Union	$\mu_{\underset{\sim}{A} \cup \underset{\sim}{B}}(x) = \mu_{\underset{\sim}{A}}(x) \vee \mu_{\underset{\sim}{B}}(x).$
Intersection	$\mu_{\underset{\sim}{A} \cap \underset{\sim}{B}}(x) = \mu_{\underset{\sim}{A}}(x) \wedge \mu_{\underset{\sim}{B}}(x).$
	$\mu_{\overline{\underset{\sim}{A}}}(x) = 1 - \mu_{\underset{\sim}{A}}(x).$



Union of fuzzy sets A and B.



Intersection of fuzzy sets A and B.



Complement of fuzzy set A.

De Morgan's principles for classical sets also hold for fuzzy sets, as denoted by the following expressions:

$$\overline{A \cap B} = \overline{A} \cup \overline{B}.$$
$$\overline{A \cup B} = \overline{A} \cap \overline{B}.$$

As enumerated before, 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,

$$A \cup \overline{A} \neq X.$$
$$A \cap \overline{A} \neq \emptyset.$$

Properties of Fuzzy Sets:

Fuzzy sets follow the same properties as crisp sets. Because of this fact and because the membership values of a crisp set are a subset of the interval [0,1], classical sets can be thought of as a special case of fuzzy sets.

1. A□ □ B□ is the set of loadings for which one expects that either material B or materialD will be "safe."

2. A□ ∩ B □ is the set of loadings for which one expects that both material B and materialD are "safe." 3. A □ and B □ are the sets of loadings for which material D and material B are unsafe, respectively.

4. A□ |B □ is the set of loadings for which the ductile material is safe but the brittle material is in jeopardy.

5. B□ |A □ is the set of loadings for which the brittle material is safe but the ductilematerial is in jeopardy.

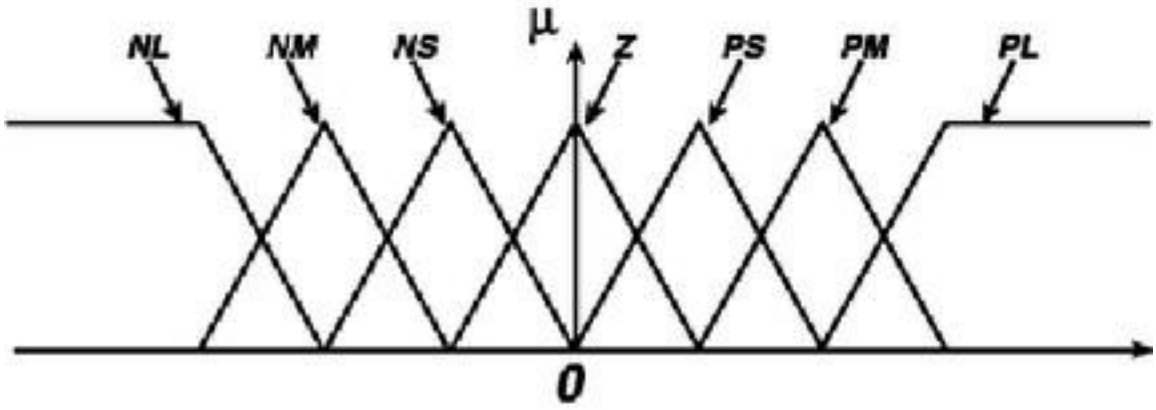
6. De Morgan's principle A □ ∩ B □ = A□ □ B □ asserts that the loadings that are not safe with respect to both materials are the union of those that are unsafe with respect to the brittle material with those that are unsafe with respect to the ductile material.

7. De Morgan's principle A □ □ B □ = A □ ∩ B □ asserts that the loads that are safe for neither material D nor material B are the intersection of those that are unsafe for material D with those that are unsafe for material B.

Fuzzy sets vs. crisp sets

Crisp sets are the sets that we have used most of our life. In a crisp set, an element is either a member of the set or not. For example, a jelly bean belongs in the class of food known as candy. Mashed potatoes do not.

Fuzzy sets, on the other hand, allow elements to be *partially* in a set. Each element is given a degree of membership in a set. This membership value can range from 0 (not an element of the set) to 1 (a member of the set). It is clear that if one only allowed the extreme membership values of 0 and 1, that this would actually be equivalent to crisp sets. A membership function is the relationship between the values of an element and its degree of membership in a set. An example of membership functions are shown in the figure. In this example, the sets (or classes) are numbers that are negative large, negative medium, negative small, near zero, positive small, positive medium, and positive large. The value, μ, is the amount of membership in the set.



CARTESIAN PRODUCT

An ordered sequence of r elements, written in the form (a1, a2, a3,...,ar), is called an ordered r-tuple; an unordered r-tuple is simply a collection of r elements without restrictions on order. In a ubiquitous special case where r = 2, the r-tuple is referred to as an ordered pair. For crisp sets A1, A2,..., Ar, the set of all r-tuples (a1, a2, a3,...,ar), where a1 □ A1, a2 □ A2, and ar □ Ar, is called the Cartesian product of A1, A2,..., Ar, and is denoted by A1 × A2 ×⋯× Ar. The Cartesian product of two or more sets is not the same thing as the arithmetic product of two or more sets. The latter is dealt with in Chapter 12, when the extension principle is introduced. When all the Ar are identical and equal to A, the Cartesian product A1 × A2 × ⋯× Ar can be denoted as Ar .

FUZZY RELATIONS

Fuzzy relations also map elements of one universe, say X, to those of another universe, say Y, through the Cartesian product of the two universes. However, the "strength" of the relation

between ordered pairs of the two universes is not measured with the characteristic function, but rather with a membership function expressing various "degrees" of strength of the relation on the unit interval [0,1]. Hence, a fuzzy relation 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 of the relation for ordered pairs from the two universes, or μR □ (x, y).

Cardinality of Fuzzy Relations

Since the cardinality of fuzzy sets on any universe is infinity, the cardinality of a fuzzy relation between two or more universes is also infinity. Operations on Fuzzy Relations 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	$\underset{\sim}{R} \subset \underset{\sim}{S} \Rightarrow \mu_{\underset{\sim}{R}}(x, y) \leq \mu_{\underset{\sim}{S}}(x, y).$

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□ is also a fuzzy set, there is overlap between a relation and its complement; hence,

$$\underset{\sim}{R} \cup \overline{\underset{\sim}{R}} \neq \mathbf{E}.$$
$$\underset{\sim}{R} \cap \overline{\underset{\sim}{R}} \neq \mathbf{O}.$$

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

Fuzzy Cartesian Product and Composition

Because fuzzy relations in general are fuzzy sets, we can define the Cartesian product to be a relation between two or more fuzzy sets. Let A□ be a fuzzy set on universe X and B□ be a fuzzy set on universe Y, then the Cartesian product between fuzzy sets A□ and B□ will result in a fuzzy relation R□ , which is contained within the full Cartesian product space, or

$$\underset{\sim}{A} \times \underset{\sim}{B} = \underset{\sim}{R} \subset X \times Y,$$

where the fuzzy relation R□ has membership function

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

Example .

$$X = \{x_1, x_2\}, \quad Y = \{y_1, y_2\}, \quad \text{and} \quad Z = \{z_1, z_2, z_3\}.$$

$$\underset{\sim}{R} = \begin{matrix} \\ x_1 \\ x_2 \end{matrix} \begin{matrix} y_1 & y_2 \end{matrix} \begin{bmatrix} 0.7 & 0.5 \\ 0.8 & 0.4 \end{bmatrix} \quad \text{and} \quad \underset{\sim}{S} = \begin{matrix} \\ y_1 \\ y_2 \end{matrix} \begin{matrix} z_1 & z_2 & z_3 \end{matrix} \begin{bmatrix} 0.9 & 0.6 & 0.2 \\ 0.1 & 0.7 & 0.5 \end{bmatrix}.$$

Then, the resulting relation, T□ , which relates elements of universe X to elements of universe Z, that is, defined on Cartesian space X × Z, can be found by max–min composition

$$\mu_{\underset{\sim}{T}}(x_1, z_1) = \max[\min(0.7, 0.9), \min(0.5, 0.1)] = 0.7,$$

The rest,

$$\underset{\sim}{T} = \begin{matrix} \\ x_1 \\ x_2 \end{matrix} \begin{matrix} z_1 & z_2 & z_3 \end{matrix} \begin{bmatrix} 0.7 & 0.6 & 0.5 \\ 0.8 & 0.6 & 0.4 \end{bmatrix},$$

and by max–product composition

$$\mu_{\underset{\sim}{T}}(x_2, z_2) = \max[(0.8 \cdot 0.6), (0.4 \cdot 0.7)] = 0.48,$$

The rest,

$$\underset{\sim}{T} = \begin{matrix} \\ x_1 \\ x_2 \end{matrix} \begin{matrix} z_1 & z_2 & z_3 \end{matrix} \begin{bmatrix} 0.63 & 0.42 & 0.25 \\ 0.72 & 0.48 & 0.20 \end{bmatrix}.$$

FUZZY TOLERANCE AND EQUIVALENCE RELATIONS

A fuzzy relation, R□ , on a single universe X is also a relation from X to X. It is a fuzzy equivalence relation if all three of the following properties for matrix relations define it:

Reflexivity $\mu_{\underset{\sim}{R}}(x_i, x_i) = 1.$

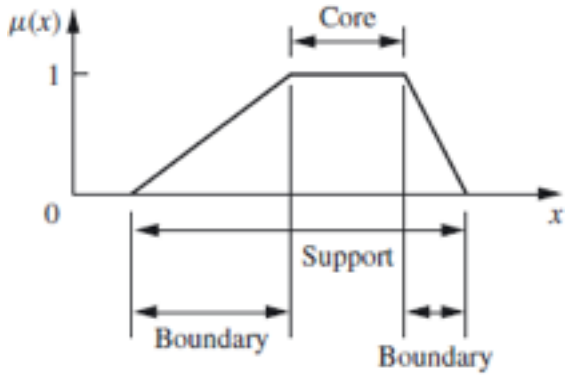
Symmetry $\mu_{\underset{\sim}{R}}(x_i, x_j) = \mu_{\underset{\sim}{R}}(x_j, x_i).$

Transitivity $\mu_{\underset{\sim}{R}}(x_i, x_j) = \lambda_1 \quad \text{and} \quad \mu_{\underset{\sim}{R}}(x_j, x_k) = \lambda_2 \longrightarrow \mu_{\underset{\sim}{R}}(x_i, x_k) = \lambda,$

where $\lambda \geq \min[\lambda_1, \lambda_2]$.

FEATURES OF THE MEMBERSHIP FUNCTION

Since all information contained in a fuzzy set is described by its membership function, it is useful to develop a lexicon of terms to describe various special features of this function. For purposes of simplicity, the functions shown in the figures will all be continuous, but the terms apply equally for both discrete and continuous fuzzy sets.



The core of a membership function for some fuzzy set A□ is defined as that region of the universe that is characterized by complete and full membership in the set A□ . That is, the core comprises those elements x of the universe such that μA □ (x) = 1.

The support of a membership function for some fuzzy set A□ is defined as that region of the universe that is characterized by nonzero membership in the set A□ . That is, the support comprises those elements x of the universe such that μA □ (x) > 0.

The boundaries of a membership function for some fuzzy set A□ are defined as that region of the universe containing elements that have a nonzero membership but not complete membership. That is, the boundaries comprise those elements x of the universe such that 0 < μA □ (x) < 1. These elements of the universe arethose with some degree of fuzziness, or only partial membership in the fuzzy set A□ .

FUZZIFICATION

Fuzzification is the process of making a crisp quantity fuzzy. We do this by simply recognizing that many of the quantities that we consider to be crisp and deterministic are actually not deterministic at all; they carry considerable uncertainty. If the form of uncertainty happens to arise because of imprecision, ambiguity, or vagueness, then the variable is probably fuzzy and can be represented by a membership function.

DEFUZZIFICATION TO CRISP SETS

We begin by considering a fuzzy set A□ , then define a lambda-cut set, Aλ, where 0 ≤ λ ≤ 1. The set Aλ is a crisp set called the lambda (λ)-cut (or alpha-cut) set of the fuzzy set A□ , where Aλ = {x|μA □ (x) ≥ λ}. Note that the λ-cut set Aλ does not have a tilde underscore; it is a crisp set derived from its parent fuzzy set, A□ . Any particular fuzzy set A□ can be transformed into an infinite number of λ-cut sets, because there are an infinite number of values λ on the interval [0, 1].

Any element x □ Aλ belongs to A□ with a grade of membership that is greater than or equal to the value λ.

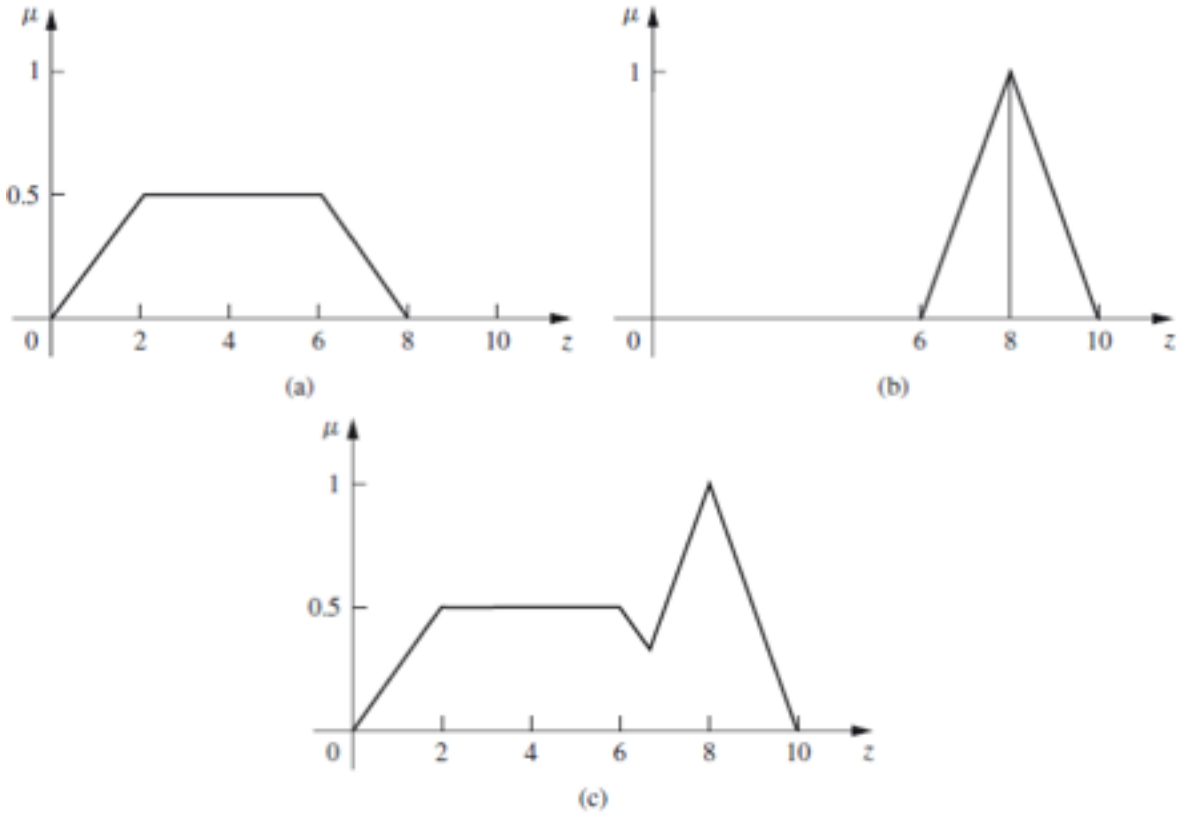
DEFUZZIFICATION TO SCALARS

As mentioned in the introduction, there may be situations where the output of a fuzzy process needs to be a single scalar quantity as opposed to a fuzzy set. Defuzzification is the conversion of a fuzzy quantity to a precise quantity, just as fuzzification is the conversion of a precise quantity to a fuzzy quantity. The output of a fuzzy process can be the logical union of two or more fuzzy membership functions defined on the universe of discourse of the output variable. For example, suppose a fuzzy output comprises two parts:

(1) C□ 1, a trapezoidalshape

(2) C□ 2, a triangular membership shape.

The union of these two membership functions, that is, C□ = C □ 1 □ C □ 2, involves the max operator, which graphically is the outer envelope of the two shapes shown



Typical fuzzy process output: (a) first part of fuzzy output; (b) second part of fuzzy output; and (c) union of both parts.

Of course, a general fuzzy output process can involve many output parts (more than two), and the membership function representing each part of the output can have shapes other than triangles and trapezoids.

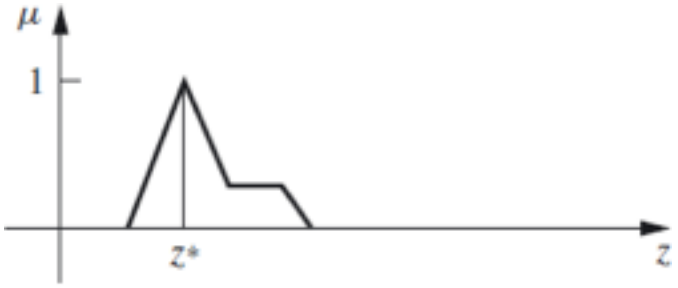
$$\underset{\sim}{C}_k = \bigcup_{i=1}^{k} \underset{\sim}{C}_i = \underset{\sim}{C}.$$

Among the many methods that have been proposed in the literature in recent years, seven are described here for defuzzifying fuzzy output functions (membership functions).

1. Max membership principle: Also known as the height method, this scheme is limited to peaked output functions. This method is given by the algebraic expression,

$$\mu_{\tilde{C}}(z^*) \geq \mu_{\tilde{C}}(z), \qquad \text{for all } z \in Z,$$

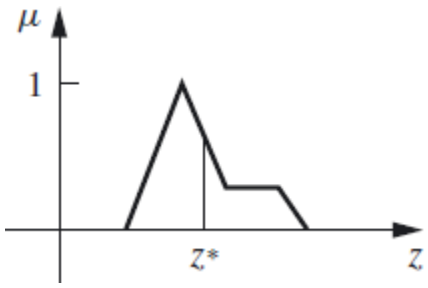
where z□ is the defuzzified value.



2. Centroid method: This procedure (also called center of area or center of gravity) is the most prevalent and physically appealing of all the defuzzification methods.

$$z^* = \frac{\int \mu_{\tilde{C}}(z) \cdot z \, dz}{\int \mu_{\tilde{C}}(z) \, dz},$$

where ∫ denotes an algebraic integration.

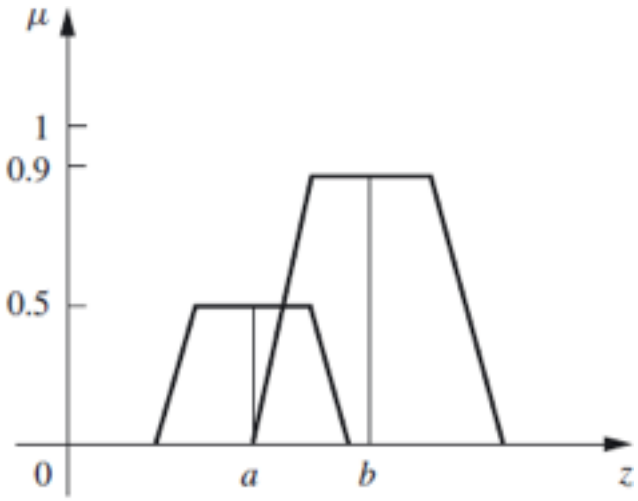


3. Weighted average method: The weighted average method is the most frequently used in fuzzy applications since it is one of the more computationally efficient methods. Unfortunately, it is usually restricted to symmetrical output membership functions. It is given by the algebraic expression,

$$z^* = \frac{\sum \mu_{\tilde{C}}(\bar{z}) \cdot \bar{z}}{\sum \mu_{\tilde{C}}(\bar{z})},$$

Where ∑ denotes the algebraic sum and where z is the centroid of each symmetric

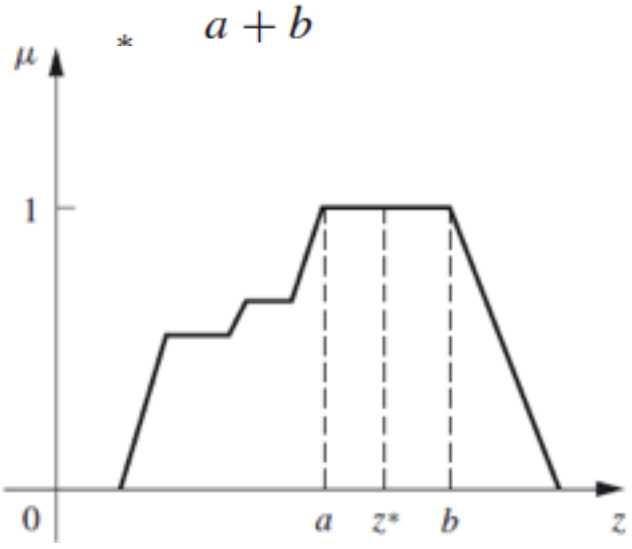
membership function.



The weighted average method is formed by weighting each membership function in the output by its respective maximum membership value.

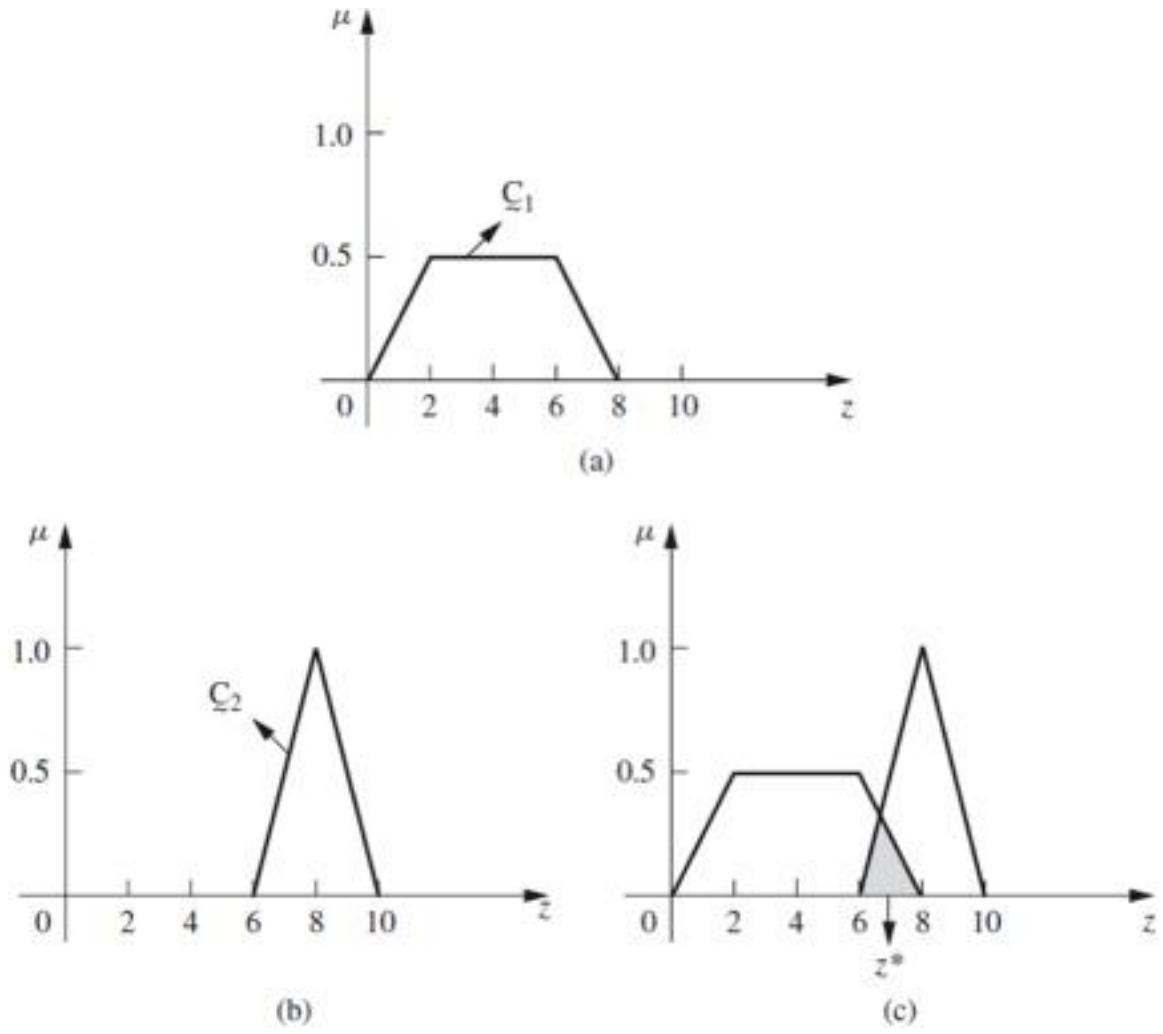
$$z^* = \frac{a(0.5) + b(0.9)}{0.5 + 0.9}.$$

4. Mean max membership: This method (also called middle-of-maxima) is closely related to the first method, except that the locations of the maximum membership can be nonunique (i.e., the maximum membership can be a plateau rather than a single point).



5. Center of sums: This is faster than many defuzzification methods that are currently in use, and the method is not restricted to symmetric membership functions. This process involves the algebraic sum of individual output fuzzy sets, say C□ 1 and C□ 2, instead of their union. Two drawbacks to this method are that the intersecting areasare added twice, and the method also involves finding the centroids of the individual membership functions. The defuzzified value z□ is given as folbws:

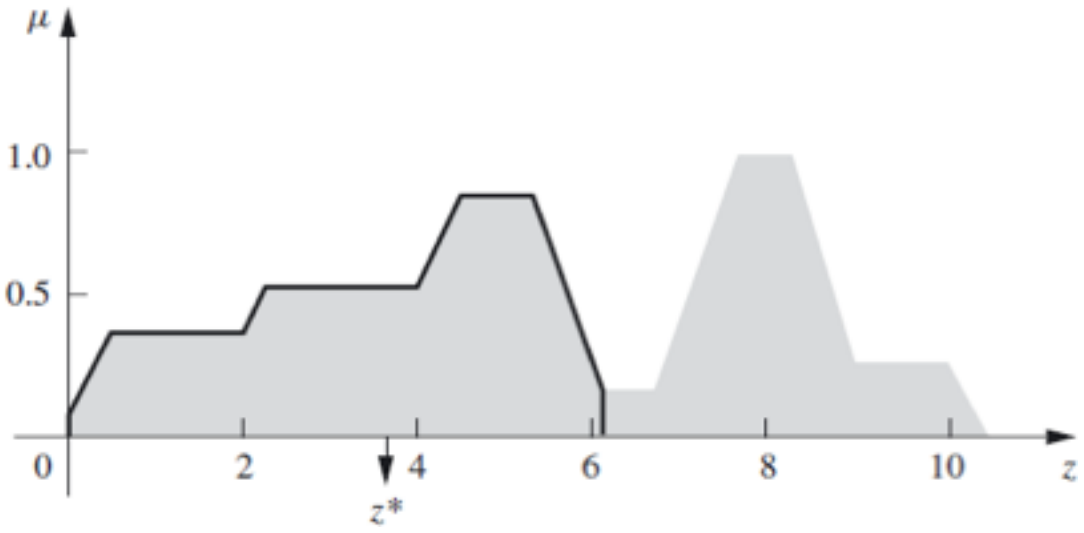
$$z^* = \frac{\sum_{k=1}^{n} \mu_{\underset{\sim}{C}_k}(z) \int_z \bar{z}\, dz}{\sum_{k=1}^{n} \mu_{\underset{\sim}{C}_k}(z) \int_z dz},$$



Center of sums method: (a) first membership function; (b) second membership function; and (c) defuzzification step.

6. Center of largest area: If the output fuzzy set has at least two convex subregions, then the center of gravity (i.e., z□ is calculated using the centroid method, Equation 4.5)of the convex fuzzy subregion with the largest area is used to obtain the defuzzified value z□ of the output.

$$z^* = \frac{\int \mu_{\underset{\sim}{C}_m}(z) z \, dz}{\int \mu_{\underset{\sim}{C}_m}(z) \, dz},$$



7. First (or last) of maxima: This method uses the overall output or union of all individual output fuzzy sets C□ k to determine the smallest value of the domain with maximized membership degree in C□ k. The equations for z□ are asfollows.

   First, the largest height in the union (denoted hgt(C □ k)) is determined,

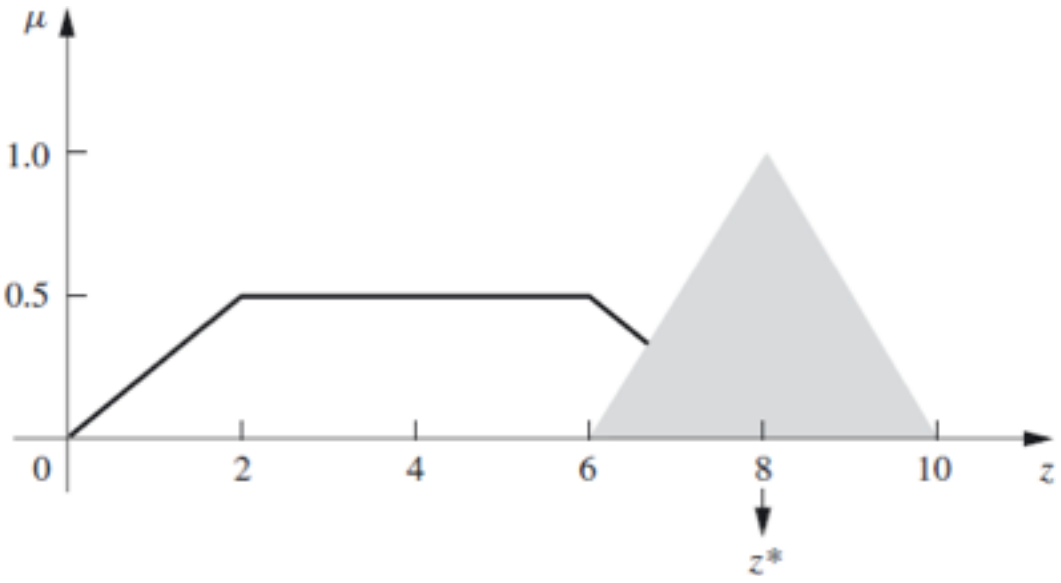
$$\text{hgt}(\underset{\sim}{C}_k) = \sup_{z \in Z} \mu_{\underset{\sim}{C}_k}(z).$$

   Then, the first of the maxima is found,

$$z^* = \inf_{z \in Z} \{z \in Z | \mu_{\underset{\sim}{C}_k}(z) = \text{hgt}(\underset{\sim}{C}_k)\}.$$

   An alternative to this method is called the last of maxima, and it is given as

$$z^* = \sup_{z \in Z} \{z \in Z | \mu_{\tilde{C}_k}(z) = \text{hgt}(\tilde{C}_k)\}.$$



Assumptions in a Fuzzy Control System

Design 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.

1. The plant is observable and controllable: state, input, and output variables are usually available for observation and measurement or computation.

2. 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.

3. A solution exists.

4. The control engineer is looking for a "good enough" solution, not necessarily the optimum one.

5. The controller will be designed within an acceptable range of precision.

6. The problems of stability and optimality are not addressed explicitly; such issues are still open problems in fuzzy controller design.

The following section discusses the procedure for obtaining the control surface, h(·), from approximations based on a collection of fuzzy IF–THEN rules that describe the dynamics of the controller.

Simple Fuzzy Logic Controllers

First-generation (nonadaptive) simple fuzzy controllers can generally be depicted by a block diagram such as that shown in Figure. The knowledge-base module in Figure contains knowledge about all the input and output fuzzy partitions. It will include the term set and the corresponding membership functions defining the input variables to the fuzzy rule-base system and the output variables, or control actions, to the plant under control.

The steps in designing a simple fuzzy control system are as follows:

1. Identify the variables (inputs, states, and outputs) of the plant.

2. Partition the universe of discourse or the interval spanned by each variable into a number of fuzzy subsets, assigning each a linguistic label (subsets include all the elements in the universe).

3. Assign or determine a membership function for each fuzzy subset.

4. Assign the fuzzy relationships between the inputs' or states' fuzzy subsets on the one hand and the outputs' fuzzy subsets on the other hand, thus forming the rule-base.

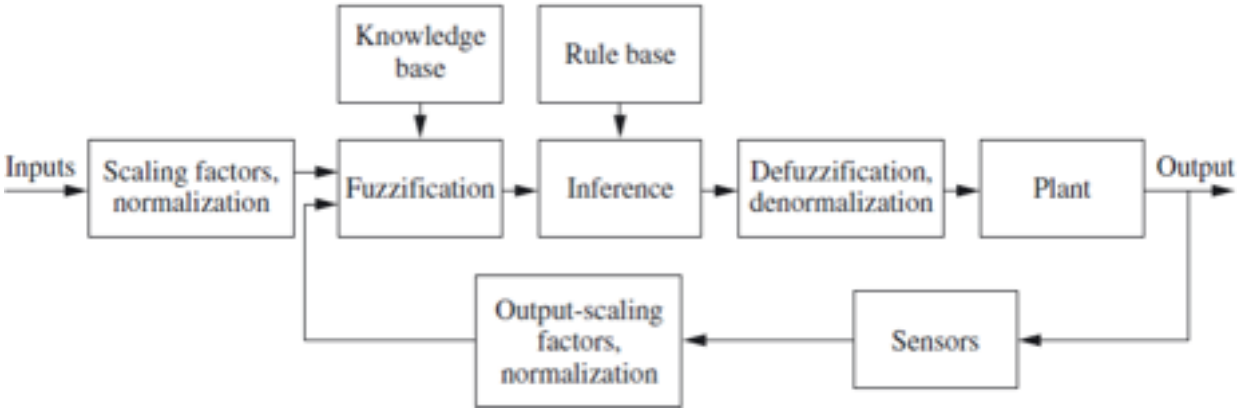
5. Choose appropriate scaling factors for the input and output variables in order to normalize the variables to the [0, 1] or the [−1, 1] interval.

6. Fuzzify the inputs to the controller.

7. Use fuzzy approximate reasoning to infer the output contributed from each rule.

8. Aggregate the fuzzy outputs recommended by each rule.

9. Apply defuzzification to form a crisp output.



EXAMPLES OF FUZZY CONTROL SYSTEM DESIGN

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. 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, $\dot{(e)}$. 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 $\dot{(e)}$ is "negative small (NS)," THEN heat input change is "negative medium (NM)."

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).

TEXT / REFERENCE BOOKS

1. Laurene Fausett, " Fundamentals of Neural Networks: Architectures, Algorithms and Applications", 2008.
2. Timothy J. Ross , "Fuzzy Logic with Engineering Applications", McGraw - Hill International Editions, 2004.
3. Jang J.S.R., Sun C.T. and Mizutani E, "Neuro-Fuzzy and soft computing", Pearson Education, 2003.
4. Rajasekaran. S, Pai. G.A.V. "Neural Networks, Fuzzy Logic and Genetic Algorithms", Prentice Hall of India, 2003.

Part- A

1. Differentiate between crisp and fuzzy logic
2. Define fuzzy set
3. Define membership function
4. What is fuzzification
5. Define defuzzification
6. What is the advantage of fuzzy logic
7. What is FLC
8. What are Demorgan's laws
9. What is meant by decision making
10. Define fuzzy inference system
11. What is meant by Linguistic pharses
12. What is meant by alpha cut
13. Give the different types of membership functions
14. What are the basic elements of a fuzzy logic control system

Part-B

1. Discuss in detail about the fuzzy set theory
2. Write the operations and properties of fuzzy set
3. Differentiate classical set theory and fuzzy set using examples
4. Discuss in detail about the Fuzzy logic controller
5. Write in detail about the fuzzy relations and fuzzy rules
6. Explain in detail about the Defuzzification methods
7. Find max-min composition for the function

$$\tilde{R} = \begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \end{array} \begin{array}{c} \begin{array}{ccc} y_1 & y_2 & y_3 \end{array} \\ \begin{bmatrix} 0.7 & 0.8 & 0.5 \\ 0.2 & 0.4 & 0.6 \\ 0.1 & 0.9 & 0.4 \end{bmatrix} \end{array}$$

$$\tilde{T} = \begin{array}{c} \\ y_1 \\ y_2 \\ y_3 \end{array} \begin{array}{c} \begin{array}{ccc} z_1 & z_2 & z_3 \end{array} \\ \begin{bmatrix} 0.3 & 0.2 & 0.5 \\ 0.2 & 0.8 & 0.6 \\ 0.6 & 0.9 & 0.1 \end{bmatrix} \end{array}$$

8. If $\widetilde{D_1} = \{\frac{1}{1} + \frac{0.6}{1.5} + \frac{0.2}{2} + \frac{0.1}{2.5} + \frac{0}{3}\}$

$$\widetilde{D_2} = \left\{\frac{1}{1} + \frac{0.6}{1.5} + \frac{0.2}{2} + \frac{0.1}{2.5} + \frac{0}{3}\right\}$$

Find Complement, Union, Intersection and Difference

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

SCHOOL OF ELECTRICAL AND ELECTRONICS

DEPARTMENT OF ELECTRONICS AND INSTRUMENTATION ENGINEERING

UNIT – IV – FUZZY LOGIC CONTROLLER AND ITS APPLICATION – SIC 1614

Fuzzy Logic Controller - Fuzzification Interface - Knowledge Base- Decision Making Logic – Defuzzification Interface- Application of Fuzzy Logic to Water Level Controller - Temperature Controller - Control of Blood Pressure during Anaesthesia. Introduction to Neuro - Fuzzy Systems - Fuzzy System Design Procedures - Fuzzy Sets and Logic Background - Fuzzy / ANN Design and Implementation

General fuzzy logic controller:

- The principle design elements in a general fuzzy logic control system.

1. Fuzzification strategies and the interpretation of a Fuzzification operator, or fuzzifier.
2. Knowledge base:

   Discretization/ normalization of the universe of

   discourse. Fuzzy partitions of the input and output

   spaces.

   Completeness of the partitions.

   Choice of membership functions of a primary fuzzy set.
3. Rule base:

   Choice of process state (input) variables and control (output)

   variables. Source of derivation of fuzzy control rules

   Types of fuzzy control rules

   Consistency, interactivity and completeness of fuzzy control rules
4. Decision making logic:

   Definition of a fuzzy implication

   Interpretation of the sentence

   connective and Inference mechanism
5. Defuzzification strategies and the interpretation of a Defuzzification operator (defuzzifier).

B.4.1 Fuszy-F.o$pc *n*.ecl>Io« 'on ction Crezttterl Lry Fuaaifyiog
RnnLmn Rm

This performs a transformation of the Boolean to smoothen their boundaries, creating a probabilistic region. The algorithm consists of the following steps. (i) Quantize the patient parameters. For example, heart

rat ‹r in 1›n:›kmJ ‹JHH”\I hlt ‹› IH st‹'Ia ‹›f I \J lJI’mI / ‹'I› 1›ku›‹1 I›r 'I .tir‹' in \›r‹*kk‹n* rt‹e 'ti il I › I ‹› ȷ•t \1w t›{ I i1I roll t/\lp/ nt \ [›. ‹Mt'¿‹qI xnlt tr I ir›l i» I iFi kt•lj ‹Ir in.'jɪ it›t i› 1(I nl‹1x ‹y} .'›* \ ‹'r nt‹•]› IM t‹i r wtiitzti orv rJ‹t \• in 1›n kt'lI ‹ l \›a'JI ilJt‹› I ‹j nl ‹•] n
of 10/step. (ii) A 4-dimensional matrix is created with one quantized patient

parameter on each dimension. Each element in matrix is assigned a probability P_r of either 100% or 0% based on the Boolean rules. (iii)A new 4-dimensional iJ ztlrix La t'r •J‹t‹s I •'itI Il\t• ‹lii cxiv j¥,s I h•• fiMt. fi c IJ ••l 'Hit'H\ ‹ s{ tual is fuzzified over window of size ‘w’ and assigned a probability P_f using the Eq. 6.10
cix I ,

^y /'IUII^'II*I— Z Z I Z 8.U'I['/Il•I1•I/'* (•*‹›)

where d is given by Eq. 6.11, and p, q, r, s indicate the four patient parameters and i, j, k, l indicate the dimension of first, second, third and fourth parameters respectively.

$$d = [1 + \sqrt{(p^2 + q^2 + r^2 + s^2)}] \qquad (\tilde{n}.\ I\ l\ )$$

$P_f(\max)$ is the maximum value in Eq. (6.12)

$$P_{f'}[i][j][k][l] = (P_f[i][j][k][l]/P_f(\max))x100\% \qquad (6.12)$$

Eq. 6.12 is used to normalize the probabilities P_f so that the highest probability is 100%.
P_f (max) is obtained after calculating all of the probabilities, P_f.

Currently, the algorithm is implemented in 4-space with four patient parameters and the fuzzification is based on the distances between parameters

over a four dimensional window of size ‘w’. It can be extended
mensions by considering ‘n’ parameters and fuzzification can be done over a ‘n’ dimensional window. In implementation, the fuzzification of probabilities

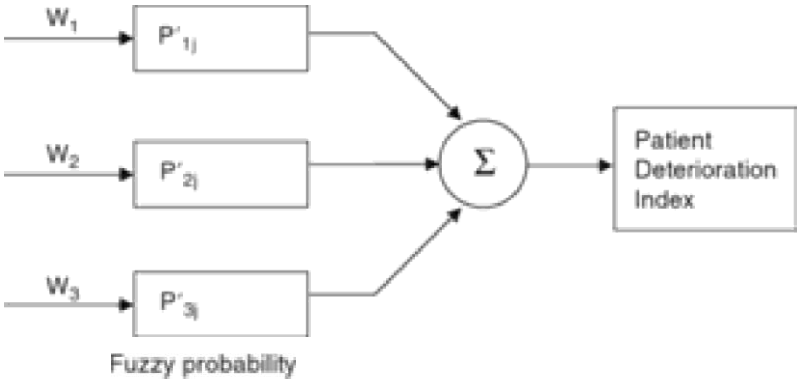
is not done over entire matrix but only over elements in the neighborhood of window 'w'. This is done to reduce computational time. Also, by reducing the size of the neighborhood, the fuzzified boundaries become sharper.

8. 4.2 Eiiaay-Logic Patient DeterlorRtion I Hdex

I. i f‹• I fln •tit ‹ ' Billy ‹ ' 'hI,s 1 i k‹' C•h S"i'i it ri‹ "H1«r L"nilJ tr •. I' Ill l i1 nrt' ‹w \‹•tMti. a tl i i It iIz;l I ¥'t'liI ris1tl r L li ltiu• ztn• J‹ i has I \r' il J I u'\ñ Its I zu•ts I ‹A i IN t‹' r •k fHs1‹›r. l'ztt it•til U‹•t «i‹/rJ‹t i\/l\ Ill I 'x ir tto lt'I\<l ru• nlJvV' tl iJ \ till' Lai . G.• . tt"‹•iphl n +- i¥,Its^I t it F}s|*t {A -I c UI JsTi' /i t"‹"[l i i› n) i]‹• Is. •}. {'III j‹ 'III ‹ i•< 'ri r•t i ›i› tI1‹]I'X > f‹ ›£It t\I) tt‹ n I t‹ › JLwwas I] t‹ ' t ¥1 I I‹-It I I I t' i f t lJ‹• -‹ ›lH {1t ic ij1 c i L t 1 H' [J‹tl H'llf - Al jn I u\.w s}

i III I iii • fi izx;' }r4i',ic-]iy‹›f u\I ›i [i\ li',s c**j t lan x • ‹ \ifF •ri'J It c-rit ice\I ‹ zl i‹ lil i i lj,s ‹*} t lz •

li‹ 'art . l'nt i ml IN 't‹ 'ri‹ ›r ztl i‹ ›ti JT t‹ li x (/r,i,) is cixx'H 1 i' t I ‹ ' fia I' G..1 \.



£"i:s- 6.5. Pnti‹V il 0•'l •'ri ›rn11MIl I\1‹I‹'x Fi tri‹ 1 i‹ II at r Art s¥yi•

Table 6.4. Weights assigned to Cardiovascular Problems

Coizsll£ lu M	R\sk E'nrter{Wx)
Left Ventricular Failure	0.40
Right Ventricular Failure	0.10
Pulmonary edema	0.50

Where k ref ft ventricular failure, right v P'_{kj} is the corresponding fu x ranges between 0 and 1 where 0 indicates that patient has no deterioration and 1 indicates maximum deterioration.

The weights (W_k) are assi

$$\mu_{di}(j) = \sum_k W_k P'_{kj} \qquad (6.13)$$

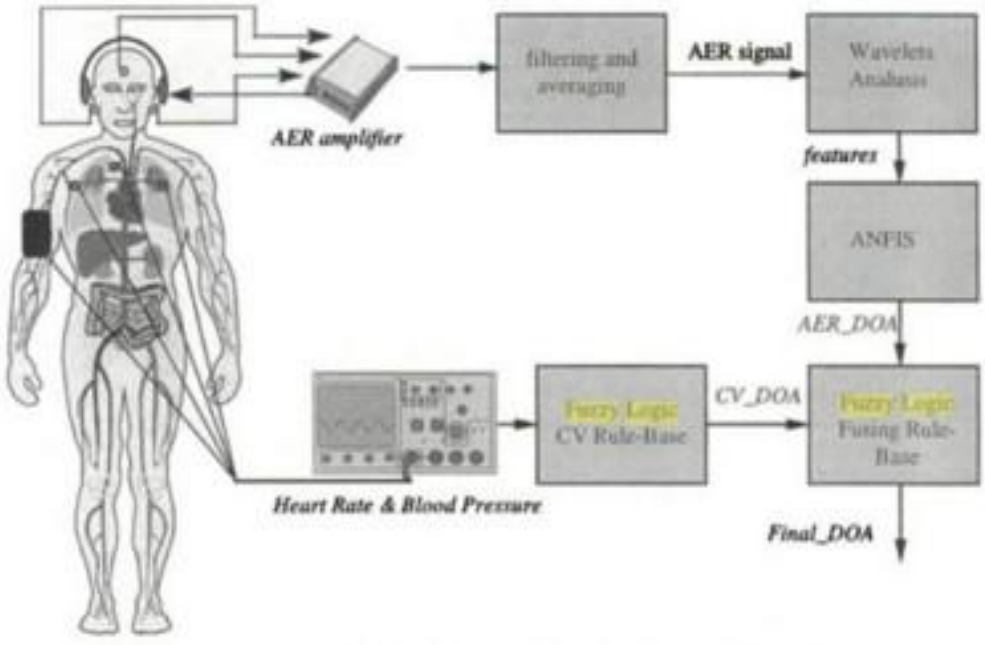
Among the four cardiac abnormalities, pulmonary edema is considered to be most critical and hence assigned a higher weight. Patient Deterioration Index (μ_{di}) value depends on the weights assigned in Table 6.4. Normal subjects do not have any risk factors, so the weight assigned is 0. Fuzzy probability (P'_{kj}) will yield the percentage of the disorders in the subject considered. The weighted sum of these probabilities will yield a single index, which indicates the cardiac health state.

Application Of Fuzzy Logic To Control Of Blood Pressure During Anaesthesia – Cardio Vascular Signals

The field of Depth of Anaesthesia (DOA) is a challenging area for fuzzy control since direct measurements are unavailable. Thus, a hierarchical structure containing

fuzzy reasoning at each level has being developed [16]. This provides self-organising fuzzy modelling (SOFM) at the lowest level for inferential monitoring of anaesthetic depth. At the next level, regulatory control is achieved for the anaesthesia maintenance phase via a self-organising fuzzy logic controller (SOFLC). This hierarchy is extended to higher several levels for adaptation, supervision, surgical evaluation and alarm monitoring. Thus, it demonstrates a totally fuzzy-logic-based architecture for manipulating complex process (i.e. the patient) [25].

An existing fuzzy logic-based programme named "SADA" provides good control algorithms with self-learning capabilities, but only limited measurement inference based mostly on cardiovascular indicators (i.e. changes in blood pressure (BP) and heart rate (HR)) [16,17]. However, these changes are subject to moderation by drugs used in anaesthesiology (e.g. opioids and anticholinergics), by surgical situation (e.g. body position) or patient's circumstance (e.g. fluid loss and blood loss), so they are unreliable. In the search for a reliable monitor of DOA the electroencephalogram (EEG) is an obvious signal to investigate. The EEG, generated from within the central nervous system (CNS), is not affected by neuromuscular blockers, and the raw signal has been known for some time to show graded changes with increasing concentration of anaesthetics. However, these changes vary for different agents and no unifying feature has been identified for measurement of DOA in spite of extensive signal processing.



EXT / REFERENCE BOOKS

1. Laurene Fausett, " Fundamentals of Neural Networks: Architectures, Algorithms and Applications", 2008.
2. Timothy J. Ross , “Fuzzy Logic with Engineering Applications”, McGraw - Hill International Editions, 2004.
3. Jang J.S.R., Sun C.T. and Mizutani E, "Neuro-Fuzzy and soft computing", Pearson Education, 2003.
4. Rajasekaran. S, Pai. G.A.V. “Neural Networks, Fuzzy Logic and Genetic Algorithms”, Prentice Hall of India, 2003.

Part- A

1. What is a fuzzy logic controller
2. What is fuzzy decision making
3. What is meant by defuzzification interface
4. Write about knowledge base
5. What is meant by neuro fuzzy system
6. What is meant by fuzzification interface

Part-B

1. Explain FLC with neat diagram and discuss advantages and disadvantages
2. Elaborate in detail about application of FLC for water level Controller
3. Write in detail about FLC for Temperature Controller
4. Explain in detail about application of FLC for the control of Blood Pressure during Anesthesia
5. Explain in detail about the design and Implementation of Neuro fuzzy system

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

SCHOOL OF ELECTRICAL AND ELECTRONICS
DEPARTMENT OF ELECTRONICS AND INSTRUMENTATION ENGINEERING

UNIT – V – GENETIC ALGORITHMS – SIC1614

Introduction - Robustness of Traditional Optimization and Search Techniques - The goals of optimization - Survival of the Fittest - Fitness Computations - Cross over - Mutation -Reproduction- Rank method- Rank space method

1.Introduction

The genetic algorithm (GA), developed by John Holland and his collaborators in the 1960s and 1970s (Holland, 1975; De Jong, 1975), is a model or abstraction of biological evolution based on Charles Darwin's theory of natural selection. Holland was probably the first to use the crossover and recombination, mutation, and selection in the study of adaptive and artificial systems. These genetic operators form the essential part of the genetic algorithm as a problem-solving strategy. Since then, many variants of genetic algorithms have been developed and applied to a wide range of optimization problems, from graph coloring to pattern recognition, from discrete systems (such as the travelling salesman problem) to continuous systems (e.g., the efficient design of airfoil in aerospace engineering), and from financial markets to multi-objective engineering optimization.

There are many advantages of genetic algorithms over traditional optimization algorithms. Two most notable are: the ability of dealing with complex problems and parallelism. Genetic algorithms can deal with various types of optimization, whether the objective (fitness) function is stationary or non-stationary (change with time), linear or nonlinear, continuous or discontinuous, or with random noise. Because multiple offsprings in a population act like independent agents, the population (or any subgroup) can explore the search space in many directions simultaneously. This feature makes it ideal to parallelize the algorithms for implementation. Different parameters and even different groups of encoded strings can be manipulated at the same time.

However, genetic algorithms also have some disadvantages. The formulation of fitness function, the use of population size, the choice of the important parameters such as the rate of mutation and crossover, and the selection criteria of the new population should be carried out carefully. Any inappropriate choice will make it difficult for the algorithm to converge or it will simply produce meaningless results. Despite these drawbacks, genetic algorithms remain one of the most widely used optimization algorithms in modern nonlinear optimization.

2.Robustness of Traditional Optimization and Search Techniques

GA is a stochastic search algorithm based on principles of natural competition between individuals for appropriating limited natural sources. Success of the winner normally depends on their genes, and reproduction by such individuals causes the spread of their genes. By successive selection of superior individuals and reproducing them, the population will be led to obtain more

natural resources. The GA simulates this process and calculates the optimum of objective functions. In general, the standard GA is not convenient for finding the solutions to complex problems. The VSP method (Gholizadeh and Salajegheh, 2010; Gholizadeh and Samavati, 2011; Salajegheh and Gholizadeh, 2005) is an alternative to overcome this shortcoming of GA.

In this modified GA, an initial population with a small number of individuals is selected; this population is much smaller than that in standard GA. Then, all the necessary operations of the standard GA are carried out and the optimal solution is achieved. As the size of the population is small, the method converges to a premature solution. In each generation, the best individual is saved. Then, the best solution is repeatedly copied to create a new population and the remaining members of the population are randomly selected. Thereafter, the optimization process is repeated using standard GA with a reduced population to achieve a new solution.

3.Survival of the Fittest - Fitness Computations - Cross over - Mutation

Over the past few years, there has been a terrific buzz around Artificial Intelligence (AI). Major companies like Google, Apple, and Microsoft are actively working on the topic. In fact, AI is an umbrella that covers lots of goals, approaches, tools, and applications. Genetic Algorithms (GA) is just one of the tools for intelligent searching through many possible solutionsGA is a metaheuristic search and optimization technique based on principles present in natural evolution. It belongs to a larger class of evolutionary algorithms.GA maintains a **population of chromosomes**—a set of potential solutions for the problem. The idea is that "evolution" will find an optimal solution for the problem after a number of successive generations—similar to natural selection.GA mimics three evolutionary processes: selection, gene crossover, and mutation.Similar to natural selection, the central concept of GA selection is fitness. The chromosomes that are more fit have a better chance for survival. Fitness is a function that measures the quality of the solution represented by the chromosome. In essence, each chromosome within the population represents the input parameters. For example, if your problem contains two input parameters, such as price and volume in trading, each chromosome will logically consist of two elements. How the elements are encoded within the chromosome is a different topic.

During the selection, chromosomes form pairs of **parents** for **breeding**. Each **child** takes characteristics from its parents. Basically, the child represents a recombination of characteristics

from its parents: Some of the characteristics are taken from one parent and some from another. In addition to the recombination, some of the characteristics can **mutate**.Because fitter chromosomes produce more children, each subsequent generation will have better fitness. At some point, a generation will contain a chromosome that will represent a good enough solution for our problem.

GA is powerful and broadly applicable for complex problems. There is a large class of optimization problems that are quite hard to solve by conventional optimization techniques. Genetic algorithms are efficient algorithms whose solution is approximately optimal. The well-known applications include scheduling, transportation, routing, group technologies, layout design, neural network training, and many others.

4.Selection

selection is a process of finding successors to the current chromosomes—the chromosomes that are more fit for our problem. During the selection, we need to ensure that the chromosomes with better fitness have a better chance for survival.

```
private List<T> selection(List<T> population) {
   final double[] fitnesses = population.stream()
                                .mapToDouble(fitness)
                                .toArray();
   final double totalFitness = DoubleStream.of(fitnesses).sum();

   double sum = 0;
   final double[] probabilities = new double[fitnesses.length];
   for (int i = 0; i < fitnesses.length; i++) {
      sum += fitnesses[i] / totalFitness;
      probabilities[i] = sum;
   }
   probabilities[probabilities.length - 1] = 1;

   return range(0, probabilities.length).mapToObj(i -> {
      int index = binarySearch(probabilities, random());
```

```
if (index < 0 ) {
        index = -(index + 1 );
    }
    return population.get(index);
  }).collect(toList());
}
```

The idea behind this implementation is the following: The population is represented as consequent ranges on the numerical axis. The whole population is between 0 and 1.

5.Crossover

```
private void crossover(List<T> population) {
    final int [] indexes = range( 0 , population.size())
      .filter(i-> random() < crossoverProbability)
      .toArray();

  shuffle(Arrays.asList(indexes));

    for ( int i = 0 ; i < indexes.length / 2 ; i++) {
      final int index1 = indexes[ 2 * i];
      final int index2 = indexes[ 2 * i + 1 ];
      final T value1 = population.get(index1);
```

```
        final T value2 = population.get(index2);
        population.set(index1, crossover.apply(value1, value2));
        population.set(index2, crossover.apply(value2, value1));
    }
}
```

With the predefined probability `crossoverProbability`, we select parents for breeding. The selected parents are shuffled, allowing any combinations to happen. We take pairs of parents and apply the `crossover` operator. We apply the operator twice for each pair because we need to keep the size of the population the same. The children replace their parents in the population.

Mutation

Finally, we perform recombination of the characteristics.

```
private void mutation(List<T> population) {
    for (int i = 0; i < population.size(); i++) {
        if (random() < mutationProbability) {
            population.set(i, mutation.apply(population.get(i)));
        }
    }
}
```

With predefined probability `mutationProbability`, we perform “mutation” on the chromosomes. The mutation itself is defined by `mutation`.

6.Reproduction- Rank method

Selection is the stage of a genetic algorithm in which individual genomes are chosen from a population for later breeding (using the crossover operator).

A generic selection procedure may be implemented as follows:

1. The fitness function is evaluated for each individual, providing fitness values, which are then normalized. Normalization means dividing the fitness value of each individual by the sum of all fitness values, so that the sum of all resulting fitness values equals 1.

2. Accumulated normalized fitness values are computed: the accumulated fitness value of an individual is the sum of its own fitness value plus the fitness values of all the previous individuals; the accumulated fitness of the last individual should be 1, otherwise something went wrong in the normalization step.

3. A random number R between 0 and 1 is chosen.

4. The selected individual is the first one whose accumulated normalized value is greater than or equal to R.

For many problems the above algorithm might be computationally demanding. A simpler and faster alternative uses the so-called stochastic acceptance.

If this procedure is repeated until there are enough selected individuals, this selection method is called fitness proportionate selection or roulette-wheel selection. If instead of a single pointer spun multiple times, there are multiple, equally spaced pointers on a wheel that is spun once, it is called stochastic universal sampling. Repeatedly selecting the best individual of a randomly chosen subset is tournament selection. Taking the best half, third or another proportion of the individuals is truncation selection.

There are other selection algorithms that do not consider all individuals for selection, but only those with a fitness value that is higher than a given (arbitrary) constant. Other algorithms select from a restricted pool where only a certain percentage of the individuals are allowed, based on fitness value.

Retaining the best individuals in a generation unchanged in the next generation, is called elitism or elitist selection. It is a successful (slight) variant of the general process of constructing a new population.

7.Rank Selection

Rank Selection also works with negative fitness values and is mostly used when the individuals in the population have very close fitness values (this happens usually at the end of the run). This leads to each individual having an almost equal share of the pie (like in case of fitness proportionate selection) and hence each individual no matter how fit relative to each other has an approximately same probability of getting selected as a parent. This in turn leads to a loss in the selection pressure towards fitter individuals, making the GA to make poor parent selections in such situations.

TEXT / REFERENCE BOOKS

5. Laurene Fausett, " Fundamentals of Neural Networks: Architectures, Algorithms and Applications", 2008.
6. Timothy J. Ross , "Fuzzy Logic with Engineering Applications", McGraw - Hill International Editions, 2004.
7. Jang J.S.R., Sun C.T. and Mizutani E, "Neuro-Fuzzy and soft computing", Pearson Education, 2003.
8. Rajasekaran. S, Pai. G.A.V. "Neural Networks, Fuzzy Logic and Genetic Algorithms", Prentice Hall of India, 2003.

Part- A

1. What is meant by genetic Algorithm
2. What is rank space method
3. What is population in GA
4. What is meant by mutation
5. What is meant by rank method
6. What is crossover
7. Give the different types of crossover technique
8. What is meant by Roulette wheel selection2

Part-B

1) Explain with an example about the survival of the fittest
2) With the neat flowchart explain the operation of Genetic Algorithm
3) Explain Crossover Mutation
4) Describe in detail about Rank Method
5) Explain in detail about the selection operation in reproduction