CCS364 - notes

UNIT I

1 Introduction to Soft Computing and Fuzzy Logic

Syllabus

Introduction - Fuzzy Logic - Fuzzy Sets, Fuzzy Membership Functions, Operations on Fuzzy sets, Fuzzy Relations, Operations on Fuzzy Relations, Fuzzy Rules and Fuzzy Reasoning, Fuzzy Inference Systems.

Contents

1.1 Introduction

- Soft computing is likely to play an especially important role in science and engineering, but eventually its influence may extend much farther. Building human-centered systems is an imperative task for scientists and engineers in the new millennium.
- Soft Computing (SC) is a concept that was introduced by Zadeh (1992), the discoverer of fuzzy logic.
- **Definition :** "Soft computing is an emerging approach to computing which parallel the remarkable ability of human mind to reason and learn in a environment of uncertainty and imprecision".
- The idea behind soft computing is to model cognitive behavior of human mind. Soft computing is foundation of conceptual intelligence in machines. Unlike hard computing, soft computing is tolerant of imprecision, uncertainty, partial truth, and approximation.
- Hard computing requires a precisely stated analytical model and often a lot of computation time. Many analytical models are valid for ideal cases.
- Soft computing is a collection of methodologies that aim to exploit the tolerance for imprecision and uncertainty to achieve tractability, robustness, and low solution cost.
- Soft computing is not precisely defined. It consists of distinct concepts and techniques which aim to overcome the difficulties encountered in real world problems. These problems result from the fact that our world seems to be imprecise, uncertain and difficult to categorize.
- The basic methods included in cognitive computing are fuzzy logic, neural networks and genetic algorithms, the methods which do not derive from classical theories.
- Soft Computing combines knowledge, techniques, and methodologies from the sources above to create intelligent systems.
- The guiding principle of soft computing is : Exploit the tolerance for imprecision, uncertainty, partial truth, and approximation to achieve tractability, robustness and low solution cost.
- Soft computing does not perform much symbolic manipulation, so user can view it as a new discipline that complements conventional artificial intelligence approaches and vice versa.
- Hard computing is also called as conventional computing. It requires a precisely stated analytical model and often a lot of computation time. Many analytical models are valid for ideal cases. Real world problems exist in a non-ideal environment.

Unique Property of soft computing

- Learning from experimental data.
- Soft computing techniques derive their power of generalization from approximating or interpolating to produce outputs from previously unseen inputs by using outputs from previous learned inputs.
- Generalization is usually done in a high-dimensional space.

Goal of soft computing

- It is a new multidisciplinary field, to construct a new generation of Artificial Intelligence, known as **Computational Intelligence**.
- The main **goal** 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 develop the tolerance for Approximation, Uncertainty, Imprecision, and Partial Truth in order to achieve close resemblance with human like decision making.

1.1.1 Characterize the Constituents of Soft Computing

1. One of the characteristics of soft computing methods is that they are typically used in problems where mathematical models are not available or are intractable or too cumbersome to be viable.
2. Another characteristic is that uncertainty inherent in many situations under study is taken into account rather than ignored.
3. Soft computing often provides a good solution as opposed to an optimal solution.
4. Soft computing uses human concept like if-then rules, cases and conventional knowledge representations.
5. Biologically inspired computing models (NN).
6. New optimization techniques are applied by soft computing.
7. Model free learning : Models are constructed based on the target system only.
8. New application domains : Mostly computation intensive like adaptive signal processing, adaptive control, nonlinear system identification etc.
9. Fault tolerance: Deletion of a neuron or a rule does not destroy the system. The system performs with lesser quality.
10. Goal driven characteristics : Only the goal is important and not the path.

1.1.2 Strength and Weakness of Soft Computing

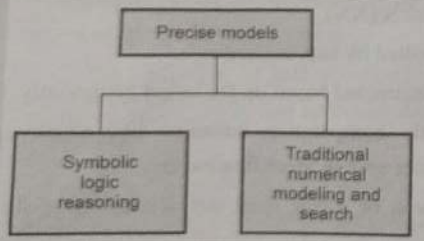
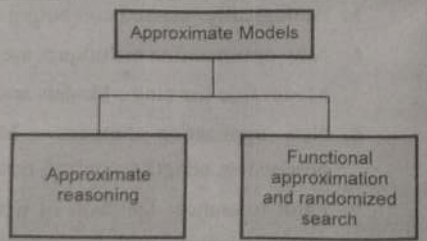
Strength :

1. Low cost and less time-consuming solutions are provided by soft computing.
2. Soft computing techniques can tolerate imprecision, uncertainty and partial truth to produce high machine intelligent quotient.
3. It provides good solutions.
4. It uses human concept.

Weakness :

1. It requires extensive computation.
2. Soft computing does not perform much symbolic manipulation.
3. Soft computing is not precisely defined.

1.1.3 Difference between Hard Computing and Soft Computing

Sr. No.	Hard computing	Soft computing
1.	It uses binary logic.	It uses fuzzy logic.
2.	It is based on numerical analysis.	It is based on genetic algorithms.
3.	Crisp system is used in hard computing.	Neurocomputing is used in soft computing.
4.	It takes help of differential equations.	It takes help of probabilistic reasoning.
5.	Approximation theory.	Evidential reasoning.
6.	Functional analysis.	Machine learning.
7.		

1.2 Constituents of Soft Computing

- The principal constituents, i.e., tools, techniques, of Soft Computing (SC) are Fuzzy Logic (FL), Neural Networks (NN), Support Vector Machines (SVM), Evolutionary Computation (EC), and Machine Learning (ML) and Probabilistic Reasoning (PR).

- Fuzzy theory plays a leading role in soft computing and this stems from the fact that human reasoning is not crisp and admits degrees.

1.2.1 Fuzzy Logic

- Fuzzy Logic (FL) is a multivalued logic, that allows intermediate values to be defined between conventional evaluations like true/false, yes/no, high/low, etc.
- Fuzzy Logic is based on fuzzy set theory and provides methods for modeling and reasoning under uncertainty, a characteristic present in many problems, which makes FL a valuable approach.
- It allows data to be represented in intuitive linguistic categories instead of using precise (crisp) numbers which might not be known, necessary or in general may be too restrictive.
- Fuzzy logic offers a practical way for designing nonlinear control systems. It achieves nonlinearly through piece-wise linear approximation. The basic building blocks of a fuzzy logical control system are set of fuzzy if-then (i.e., fuzzy rule based models) that approximate a functional mapping.
- Fuzzy logic provides an inference morphology that enables approximate human reasoning capabilities to be applied to knowledge-based systems. The theory of fuzzy logic provides a mathematical strength to capture the uncertainties associated with human cognitive processes, such as thinking and reasoning.
- 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.
- Fuzzy logic is viewed as a formal mathematical theory for the representation of uncertainty. Uncertainty is crucial for the management of real systems : if you had to park your car precisely in one place, it would not be possible. Instead, you work within, say, 10 cm tolerances.
- The presence of uncertainty is the price you pay for handling a complex system. Nevertheless, fuzzy logic is a mathematical formalism, and a membership grade is a precise number. What's crucial to realize is that fuzzy logic is a logic of fuzziness, not a logic which is itself fuzzy.

1.2.2 Genetic Algorithms

- Genetic algorithms are inspired by Darwin's theory of natural evolution. In the natural world, organisms that are poorly suited for an environment die off, while those well-suited, prosper.
- Genetic algorithms search the space of individuals for good candidates. The chance of an individual's being selected is proportional to the amount by which its fitness is greater or less than its competitors' fitness.
- It is basically a method for moving from one population of chromosomes to a new generated population using selection together with the genetic inspired operators of crossover, mutation and inversion.
- It is basically a search technique that will map data for the problems where no particular formula can be implemented.
- Genetic algorithms are broadly applicable and have the advantage that they require little knowledge encoded in the system.

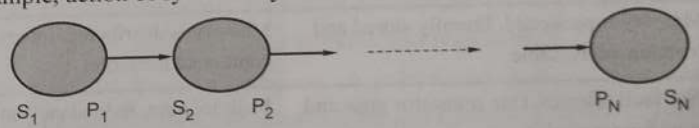
1.2.3 Neural Networks

- Neural networks consists of many number of simple elements (neurons) connected between them in system. Whole system is able to solve of complex tasks and to learn for it like a natural brain.
- For user NN is black box with Input vector (source data) and Output vector (result).
- A Neural network is usually structured into an input layer of neurons, one or more hidden layers and one output layer.
- Neurons belonging to adjacent layers are usually fully connected and the various types and architectures are identified both by the different topologies adopted for the connections as well by the choice of the activation function.
- The values of the functions associated with the connections are called "weights".
- The whole game of using NNs is in the fact that, in order for the network to yield appropriate outputs for given input, the weight must be set to suitable values. The way this is obtained allows a further distinction among modes of operations.
- A neural network is a processing device, either an algorithm or actual hardware, whose design was motivated by the design and functioning of human brains and components thereof.
- Most neural networks have some sort of "training" rule whereby the weights of connections are adjusted on the basis of presented patterns.

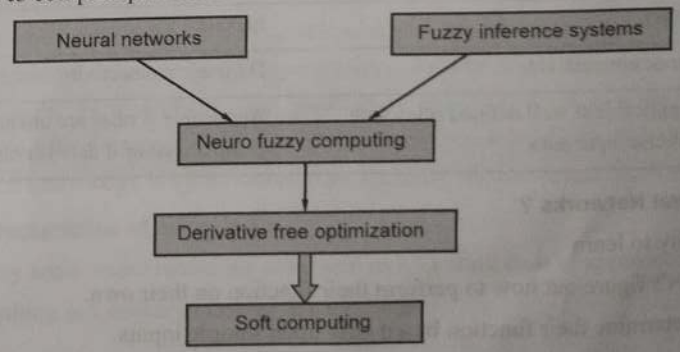
- In other words, neural networks "learn" from examples, just like children learn to recognize dogs from examples of dogs, and exhibit some structural capability for generalization.
- Neural networks normally have great potential for parallelism, since the computations of the components are independent of each other.
- Neural networks are a different paradigm for computing :
  1. Vone neumann machines are based on the processing/memory abstraction of human information processing.
  2. Neural networks are based on the parallel architecture of animal brains.
- Neural networks are a form of multiprocessor computer system, with
  a. Simple processing elements b. A high degree of interconnection
  c. Simple scalar messages d. Adaptive interaction between elements

1.2.4 Probabilistic Reasoning

- Uncertainty is described by probabilities. Probability may be use for simulation of fuzziness
- Relations between events are described as conditional probabilities (Bayesian nets) or probabilities of transition probabilities (Markovian process).
- For example, action of system may be described as graph of states -



- Hard computing deals with precise models where accurate solutions are achieved quickly. On the other hand, soft computing deals with approximate models and gives solution to complex problems.



- Soft computing methods have been applied to many real-world problems. Applications can be found in signal processing, pattern recognition, quality assurance and industrial inspection, business forecasting, speech processing, credit rating, adaptive process control, robotics control, natural language understanding, etc.
- Possible new application areas are programming languages, user friendly application interfaces, computer networks, database management, fault diagnostics and information security etc.
- For learning and adaptations, soft computing requires extensive computation. In this sense, soft computing shares the same characteristics as computational intelligence.
- Soft computing does not perform much symbolic manipulation, so user can view it as a new discipline that complements conventional artificial intelligence approaches and vice versa.

1.2.5 Difference between Digital Computer and Neural Network

Sr. No.	Digital Computers	Neural Networks
1.	Deductive reasoning : We apply known rules to input data to produce output.	Inductive reasoning : Given input and output data (training examples), we construct the rules.
2.	Computation is centralized, synchronous and serial.	Computation is collective, asynchronous and parallel.
3.	Memory is packetted, literally stored and location addressable.	Memory is distributed, internalized and content addressable.
4.	Not fault tolerant. One transistor goes and it no longer works.	Fault tolerant, redundancy and sharing of responsibilities.
5.	Fast. Measured in millionths of a second.	Slow. Measured in thousandths of a second.
6.	Exact.	Inexact.
7.	Static connectivity.	Dynamic connectivity.
8.	Applicable if well defined rules with precise input data.	Applicable if rules are unknown or complicated or if data is noisy or partial.

Why Use Neural Networks ?

1. Ability to learn
   a. NN's figure out how to perform their function on their own.
   b. Determine their function based only upon sample inputs.
2. Ability to generalize i.e. produce reasonable outputs for inputs, it has not been taught how to deal with.

1.3 Introduction and Characteristics of Fuzzy Logic

- Fuzzy set theory was developed by Lotfi A. Zadeh, professor for computer science at the University of California in Berkeley, to provide a mathematical tool for dealing with the concepts used in natural language.
- Fuzzy logic is not logic that is fuzzy, but logic that is used to describe fuzziness. Fuzzy logic is the theory of fuzzy sets, sets that calibrate vagueness.
- Fuzzy Logic is basically a multi-valued logic that allows intermediate values to be defined between conventional evaluations. Fuzzy logic is a form of many-valued logic in which the truth values of variables may be any real number between 0 and 1.
- Fuzzy logic is an extension of Boolean logic used to describe environments where there is no absolute truth and there is uncertainty.
- Fuzzy logic was specifically designed to mathematically represent uncertainty and vagueness and to provide formalized tools for dealing with the imprecision intrinsic to many problems.
- Fuzzy logic provides an inference morphology that enables approximate human reasoning capabilities to be applied to knowledge-based systems. The theory of fuzzy logic provides a mathematical strength to capture the uncertainties associated with human cognitive processes, such as thinking and reasoning.
- Fuzzy systems are **knowledge-based or rule-based systems**. The heart of a fuzzy system is a knowledge base consisting of the so-called fuzzy IF-THEN rules.
- FL is a problem-solving control system methodology that lends itself to implementation in systems ranging from simple, small, embedded micro-controllers to large, networked, multi-channel PC or workstation-based data acquisition and control systems.
- The conventional approaches to knowledge representation lack the means for representation the meaning of fuzzy concepts. As a consequence, the approaches based on first order logic and classical probability theory do not provide an appropriate conceptual framework for dealing with the representation of commonsense knowledge, since such knowledge is by its nature both lexically imprecise and non-categorical.

Essential characteristics of fuzzy logic :

1. In fuzzy logic, exact reasoning is viewed as a limiting case of approximate reasoning.
2. Everything is a matter of degree in fuzzy logic.

3. Knowledge is interpreted a collection of elastic or, equivalently, fuzzy constraint on a collection of variables.
4. Inference is viewed as a process of propagation of elastic constraints.
5. Any logical system can be fuzzified.

- 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.
- Fuzzy logic is based on the idea that all things admit of degrees. For example, temperature, height, speed, distance and beauty all come on a sliding scale. The motor is running *really* hot. Ram is a very tall guy.

What is fuzziness ?

- For example, if pressure takes values between 0 and 50, one might label the range 20 to 30 as medium pressure.
- Medium is known as a linguistic variable. Therefore, with Boolean logic 15.0 (or even 19.99) is not a member of the medium pressure range. As soon as the pressure equals 20, then it becomes a member.
- Fuzziness is not vague and Multi-valued logic.
- Fuzzy logic may be considered as an extension of multi-valued logic but they are somewhat different. Multi-valued logic is still based on exact reasoning whereas fuzzy logic is approximate reasoning.

1.3.1 Difference between Boolean Algebra and Fuzzy Logic

Boolean algebra	Fuzzy logic
A system of logic that is based on Boolean algebra, named after George Boole. It deals with two truth values : 'true' and 'false'.	It is very easy to extend Boolean logic into multi-valued logic. Unlike Boolean logic, fuzzy logic is multi-valued and handles the concept of partial truth.
Something is part of A or Not A. It can not be A and Not-A at the same time.	Something can be part of A and part of Not-A at the same time.
It does not mimic human thinking very good.	Mimics human thinking and decisions very well.
YES or NO Logic (0 or 1)	Fuzzy sets

1.4 Fuzzy Sets

- A fuzzy set is a collection of objects with graded membership.
- Two examples of "Sets"
  1. All employees of XYZ who are over 1.8 meter in height.
  2. All employees of XYZ who are tall.
- The first example is a **classical set**, we have a universe (all XYZ employees) and a membership rule that divides the universe into members (those over 1.8 m) and nonmembers.
- The second example is a **fuzzy set**, some employees are definitely in the set and some are definitely not in the set, but some are "borderline". This distinctions between the "ins", the "outs", and the "borderlines" is made more exact by the membership function, $\mu A(x)$.
- $\mu A(x)$: If we return to our second example and let A represent the fuzzy set of all tall employees and x represent a member of the universe X (i.e. all employees), what would the function $\mu A(x)$ look like ?

$$\mu A(x) = 1 \text{ if x is definitely tall}$$

$$\mu A(x) = 0 \text{ if x is definitely not tall}$$

$$0 < \mu A(x) < 1 \text{ for borderline cases}$$

i.e. anyone over 2.0 m is definitely tall, anyone under 1.75 m is definitely not tall and anyone between 1.75 m and 2.0 m is partly tall and partly not tall.

1.5 Fuzzy Membership Functions

- Membership Function (MF) is a curve that defines how each point in the input space is mapped to a membership value (or degree of membership) between 0 and 1. The input space is sometimes referred to as the universe of discourse.

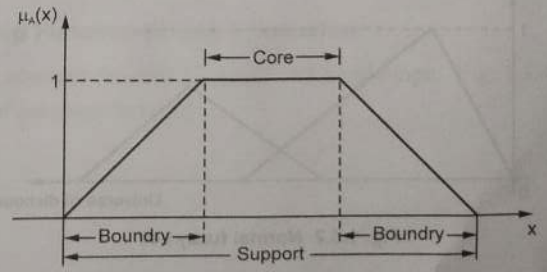


Fig. 1.5.1 Membership function feature

- Feature of membership function is defined by three properties :
  1. Core 2. Support 3. Boundary

1. **Core :** If the region of universe is characterized by full membership in the set A then this gives the core of the membership function of fuzzy at A. The elements which have the membership function as 1 are the elements of the core. Here

$$\mu_A(x) = 1$$

2. **Support :** If the region of universe is characterized by nonzero membership in the set A. This defines the support of a membership function for fuzzy set A. The support has the elements whose membership is greater than 0.

$$\mu_A(x) > 1$$

3. **Boundary :** If the region of universe has a nonzero membership but not full membership, this defines the boundary of a membership; this define the boundary of a membership function for fuzzy set A. The boundary has the elements whose membership is between 0 and 1.

$$0 < \mu_A(x) < 1$$

- **Normal fuzzy set :** A fuzzy set A is normal if its maximal degree of membership is unity, i.e., there must exist at least one x for which $\mu_A(x) = 1$. On the other hand, non-normal fuzzy sets have maximum degree of membership less than one. Fig. 3.6.2 shows normal fuzzy set.
- The height of a fuzzy set is the largest membership degree among all elements of the universe. Fuzzy sets whose height equals one for at least one element x in the domain X are called **normal fuzzy sets**. The height of sub-normal fuzzy sets is thus smaller than one for all elements in the domain.

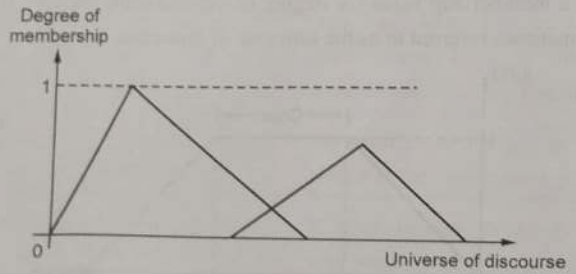


Fig. 1.5.2 Normal fuzzy set

Various types of membership functions

1. S-shaped function
2. Trapezoidal membership function
3. Gaussian membership function
4. Exponential - like function
5. Triangular membership function

- Fig. 1.5.3 shows various shapes of membership function.

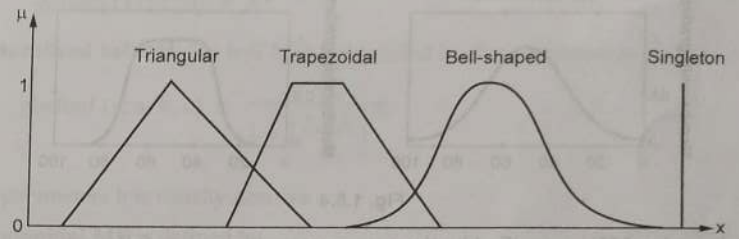


Fig. 1.5.3 Various shapes of membership function

- Following criteria are valid for all membership functions :
  1. The membership function must be a real valued function whose values are between 0 and 1.
  2. The membership values should be 1 at the center of the set, i.e., for those members that definitely belong to the set.
  3. The membership function should fall off in an appropriate way from the center through the boundary.
  4. The points with membership value 0.5 (crossover point) should be at the boundary of the crisp set, i.e., if we would apply a crisp classification, the class boundary should be represented by the crossover points.

1.5.1 Membership Function of One Dimension

- Here we will consider member function with single input. Fig. 1.5.4 shows examples of four classes of parameterized MF.

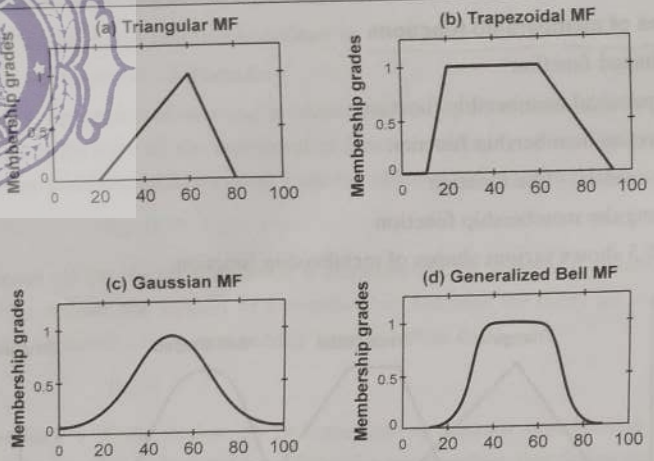


Fig. 1.5.4

- A **triangular MF** is specified by three parameters {a, b, c} as follows :

$$\text{triangle}(x; a, b, c) = \begin{cases} 0 & x \le a \\ \dfrac{x-a}{b-a} & a \le x \le b \\ \dfrac{c-x}{c-b} & b \le x < c \\ 0 & c \le x \end{cases}$$

- By using min and max, we have an alternative expression for the preceding equation :

$$\text{Trimf}(x; a, b, c) = \max\left(\min\left(\frac{x-a}{b-a}, \frac{c-x}{c-b}\right), 0\right)$$

- The parameters {a, b, c} determine the x coordinates of the three corners of the underlying **triangular MF**.
- A **trapezoidal MF** is specified by four parameters {a, b, c, d} as follows :

$$\text{Trapezoid}(x; a, b, c, d) = \begin{cases} 0 & x \le a \\ \dfrac{x-a}{b-a} & a \le x \le b \\ 1 & b \le x < c \\ \dfrac{d-x}{d-c} & c \le x \le d \\ 0 & d \le x \end{cases}$$

- By using min and max, rewrite the equation :

$$\text{trapmf}(x; a, b, c, d) = \max\left(\min\left(\frac{x-a}{b-a}, 1, \frac{d-x}{d-c}\right), 0\right)$$

- The parameters {a, b, c, d} determine the x coordinates of the four corners of the underlying **trapezoidal MF**.
- A **Gaussian MF** is specified by two parameters {c, σ};

$$\text{gaussmf}(x; c, \sigma) = e^{-\frac{1}{2}\left(\frac{x-c}{\sigma}\right)^2}$$

- **Generalized bell MF (or bell MF)** is specified by three parameters {a, b, c} :

$$\text{gbellmf}(x; a, b, c) = \frac{1}{1 + \left|\dfrac{x-c}{b}\right|^{2b}}$$

Where parameters b is usually positive.

- A **sigmoidal MF** is defined by

$$\text{sigmf}(x; a, b, c) = \frac{1}{1 + e^{-a(x-c)}}$$

Where a controls the slope at the crossover point x = c.

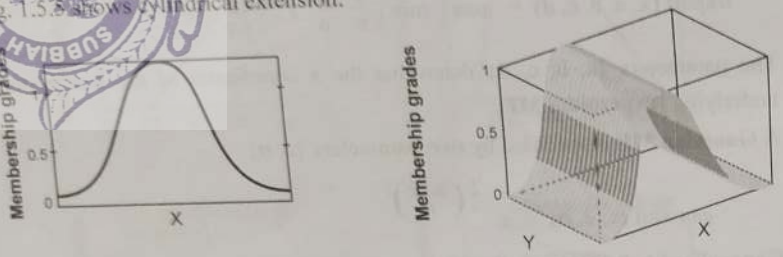
- Gaussian MFs and bell MFs achieve smoothness, they are unable to specify asymmetric Mfs which are important in many applications.
- A sigmoidal MF is inherently open right or left and thus, it is appropriate for representing concepts such as "very large" or "very negative".
- Sigmoidal MF mostly used as activation function of artificial Neural Networks (NN). A NN should synthesize a close MF in order to simulate the behavior of a fuzzy inference system.

1.5.2 Membership Function of Two Dimension

- In this case, there are two inputs assigned to an MF : This MF is a two dimensional MF. A one input MF is called ordinary MF.
- Extension of a one-dimensional MF to a two-dimensional MF via cylindrical extensions.
- If A is a fuzzy set in X, then its cylindrical extension in X*Y is a fuzzy set C(A) defined by :

$$C(A) = \int_{x \cdot y} \mu_A(x) \mid (x, y)$$

- C(A) can be viewed as a two-dimensional fuzzy set.
- Fig. 1.5.5 shows cylindrical extension.



(a) Base set A (b) Its cylindrical extension

Fig. 1.5.5

Projection of fuzzy sets (Decrease dimension) :

- Let R be a two-dimensional fuzzy set on X*Y. Then the projections of R onto X and Y are defined as :

$$R_x = \int_x [\max_y \mu_R(x, y)] \mid x$$

And

$$R_Y = \int_Y [\max_x \mu_R(x, y)] \mid y$$

respectively.

- Two dimensional MF falls into two categories : **Composite and non-composite MFs.**
- If an MF of two dimensions can be expressed as an analytic expression of two MFs of one dimension, then it is composite; otherwise it is noncomposite.
- Suppose that the fuzzy A = "(x, y) is near (3, 4)" is defined by :

$$\mu_A(x, y) = \exp\left[-\left(\frac{x-3}{2}\right)^2 - (y-4)^2\right]$$

$$= \exp\left[-\left(\frac{x-3}{2}\right)^2\right] \exp\left[-\left(\frac{y-4}{1}\right)^2\right]$$

$$= G(x; 3, 2) * G(y; 4, 1)$$

- This two-dimensional MF is composite. The fuzzy set A is composed of two statements :

"x is near 3" and "y is near 4"

- These two statements are respectively defined as :

$$\mu_{near3}(x) = G(x; 3, 2)$$

And

$$\mu_{near4}(x) = G(x; 4, 1)$$

- If a fuzzy set is defined by :

$$\mu_4(x, y) = \frac{1}{1 + |x-3|\,|y-4|^{2.5}}$$

it is non-composite.

- A composite two-dimensional MF is usually the result of two statements joined by the AND or OR connectives.
- Composite two-dimensional based on min and max operations.
- Let trap(x) = trapezoid (x; –6, –2, 2, 6)

  trap(y) = trapezoid (y; –6, –2, 2, 6)

be two trapezoidal on X and Y respectively

- By applying the min and max operators, we obtain two-dimensional on X*Y.
- When the min operator is used to aggregate one dimensional MF's , the resulting two-dimensional MF can be viewed either as the result of applying classical fuzzy MF or as a Cartesian product of two one-dimensional fuzzy sets.

1.6 Operations on Fuzzy Sets

- Union, intersection, and complement are the most basic operations on classical sets. On the basis of these three operations, a number of identities can be established. Corresponding to the ordinary set operations of union, intersection, and complement, fuzzy sets have similar operations.

1. **Containment or subset :** Fuzzy set A is **contained** in fuzzy set B (or, equivalently, A is a **subset** of B, or A is smaller than or equal to B, $A \subseteq B$) if and only if $\mu_A(x) \leq \mu_B(x)$ for all x.

$$A \subseteq B \Leftrightarrow \mu_A \leq \mu_B$$

A is contained in B

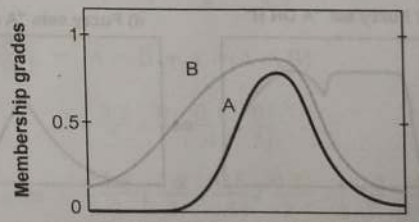


Fig. 1.6.1 Concept of $A \subseteq B$

2. **Union (disjunction) :** The **union** of two fuzzy sets A and B is a fuzzy set C, written as

C = A ∪ B or C = A OR B, whose MF is related to those of A and B by $\mu_C(x) = \max(\mu_A(x), \mu_B(x))$.

$$C = A \cup B \Leftrightarrow \mu_C(x) = \max(\mu_A(x), \mu_B(x)) = \mu_A(x) \vee \mu_B(x)$$

3. **Intersection (conjunction) :** The **intersection** of two fuzzy sets A and B is a fuzzy set C, written as C = A ∩ B or C = A AND B, whose MF is related to those of A and B by $\mu_C(x) = \min(\mu_A(x), \mu_B(x))$.

$$C = A \cap B \Leftrightarrow \mu_C(x) = \min(\mu_A(x), \mu_B(x)) = \mu_A(x) \wedge \mu_B(x)$$

4. **Complement (negation) :** The complement of fuzzy set A, denoted by $\overline{A}$ or NOT A, is defined as $\mu_A(x) = 1 - \mu_A(x)$.

5. **Cartesian product and co-product :** Let A and B be the fuzzy sets in X and Y respectively. The cartesion product of A and B denoted by A × B, is a fuzzy set in the product space X × Y with the membership function

$$\mu_{A \times B}(x, y) = \min(\mu_A(x), \mu_B(x)).$$

Cartesian co-product A + B is a fuzzy set with the membership function

$$\mu_{A+B}(x, y) = \max(\mu_A(x), \mu_B(x)).$$

Fig. 1.6.2 shows the basic three operation on two fuzzy set A and B.

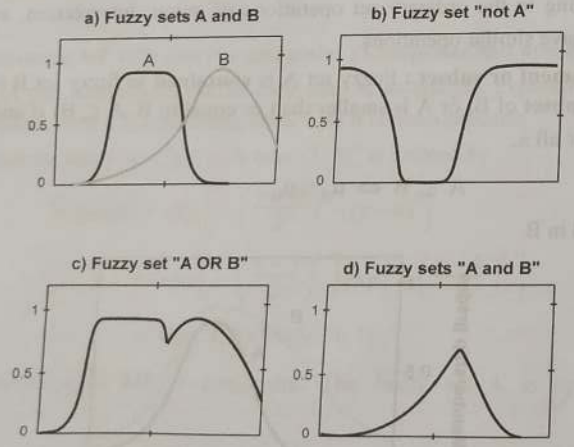


Fig. 1.6.2 Operation on fuzzy sets

Solved Examples

Example 1.6.1 *Consider two fuzzy sets A and B*

$$A = \left\{\frac{1}{2} + \frac{0.5}{3} + \frac{0.3}{4} + \frac{0.2}{5}\right\} \quad B = \left\{\frac{0.5}{2} + \frac{0.7}{3} + \frac{0.2}{4} + \frac{0.4}{5}\right\}$$

Perform the following operating on fuzzy sets

i) $A \cup B$ *ii)* $A \cap B$

iii) Component of fuzzy set A

iv) Difference $\left(\frac{A}{B}\right)$

v) Algebraic sum of given fuzzy sets.

vi) Bounded sum of the given fuzzy set.

vii) Algebraic product of the given fizzy sets.

viii) $\overline{A \cup B}$

ix) $A \cup \overline{B}$

Solution :

i) $$A \cup B = \frac{1}{2} + \frac{0.7}{3} + \frac{0.3}{4} + \frac{0.4}{5}$$

ii) $$A \cap B = \frac{0.5}{2} + \frac{0.5}{3} + \frac{0.2}{4} + \frac{0.2}{5}$$

iii) Complement of fuzzy set A : $1 - \mu_A(x)$

$$= \left\{\frac{0}{2} + \frac{0.5}{3} + \frac{0.7}{4} + \frac{0.8}{5}\right\}$$

iv) $$\text{Difference}\left(\frac{A}{B}\right) = A \cap \overline{B} = A - (A \cap B)$$

$$= \left\{\frac{0.5}{2} + \frac{0}{3} + \frac{0.1}{4} + \frac{0}{5}\right\}$$

v) $$\text{Algebraic sum} = \left\{\frac{1.5}{2} + \frac{1.2}{3} + \frac{0.5}{4} + \frac{0.6}{5}\right\} - \left\{\frac{0.5}{2} + \frac{0.35}{3} + \frac{0.06}{4} + \frac{0.08}{5}\right\}$$

$$= \left\{\frac{1}{2} + \frac{0.85}{3} + \frac{0.44}{4} + \frac{0.52}{5}\right\}$$

vi) Bounded sum $= \min\left\{1, \left\{\frac{1.5}{2}+\frac{1.2}{3}+\frac{0.5}{4}+\frac{0.6}{5}\right\}\right\}$

$= \left\{\frac{1.5}{2}+\frac{1.2}{3}+\frac{0.5}{4}+\frac{0.6}{5}\right\}$

vii) Algebraic product $= \left\{\frac{0.5}{2}+\frac{0.35}{3}+\frac{0.06}{4}+\frac{0.08}{5}\right\}$

viii) $\overline{A \cup B} = 1 - \max\{\mu_A(x), \mu_B(x)\}$

$= 1 - \left\{\frac{1}{2}+\frac{0.7}{3}+\frac{0.3}{4}+\frac{0.4}{5}\right\} = \left\{\frac{0}{2}+\frac{0.3}{3}+\frac{0.7}{4}+\frac{0.6}{5}\right\}$

ix) $A \cup \bar{B} = \max[\mu_A(x), \mu_B(x)]$

Calculate, $\bar{B} = 1 - \mu_B(x)$

$= 1 - \left\{\frac{0.5}{2}+\frac{0.7}{3}+\frac{0.2}{4}+\frac{0.4}{5}\right\} = \left\{\frac{0.5}{2}+\frac{0.3}{3}+\frac{0.8}{4}+\frac{0.6}{5}\right\}$

$A \cup \bar{B} = \left\{\frac{1}{2}+\frac{0.5}{3}+\frac{0.8}{4}+\frac{0.6}{5}\right\}$

Example 1.6.2 *Consider two fuzzy sets X and Y, find Complement, Union, Intersection, Difference :*

$X = \left\{\frac{0.5}{2}, \frac{0.4}{3}, \frac{0.1}{4}, \frac{0.1}{5}, \frac{0.3}{6}, \frac{0.7}{7}, \frac{0.8}{8}\right\}$

$Y = \left\{\frac{0.3}{2}, \frac{0.8}{3}, \frac{0.6}{4}, \frac{0.8}{5}, \frac{0.2}{6}, \frac{0.2}{7}, \frac{0.3}{8}\right\}$

Solution :

1) Complement :

$\text{Set } X = \left\{\frac{0.5}{2}, \frac{0.6}{3}, \frac{0.9}{4}, \frac{0.9}{5}, \frac{0.7}{7}, \frac{0.3}{7}, \frac{0.2}{8}\right\}$

$\text{Set } Y = \left\{\frac{0.7}{2}, \frac{0.2}{3}, \frac{0.4}{4}, \frac{0.2}{5}, \frac{0.8}{6}, \frac{0.8}{7}, \frac{0.7}{8}\right\}$

2) Untion :

$X \cup Y = \left\{\frac{0.5}{2}, \frac{0.8}{3}, \frac{0.6}{4}, \frac{0.8}{5}, \frac{0.3}{6}, \frac{0.7}{7}, \frac{0.8}{8}\right\}$

3) Intersection :

$X \cap Y = \left\{\frac{0.3}{2}, \frac{0.4}{3}, \frac{0.1}{4}, \frac{0.1}{5}, \frac{0.2}{6}, \frac{0.2}{7}, \frac{0.3}{8}\right\}$

4) Difference

$(X/Y) = X \cap \bar{Y} = X - (X \cap Y)$

$= \left\{\frac{0.2}{2}, \frac{0}{3}, \frac{0}{4}, \frac{0}{5}, \frac{0.1}{6}, \frac{0.5}{7}, \frac{0.5}{8}\right\}$

1.7 Fuzzy Relations

- Relations are the way in which two or more things are connected. In natural language, relations are kinds of links existing between objects such as "mother of", "neighbor of", "part of", etc.
- Mathematically, a relation implies the presence of an association between elements of different sets (at least two fuzzy sets). Binary relation is a relation between two sets. Tertiary relation is a relation between three sets and n-ary relation is a relation between n elements.
- Classical relations are structure that represents the presence or absence of correlation or interaction among various sets. Two degrees of relationship in a crisp relations : *Completely related and Not related.*
- A binary relation is a relation between two sets X and Y denoted by R(X,Y). A binary fuzzy relation can be expressed as
  1. Linguistically, i.e. through statement
  2. By listing the set of all ordered pairs
  3. Directed graph
  4. Matrix relation R
  5. Table
- **A fuzzy relation is** based on the philosophy that everything is related to some extend or unrelated. Unlike crisp relations, the strength of the relation between ordered pair of two universes is not measured with the characteristic function, but rather with a membership function expression expressing various degree of strength of the relation on the unit interval [0,1].
- Crisp relations and fuzzy relations can be defined in terms of subsets.
- A binary fuzzy relation is a fuzzy relation between two sets X and Y denoted by R(X,Y). A binary fuzzy relation can be expressed as

a. By listing the set of all ordered pairs

b. Directed graph

c. Matrix (Fuzzy R

d. Table (Tabular form)

- Let X and Y be two universes of discourse, then

$$R = \{ ((x, y), \mu_R (x, y)) \mid (x, y) \in X \times Y \}$$

is a **binary fuzzy relation in X × Y**.

- Fuzzy relations represents the strength of association between elements of the two sets

Example : R = "x is considerably larger than y"

$R(X, Y)$ = Relation between sets X and Y

$R(x, y)$ = Membership function for the relation R (X, Y)

$$R(X, Y) = \{R(x, y) / (x, y) \setminus (x, y) \in (X \times Y)\}$$

$$R(x, y) = \begin{cases} 0 & \text{for } x \le y \\ (x-y)/(10-y) & \text{for } y < x \le 11y \\ 1 & \text{for } x > 11y \end{cases}$$

- Let X = Y = R+ and R = "y is much greater than x" . The member function of the fuzzy relation R can be subjectively defined as

$$\mu_R(x, y) = \begin{cases} \dfrac{y-x}{x+y+2} & \text{if } y \cdot x \\ 0 & \text{if } y \le x \end{cases}$$

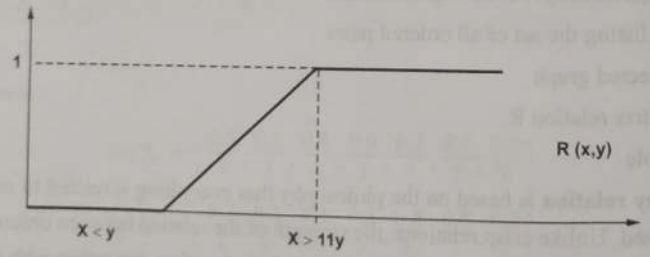


Fig. 1.7.1

- Assume two universes : A = {3, 4, 5} and B = {3, 4, 5, 6, 7}

$$\mu_R(x, y) = \begin{cases} (y-x)/(y+x+2) & \text{if } y > x \\ 0 & \text{if } y \le x \end{cases}$$

- This can be expressed as follow :

$$R = \begin{array}{c} \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 3 & 4 & 5 & 6 & 7 \end{array} \\ \begin{bmatrix} 0 & 0.11 & 0.2 & 0.27 & 0.33 \\ 0 & 0 & 0.09 & 0.17 & 0.23 \\ 0 & 0 & 0 & 0.08 & 0.14 \end{bmatrix} \end{array}$$

- Fuzzy relation example :

$$R = \begin{array}{c} \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 3 & 4 & 5 & 6 & 7 \end{array} \\ \begin{bmatrix} 0 & 0.11 & 0.2 & 0.27 & 0.33 \\ 0 & 0 & 0.09 & 0.17 & 0.23 \\ 0 & 0 & 0 & 0.08 & 0.14 \end{bmatrix} \end{array}$$

- This matrix represents the membership grades between elements in X and Y

$$\mu_R(x, y) = \{[0/(3, 3)], [0.11/(3, 4)], [0.2/(3, 5)], \ldots\ldots\ldots\ldots, [0.14/(5, 7)]\}$$

Example 1.7.1 *Assume two fuzzy sets : $A = \{0.2/x_1 + 0.5/x_2 + 1/x_3\}$*

$$B = \{0.3/y_1 + 0.9/y_2\}$$

Find the fuzzy relation (the Cartesian product).

Solution : The fuzzy relation :

$$A \times B = R = \begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \end{array} \begin{array}{c} \begin{array}{cc} y_1 & y_2 \end{array} \\ \begin{bmatrix} 0.2 & 0.2 \\ 0.3 & 0.5 \\ 0.3 & 0.9 \end{bmatrix} \end{array}$$

Example 1.7.2 *Consider the fuzzy relation. Express R using the resolution principle.*

$$R = \begin{bmatrix} 0.4 & 0.5 & 0 \\ 0.9 & 0.5 & 0 \\ 0 & 0 & 0.3 \\ 0.3 & 0.9 & 0.4 \end{bmatrix}$$

By using, $R = 0.3\, R_{0.3} + 0.4\, R_{0.4} + 0.5\, R_{0.5} + 0.9\, R_{0.9}$

Solution :

$$R = 0.3 \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix} + 0.4 \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 1 \end{pmatrix} + 0.5 \begin{pmatrix} 0 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} + 0.9 \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

Max-min composition

- The max-min composition can be interpreted as indicating the strength of the existence of a relation between the elements of X and Z.
- When R_1 and R_2 are expressed as relation matrices, the calculation of R1 O R2 is almost the same as matrix multiplication, except that X and + are replaced by $\wedge$ and $\vee$ respectively. For this reason, the max-min composition is also called the **max-min product**.

Max-min based compositional operation

- The composition operator is the net effect of applying one relation after another. It is defined in most application, using max-min as follows. Let P and Q be defined as :

$$P = \begin{bmatrix} 0.1 & 0.5 \\ 0.6 & 0.9 \end{bmatrix} \qquad Q = \begin{bmatrix} 0.3 & 0.6 & 0.8 \\ 0.7 & 0.5 & 0.4 \end{bmatrix}$$

Then the relation R can be obtained using max-min compositional operations :

$$R = P \circ Q = \begin{bmatrix} 0.2 & 0.5 \\ 0.6 & 0.9 \end{bmatrix} \circ \begin{bmatrix} 0.3 & 0.6 & 0.8 \\ 0.7 & 0.5 & 0.4 \end{bmatrix}$$

$$= \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \end{bmatrix}$$

Min-max composition

- Min-max composition is similar to max-min composition with the difference that the roll of max and min are interchanged.
- The Min-Max composition is denoted by $R_1 \square R_2$ with membership function $\mu R_1 \square R_2$.

Fuzzy sets examples

1. Example of **Equality** : A = 0.3/1 + 0.5/2 + 1/3, B = 0.3/1 + 0.5/2 + 1/3,
   Therefore **A = B**.
2. Example of **Inclusion** : Consider X = {1, 2, 3} and sets A and B

   A = 0.3/1 + 0.5/2 + 1/3; B = 0.5/1 + 0.55/2 + 1/3

   then A is a subset of B, or $A \subseteq B$
3. Example of **Cardinality** : Consider X = {1, 2, 3} and sets A and B

   A = 0.3/1 + 0.5/2 + 1/3; B = 0.5/1 + 0.55/2 + 1/3

   Then card A = 1.8

   card B = 2.05

4. Example : Consider X = {1, 2, 3} and fuzzy set A = 0/1 + 0/2 + 0/3, **then A is empty.**
5. Consider two fuzzy subsets of the set X,

   X = {a, b, c, d, e } referred to as A and B

   A = {1/a, 0.3/b, 0.2/c 0.8/d, 0/e} and B = {0.6/a, 0.9/b, 0.1/c, 0.3/d, 0.2/e}

Solution : i) Support :

supp(A) = {a, b, c, d }

supp(B) = {a, b, c, d, e }

ii) Core :

core(A) = {a}

core(B) = {}

iii) Cardinality :

card(A) = 1 + 0.3 + 0.2 + 0.8 + 0 = 2.3

card(B) = 0.6 + 0.9 + 0.1 + 0.3 + 0.2 = 2.1

iv) Complement :

A = {1/a, 0.3/b, 0.2/c 0.8/d, 0/e}

A = {0/a, 0.7/b, 0.8/c 0.2/d, 1/e}

v) Union :

$A \cup B$ = {1/a, 0.9/b, 0.2/c, 0.8/d, 0.2/e}

vi) Intersection :

$A \cap B$ = {0.6/a, 0.3/b, 0.1/c, 0.3/d, 0/e}

6. Two fuzzy subsets of the set X = {a, b, c, d, e } :

   A = {1/a, 0.3/b, 0.2/c 0.8/d, 0/e}

   and B = {0.6/a, 0.9/b, 0.1/c, 0.3/d, 0.2/e}

Solutions :

- kA :

for k = 0.5

kA = {0.5/a, 0.15/b, 0.1/c, 0.4/d, 0/e}

for $m = 2$

$A_m = \{1/a, 0.09/b, 0.04/c, 0.64/d, 0/e\}$

$A_{0.2} = \{a, b, c, d\}$

$A_{0.3} = \{a, b, d\}$

$A_{0.8} = \{a, d\}$

$A_1 = \{a\}$

1.7.1 Tolerance and Equivalence Relation

- A fuzzy relation R has :

1. Reflexivity $\mu_R(x_i, x_i) = 1$
2. Symmetry $\mu_R(x_i, x_j) = \mu_R(x_j, x_i)$
3. Transitivity $\mu_R(x_i, x_j) = \lambda_1$

$\mu_R(x_j, x_k) = \lambda_2 \rightarrow \mu_R(x_i, x_k) = \lambda$

where $\lambda \geq \min\{\lambda_1, \lambda_2\}$

- Fuzzy tolerance relation R_1 has reflexivity, symmetry. It can be transformed into a fuzzy equivalence relation by at most $(n-1)$ compositions.
- A binary relation R(X, X) that is reflexive and symmetric is usually called a compatibility relation or tolerance relation. When R(X, X) is a reflexive and symmetric fuzzy relation it is sometimes called proximity relation.
- Unlike transitivity, checking reflexivity or symmetry does not refer to any logical connective. That is, whether or not a binary L-relation is reflexive (symmetric) does not depend on logical connectives on L.
- **Example :** Consider a fuzzy relation R(X, X) defined on $X = \{x_1, x_2, \ldots, x_8\}$ by the following membership matrix :

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
x_1	1	.3	0	0	.4	0	0	.6
x_2	.3	1	.5	.3	0	0	0	0
x_3	0	.5	1	0	0	.7	.6	.8
x_4	0	.3	0	1	.2	0	.7	.5
x_5	.4	0	0	.2	1	0	0	0
x_6	0	0	.7	0	0	1	.2	0
x_7	0	0	.6	.7	0	.2	1	.8
x_8	.6	0	.5	.5	0	0	.8	1

Since the matrix is symmetric and all entries on the main diagonal are equal to 1, the relation represented is reflexive, and symmetric therefore it is a compatibility relation.

1.8 Fuzzy If-Then Rules

- Fuzzy sets and fuzzy sets operations are the subjects and verbs of fuzzy logic. If-Then rule statements are used to formulate the conditional statements that comprise fuzzy logic.
- A fuzzy if-then rule is also called fuzzy rule, fuzzy implication or fuzzy conditional statement. It assume the form

If x is A then y is B

Where A and B are linguistic values defined by fuzzy sets on universes of discourse X and Y respectively.

- Often "x is A" is called the **antecedent or premise** while "y is B" is called the **consequence or** conclusion.
- The antecedent describes to what degree the rule applies, while the conclusion assigns a fuzzy function to each of one or more output variables.

Examples :

- If pressure is high, then volume is small.
- If the road is slippery, then driving is dangerous.
- If a tomato is red, then it is ripe.
- If the speed is high, then apply the brake a little.

Example :

Speed and pressure of a steam engine can be expressed with the following linguistic conditional statement

If **Speed** is **Slow** Then **Pressure** should be **High**.

Graphically, this statement looks like

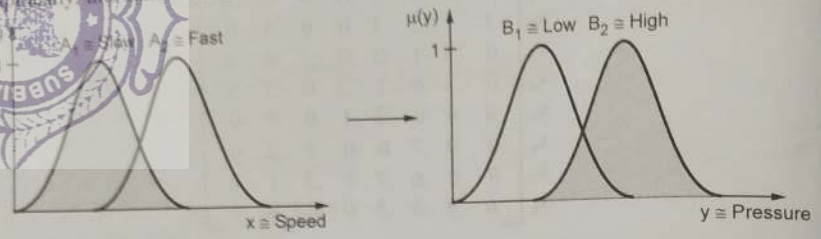


Fig. 1.8.1 If-then rule

- The fuzzy rule "If x is A then y is B" may be abbreviated as $A \rightarrow B$ and is interpreted as $A \times B$. A fuzzy if then rule may be defined (Mamdani) as a binary fuzzy relation R on the product space $X \times Y$.

$$R = A \rightarrow B = A \times B = \int \mu_A(x) * \mu_A(y) \mid (x, y)$$

- If $A \rightarrow B$ is interpreted as **A entails B**, then it can be written as four different formulas :
  - Material implication

$$R = A \rightarrow B = \neg A \cup B$$

  - Propositional calculus

$$R = A \rightarrow B = \neg A \cup (A \cap B)$$

  - Extended propositional calculus

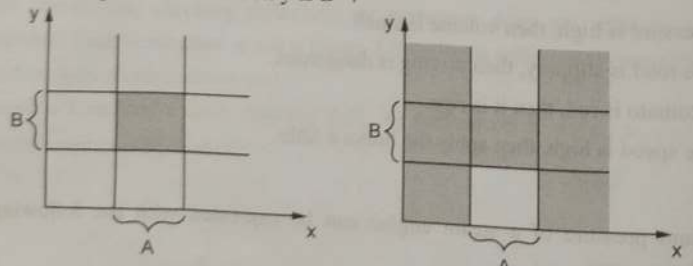
$$R = A \rightarrow B = (\neg A \cap \neg B) \cup B)$$

  - Generalization of modus ponens

$$\mu_R(x, y) = \sup \{ c \mid \mu_A(x) \bar{X} c \leq \mu B(y) \text{ and } 0 \leq c \leq 1 \}$$

- Fig. 1.8.2 shows two interpretation of a fuzzy rule $A \rightarrow B$.

Two ways to interpret "If x is A then y is B" :



(a) A coupled with B (b) A entails B

Fig. 1.8.2 Two interpretation of a fuzzy rule $A \rightarrow B$

- Based on these two interpretations and various T-norm and T-conorm operators, a number of qualified methods can be formulated to calculate the fuzzy rule $A \rightarrow B$. R can be viewed as a fuzzy set with a two dimensional MF :

$$\mu_R(x, y) = f(\mu_A(x), \mu_B(y)) = f(a, b)$$

Here f is called **fuzzy implication function** performs the task of transferring the membership grades of x in A and y in B into those of (x, y) in $A \rightarrow B$.

Example 1.8.1 *For a given set A = {0.2/a, 0.4/b, 1/c, 0.8/d, 0/e}*

B = {0/a, 0.9/b, 0.3/c, 0.2/d, 0.1/e}. Calculate the following :

1. *Support, Core, Cardinality, and Complement for A and B independently,*
2. *Union and Intersection of A and B,*
3. *The new set $C = A^2$*
4. *The new set $D = 0.5 \times B$*
5. *The new set E, which is the alpha cut at $A_{0.5}$*

Solution :

Support

$$\text{Supp}(A) = \{a, b, c, d\}$$

$$\text{Supp}(B) = \{b, c, d, e\}$$

Core

$$\text{Core}(A) = \{c\}$$

$$\text{Core}(B) = \{\}$$

Cardinality

$$\text{Card}(A) = 0.2 + 0.4 + 1 + 0.8 + 0 = 2.4$$

$$\text{Card}(B) = 0 + 0.9 + 0.3 + 0.2 + 0.1 = 1.5$$

Complement

$$\text{Comp}(A) = \{0.8/a, 0.6/b, 0/c, 0.2/d, 1/e\}$$

$$\text{Comp}(B) = \{1/a, 0.1/b, 0.7/c, 0.8/d, 0.9/e\}$$

Union

$$A \cup B = \{0.2/a, 0.9/b, 1/c, 0.8/d, 0.1/e\}$$

Intersection

$A \cap B = \{0/a, 0.4/b, 0.3/c, 0.2/d, 0/e\}$

$C = A^2$

$C = \{0.04/a, 0.16/b, 1/c, 0.64/d, 0/e\}$

$\mathbf{D = 0.5 \times B}$

$D = \{0/a, 0.45/b, 0.15/c, 0.1/d, 0.05/e\}$

$E = A_{0.5}$

$E = \{c, d\}$

Example 1.8.2 *Let A be a fuzzy set defined as A = {0.9/1 + 0.4/2 + 0/3} and given is the fuzzy relation R via the following relational matrix :*

$$R = A \times B = \begin{bmatrix} 1 & 0.8 & 0.1 \\ 0.8 & 0.6 & 0.3 \\ 0.6 & 0.3 & 0.1 \end{bmatrix}$$

Find fuzzy output B in Y using max-min operation.

Solution : Fuzzy output B in Y using max-min operation will be

$$B = A \circ R = \begin{bmatrix} \frac{0.9}{1} & \frac{0.4}{2} & \frac{0}{3} \end{bmatrix} \circ \begin{bmatrix} 1 & 0.8 & 0.1 \\ 0.8 & 0.6 & 0.3 \\ 0.6 & 0.3 & 0.1 \end{bmatrix}$$

$$B = \begin{bmatrix} (0.9, 1) & (0.9,0.8) & (0.9,0.1) \\ (0.4,0.8) & (0.4,0.6) & (0.4,0.3) \\ (0,0.6) & (0,0.3) & (0,0.1) \end{bmatrix}$$

Taking the min values row wise and max values column wise, we get

$$B = \begin{bmatrix} 0.9 & 0.8 & 0.1 \\ 0.4 & 0.4 & 0.3 \\ 0 & 0 & 0 \end{bmatrix} = [\,0.9 \quad 0.8 \quad 0.3\,]$$

1.8.1 Linguistic Variable

- Linguistic variable is "a variable whose values are words or sentences in a natural or artificial language".
- Each linguistic variable may be assigned one or more linguistic values, which are in turn connected to a numeric value through the mechanism of membership functions.

- Conventional techniques for system analysis are intrinsically unsuited for dealing with systems based on human judgment, perception and emotion.
- A linguistic variable is characterized by a quintuple (x, T(x), X, G, M) in which x is the name of the variable, T(x) is the term set of x, X is the universe of discourse; G is syntactic rule which generates the terms in T(x) ; and M is a semantic rule which associates with each linguistic value A its meaning M(A) , where M(A) denotes a fuzzy set in X.
- A fuzzy rule can be defined as a conditional statement in the form :

  If x is A THEN y is B

  where x and y are linguistic variables; and A and B are linguistic values determined by fuzzy sets on the universe of discourses X and Y, respectively.
- Example : If temperature is cold and oil is cheap then heating is high
- If age is interpreted as a linguistic variable, then its term set T(age) could be

  T(age) = { young, not young, very young, not very young, ...
  middle aged, not middle aged,.....,
  old, not old, very old, more or less old, not very old, ...,
  not very young and not very old, ...},

  where each term in T(age) is characterized by a fuzzy set of a universe of discourse X = [0, 100] . It is shown in Fig. 1.8.3.

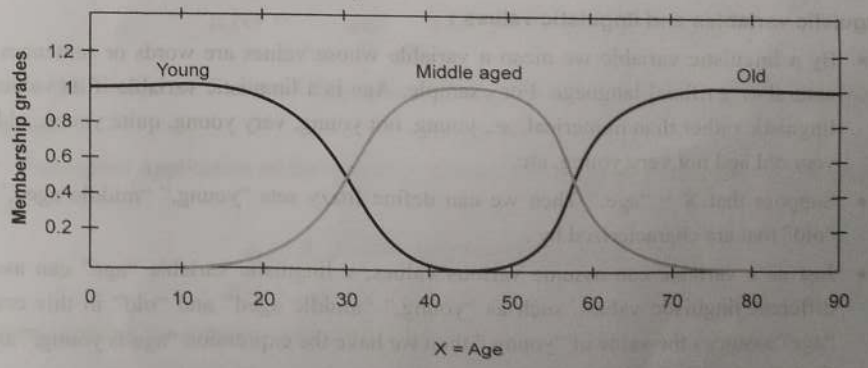


Fig. 1.8.3 Membership function of the term set T(age)

- Here "age is young" is used to denote the assignment of the linguistic value "young" to the linguistic age.

- Let A be a linguistic value characterized by a fuzzy set with membership function $\mu_A(.)$. Then A^k is interpreted as a modified version of the original linguistic value expressed as

$$A^k = x\,[\mu_A(x)]^k / x$$

- The operation of **concentration** is defined as

$$CON(A) = A^2$$

While that of dilation is expressed by

$$DIL(A) = A^{0.5}$$

Constructing MF for composite linguistic terms

- Meaning of the linguistic terms young and old be defined by the following membership functions :

$$\mu_{young}(x) = bell(x, 20, 2, 0) = \frac{1}{1 + (x/20)^4}$$

$$\mu_{old}(x) = bell(x, 30, 3, 100) = \frac{1}{1 + ((x - 100)/30)^6}$$

Where x is the age of a given person, with the interval [0, 100] as the universe of discourse. Now we construct for the following composite linguistic terms :

1. More or less old = DIL(old) = $old^{0.5}$
2. Not young and not old

Linguistic variables and linguistic values :

- By a linguistic variable we mean a variable whose values are words or sentences in a natural or artificial language. For example, Age is a linguistic variable if its values are linguistic rather than numerical, .e., young, not young, very young, quite young, old, not very old and not very young, etc.
- Suppose that X = "age." Then we can define fuzzy sets "young," "middle aged," and "old" that are characterized by .
- Just as a variable can assume various values, a linguistic variable "age" can assume different linguistic values, such as "young," "middle aged" and "old" in this case. If "age" assumes the value of "young," then we have the expression "age is young," and so forth for the other values.
- The name of a linguistic variable is its **label**. The set of values that it can take is called its term set. Each value in the term set is a linguistic value or term defined over a universe.

- In summary : A linguistic variable takes a linguistic value, which is a fuzzy set defined on the universe.
- Example : Let x be a linguistic variable labeled 'Age'. Its term set T could be defined as

T (age) = {very young, young, not very young, more or less old, old}

- Each term is defined on the universe, for example the integers from 0 to 100 years.
- The **support** of a fuzzy set A is the set of all points x in X such that $\mu_A(x) > 0$.
- The **core** of a fuzzy set A is the set of all points x in X such that $\mu_A(x) = 1$.
- A fuzzy set A is **normal** if its core is nonempty. In other words, we can always find at least a point $x \in X$ such that $\mu_A(x) = 1$.
- A **crossover point** of a fuzzy set A is a point $x \in X$ at which $\mu_A(x) = 0.5$.
- A fuzzy set whose support is a single point in X with $\mu_A(x) = 1$ is called a **fuzzy singleton**.

1.8.2 Extension Principle

- Extension principle provides a general procedure for extending crisp domains of mathematical repressions to fuzzy domains. This procedure generalizes a common point to point mapping of a function f(.) to a mapping between fuzzy sets.
- Let $A_1, \ldots, A_n$ be fuzzy sets, defined on $X_1 \ldots X_n$ and let f be a function $f : X_i \ldots\ldots X_n$ V. The extension of f operating on $A_1, \ldots, A_n$ gives a membership function (fuzzy set F) :

$$\mu_F(v) = \sup_{u_1, \ldots u_n \in f^{-1}(v)} \min(\mu A_1(u_1), \ldots, \mu A_n(u_n))$$

when the inverse of f exists. Otherwise define $\mu_F(v) = 0$. Function f is called **inducing mapping.**

- **Example :** Application of the extension principle of fuzzy sets with discrete universes

Let,

$$A = 0.1/-2 + 0.4/-1 + 0.8/0 + 0.9/1 + 0.3/2$$

and

$$f(x) = x^2 - 3$$

Upon applying the extension principle, we have

$$B = 0.1/1 + 0.4/-2 + 0.8/-3 + 0.9/-2 + 0.3/1$$

$$= 0.8/-3 + (0.4 \vee 0.9)/-2 + (0.1 \vee 0.3)/1$$

$$= 0.8/-3 + 0.9/-2 + 0.3/1$$

Where V represents max.

- Suppose that f is the function from X to Y and A is a fuzzy set on X defined as :

$$A = \mu_A(x_1)/x_1 + \mu_A(x_2)/x_2 + \ldots + \mu_A(x_n)/x_n$$

- Then the extension principle states that the image of fuzzy set A under the mapping f(.) can be expressed as a fuzzy set B,

$$B = f(A) = \mu_A(x_1)/y_1 + \mu_A(x_2)/y_2 + \ldots + \mu_A(x_n)/x_n$$

Where $y_i = f(x_i), \quad i = 1, \ldots n.$

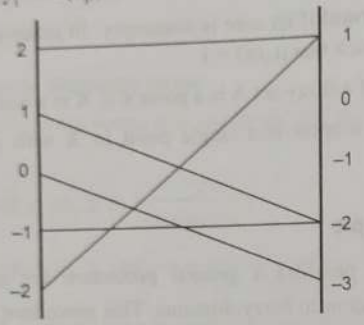


Fig. 1.8.4 Extension principle

- If f(.) is a many to one mapping, then there exist $x_1, x_2 \in X$, $x_1 \neq x_2$, such that $f(x_1) = f(x_2) = y^*, y^* \in Y$. So generally we can write as follows :

$$\mu_B(y) = \max_{x = f^{-1}(y)} \mu_A(x)$$

1.9 Fuzzy Reasoning

- The goal of fuzzy logic is to provide the basis for reasoning under non-binary information. The ensuing reasoning system often this is referred to as "**Approximate reasoning**" or "**Fuzzy reasoning**". Fuzzy reasoning might be considered more exact precisely because it does not assume a binary universe.

1.9.1 Compositional Rule of Inference

- Suppose a curve y = f(x) that regulates the relation between x and y. For given x = a, then from y = f(x) we can infer that y = b = f(a) which is shown in, Fig. 1.9.1.

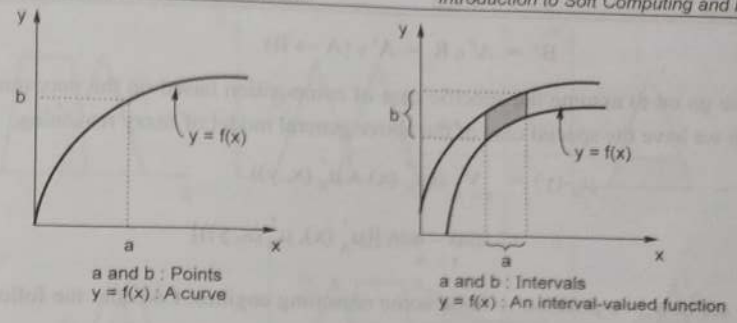


Fig. 1.9.1

- To find resulting interval y = b corresponding to the interval x = a, we first construct a cylindrical extension of "a" and then find its intersection I with the interval valued curve. The projection of I onto the y-axis yields the interval y = b.

1.9.2 Approximate Reasoning

- The basis for formal reasoning is an inference procedure, itself based upon an appropriate model for 'if-then' rules, or modus ponens. The general goal is to infer the associated with a proposition, B, from the implication, A, or $A \rightarrow B$.
- "Consider, 'A' denotes "sharp corner" and 'B' "approach slowly" then we can naturally express the implication by,

premise 1 (fact)	x is A
premise 2 (fact)	IF x is A, THEN y is B
consequence (conclusion)	y is B

- However, we also take for granted that there is an implicit monotonic relationship between the degree of satisfaction in the premise. Thus, a "somewhat sharp" corner enables us to generalize to inferring that the approach should be "more or less" slow, or

premise 1 (fact)	:	x is A'
premise 2 (fact)	:	IF x is A, THEN y is B
consequence (conclusion)	:	y is B'

Definition of fuzzy reasoning : Let A and A' be fuzzy sets on the universe X and B a fuzzy set on Y. Implication, $A \rightarrow B$, is defined in terms of a fuzzy relation R on the Cartesian product, $X \times Y$. The fuzzy proposition B' induced by the premise "x is A" and the fuzzy rule "if x is A then y is B" is defined in the form of the fuzzy composition,

$$B' = A' \circ R = A' \circ (A \rightarrow B)$$

- If we go on to assume the specific case of composition based on the max-min operator, then we have the special case of the above general model of fuzzy reasoning,

$$\mu_{B'}(y) = \bigvee_{x \in A} ((\mu_{A'}(x) \wedge \mu_{R'}(x, y))$$

$$= \max_{x \in A} - \min [(\mu_{A'}(x), \mu_{R'}(x, y))]$$

- We are now in a position to build some reasoning engines. Consider the following three special cases,

1. **Single rule with single premise :** The premise simplifies to the special case of a scalar thresholds.

$$\mu_{B'} = w \wedge \mu_B(y)$$

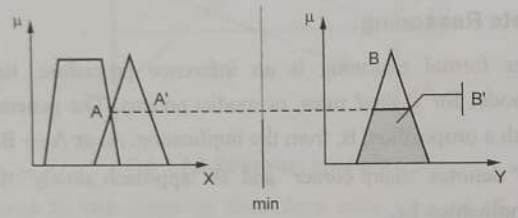


Fig. 1.9.2 Single rule with single premise

2. **Single rule with multiple premises :** The premise simplifies to the special case of the minimum of two scalar thresholds. Thus, the compositional rule implies

$$C' = (A' \times B') \circ (A \times B \rightarrow C) \text{ or } \mu_B = w_1 \wedge w_2 \wedge \mu_C(z)$$

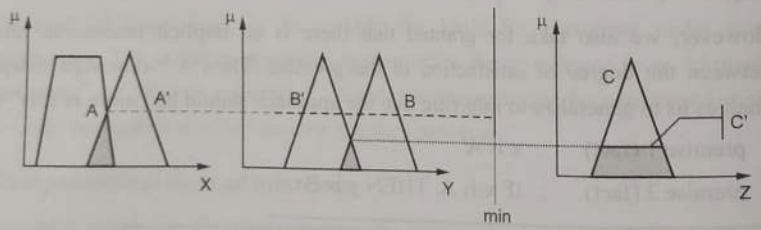


Fig. 1.9.3 Single rule with multiple premises

3. **Multiple rules with multiple premises :** The 'max' operator of equation now applies, thus the area of the implication is the maximum of each minimally thresholded premise.

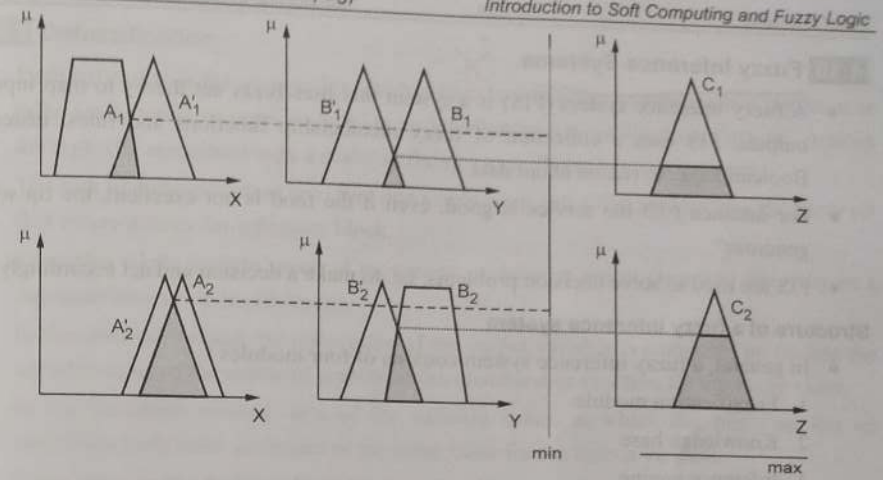


Fig. 1.9.4 Multiple rules with multiple premises

- Process of fuzzy reasoning is divided into four steps :
  1. Degrees of compatiability
  2. Firing strength
  3. Qualified consequent MF
  4. Overall output MF

Advantages of fuzzy system

1. Robust approach to solve many real-world problems.
2. Employable in very complex systems, when there is no simple mathematical model for highly nonlinear processes.
3. Hence, low computational costs and ease at using it in embedded systems.
4. Expert knowledge in complex systems can be formulated in ordinary language.

Disadvantages of fuzzy system

1. The number of rules can grow exponentially inverse with the accuracy level. Undesirable high complexity and rule-chaining problem (Castro, 1999).
2. The rules and the membership function for (imprecise) data must be (accurately) known and defined.
3. Must be combined with an adaptive system (such as neural networks) if some heuristics is desired.

1.10 Fuzzy Inference Systems

- A fuzzy inference system (FIS) is a system that uses fuzzy set theory to map inputs to outputs. FIS uses a collection of fuzzy membership functions and rules, instead of Boolean logic, to reason about data.
- For instance : "If the service is good, even if the food is not excellent, the tip will be generous".
- FIS are used to solve decision problems, i.e. to make a decision and act accordingly.

Structure of a fuzzy inference system

- In general, a fuzzy inference system consists of four modules :
  1. Fuzzification module
  2. Knowledge base
  3. Inference engine
  4. Defuzzification module
- Fig. 1.10.1 shows block diagram of FIS.

1. **Fuzzification module :** Transforms the system inputs, which are crisp numbers, into fuzzy sets. This is done by applying a fuzzification function.
2. **Knowledge base :** Stores IF-THEN rules provided by experts.
3. **Inference engine :** Simulates the human reasoning process by making fuzzy inference on the inputs and IF-THEN rules.

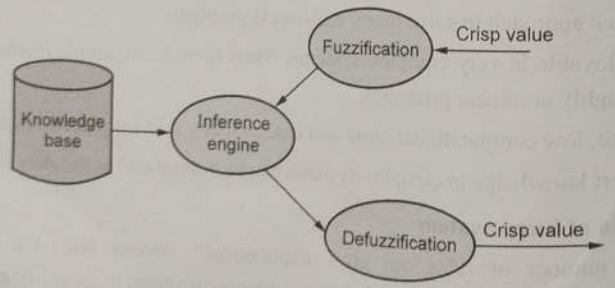


Fig. 1.10.1 Block diagram of FIS

4. **Defuzzification module :** Transforms the fuzzy set obtained by the inference engine into a crisp value. In defuzzification, the fuzzy output set is converted to a crisp number. Some commonly used techniques are the centroid and maximum methods.

1.10.1 Defuzzification

- Defuzzification is the process by which a fuzzy consequent is reduced to a singleton or crisp scalar value in order to provide an interface to a typically scalar 'action'. Actions are typically associated with a scalar defining a specific measurable property.
- The task of defuzzification is to find one single crisp value that summarizes the fuzzy set that enters it from the inference block.
- Examples might include speed of movement, volume of voice, (angular) direction, or a decrease/increase in health/energy.
- In the centroid method, the crisp value of the output variable is computed by finding the variable value of the centre of gravity of the membership function for the fuzzy value.
- In the maximum method, one of the variable values at which the fuzzy set has its maximum truth value is chosen as the crisp value for the output variable.
- Some other methods for defuzzification are : Bisector, middle of maximum, largest of maximum, and smallest of maximum, etc.

Centroid method :

- This most familiar defuzzification scheme. The composite output fuzzy set is built by taking the union of all clipped or scaled output fuzzy sets.
- The crisp output value is then obtained by deriving the centroid or centre of mass of the shape represented by the composite output fuzzy set.
- The x co-ordinate of the centre of mass is the required crisp output value. Because of the variety of shapes that are encountered, the method is computationally intensive.

Centre of sums

- As with the centroid method, the clipped or scaled output fuzzy sets are combined to form a composite output fuzzy set.
- The contribution of each clipped or scaled fuzzy set, however, is considered individually.
- Unlike the Centroid method which builds the output Composite set using the Union operation the Centre of Sums method takes the SUM of the clipped/scaled output Fuzzy sets and then computes the centre of mass of the resulting shape.
- Thus overlapping areas, if they exist are reflected more than once by this technique

Weighted average method

- Weighted average defuzzification instead of using a composite output membership function, uses the individual clipped or scaled output fuzzy sets.

- The method takes the peak value of each clipped/ scaled output fuzzy set and builds the (weighted (with respect to the peak heights) sum of these peak values

Singleton method

- The singleton method is quite a simple method that is widely used in hardware implementations.
- The technique uses output membership functions which are just vertical lines of height 1 in output space, or singletons.
- Each singleton has a single crisp output value associated with it.
- The final crisp output is a weighted average of all qualified output sets.

Why should we use fuzzy inference systems ?

1. Fuzzy logic does not solve new problems. It uses new methods to solve everyday problems.
2. Mathematical concepts within fuzzy reasoning are very simple.
3. Fuzzy logic is flexible : It is easy to modify a FIS just by adding or deleting rules. There is no need to create a new FIS from scratch.
4. Fuzzy logic allows imprecise data (it does NOT work with uncertainty) : It handles elements in a fuzzy set, i.e. membership values. For instance, fuzzy logic works with 'He is tall to the degree 0.8' instead of 'He is 180 cm tall'.
5. Fuzzy logic is built on top of the knowledge of experts : It relies on the know-how of the ones who understand the system.
6. Fuzzy logic can be blended with other classic control techniques.

Classification of fuzzy inference methods

- Fuzzy inference methods are classified in **direct methods** and **indirect methods.** Direct methods, such as Mamdani's and Sugeno's, are the most commonly used. Indirect methods are more complex.

1.10.2 Mamdani Fuzzy Models

- **Goal :** To control a steam engine and boiler combination by a set of linguistic control rules obtained from experienced human operators.
- To compute the output of this FIS given the inputs, one must go through six steps :
  1. Determining a set of fuzzy rules
  2. Fuzzifying the inputs using the input membership functions,
  3. Combining the fuzzified inputs according to the fuzzy rules to establish rule strength.

4. Finding the consequence of the rule by combining the rule strength and the output membership function.
5. Combining the consequences to get an output distribution and
6. Defuzzifying the output distribution (this step is only if a crisp output (class) is needed).

- Fig. 1.10.2 shows the Mamdani fuzzy models.

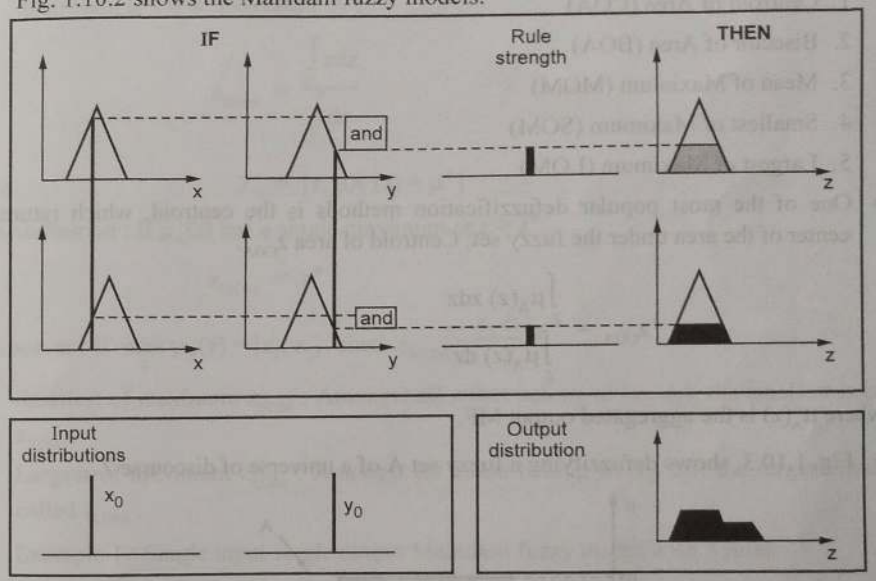


Fig. 1.10.2 Mamdani fuzzy models

- The 1.10.3 illustrations of how a two-rule Mamdani fuzzy inference system derives the overall output z when subjected to two crisp input x and y.
- In Mamdani application, two fuzzy inference systems were used as two controllers to generate the heat input to the boiler and throttle opening of the engine cylinder respectively, to regulate the stream pressure in the boiler and the speed of the engine.
- The plant takes only crisp values as inputs; we have to use a defuzzifier to convert a fuzzy set to a crisp value.
- In Mamdani's application, two fuzzy inference systems were used as two controllers to generate the heat input to the boiler and throttle opening of the engine cylinder, respectively, to regulate the steam pressure in the boiler and the speed of the engine. Since the plant takes only crisp values as inputs, we have to use a defuzzifier to convert a fuzzy set to a crisp value.

Defuzzification

- It converts the fuzzy output of the inference engine to crisp using membership functions analogous to the ones used by the fuzzifier.
- It refers to the way a crisp value is extracted from a fuzzy set as a representative value.
- Five commonly used defuzzifying methods :
  1. Centroid of Area (COA)
  2. Bisector of Area (BOA)
  3. Mean of Maximum (MOM)
  4. Smallest of Maximum (SOM)
  5. Largest of Maximum (LOM)
- One of the most popular defuzzification methods is the centroid, which returns the center of the area under the fuzzy set. Centroid of area z_{COA}.

$$z_{COA} = \frac{\int_Z \mu_A(z)\, z\, dz}{\int_Z \mu_A(z)\, dz}$$

where $\mu_A(z)$ is the aggregated output MF.

- Fig. 1.10.3 shows defuzzifying a fuzzy set A of a universe of discourse Z.

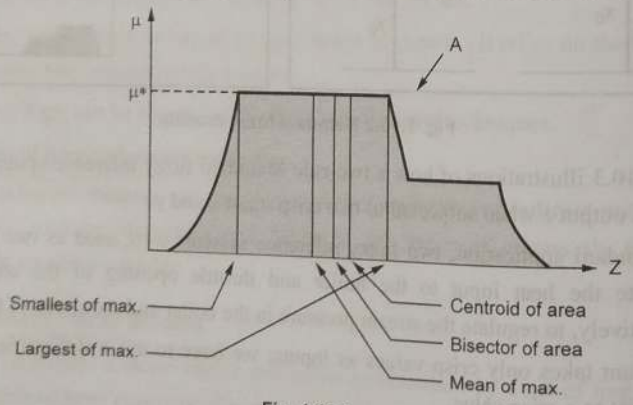


Fig. 1.10.3

- **Bisector of area z_{BOA} :** This operator satisfies the following ;

$$\int_a^{z_{BOA}} \mu_A(z)\, dz = \int_{z_{BOA}}^{\beta} \mu_A(z)\, dz$$

where $\alpha = \min\{z; z \in Z\}$ and $\beta = \max\{z; z \in Z\}$.

The vertical line $z = z_{BOA}$ partitions the region between $z = \alpha$, $z = \beta$, $y = 0$ and $y = \mu_A(z)$ into two regions with the same area.

Mean of maximum z_{MOM}

- This operator computes the average of the maximizing z at which the MF reaches a maximum μ^*. It is expressed by :

$$z_{MOM} = \frac{\int_{Z'} z\, dz}{\int_{Z'} dz}$$

where $Z' = [z; \mu A(z) = \mu^*]$

By definition : If $\mu_A(z)$ has a single maximum at $z = z^*$.

then $z_{MOM} = z^*$

However : if $\max_z \mu_A(z) = [z_1, z_2]$ then $z_{MOM} = \frac{z_1 + z_2}{2}$

- Smallest of maximum z_{SOM} : Amongst all z that belong to $[z_1, z_2]$, the smallest is called z_{SOM}.
- Largest of maximum z_{LOM} : Amongst all z that belong to $[z_1, z_2]$, the largest value is called z_{LOM} .
- Example 1 : Single input single output Mamdani fuzzy model with 3 rules :

If X is small then Y is small $\rightarrow R_1$

If X is medium then Y is medium $\rightarrow R_2$

Is X is large then Y is large $\rightarrow R_3$

X = input $\in [-10, 10]$

Y = output $\in [0, 10]$

Using max-min composition (R_1 o R_2 o R_3) and centroid defuzzification, we obtain the following.

- Example 2 : Two input single-output Mamdani fuzzy model with 4 rules :

If X is small and Y is small then Z is negative large

If X is small and Y is large then Z is negative small

If X is large and Y is small then Z is positive small

If X is large and Y is large then Z is positive large

[– 5, 5]; Z = [– 5, 5] with max-min composition and centroid

we can determine the overall input output surface.

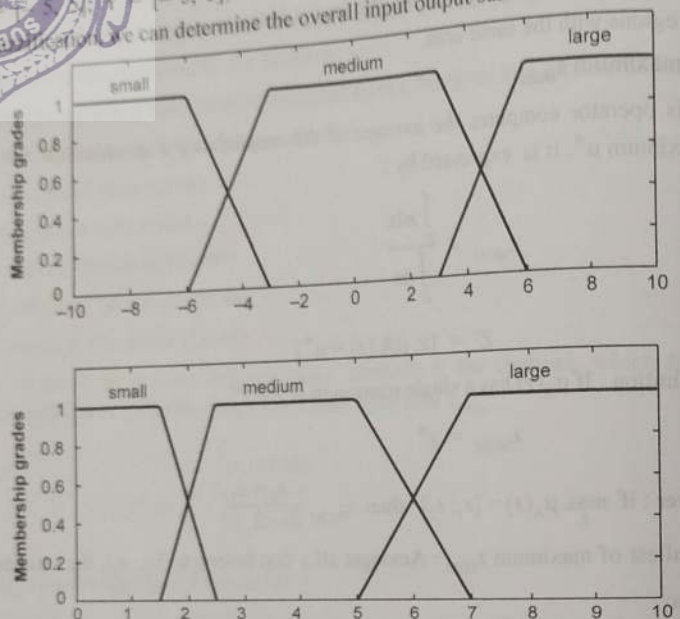


Fig. 1.10.4 Single input single output model

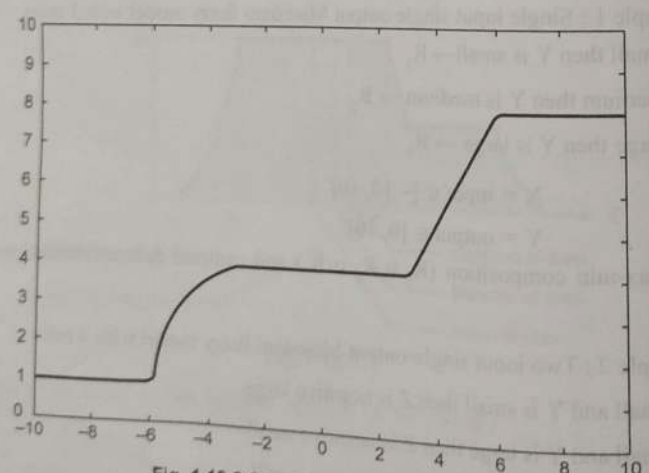


Fig. 1.10.5 Overall input output curve

Other variants

- Classical fuzzy reasoning is "not" tractable, difficult to compute. In practice, a fuzzy inference system may have a certain reasoning mechanism that does not follow the strict definition of the compositional rule of inference.
- To completely specify the operation of a Mamdani fuzzy inference system, we need to assign a function for each of the following operators :
- AND operator (usually T-norm) for the rule firing strength computation with AND'ed antecedents.
- OR operator (usually T-conorm) for calculating the firing strength of a rule with OR'ed antecedents.
- Implication operator (usually T-norm) for calculating qualified consequent MFs based on given firing strength.
- Aggregate operator (usually T-conorm) for aggregating qualified consequent MFs to generate an overall output MF composition of facts and rules to derive a consequent.
- Defuzzification operator for transforming an output MF to a crisp single output value.

Advantages of the Mamdani method

1. It is intuitive.
2. It has widespread acceptance.
3. It is well-suited to human input.

- It is worth mentioning that Mamdani's method is useful when there is a small number of variables. Otherwise, we will find the following difficulties :
  1. The number of rules increases exponentially with the number of variables in the antecedent.
  2. The more rules we construct, the harder is to know if they are suitable for our problem.
  3. If the number of variables in the antecedent is too large, it will be difficult to grasp the casual relationship between the antecedents and the consequents. Therefore, constructing new rules will become harder.
- There are other methods that try to solve these difficulties, such as Sugeno's method.

1.10.3 Sugeno Fuzzy Models

- Also known as TSK fuzzy model. Takagi, Sugeno and Kang was proposed generation of fuzzy rules from a given input-output data set.

- A typical fuzzy rule is of the form : **"If x is A and y is B then z = f(x, y)"**
- A and B are fuzzy sets in the antecedent, while z = f(x, y) is a crisp function in the consequent.
- f(...) is very often a polynomial function w.r.t. x and y
- If f(...) is a first order polynomial, then the resulting fuzzy inference is called a first order Sugeno fuzzy model.
- If f(...) is a constant then it is a zero-order Sugeno fuzzy model (special case of Mamdani model).
- Case of two rules with a first-order Sugeno fuzzy model.
  1. Each rule has a crisp output.
  2. Overall output is obtained via weighted average.
  3. No defuzzyfication required.

Example 1 : Single output-input Sugeno fuzzy model with three rules.

If X is small then Y = 0.1X + 6.4

If X is medium then Y = – 0.5X + 4

If X is large then Y = X – 2

If "small", "medium" and "large" are nonfuzzy sets then the overall input-output curve is a piece wise linear. Following Fig. 1.10.6 (See Fig. 1.10.6 on next page) shows comparison between fuzzy and nonfuzzy rules of example 1.

- However, if we have smooth membership functions (fuzzy rules) the overall input-output curve becomes a smoother one.

Example 2 : Two-input single output fuzzy model with 4 rules

R_1 : If X is small and Y is small then z = – x + y + 1

R_2 : If X is small and Y is large then z = – y + 3

R_3 : If X is large and Y is small then z = – x + 3

R_4 : If X is large and Y is large then z = x + y + 2

$R_1 \rightarrow (x \wedge s)$ and $(y \wedge s) \rightarrow w_1$

$R_2 \rightarrow (x \wedge s)$ and $(y \wedge l) \rightarrow w_2$

$R_3 \rightarrow (x \wedge l)$ and $(y \wedge s) \rightarrow w_3$

$R_4 \rightarrow (x \wedge l)$ and $(y \wedge l) \rightarrow w_4$

Aggregated consequent $\rightarrow F[(w_1, z_1); (w_2, z_2); (w_3, z_3); (w_4, z_4)]$

= Weighted average

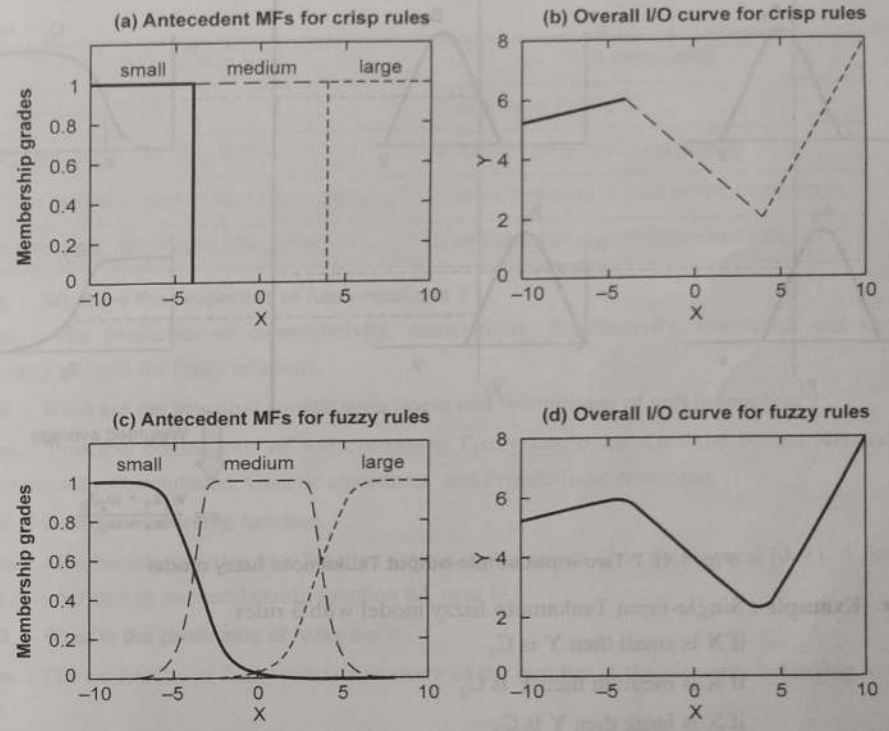


Fig. 1.10.6

1.10.4 Tsukamoto Fuzzy Models

- In the Tsukamoto fuzzy model, the consequent of each fuzzy if-then rule is represented by a fuzzy set with a monotonic MF.
- The inferred output of each rule is defined as crisp value included by the rule's firing strength. The overall output is taken as the weighted average of each rule's output.
- Since each rule infers a crisp output, the Tsukamoto fuzzy model aggregate each rule's output by the method of weighted average and thus avoids the time-consuming process of defuzzification.
- Fig. 1.10.7 shows two-input single-output Tsukamoto fuzzy model.

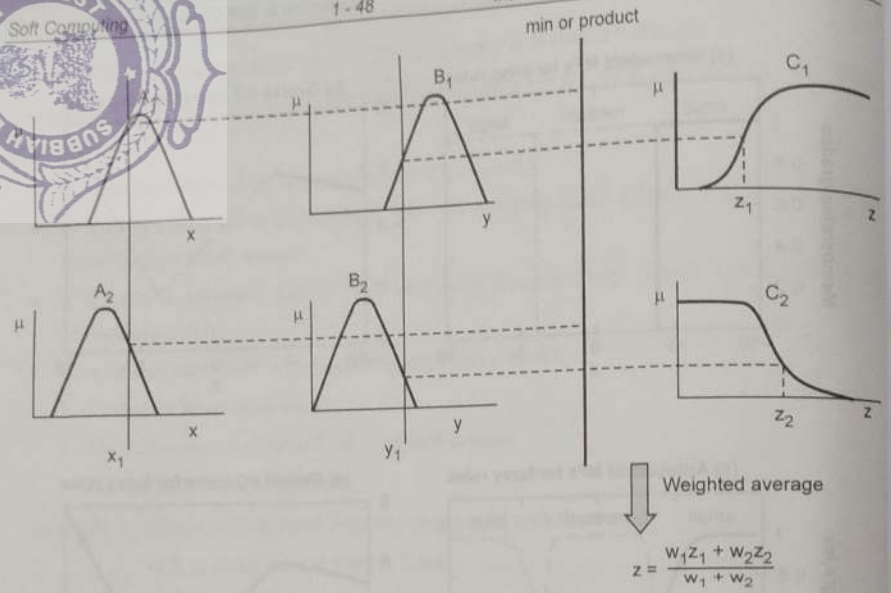


Fig. 1.10.7 Two-input single-output Tsukamoto fuzzy model

- **Example :** Single-input Tsukamoto fuzzy model with 3 rules
  if X is small then Y is C_1
  if X is medium then Y is C_2
  if X is large then Y is C_3

1.11 Two Marks Questions with Answers

Q.1 What is soft computing ?

Ans. : Soft computing is a collection of methodologies that aim to exploit the tolerance for imprecision and uncertainty to achieve tractability, robustness, and low solution cost.

Q.2 Explain unique property of soft computing.

Ans. : Property are :

a. Learning from experimental data

b. Soft computing techniques derive their power of generalization from approximating

c. Generalization is usually done in a high-dimensional space

Q.3 What are the techniques used in soft computing ?

Ans. : Techniques used in soft computing are neural networks, support vector machines, fuzzy logic and genetic algorithms in evolutionary computation.

Q.4 Explain difference between Hard computing and Soft computing

Ans. :

Hard computing	Soft computing
It uses binary logic.	It uses fuzzy logic.
It is based on numerical analysis.	It is based on genetic algorithms.
Crisp system is used in hard computing.	Neuro computing is used in soft computing.
It taken help of differential equations.	It takes help of probabilistic reasoning.

Q.5 What are the properties of fuzzy relations ?

Ans. : The properties of commutativity, associativity, distributivity, involution and idempotency all hold for fuzzy relations.

Q.6 What are the principal constituents (tools and techniques) of soft computing ?

Ans. : Principal constituents of soft computing Fuzzy Logic are Artificial Neural Networks, Evolutionary Computation, Genetic algorithms and Probabilistic reasoning.

Q.7 Define membership function.

Ans. : Membership function is a function from a universal set U to the interval [0, 1]. A fuzzy set A is defined by its membership function ϕA over U.

Q.8 What is the cardinality of fuzzy set ?

Ans. : The cardinality of fuzzy sets is a measure of the number of the elements belonging to the set.

Q.9 What is called the principle of incompatibility ?

Ans. : Conventional techniques for system analysis are intrinsically unsuited for dealing with humanistic systems, whose behavior is strongly influenced by human judgment, perception and emotion. This is the manifestation of what might be called the principle of incompatibility.

Q.10 What is a classical set and fuzzy set ?

Ans. : Classical set is a set with crisp boundary and fuzzy set is a set without crisp boundary.

Q.11 What is degree of membership ?

Ans. : For any element x of universe X, membership function $\mu_A(x)$ equals the degree to which x is an element of set A. This degree, a value between 0 and 1, represents the degree of membership, also called membership value, of element x in set A.

Q.12 How to represent a fuzzy set in a computer ?

Ans. : The membership function must be determined first. A number of methods learned from knowledge acquisition can be applied here. For example, one of the most practical approaches for forming fuzzy sets relies on the knowledge of a single expert.

Q.13 What is a fuzzy rule ?

Ans. : A fuzzy rule can be defined as a conditional statement in the form :

IF x is A

THEN y is B

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

Q.14 State fuzzy inference.

Ans. : Fuzzy inference can be defined as a process of mapping from a given input to an output, using the theory of fuzzy sets.

Q.15 What do you mean defuzzification ?

Ans. : The last step in the fuzzy inference process is defuzzification. Fuzziness helps us to evaluate the rules, but the final output of a fuzzy system has to be a crisp number. The input for the defuzzification process is the aggregate output fuzzy set and the output is a single number.

Q.16 What is linguistic information ?

Ans. : An experienced human operator can usually summarize his or her reasoning process in arriving at final control actions or decisions as a set of fuzzy if-then rules with imprecise but correct membership functions.

Q.17 What is called the principle of incompatibility ?

Ans. : Conventional techniques for system analysis are intrinsically unsuited for dealing with humanistic systems, whose behavior is strongly influenced by human judgment, perception and emotion. This is the manifestation of what might be called the principle of incompatibility.

Q.18 Explain difference between Crisp set and Fuzzy set.

Ans. :

Parameters	Crisp set	Fuzzy set
Basic	Defined by precise and certain characteristics	Prescribed by vague or ambiguous properties
Property	Element is either the member of a set or not	Elements are allowed to be partially included in the set
Application	Digital design	Used in fuzzy controllers
Logic	Bi-valued	Infinite-valued

UNIT II

2 Neural Networks

Syllabus

Supervised Learning Neural Networks - Perceptrons - Backpropagation - Multilayer Perceptrons - Unsupervised Learning Neural Networks - Kohonen Self - Organizing Networks.

Contents

2.1 Introduction to Supervised Learning Neural Networks

- Supervised learning is the machine learning task of inferring a function from supervised training data. The training data consist of a set of training examples. The task of the supervised learner is to predict the output behavior of a system for any set of input values, after an initial training phase.
- **Supervised learning** in which the network is trained by providing it with input and matching output patterns. These input-output pairs are usually provided by an external teacher.
- Human learning is based on the past experiences. A computer does not have experiences.
- A computer system learns from data, which represent some "past experiences" of an application domain.
- To learn a target function that can be used to predict the values of a discrete class attribute, e.g., approve or not-approved and high-risk or low risk. The task is commonly called : Supervised learning, Classification or inductive learning.
- Training data includes both the input and the desired results. For some examples the correct results (targets) are known and are given in input to the model during the learning process. The construction of a proper training, validation and test set is crucial. These methods are usually fast and accurate.
- Have to be able to generalize : give the correct results when new data are given in input without knowing a priori the target.
- Supervised learning is the machine learning task of inferring a function from supervised training data. The training data consist of a set of training examples. In supervised learning, each example is a pair consisting of an input object and a desired output value.
- A supervised learning algorithm analyzes the training data and produces an inferred function, which is called a classifier or a regression function. Fig. 2.1.1 shows supervised learning process.

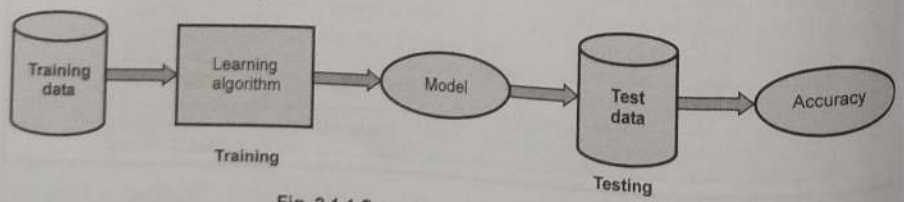


Fig. 2.1.1 Supervised learning process

- The learned model helps the system to perform task better as compared to no learning.
- Each input vector requires a corresponding target vector.

Training Pair = (Input Vector, Target Vector)

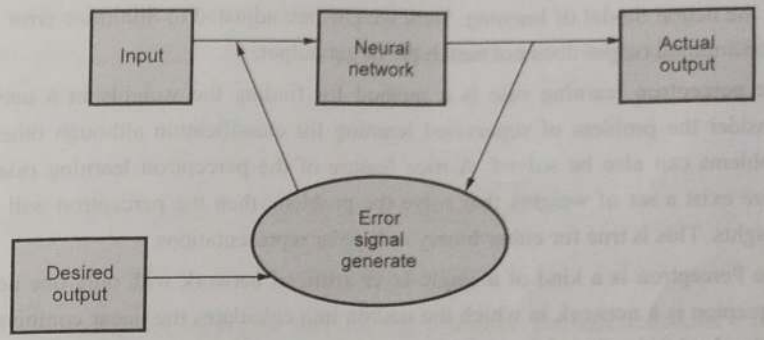


Fig. 2.1.2

- Supervised learning denotes a method in which some input vectors are collected and presented to the network. The output computed by the net-work is observed and the deviation from the expected answer is measured. The weights are corrected according to the magnitude of the error in the way defined by the learning algorithm.
- Supervised learning is further divided into methods which use reinforcement or error correction. The perceptron learning algorithm is an example of supervised learning with reinforcement.
- In order to solve a given problem of supervised learning, following steps are performed :
  1. Find out the type of training examples.
  2. Collect a training set.
  3. Determine the input feature representation of the learned function.
  4. Determine the structure of the learned function and corresponding learning algorithm.
  5. Complete the design and then run the learning algorithm on the collected training set.
  6. Evaluate the accuracy of the learned function. After parameter adjustment and learning, the performance of the resulting function should be measured on a test set that is separate from the training set.

2.2 Perceptron

- The perceptron is a feed-forward network with one output neuron that learns a separating hyper-plane in a pattern space. The perceptron is a classic learning algorithm for the neural model of learning. Here weights are adjusted to minimize error whenever the computed output does not match the target output.
- The perceptron learning rule is a method for finding the weights in a network. We consider the problem of supervised learning for classification although other types of problems can also be solved. A nice feature of the perceptron learning rule is that if there exist a set of weights that solve the problem, then the perceptron will find these weights. This is true for either binary or bipolar representations.
- The Perceptron is a kind of a single-layer artificial network with only one neuron. The Percepton is a network in which the neuron unit calculates the linear combination of its real-valued or boolean inputs and passes it through a threshold activation function.
- Fig. 2.2.1 shows the basic perceptron. The Perceptron is sometimes referred to a Threshold Logic Unit (TLU) since it discriminates the data depending on whether the sum is greater than the threshold value.

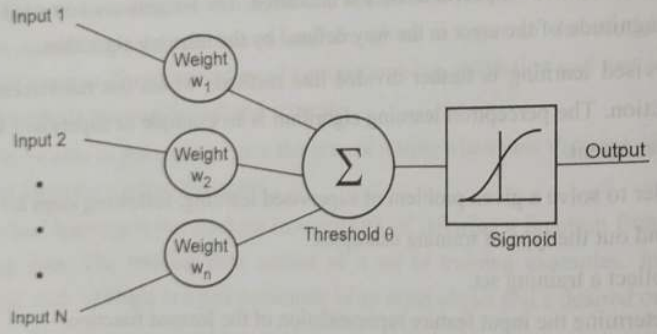


Fig. 2.2.1 Basic perceptron

- The output of the neuron is a linear combination of the inputs rescaled by the synaptic weights.
- Learning is initiated by making adjustments to the relevant connection strengths and a threshold value.
- Here we consider only two class problem. Here output layer usually has only a single node. For an n-class problem (n > 3), the output layer usually has n-nodes, each corresponding to a class and the output node with the largest value indicates which class the input vector belongs to.

- In the first stage, the linear combination of inputs is calculated. Each value of input array is associated with its weight value, which is normally between 0 and 1. Also, the summation function often takes an extra input value Theta with weight value of 1 to represent threshold or bias of a neuron.
- In the simplest case the network has only two inputs and a single output. The output of the neuron is :

$$y = f\left(\sum_{i=1}^{2} w_i x_i + b\right)$$

- Suppose that the activation function is a threshold then

$$f = \begin{cases} 1 & \text{if } s > 0 \\ -1 & \text{if } s \le 0 \end{cases}$$

- The Perceptron can represent most of the primitive boolean functions : AND, OR, NAND and NOR but cannot represent XOR.
- In single layer perceptron, initial weight values are assigned randomly because it does not have previous knowledge. It sum all the weighted inputs. If the sum is greater than the threshold value then it is activated i.e output = 1.

Output

$$w_1 x_1 + w_2 x_2 + \ldots + w_n x_n > \theta \Rightarrow 1$$

$$w_1 x_1 + w_2 x_2 + \ldots + w_n x_n \le \theta \Rightarrow 0$$

- The input values are presented to the perceptron, and if the predicted output is the same as the desired output, then the performance is considered satisfactory and no changes to the weights are made.
- If the output does not match the desired output, then the weights need to be changed to reduce the error.
- The weight adjustment is done as follows :

$$\Delta w = \eta \times d \times x$$

where , x = Input data

d = Predicted output and desired output

η = Learning rate

- If the output of the perceptron is correct then we do not take any action. If the output is incorrect then the weight vector is $w \rightarrow w + \Delta w$
- The process of weight adaptation is called **learning**.

- **Perceptron Learning Algorithm**
  1. Select random sample from training set as input
  2. If classification is correct, do nothing
  3. If classification is incorrect, modify the weight vector w using

$$w_i = w_i + \eta\, d(n)\, x_i(n)$$

- Repeat this procedure until the entire training set is classified correctly.
  a. The term x_i is referred to as **active or excitatory** if its value is 1.
  b. If the value is 0 then it is **inactive**.
  c. If the value is – 1 then it is **inhibitory**.
- The output unit is a linear threshold element with a threshold value θ.

$$0 = f\left(\sum_{i=1}^{n} w_i x_i - \theta\right)$$

$$= f\left(\sum_{i=1}^{n} w_i x_i + w_0\right), w_0 = -\theta$$

$$= f\left(\sum_{i=1}^{n} w_i x_i\right), x_0 = 1$$

where w_i is a modifiable weight associated with an incoming signal x_i.

- Fig. 2.2.2 shows the bias term w_0.

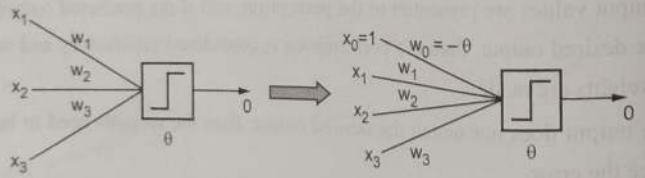


Fig. 2.2.2 Bias term B_0

- The function y = f(x) describes relationship, an input-output mapping from x to y.
- In equation (2.2.1), the f(.) is the **activation function** of the perceptron and it is typically either a **signum function** sgn(x) or **step function** step (x) :

$$\text{sgn}(x) = \begin{cases} 1 & \text{if } x > 0 \\ -1 & \text{otherwise} \end{cases}$$

$$\text{step}(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases} \quad \ldots (2.2.1)$$

- The sum-of-product value is then passed into the second stage to perform the activation function which generates the output from the neuron. The activation function "squashes" the amplitude of the output in the range of [0, 1] or [–1, 1] alternately. The behavior of the activation function will describe the characteristics of an artificial neuron model.
- The basic learning algorithm for a single layer perceptron repeats the following steps until the weights converge :
  1. Select an input vecto x from the training data set.
  2. If the perceptron gives an incorrect response, modify all connection weights w_i according to

$$\Delta w_i = \eta\, t_i\, x_i$$

where t_i is a target output and η is a learning state.

Perceptron convergence theorem

Theorem : If there is a set of weights that correctly classify the (linearly separable) training patterns, then the learning algorithm will find one such weight set, w* in a finite number of iterations.

2.2.1 Single Layer Perceptron

- The perceptron is a feed-forward network with one output neuron that learns a separating hyper-plane in a pattern space. The "n" linear F_x neurons feed forward to one threshold output F_y neuron. The perceptron separates linearly separable set of patterns.
- SLP is the simplest type of artificial neural networks and can only classify linearly separable cases with a binary target (1, 0).
- We can connect any number of McCulloch-Pitts neurons together in any way we like. An arrangement of one input layer of McCulloch-Pitts neurons feeding forward to one output layer of McCulloch-Pitts neurons is known as a **Perceptron.**
- A single layer feed-forward network consists of one or more output neurons, each of which is connected with a weighting factor to all of the inputs X_i.
- The Perceptron is a kind of a single-layer artificial network with only one neuron. The Percepton is a network in which the neuron unit calculates the linear combination of its real-valued or boolean inputs and passes it through a threshold activation function. Fig. 2.2.3 shows Perceptron.

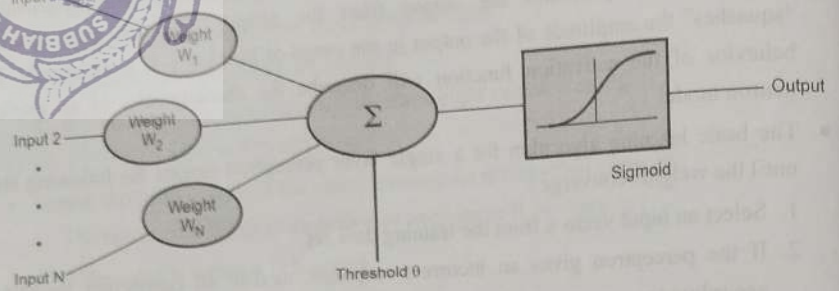


Fig. 2.2.3

- The Perceptron is sometimes referred to a Threshold Logic Unit (TLU) since it discriminates the data depending on whether the sum is greater than the threshold value.

2.2.2 Exclusive-OR Problem

- The XOR function is an operation on two binary values, x_1 and x_2. When exactly one of these binary values is equal to 1, the XOR function returns 1. Otherwise, it returns 0.
- The XOR function provides the target function $y = f^*(x)$ that we want to learn. Our model provides a function $y = f(x; \theta)$ and our learning algorithm will adapt the parameters θ to make f as similar as possible to f^*.
- Neural networks can be used to classify Boolean functions depending on their desired outputs.
- The XOR problem is not **linearly separable**. We cannot use a single layer perceptron to construct a straight line to partition the two dimensional input space into two regions, each containing only data points of the same class.
- However, we may solve the XOR problem by using a single hidden layer with two neurons, as in Fig. 2.2.4.

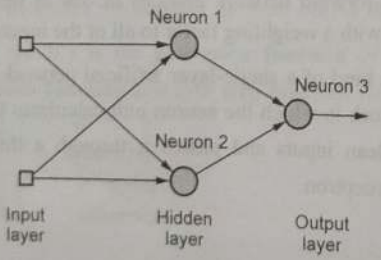


Fig. 2.2.4 Architectural graph of network for solving the XOR problem

- The top neuron, labeled as "Neuron 1" in the hidden layer, is characterized as $w_{11} = w_{12} = +1$
- The slope of the decision boundary constructed by this hidden neuron is equal to –1. The bottom neuron, labeled as "Neuron 2" in the hidden layer, is characterized as $w_{21} = w_{22} = +1$
- The output neuron, labeled as "Neuron 3" is characterized as $w_{31} = -2$, $w_{32} = +1$
- The following assumptions are made here :
  a) Each neuron is represented by a McCulloch–Pitts model, which uses a threshold function for its activation function.
  b) Bits 0 and 1 are represented by the levels 0 and +1, respectively.
- The function of the output neuron is to construct a linear combination of the decision boundaries formed by the two hidden neurons. The bottom hidden neuron has an excitatory (positive) connection to the output neuron, whereas the top hidden neuron has an inhibitory (negative) connection to the output neuron.
- When both hidden neurons are off, which occurs when the input pattern is (0, 0), the output neuron remains off. When both hidden neurons are on, which occurs when the input pattern is (1, 1), the output neuron is switched off again because the inhibitory effect of the larger negative weight connected to the top hidden neuron over powers the excitatory effect of the positive weight connected to the bottom hidden neuron.
- When the top hidden neuron is off and the bottom hidden neuron is on, which occurs when the input pattern is (0, 1) or (1, 0), the output neuron is switched on because of the excitatory effect of the positive weight connected to the bottom hidden neuron.

2.2.3 Madaline Network

- A multilayer network of ADALINE units is known as a MADALINE.
- A Multi-Layer Perceptron (MLP) has the same structure of a single layer perceptron with one or more hidden layers. An MLP is a network of simple neurons called perceptrons.
- A typical multilayer perceptron network consists of a set of source nodes forming the input layer, one or more hidden layers of computation nodes, and an output layer of nodes.
- It is not possible to find weights which enable single layer perceptrons to deal with non-linearly separable problems like XOR.
- Multi-layer perceptrons are able to cope with non-linearly separable problems.

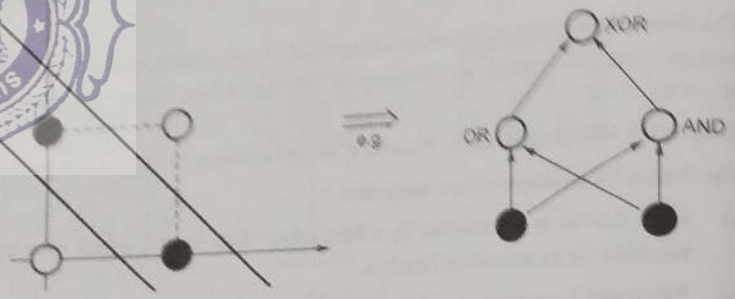


Fig. 2.2.5

- Each neuron in one layer has direct connections to all the neurons of the subsequent layer. MLP can implement nonlinear discriminants (for classification) and nonlinear regression functions (for regression).
- Historically, the problem was that there were no known learning algorithms for training MLPs. Fortunately; it is now known to be quite straightforward. The procedure for finding a gradient vector in the network structure is generally referred to as **backpropagation**. Because the gradient vector is calculated in the direction opposite to the flow of the output of each node.
- **Procedure of backpropagation :**
  1. The output values are compared with the target to compute the value of some predefined error function.
  2. The error is then fedback through the network.
  3. Using this information, the algorithm adjusts the weights of each connection in order to reduce the value of the error function.
- Continue this process until the connection weights in the network have been adjusted so that the network output has converged, to an acceptable level, with the desired output.
- If we use the gradient vector in a simple steepest descent method, the resulting learning paradigm is often referred to as the **backpropagation** learning rule. Backpropagation works by approximating the non-linear relationship between the input and the output by adjusting the weight values internally.
- Generally, the backpropagation network has two stages, training and testing. During the training phase, the network is "shown" sample inputs and the correct classifications. For example, the input might be an encoded picture of a face, and the output could be represented by a code that corresponds to the name of the person.

- Fig. 2.2.6 shows three most commonly used activation functions in backpropagation MLPs.

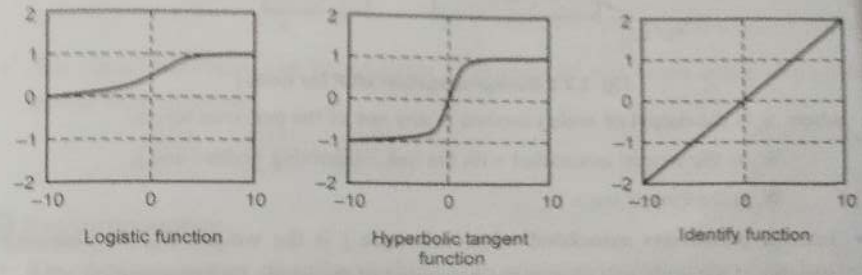


Fig. 2.2.6 Activation function

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

Hyperbolic tangent function : $f(x) = \tanh(x/2) = \frac{1 - e^{-x}}{1 + e^{-x}}$

Identity function : $f(x) = x$

- Both the hyperbolic tangent function and logistic function approximate the signum and step function respectively. Sometimes these two function are referred to as **squashing functions** since the inputs to these functions are squashed to the range [0, 1] or [– 1, 1].
- These functions are also called **sigmoidal functions** because their S-shaped curves exhibits smoothness and asymptotic properties.
- A learning process is organized through a learning algorithm, which is a process of updating the weights in such a way that a machine learning tool implements a given input/output mapping with no errors or with some minimal acceptable error.
- Any learning algorithm is based on a certain learning rule, which determines how the weights shall be updated if the error occurs.

Backpropagation Learning Rule

- The **net input** of a node is defined as the weighted sum of the incoming signals plus a bias term. Fig. 2.2.7 shows the backpropagation MLP for node j. The net input and output of node j is as follows :

$$\overline{X_j} = \Sigma_i + W_{0j} + W_j$$

$$x_j = f\left(\overline{X_j}\right) = \frac{1}{1 + \exp\left(-\overline{X_j}\right)}$$

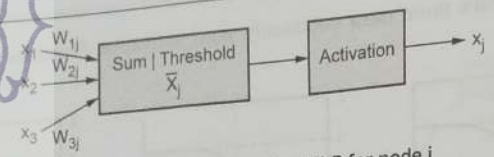


Fig. 2.2.7 Backpropagation MLP for node j

where x_j is the output of node j located in any one of the pervious layers,

W_{ij} is the weight associated with the link connecting nodes i and j,

W_j is the bias of node j.

- Internal parameters associated with each node j is the weight W_{ij}. So changing the weights of the node will change the behaviour of the whole backpropagation MLP.
- Fig. 2.2.8 shows two layer backpropagation MLP.

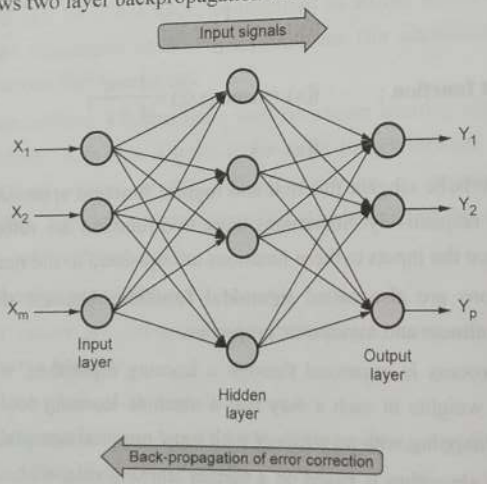


Fig. 2.2.8 Two layer backpropagation MLP

- The above backpropagation MLP will refer to as a 3-4-3 network, corresponding to the number of nodes in each layer.
- The backward error propagation also known as the backpropagation (BP) or the Generalized Delta Rule (GDR). A squared error measure for the p^{th} input-output pair is defined as

$$E_p = \sum_k (d_k - x_k)^2$$

where d_k is the desired output for node k and x_k is the actual output for node k when the input part of the p^{th} data pair is presented.

- To find the gradient vector, an error term $\bar{\in}_i$ for node i is defined as :

$$\bar{\in}_i = \frac{\partial + E_p}{\partial \bar{X}_i}$$

- The partial derivative can be rewritten as product of two terms using chain rule for partial differentiation :

$$\frac{\partial E(t)}{\partial w_{ij}(t)} = \frac{\partial E(t)}{\partial a_i(t)} \bullet \frac{\partial a_i(t)}{\partial w_{ij}(t)}$$

2.3 Backpropagation

- The Backpropagation algorithm looks for the minimum value of the error function in weight space using a technique called the delta rule or gradient descent. The weights that minimize the error function is then considered to be a solution to the learning problem.
- Back propagation is a systematic method for training multiple layer ANN. It is a generalization of Widrow-Hoff error correction rule. 80% of ANN applications uses back propagation.
- Fig. 2.3.1 shows backpropagation network.

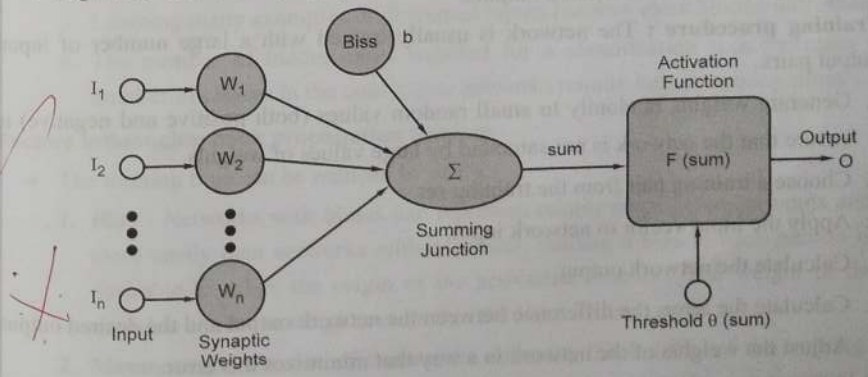


Fig. 2.3.1 Backpropagation network

- **Consider a simple neuron :**
  a. Neuron has a summing junction and activation function.
  b. Any non linear function which differentiable every where and increases everywhere with sum can be used as activation function.
  c. Examples : Logistic function, Arc tangent function, Hyperbolic tangent activation function.

- The activation function makes the multilayer network to have greater representational power than single layer network only when non-linearity is introduced.
- The input to the activation function is sum which is defined by the following equation

$$sum = I_1 W_1 + I_2 W_2 + \ldots + I_n W_n = \sum_{j=1}^{n} I_j W_j + b$$

Need of hidden layers :

1. A network with only two layers (input and output) can only represent the input with whatever representation already exists in the input data.
2. If the data's are discontinuous or non-linearly separable, the innate representation is inconsistent, and the mapping cannot be learned using two layers(Input and Output).
3. Therefore, hidden layer (s) are used between input and output layers

- **Weights** connects unit(neuron) in one layer only to those in the next higher layer. The output of the unit is scaled by the value of the connecting weight, and it is fed forward to provide a portion of the activation for the units in the next higher layer.
- Back propagation can be applied to an artificial neural network with any number of hidden layers. The training objective is to adjust the weights so that the application of a set of inputs produces the desired outputs.
- **Training procedure :** The network is usually trained with a large number of input-output pairs.
  1. Generate weights randomly to small random values (both positive and negative) to ensure that the network is not saturated by large values of weights.
  2. Choose a training pair from the training set.
  3. Apply the input vector to network input.
  4. Calculate the network output.
  5. Calculate the error, the difference between the network output and the desired output.
  6. Adjust the weights of the network in a way that minimizes this error.
  7. Repeat steps 2 - 6 for each pair of input-output in the training set until the error for the entire system is acceptably low.

Forward pass and backward pass :

- Back propagation neural network training involves two passes.
  1. In the forward pass, the input signals moves forward from the network input to the output.

  2. In the backward pass, the calculated error signals propagate backward through the network, where they are used to adjust the weights.
  3. In the forward pass, the calculation of the output is carried out, layer by layer, in the forward direction. The output of one layer is the input to the next layer.

- In the reverse pass,
  a. The weights of the output neuron layer are adjusted first since the target value of each output neuron is available to guide the adjustment of the associated weights, using the delta rule.
  b. Next, we adjust the weights of the middle layers. As the middle layer neurons have no target values, it makes the problem complex.
- **Selection of number of hidden units :** The number of hidden units depends on the number of input units.
  1. Never choose h to be more than twice the number of input units.
  2. You can load p patterns of I elements into $\log_2$ p hidden units.
  3. Ensure that we must have at least 1/e times as many training examples.
  4. Feature extraction requires fewer hidden units than inputs.
  5. Learning many examples of disjointed inputs requires more hidden units than inputs.
  6. The number of hidden units required for a classification task increases with the number of classes in the task. Large networks require longer training times

Factors influencing back propagation training

- The training time can be reduced by using
  1. **Bias :** Networks with biases can represent relationships between inputs and outputs more easily than networks without biases. Adding a bias to each neuron is usually desirable to offset the origin of the activation function. The weight of the bias is trainable similar to weight except that the input is always +1.
  2. **Momentum :** The use of momentum enhances the stability of the training process. Momentum is used to keep the training process going in the same general direction analogous to the way that momentum of a moving object behaves. In back propagation with momentum, the weight change is a combination of the current gradient and the previous gradient

2.3.1 Advantages and Disadvantages of Backpropagation

Advantages of backpropagation :

1. It is simple, fast and easy to program
2. Only numbers of the input are tuned and not any other parameter
3. No need to have prior knowledge about the network
4. It is flexible
5. A standard approach and works efficiently
6. It does not require the user to learn special functions

Disadvantages of backpropagation :

1. Backpropagation possibly be sensitive to noisy data and irregularity
2. The performance of this is highly reliant on the input data
3. Needs excessive time for training
4. The need for a matrix-based method for backpropagation instead of mini-batch

2.4 Multilayer Perceptrons

- A Multi-Layer Perceptron (MLP) has the same structure of a single layer perceptron with one or more hidden layers. An MLP is a network of simple neurons called perceptrons.
- A typical multilayer perceptron network consists of a set of source nodes forming the input layer, one or more hidden layers of computation nodes and an output layer of nodes.
- It is not possible to find weights which enable single layer perceptrons to deal with non-linearly separable problems like XOR : See Fig. 2.4.1.

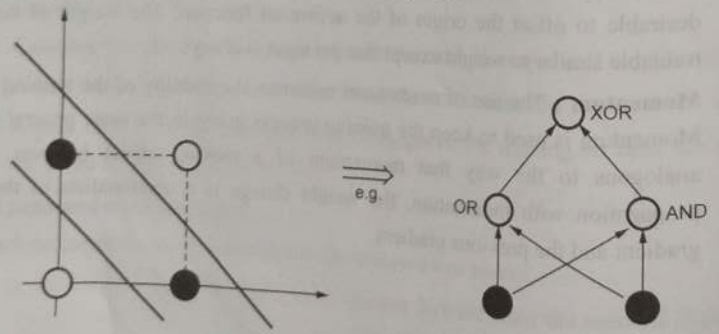


Fig. 2.4.1

2.4.1 Limitation of Learning in Perceptron : Linear Separability

- Consider two-input patterns (X_1, X_2) being classified into two classes as shown in Fig. 2.4.2. Each point with either symbol of × or 0 represents a pattern with a set of values (X_1, X_2).
- Each pattern is classified into one of two classes. Notice that these classes can be separated with a single line L. They are known as linearly separable patterns.

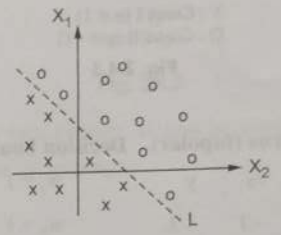
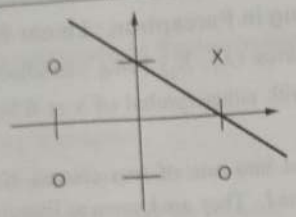


Fig. 2.4.2 Two class

- Linear separability refers to the fact that classes of patterns with - dimensional vector $x = (x_1, x_2, \ldots x_n)$ can be separated with a single decision surface. In the case above, the line L represents the decision surface.
- If two classes of patterns can be separated by a decision boundary, represented by the linear equation then they are said to be linearly separable. The simple network can correctly classify any patterns.
- Decision boundary (i.e., **W, b** or **q**) of linearly separable classes can be determined either by some learning procedures or by solving linear equation systems based on representative patterns of each classes.
- If such a decision boundary does not exist, then the two classes are said to be linearly inseparable.
- Linearly inseparable problems cannot be solved by the simple network, more sophisticated architecture is needed.
- Examples of linearly separable classes

1. Logical **AND** function

Patterns (bipolar)			Decision boundary
x_1	x_2	y	$w_1 = 1$
−1	−1	−1	$w_2 = 1$
−1	1	1	$b = -1$
1	−1	1	$q = 0$
1	1	1	$-1 - x_1 + x_2 = 0$

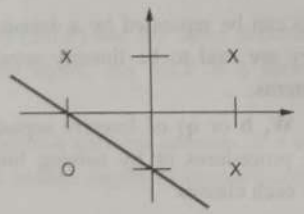


X : Class I (y = 1)
O : Class II (y = –1)

Fig. 2.4.3

2. Logical **OR** function

Patterns (bipolar)			Decision boundary
x_1	x_2	y	$w_1 = 1$
– 1	– 1	– 1	$w_2 = 1$
– 1	1	1	b = 1
1	– 1	1	q = 0
1	1	1	$1 + x_1 + x_2 = 0$



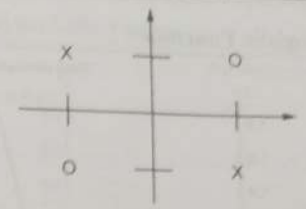
X : Class I (y = 1)
O : Class II (y = –1)

Fig. 2.4.4

- Examples of linearly inseparable classes

1. Logical **XOR** (exclusive OR) function

Patterns (bipolar)		
x_1	x_2	y
– 1	– 1	– 1
– 1	1	1
1	– 1	1
1	1	– 1



X : Class I (y = 1)
O : Class II (y = –1)

Fig. 2.4.5

- No line can separate these two classes, as can be seen from the fact that the following linear inequality system has no solution.

$$\begin{cases} b - w_1 - w_2 < 0 \ (1) \\ b - w_1 + w_2 \geq 0 \ (2) \\ b + w_1 - w_2 \geq 0 \ (3) \\ b + w_1 + w_2 < 0 \ (4) \end{cases}$$

because we have b < 0 from (1) + (4), and b > = 0 from (2) + (3), which is a contradiction.

2.4.2 Activation Functions

- Activation functions also known as transfer function is used to map input nodes to output nodes in certain fashion.
- The activation function is the most important factor in a neural network which decided whether or not a neuron will be activated or not and transferred to the next layer.
- Activation functions help in normalizing the output between 0 to 1 or – 1 to 1. It helps in the process of backpropagation due to their differentiable property. During backpropagation, loss function gets updated, and activation function helps the gradient descent curves to achieve their local minima.
- Activation function basically decides in any neural network that given input or receiving information is relevant or it is irrelevant.
- These activation function makes the multilayer network to have greater representational power than single layer network only when non-linearity is introduced.
- The input to the activation function is sum which is defined by the following equation.

$$\text{sum} = I_1 W_1 + I_1 W_2 + \ldots + I_n W_n$$

$$= \sum_{j=1}^{n} W_j + b$$

- **Activation Function : Logistic Function**

$$f(sum) = \frac{1}{(1+e^{-s^* sum})}$$

$$= (1+e^{-s \times sum})^{-1}$$

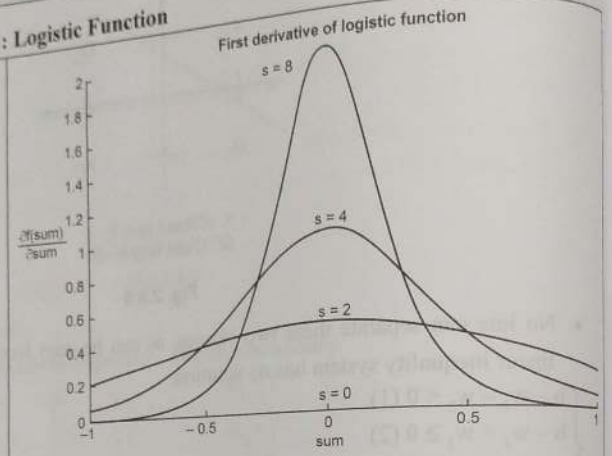


Fig. 2.4.6

- Logistic function monotonically increases from a lower limit (0 or – 1) to an upper limit (+1) as sum increases. In which values vary between 0 and 1, with a value of 0.5 when I is zero.

- **Activation Function : Arc Tangent**

$$f(sum) = \frac{2}{\pi} \tan^{-1} (s \times sum)$$

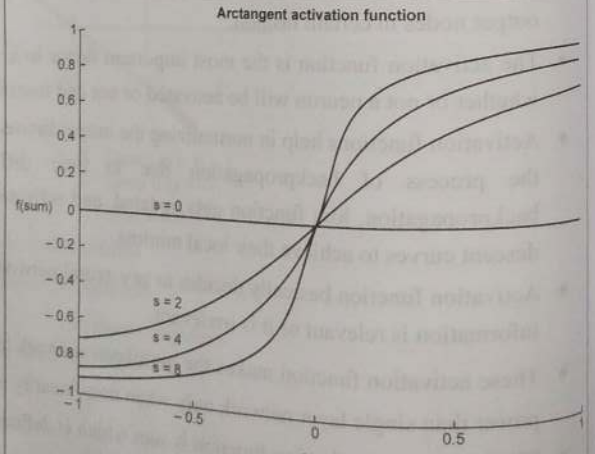


Fig. 2.4.7

- **Activation Function : Hyperbolic Tangent**

$$f(sum) = \tan h\ (s^* I)$$

$$= \frac{e^{s \times sum} - e^{-s \times sum}}{e^{s \times sum} + e^{-s \times sum}}$$

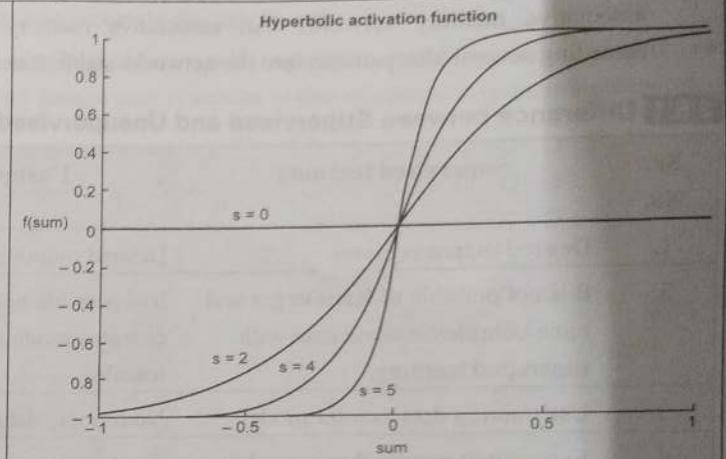


Fig. 2.4.8

2.5 Unsupervised Learning Neural Networks

- The model is not provided with the correct results during the training. It can be used to cluster the input data in classes on the basis of their statistical properties only. Cluster significance and labeling.
- The labeling can be carried out even if the labels are only available for a small number of objects representative of the desired classes. All similar inputs patterns are grouped together as clusters.
- If matching pattern is not found, a new cluster is formed. There is no error feedback.
- External teacher is not used and is based upon only local information. It is also referred to as **self-organization**.
- They are called unsupervised because they do not need a teacher or super-visor to label a set of training examples. Only the original data is required to start the analysis.
- In contrast to supervised learning, unsupervised or self-organized learning does not require an external teacher. During the training session, the neural network receives a number of different input patterns, discovers significant features in these patterns and learns how to classify input data into appropriate categories.
- Unsupervised learning algorithms aim to learn rapidly and can be used in real-time. Unsupervised learning is frequently employed for data clustering, feature extraction etc.

- Another kinds of learning called recording learning by Zurada is typically employed for associative memory networks. An associative memory networks is designed by recording several idea patterns into the networks stable states.

2.5.1 Difference between Supervised and Unsupervised Learning

Sr. No.	Supervised learning	Unsupervised learning
1.	Desired output is given.	Desired output is not given.
2.	It is not possible to learn larger and more complex models than with supervised learning.	It is possible to learn larger and more complex models with unsupervised learning.
3.	Use training data to infer model.	No training data is used.
4.	Every input pattern that is used to train the network is associated with an output pattern.	The target output is not presented to the network.
5.	Trying to predict a function from labeled data	Try to detect interesting relations in data.
6.	Supervised learning requires that the target variable is well defined and that a sufficient number of its values are given.	For unsupervised learning typically either the target variable is unknown or has only been recorded for too small a number of cases.
7.	Example : Optical character recognition.	Example : Find a face in an image.
8.	We can test our model.	We can not test our model.
9.	Supervised learning is also called classification.	Unsupervised learning is also called clustering.

2.6 Kohonen Self-Organizing Networks

- Kohonen self organizing networks are also called **Kohonen features** maps or **topology preserving maps** are used to solve competition based network paradigm for data clustering.
- The Kohonen model provides a topological mapping. It places a fixed number of input patterns from the input layer into a higher-dimensional output or Kohonen layer.

- Training in the Kohonen network begins with the winner's neighbourhood of a fairly large size. Then, as training proceeds, the neighbourhood size gradually decreases.
- Fig. 2.6.1 shows a simple Kohonen self organizing network with 2 inputs and 49 outputs. The learning feature map is similar to that of competitive learning networks.

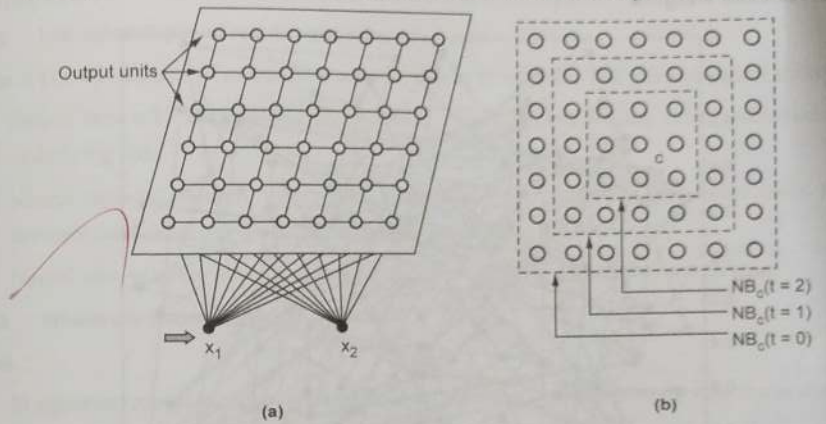


Fig. 2.6.1 Simple Kohonen self organizing network

- A similarity measure is selected and the winning unit is considered to be the one with the largest activation. For this **Kohonen features** maps all the weights in a neighborhood around the winning units are also updated. The neighborhood's size generally decreases slowly with each iteration.
- Step for how to train a Kohonen self organizing network is as follows :

  For n-dimensional input space and m output neurons :

  1. Choose random weight vector wi for neuron i, i = 1, ..., m
  2. Choose random input x
  3. Determine winner neuron k : $||w_k - x|| = \min_i ||w_i - x||$ (Euclidean distance)
  4. Update all weight vectors of all neurons i in the neighborhood of neuron $(k : w_i := w_i + \eta \cdot \varphi(i,k) \cdot (x - w_i))$ (w_i is shifted towards x)
  5. If convergence criterion met, STOP. Otherwise, narrow neighborhood function and learning parameter η and go to (2).

Competitive learning in the Kohonen network

- To illustrate competitive learning, consider the Kohonen network with 100 neurons arranged in the form of a two-dimensional lattice with 10 rows and 10 columns. The network is required to classify two-dimensional input vectors - each neuron in the network should respond only to the input vectors occurring in its region.

- The network is trained with 1000 two-dimensional input vectors generated randomly in a square region in the interval between – 1 and + 1. The learning rate parameter a is equal to 0.1.

1. Initial random weights

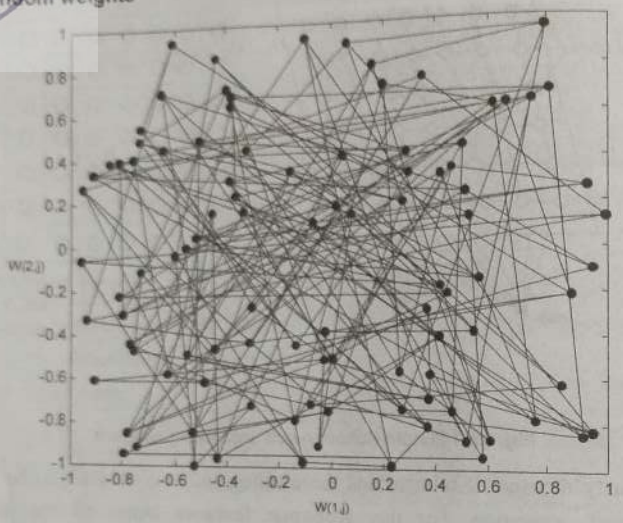


Fig. 2.6.2

2. Network after 10,000 iterations

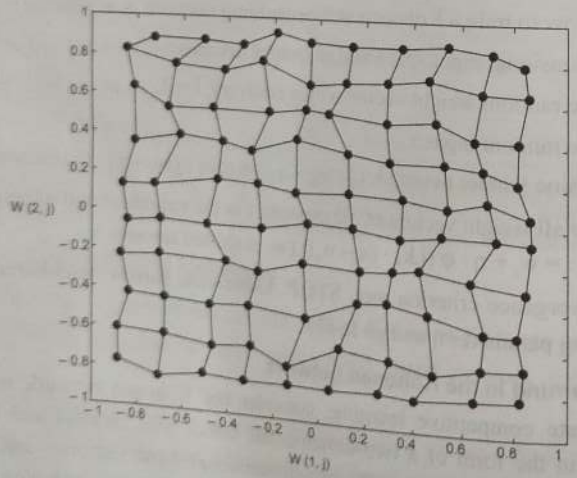


Fig. 2.6.3

2.7 Two Marks Questions with Answers

Q.1 Describe the term Perceptron.

Ans. : An arrangement of one input layer of McCulloch-Pitts neurons feeding forward to one output layer of McCulloch-Pitts neurons is known as a Perceptron.

Q.2 List advantages of neural networks.

Ans. : The advantages of neural networks are due to its adaptive and generalization ability.

a) Neural networks are adaptive methods that can learn without any prior assumption of the underlying data.

b) Neural network, namely the feed forward multilayer perception and radial basis function network have been proven to be universal functional approximations.

c) Neural networks are non-linear model with good generalization ability.

Q.3 Where are neural networks applicable ?

Ans. :

a. In signature analysis : As a mechanism for comparing signatures made with those stored.

b. In process control : There are clearly applications to be made here, most processes cannot be determined as computable algorithms.

c. In monitoring : Networks have been used to monitor the state of aircraft engines.

Q.4 What is supervised learning ?

Ans. : In supervised learning, both the inputs and the outputs are provided. The network then processes the inputs and compares its resulting outputs against the desired outputs. Errors are then propagated back through the system, causing the system to adjust the weights which control the network.

Q.5 Define the term neural network.

Ans. : Neural network is a system composed of many simple processing elements operating in parallel whose function is determined by network structure, connection strengths, and the processing performed at computing elements or nodes.

Q.6 What is unsupervised learning ?

Ans. : In an unsupervised learning, the network adapts purely in response to its inputs. Such networks can learn to pick out structure in their input.

Q.7 What is back propagation algorithm ?

Ans. : The back-prop algorithm is an iterative gradient algorithm designed to minimize the mean square error between the actual output of a multilayer feed-forward perceptron and the desired output. It requires continuous differentiable non-linearity.

Q.8 What is winner takes all ?

Ans. : In competitive learning, neurons compete among themselves to be activated. While in Hebbian learning, several output neurons can be activated simultaneously, in competitive learning, only a single output neuron is active at any time. The output neuron that wins the "competition" is called the winner-takes-all neuron.

Q.9 Discuss Kohonen self organizing networks.

Ans. : Kohonen self organizing networks are also called Kohonen features maps or topology preserving maps are used to solve competition based network paradigm for data clustering. The Kohonen model provides a topological mapping. It places a fixed number of input patterns from the input layer into a higher-dimensional output or Kohonen layer.

Q.10 What are disadvantages of backpropagation ?

Ans. :

1. Backpropagation possibly be sensitive to noisy data and irregularity.
2. The performance of this is highly reliant on the input data.
3. Needs excessive time for training.
4. The need for a matrix-based method for backpropagation instead of mini-batch.

Q.11 Explain advantages of backpropagation.

Ans. :

1. It is simple, fast and easy to program.
2. Only numbers of the input are tuned and not any other parameter.
3. No need to have prior knowledge about the network.
4. It is flexible.
5. A standard approach and works efficiently.
6. It does not require the user to learn special functions.

Q.12 What is need of hidden layers in backpropagation ?

Ans. :

1. A network with only two layers (input and output) can only represent the input with whatever representation already exists in the input data.
2. If the data's are discontinuous or non-linearly separable, the innate representation is inconsistent and the mapping cannot be learned using two layers(Input and Output).
3. Therefore, hidden layer (s) are used between input and output layers.

Q.13 What is perceptron convergence theorem ?

Ans. : If there is a set of weights that correctly classify the (linearly separable) training patterns, then the learning algorithm will find one such weight set, w* in a finite number of iterations.

Q.14 What is activation function ?

Ans. : Activation functions also known as transfer function is used to map input nodes to output nodes in certain fashion. The activation function is the most important factor in a neural network which decided whether or not a neuron will be activated or not and transferred to the next layer. It help in normalizing the output between 0 to 1 or – 1 to 1

Q.15 Define multilayer perceptron.

Ans. : An MLP is a network of simple neurons called perceptrons. A typical multilayer perceptron network consists of a set of source nodes forming the input layer, one or more hidden layers of computation nodes, and an output layer of nodes.

Q.16 Why models are called feedforward ?

Ans. : Models are called feedforward because information flows through the function being evaluated from x, through the intermediate computations used to define f and finally to the output y. There are no feedback connections in which outputs of the model are fed back into itself.

Q.17 Why feedforward neural network are called network ?

Ans. : Feedforward neural networks are called networks because they are typically represented by composing together many different functions. The model is associated with a directed acyclic graph describing how the functions are composed together.

Q.18 What is use of hidden layer ?

Ans. : Feedforward networks have introduced the concept of a hidden layer and this requires us to choose the activation functions that will be used to compute the hidden layer values.

❑❑❑

Notes

UNIT III

3 Genetic Algorithms

Syllabus

Chromosome Encoding Schemes - Population initialization and selection methods - Evaluation function - Genetic operators - Cross over - Mutation - Fitness Function - Maximizing function.

Contents

3.1 Genetic Algorithm Concept

- As early as 1962, John Holland's work on adaptive systems laid the foundation for later developments. By the 1975, the publication of the book Adaptation in Natural and Artificial Systems, by Holland and his students and colleagues.
- Early to mid-1980s, genetic algorithms were being applied to a broad range of subjects. In 1992 John Koza has used genetic algorithm to evolve programs to perform certain tasks. He called his method genetic programming.
- **Genetic Algorithm** (GA) is a search technique used in computing to find true or approximate solutions to optimization and search problems. Genetic algorithms are categorized as global search heuristics.
- Genetic algorithms are a particular class of evolutionary algorithms that use techniques inspired by evolutionary biology such as inheritance, mutation, selection and recombination.
- GA tries to perform an intelligent search for a solution from a nearly infinite number of possible solutions. It works with coding variables. Genetic algorithm and their variants are sometimes referred to as methods of population based optimization.
- Genetic algorithms are also categorized as global search heuristics. Actually they are a particular class of evolutionary algorithms which uses techniques inspired by evolutionary biology such as inheritance, mutation, selection, and crossover.
- A typical genetic algorithm requires two things to be defined :
  1. Genetic representation of the solution domain.
  2. Fitness functions to evaluate the solution domain.

Components of Genetic Algorithm

- Genetic algorithm consists of encoding schemes, fitness evaluations, parent selection, crossover operators, and mutation operators.
- After an initial population is randomly generated, the algorithm evolves the through three operators :
  1. Selection which equates to survival of the fittest;
  2. Crossover which represents mating between individuals;
  3. Mutation which introduces random modifications.
- The Genetic algorithm starts off with an initial population P(0) and it is followed by Selection, Crossover and Mutation processes. These processes are shown in Fig. 3.1.1.

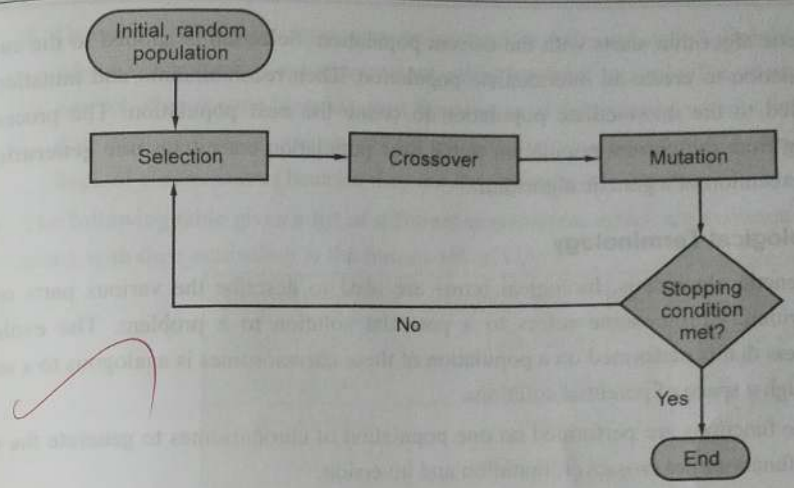


Fig. 3.1.1 One generation of GA process

Psuedocode of Genetics Algorithm

1. Choose the initial population of individuals
2. Evaluate the fitness of each individual in population
3. Repeat until termination condition satisfied :
   - 3a. Selection : Select the individuals with greater fitness for reproduction
   - 3b. Crossover : Breed new individuals through crossover
   - 3c. Mutation : Apply probabilistic mutation on new individuals
   - 3d. Form a new population with these offsprings.
4. Terminate

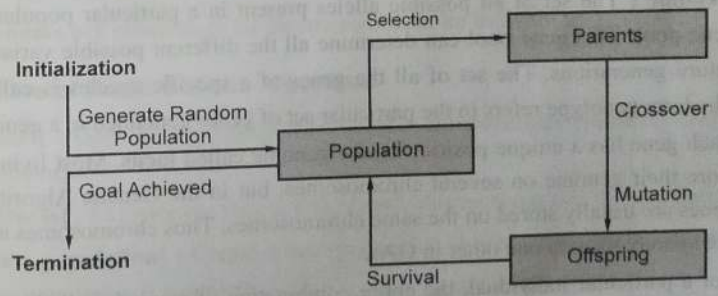


Fig. 3.1.2

- Genetic algorithm starts with the current population. Selection is applied to the current population to create an intermediate population. Then recombination and mutation are applied to the intermediate population to create the next population. The process of moving from current population to the next population constitutes **one generation** in the execution of a genetic algorithm.

3.1.1 Biological Terminology

- In genetic algorithms, biological terms are used to describe the various parts of the algorithm. Chromosome refers to a potential solution to a problem. The evolution process that is performed on a population of these chromosomes is analogous to a search through a space of potential solutions.
- Three functions are performed on one population of chromosomes to generate the next. The functions are crossover, mutation and inversion.

1. **Cell :** Animal or human cell is a complex of many small factories that work together. The center of all this is the cell nucleus. The genetic information is contained in the cell nucleus. Each cell contains the same set of one or more chromosomes.
2. **Chromosomes :** All the genetic information gets stored in the chromosomes. Each chromosome is build of Dioxy Ribo Nucleic Acid (DNA). In human body, a chromosome exists in the form of pairs (23 pairs).
3. **Genes :** A chromosome can be conceptually divided into genes. Each of which encodes a particular protein. Genes code the properties of species i.e., the characteristics of an individual. Every gene has an unique position on the chromosome. The possibilities values of the genes for one property are called **allele** and a gene can take different alleles. Alleles can be either dominant or recessive.
4. **Genome :** The set of all possible alleles present in a particular population forms a gene pool. This gene pool can determine all the different possible variations for the future generations. The set of all the genes of a specific species is called genome. The term genotype refers to the particular set of genes contained in a genome.
5. Each gene has a unique position on the genome called **locus**. Most living organisms store their genome on several chromosomes, but in the Genetic Algorithms, all the genes are usually stored on the same chromosomes. Thus chromosomes and genomes are synonyms with one other in GAs.
6. For a particular individual, the entire combination of genes is called **genotype**. The phenotype describes the physical aspect of decoding a genotype to produce the **phenotype**. A genotype develops to a phenotype. Phenotype describe physical characteristic of the genotype (smart, beautiful, healthy, etc.).

7. Organisms whose chromosomes are arrayed in pairs are called **diploid**; organisms whose chromosomes are unpaired are called **haploid**. In haploid representation, only one set of each gene is stored, thus the process of determining which allele should be dominant and which one should be recessive is avoided. Most GA concentrates on haploid chromosomes because they are much simple to construct.

- The following table gives a list of different expressions, which are common in genetics, along with their equivalent in the framework of GAs :

Natural evolution	Genetic algorithm
genotype	coded string
phenotype	uncoded point
chromosome	string
gene	string position
allele	value at a certain position
fitness	objective function value

Common term used in GA

1. **Individual :-** Any possible solution
2. **Population :-** Group of all individuals
3. **Search Space :-** All possible solutions to the problem
4. **Chromosome :-** Blueprint for an individual
5. **Trait :-** Possible aspect (features) of an individual
6. **Allele :** - Possible settings of trait (black, blond, etc.)
7. **Locus :-** The position of a gene on the chromosome
8. **Genome :-** Collection of all chromosomes for an individual

3.1.2 Outline of Basic Genetic Algorithm

1. **[Start] :** Generate random population of n chromosomes (suitable solutions for the problem).
2. **[Fitness] :** Evaluate the fitness **f(x)** of each chromosome x in the population.
3. **[New population] :** Create a new population by repeating following steps until the new population is complete.
   a. **[Selection] :** Select two parent chromosomes from a population according to their fitness (the better fitness, the bigger chance to be selected).

b. **[Crossover]** : With a crossover probability cross over the parents to form a new offspring (children). If no crossover was performed, offspring is an exact copy of parents.

c. **[Mutation]** : With a mutation probability mutate new offspring at each locus (position in chromosome).

d. **[Accepting]** : Place new offspring in a new population.

4. **[Replace]** : Use new generated population for a further run of algorithm.
5. **[Test]** : If the end condition is satisfied, stop and return the best solution in current population.
6. **[Loop]** : Go to step 2.

GA Steps

Step 1 : Set population size and probability

Step 2 : Define fitness function

Step 3 : Generate initial population

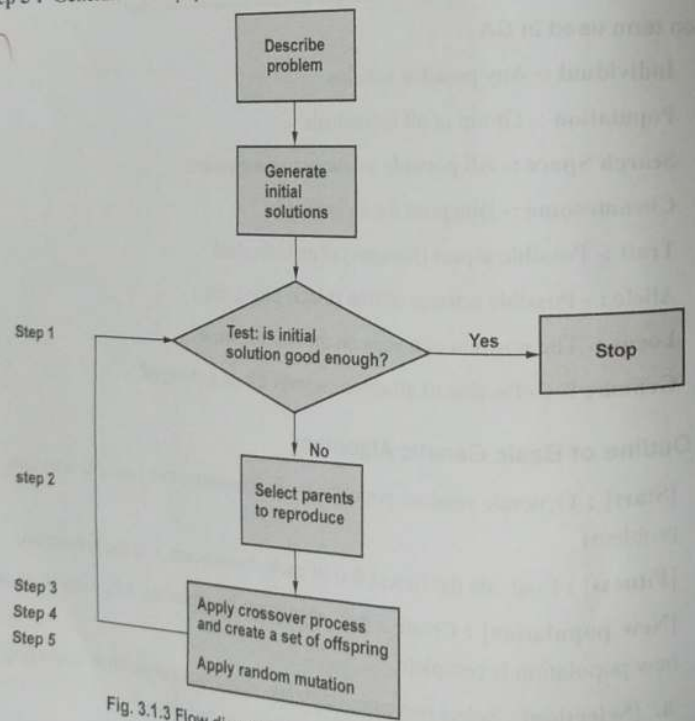


Fig. 3.1.3 Flow diagram of the genetic algorithm process

Step 4 : Calculate fitness for each chromosome

Step 5 : Mating of chromosomes

Step 6 : Create offspring-crossover and mutation

Step 7 : Offspring in new population

Step 8 : Repeat Step 5 until new = initial population

Step 9 : Replace initial population with new

Step 10 : Go to Step 4 and repeat until criteria achieved

3.1.3 Advantages of Genetic Algorithms

- Genetic Algorithm is a stochastic algorithm.
- Randomness as an essential role in both selection and reproduction phases.
- Genetic algorithms always consider a population of solutions. A population base algorithm is also very amenable for parallelization.
- There is no particular requirement on the problem before using genetic algorithms, so it can be applied to resolve any problem (optimization).
- GAs are a new field and parts of the theory have still to be properly established. We can find almost as many opinions on GAs as there are researchers in this field.

3.1.4 Limitations of GA

- GAs are not guaranteed to find the global optimum solution to a problem.
- GAs are an extremely general tool and they have no specific way for solving particular problems.
- GAs are usually used when everything else is failed or when we don't have enough knowledge of the search space.
- Even when such specialized techniques exist, it often interesting to hybridise them with a GA in order to possibly gain some improvements.

3.1.5 Differences between GAs and Traditional Methods

1. Genetic algorithms a coded form of the function values i.e. parameter set, rather than with the actual values themselves. For example, if we want to find the minimum of the function $f(x) = x^3 + x^2 + 5$, the GA would not deal directly with x or y values, but with strings that encode these values. For this case, strings representing the binary x values should be used.
2. Genetic algorithms use a set or population of points to conduct a search, not just a single point on the problem space. This gives GAs the power to search noisy spaces

... local optimum points. Instead of relying on a single point to search ... space, the GAs looks at many different areas of the problem space at ... uses all of this information to guide it.

... algorithms use only payoff information to guide themselves through the ... space. Many search techniques need a variety of information to guide themselves. Hill climbing methods require derivatives, for example. The only information a GA needs is some measure of fitness about a point in the space. Once the GA knows the current measure of "goodness" about a point, it can use this to continue searching for the optimum.

4. GAs are probabilistic in nature, not deterministic. This is a direct result of the randomization techniques used by GAs.
5. GA is inherently parallel. Here lies one of the most powerful features of genetic algorithms. GA's, by their nature, are very parallel, dealing with a large number of points simultaneously.

Parameters	Genetic algorithms	Traditional methods
Work with	Coding of parameter set	Parameters directly
Use information	Payoff i.e. objective function	Payoff plus derivatives etc.
Rules	Probabilistic	Fully deterministic
Search	A population of points	A population of points a single point

3.2 Chromosome Encoding Schemes

- Encoding is the process of representing the solution in the form of a string that conveys the necessary information. It is a process of representing individual genes. The process can be performed using bits, numbers, trees, array or any other objects.
- Just as in a chromosome, each gene controls a particular characteristic of the individual; similarly, each bit in the string represents a characteristic of the solution.
- When choosing an encoding method rely on the following key ideas
  1. Use a data structure as close as possible to the natural representation.
  2. Write appropriate genetic operators as needed.
  3. If possible, ensure that all genotypes correspond to feasible solutions.
  4. If possible, ensure that genetic operators preserve feasibility.

3.2.1 Binary Encoding

- Binary encoding is most common method of encoding. Chromosomes are strings of 1s and 0s and each position in the chromosome represents a particular characteristic of the problem. The length of the string is usually determined according to the desired solution accuracy.

  Chromosome A 110100011010

  Chromosome B 011111111100

- **Octal encoding :** This encoding uses string made up of octal numbers 0 to 7.

  Chromosome A 03467216

  Chromosome B 15723314

- **Hexadecimal encoding :** This encoding uses string made up of hexadecimal numbers 0 to 9 and A to F.

  Chromosome A 9CE7

  Chromosome B 3DBA

3.2.1.1 Permutation Encoding

- Useful in ordering problems such as the Traveling Salesman Problem (TSP). Example : In TSP, every chromosome is a string of numbers, each of which represents a city to be visited.
- Permutation encoding is used in scheduling and ordering problems. Every chromosome is a string of numbers, which represents numbers in a sequence. The kind of encoding usually needs to make some corrections after the crossover and mutation operating procedures because any transformation might create an illegal situation.

  Chromosome A 1 5 3 2 6 4 7 9 8

  Chromosome B 8 5 6 7 2 3 1 4 9

- **Example of problem :** Traveling Salesman Problem (TSP).
- **The problem :** There are cities and given distances between them. Traveling salesman has to visit all of them, but he does not to travel very much. Find a sequence of cities to minimize traveled distance. **Encoding :** Chromosome says order of cities, in which salesman will visit them.

3.2.1.2 Value Encoding

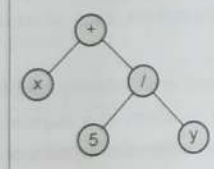
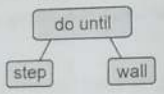
- Used in problems where complicated values, such as real numbers, are used and where binary encoding would not suffice. It is good for some problems, but often necessary to develop some specific crossover and mutation techniques for these chromosomes.

- **Example of problem :** Finding weights for neural network.
- **The problem :** There is some neural network with given architecture. Find weights for inputs of neurons to train the network for wanted output.
- **Encoding :** Real values in chromosomes represent corresponding weights for inputs.

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

3.2.1.3 Tree Encoding

- In **tree encoding** every chromosome is a tree of some objects, such as functions or commands in programming language. It is used in genetic programming.
- **Example of problem :** Finding a function from given values.
- **The problem :** Some input and output values are given. Task is to find a function, which will give the best (closest to wanted) output to all inputs.
- **Encoding :** Chromosome is functions represented in a tree.

Chromosome A	Chromosome B
Chromosome A  (+ x (/ 5 y))	Chromosome B  (do_until step wall)

3.3 Population Initialization and Selection Methods

- The process begins with a set of individuals which is called a Population. Each individual is a solution to the problem you want to solve. An individual is characterized by a set of parameters (variables) known as Genes. Genes are joined into a string to form a Chromosome (solution).
- In a genetic algorithm, the set of genes of an individual is represented using a string, in terms of an alphabet. Usually, binary values are used (string of 1s and 0s). We say that we encode the genes in a chromosome.
- Reproduction selects good strings in a population and forms a mating pool. This is one of the reasons for the reproduction operation to be sometimes known as the **selection operator**.

- **Selection** is the process of choosing two parents from the population for crossing. Thus, in reproduction operation the process of natural selection causes those individuals that encode successful structures to produce copies more frequently.
- To sustain the generation of a new population, the reproduction of the individuals in the current population is necessary. For better individuals, these should be from the fittest individuals of the previous population.
- Through reproduction, genetic algorithms produce new generations of improved solutions by selecting parents with higher fitness ratings or by giving such parents a greater probability of being contributors and by using random selection.
- If the selection pressure is too low, the convergence rate will be slow and the GA will take unnecessarily longer to find the optimal solution. If the selection pressure is too high, there is an increased change of the GA prematurely converging to an incorrect solution.
- Selection scheme are of two types : Proportionate based selection and ordinal based selection

1. Proportionate selection

- In proportionate selection, individuals are assigned a probability of being selected based on their fitness :

$$p_i = f_i \Sigma f_j$$

where p_i is the probability that individual i will be selected,

f_i is the fitness of individual i and

Σf_j represents the sum of all the fitnesses of the individuals with the population.

- This type of selection is similar to using a roulette wheel where the fitness of an individual is represented as proportionate slice of wheel. The wheel is then spun and he slice underneath the wheel when it stops determines which individual becomes a parent.
- There are a number of disadvantages associated with using proportionate selection :
  1. Cannot be used on minimization problems,
  2. Loss of selection pressure (search direction) as population converges,
  3. Susceptible to super individuals

2. Ordinal based selection

In ordinal based selection, individuals are assigned subjective fitness base on the rank within the population :

$$sf_i = \frac{(P - r_i)(max - min)}{(P - 1) + min}$$

where r_i is the rank of individual i,

P is the population size,

Max represents the fitness to assign to the best individual,

Min represents the fitness to assign to the worst individual.

- $p_i = sf_i / \Sigma sf_i$ Roulette wheel selection can be performed using the subjective fitness.
- One disadvantage associated with linear rank selection is that the population must be sorted on each cycle.
- Selection has to balance with variation from crossover and mutation.
  1. Too strong selection means sub optimal highly fit individuals will take over the population, reducing the diversity needed for change and progress.
  2. Too weak selection will result in too slow evolution.
- The various methods of selecting chromosomes for parents to crossover are :
  1. Roulette-Wheel selection
  2. Boltzmann selection
  3. Tournament selection
  4. Rank selection
  5. Steady state selection

3.3.1 Roulette-Wheel Selection

- The commonly-used reproduction operator is the proportionate reproduction operator where a string is selected for the mating pool with a probability proportional to its fitness. Thus, the i^{th} string in the population is selected with a probability proportional to F_i.
- The population size is usually kept fixed in a simple GA, the sum of the probability of each string being selected for the mating pools must be one. Therefore, the probability for selecting the i^{th} string is

$$p_i = \frac{F_i}{\sum_{i=1}^{n} F_i}$$

where n is the population size.

- The expected value of an individual is individual's fitness divided by the actual fitness of the population.

- Each current string in the population has a slot assigned to it which is in proportion to it's fitness. We spin the weighted Roulette wheel thus defined n times (where n is the total number of solutions).
- Each time Roulette Wheel stops, the string corresponding to that slot are created. Strings that are fitter are assigned a larger slot and hence have a better chance of appearing in the new population.
- One way to implement this selection scheme is to imagine a Roulette-wheel with it's circumference marked for each string proportionate to the string's fitness. The Roulette-wheel is spun n times. Each time selecting an instance of the string chosen by the Roulette-wheel pointer.

Example of Roulette wheel selection

Sr. No	String	Fitness	% of Total
1.	01101	169	14.4
2.	11000	576	49.2
3.	01000	64	5.5
4.	10011	361	30.9
Total		1170	100.0

- Fig. 3.3.1 shows a Roulette-wheel for each individual having different fitness values.

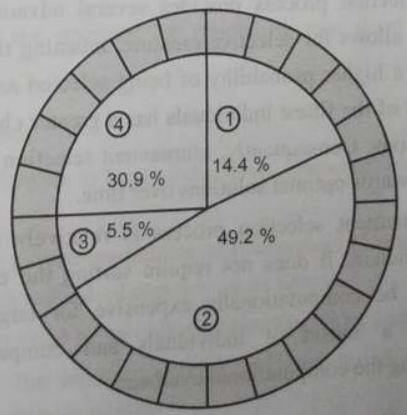


Fig. 3.3.1 Roulette-wheel for each individual having different fitness values

3.3.2 Rank Selection

- Each individual in the population is assigned a numerical rank based on fitness and selection is based on these ranking rather than absolute differences in fitness.

- The advantage of this method is that it can prevent very fit individuals from gaining dominance early at the expense of less fit ones, which would reduce the population's genetic diversity and might hinder attempts to find an acceptable solution.

3.3.3 Tournament Selection

- Subgroups of individuals are chosen from the larger population and members of each subgroup compete against each other. Only one individual from each subgroup is chosen to reproduce.
- Tournament selection works by randomly selecting a subset of individuals from the population, often referred to as the tournament size. The individuals in this subset compete against each other, and the one with the highest fitness value is selected as a parent for reproduction. This process is repeated until the desired number of parents is selected.
- The effectiveness of tournament selection heavily depends on the choice of tournament size. The tournament size determines the number of individuals that compete against each other in each tournament. A larger tournament size increases the selective pressure and promotes the selection of fitter individuals. However, it also reduces the diversity of the population, potentially leading to premature convergence or getting stuck in local optima.
- The tournament selection process provides several advantages over other selection methods. Firstly, it allows for selective pressure, meaning that individuals with higher fitness values have a higher probability of being selected as parents. This ensures that the genetic material of the fittest individuals has a greater chance of being passed on to subsequent generations. Consequently, tournament selection promotes the convergence of the population towards optimal solutions over time.
- Secondly, the tournament selection process is relatively simple to implement and computationally efficient. It does not require sorting the entire population based on fitness, which can be computationally expensive for large populations. Instead, it randomly samples a subset of individuals and compares their fitness values, significantly reducing the computational overhead.

3.3.4 Boltzmann Selection

- In this selection method, a continuously varying temperature controls the rate of selection according to a preset schedule. The temperature starts out high, which means that the selection pressure is low. The temperature is gradually lowered, which gradually

increases the selection pressure, thereby allowing the GA to narrow in more closely to the best part of the search space while maintaining the appropriate degree of diversity.

- Under Boltzmann selection, the selection weight for each population member is,

$$w_a = \exp^{(\beta F_a)}$$

Where β is the selection strength and determines the relative probability of selecting different population members.

- For zero β each population member is selected with equal probability, while for very high β only the fittest population member will be selected.

3.4 Genetic Operators

- Generation of successors is determined by a set of operators that recombine and mutate selected members of the current population. Operators correspond to idealized versions of the genetic operations found in biological evolution.

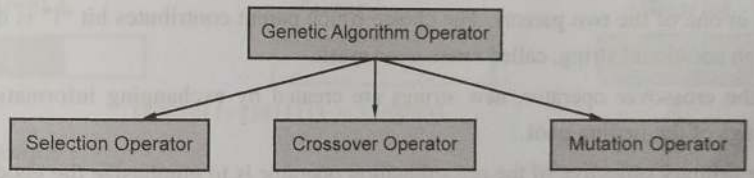


Fig. 3.4.1

- Genetic operator is the operator for generating new gene based on fitness of each individual.

3.4.1 Selection Operator

- **Key idea :** give preference to better individuals, allowing them to pass on their genes to the next generation.
- It determines which parents participate in producing offspring for the next generation. The goodness of each individual depends on its fitness.
- Fitness may be determined by an objective function or by a subjective judgement.
- Selection replicates the most successful solutions found in a population at a rate proportional to their relative quality.
- A common selection method in genetic algorithm is fitness proportionate selection, in which the number of times an individual is expected to reproduce is equal to its fitness divided by the average of fitness in the population.

- Common selection methods used in GAs are
  a. Fitness proportionate selection
  b. Rank selection
  c. Tournament selection

3.4.2 Crossover Operator

- A binary variation operator is called recombination or crossover. A method of mixing good solutions to produce better ones is called crossover.
- Crossover means choosing a random position in the string (say, after 2 digits) and exchanging the segments either to the right or to the left of this point with another string partitioned similarly to produce two new off spring.
- Crossover produces two new offspring from two parent strings by copying selected bits from each parent. Bit at position "i" in each offspring is copied from the bit at position "I" in one of the two parents. The choice which parent contributes bit "I" is determined by an additional string, called **cross-over mask**.
- In the crossover operator, new strings are created by exchanging information among strings of the mating pool.
- The primary objective of the recombination operator is to emphasize the good solutions and eliminate the bad solutions in a population, while keeping the population size constant.
- "Selects The Best, Discards The Rest". "Recombination" is different from "Reproduction".
- Identify the good solutions in a population. Make multiple copies of the good solutions. Eliminate bad solutions from the population so that multiple copies of good solutions can be placed in the population. It is the process in which two chromosomes (strings) combine their genetic material (bits) to produce a new offspring which possesses both their characteristics. Two strings are picked from the mating pool at random to cross over. The method chosen depends on the Encoding Method.
- Crossover can be rather complicated and very depends on encoding of the encoding of chromosome. Specific crossover made for a specific problem can improve performance of the genetic algorithm.

Steps of crossover :

1. The reproduction operator selects at random a pair of two individual strings for the mating.

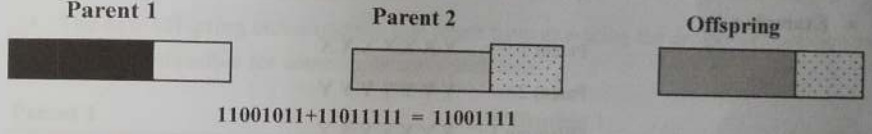
2. A cross site is selected at random along the string length.
3. Finally, the position values are swapped between the two strings following the cross site.

3.4.2.1 Crossover Types

1. Single-point crossover,
2. Two-point crossover,
3. Uniform crossover

1. One point crossover

- One point crossover is the most basic crossover operator. Crossover point on the genetic code is selected at random and two parent chromosomes are interchanged at this point. Binary string from beginning of chromosome to the crossover point is copied from one parent, the rest is copied from the second parent.



11001011+11011111 = 11001111

- **Example :**

Parent 1 : X X | X X X X X

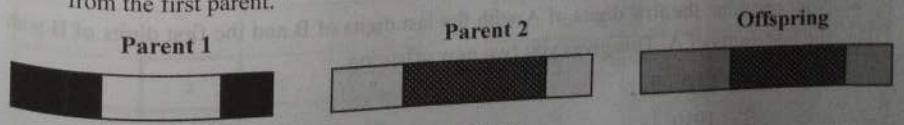
Parent 2 : Y Y | Y Y Y Y Y

Offspring 1 : X X Y Y Y Y Y

Offspring 2 : Y Y X X X X X

2. Two point crossover

- Two-point crossover is very similar to single-point crossover except that two cut-points are randomly generated instead of one.
- Two crossover points are selected and the part of the chromosome string between these two points is then swapped to generate two children. binary string from beginning of chromosome to the first crossover point is copied from one parent, the part from the first chromosome to the first crossover point is copied from the second parent and the rest is copied to the second crossover point is copied from the second parent and the rest is copied from the first parent.



11001011+ 11011111 = 11011111

- **Example :**

Parent 1 : X X | X X X | X X

Parent 2 : Y Y | Y Y Y | Y Y

Offspring 1 : X X Y Y Y X X

Offspring 2 : Y Y X X X Y Y

3. Uniform crossover

- In uniform crossover, a value of the first parent's gene is assigned to the first offspring and the value of the second parent's gene is to the second offspring with a probability value p_c.
- With probability p_c the value of the first parent's gene is assigned to the second offspring and the value of the second parent's gene is assigned to the first offspring.
- **Example :**

Parent 1 : X X X X X X X

Parent 2 : Y Y Y Y Y Y Y

Offspring 1 : X Y X Y Y X Y

Offspring 2 : Y X Y X X Y X

- Crossover between 2 good solutions MAY NOT ALWAYS yield a better or as good a solution. Since parents are good, probability of the child being good is high. If offspring is not good (poor solution), it will be removed in the next iteration during "Selection".
- One site crossover is more suitable when string length is small while two site crossover is suitable for large strings.
- **Crossover example :**

Parent A 011011

Parent B 101100

- Mate the parents by splitting each number as shown between the second and third digits (position is randomly selected)

01*1011 10*1100

- Now combine the first digits of A with the last digits of B and the first digits of B with the last digits of A. This gives you two new offspring.

011100

101011

- If these new solutions or offspring, are better solutions than the parent solutions, the system will keep these as more optimal solutions and they will become parents. This is repeated until some condition (for example number of populations or improvement of the best solution) is satisfied.

Matrix crossover

- Some real problems are naturally suitable for two-dimensional representation. If this kind of problems is to be solved by genetic algorithms, then each possible solution can very conveniently and naturally be conceptually represented as a two-dimensional table.
- Two-dimensional substring crossover generates two offspring chromosomes by choosing only one of the two crossover strategies (horizontal or vertical).
- Alternatively, the two-dimensional crossover operator can be easily modified to generate four offspring chromosomes from a pair of parents by executing the horizontal and the vertical crossovers at the same time.
- The new offspring chromosomes that result from executing the crossover operation may become infeasible for some application problems.

Parent 1

1	3	9	8
5	4	7	2
6	12	11	10

→

Offspring 1

1	3	9	8
5	4	5	3
2	12	7	8

Parent 2

4	6	11	9
10	1	5	3
2	12	7	8

→

Offspring 2

4	6	11	9
10	1	7	2
6	12	11	10

Fig. 3.4.2 Horizontal substring crossover

Parent 1

1	3	9	8
5	4	7	2
6	12	11	10

→

Offspring 1

1	3	11	9
5	4	5	3
6	12	6	8

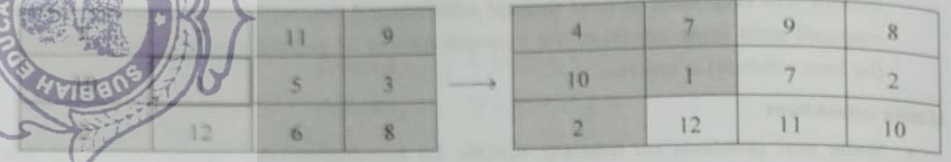


Fig. 3.4.3 Vertical substring crossover

3.4.3 Effects of Genetic Operators

- Using selection alone will tend to fill the population with copies of the best individual from the population.
- Using selection and crossover operators will tend to cause the algorithms to converge on a good but sub-optimal solution.
- Using mutation alone induces a random walk through the search space.
- Using selection and mutation creates a parallel, noise-tolerant, hill climbing algorithm.
- Only crossover can combine information from two parents.
- Only mutation can introduce new information (alleles).
- Crossover does not change the allele frequencies of the population.

3.5 Mutation

- The mutation operator introduces new genetic structures in the population by randomly changing some of its building blocks, helping the algorithm escape local minima traps.
- This operator randomly flips some of the bits in a chromosome. For example, the string 00000100 might be mutated in its second position to yield 01000100. Mutation can occur at each bit position in a string with some probability, usually very small.
- It helps to increase searching power. In the Mutation process, each element in an individual is changed with a small probability of p_m. With some low probability, a portion of the new individuals will have some of their bits flipped.
- Main purpose is to maintain diversity within the population and inhibit premature convergence. Mutation alone induces a random walk through the search space.
- The mutation rate is usually kept low so good chromosomes obtained from crossover are not lost. Mutation increases diversity in the population and thus allows the algorithm to search more distinct points.
- The mutation depends on the encoding as well as the crossover. For example when we are encoding permutations, mutation could be exchanging two genes.

- It is the process by which a string is deliberately changed so as to maintain diversity in the population set. **Mutation probability** determines how often the parts of a chromosome will be mutated.
- Mutation adds new information in a random way to the genetic search process and ultimately helps to avoid getting trapped at local optima. It is an operator that introduces diversity in the population whenever the population tends to become homogeneous due to repeated use of reproduction and crossover operators.
- Mutation may cause the chromosomes of individuals to be different from those of their parent individuals.
- If crossover is supposed to exploit the current solution to find better ones, mutation is supposed to help for exploration of the whole search space. Mutation in a way is the process of randomly disturbing genetic information. They operate at the bit level; when the bits are being copied from the current string to the new string, there is probability that each bit may become mutated.
- The need for mutation is to create a point in the neighborhood of the current point, thereby achieving a local search around the current solution. The mutation is also used to maintain diversity in the population. For example, the following population having four eight bit strings may be considered :

  01101011
  00111101
  00010110
  01111100

- It can be noticed that all four strings have a 0 in the left most bit position. If the true optimum solution requires in that position, then neither reproduction nor crossover operator described above will be able to create 1 in that position.

Example of Mutation

- For chromosomes using binary encoding, randomly selected bits are inverted.

Offspring	1101 **1** 00100 1 **1** 0110
Mutated offspring	1101 **0** 00100 1 **0** 0110

Note : The number of bits to be inverted depends on the mutation probability.

- **Exchange mutation :** The interchange of two randomly chosen elements of the permutation.

- **Shift mutation :** The movement of a randomly chosen element a random number of places to the left or right.
- **Scramble sublist mutation :** Choose two points on the string in random and randomly scrambling the elements between these two positions.
- **Flipping** of a bit involves changing 0 to 1 and 1 to 0 based on a mutation chromosome generated.

Parent	1011 0101
Mutation chromosome	1000 1001
Child	0011 1100

- **Interchanging :** Two random positions of the string are chosen and the bits corresponding to those positions are interchanged.

Parent	10110101
Child	11110001

- **Reserving :** A random position is chosen and the bits next to that position are reversed and child chromosome is produced.

Parent	10110101
Child	10110110

Example : Suppose a Genetic Algorithm uses chromosomes of the form x = abcdefgh with a fixed length of eight genes. Each gene can be any digit between 0 and 9. Let the fitness of individual x be calculated as :

$$f(x) = (a + b) - (c + d) + (e + f) - (g + h)$$

And let the initial population consist of four individuals x1, ... , x4 with the following chromosomes :

$$X1 = 6\ 5\ 4\ 1\ 3\ 5\ 3\ 2$$

$$f(x1) = (6 + 5) - (4 + 1) + (3 + 5) - (3 + 2) = 9$$

$$X2 = 8\ 7\ 1\ 2\ 6\ 6\ 0\ 1$$

$$f(x2) = (8 + 7) - (1 + 2) + (6 + 6) - (0 + 1) = 23$$

$$X3 = 2\ 3\ 9\ 2\ 1\ 2\ 8\ 5$$

$$f(x3) = (2 + 3) - (9 + 2) + (1 + 2) - (8 + 5) = -16$$

$$X4 = 4\ 1\ 8\ 5\ 2\ 0\ 9\ 4$$

$$f(x4) = (4 + 1) - (8 + 5) + (2 + 0) - (9 + 4) = -19$$

The arrangement is (assume maximization)

X2 x1 x3 x4

(the fittest individual) (least fit individual)

Put the calculations in table for simplicity

Individuals	String representation	Fitness	Arrangement assume maximization
X1	65413532	9	X2(fittest individual)
X2	87126601	23	X1(second fittest individual)
X3	23921285	– 16	X3 (third fittest individual)
X4	41852094	– 19	X4 (least fit individual)

So Average fitness : – 0.75 Best : 23 Worst : – 19

Average fitness = (9 + 23 + – 16 + – 19)/ 4 = – 0.75

X2 = 8	7	1	2	6	6	0	1
X1 = 6	5	4	1	3	5	3	2
Offspring 1 = 8	7	1	2	3	5	3	2
Offspring 2 = 6	5	4	1	6	6	0	1
X1 = 6	5	4	1	3	5	3	2
X3 = 2	3	9	2	1	2	8	5
Offspring 3 = 6	5	9	2	1	2	3	2
Offspring 4 = 2	3	4	1	3	5	8	5
X2 = 8	7	1	2	6	6	0	1
X3 = 2	3	9	2	1	2	8	5
Offspring 5 = 8	3	1	2	6	6	8	1

Offspring 9 = 2	7	1	2	6	2	8	1	
Offspring 1 = 8	7	1	2	3	5	3	2	

F (Offspring 1) = (8 + 7) – (1 + 2) + (3 + 5) – (3 + 2) = 15

Offspring 2 = 6	5	4	1	6	6	0	1	

F (offspring 2) = (6 + 5) – (4 + 1) + (6 + 6) – (0 + 1) = 17

Offspring 3 = 6	5	9	2	1	2	3	2	

F (offspring 3) = (6 + 5) – (9 + 2) + (1 + 2) – (3 + 2) = – 2

Offspring 4 = 2	3	4	1	3	5	8	5	

F (offspring 4) = (2 + 3) – (4 + 1) + (3 + 5) – (8 + 5) = – 5

Offspring 5 = 8	3	1	2	6	6	8	2	

F (offspring 5) = (8 + 3) – (1 + 2) + (6 + 6) – (8 + 1) = 11

Offspring 6 = 2	7	1	2	6	2	8	1	

F (offspring 6) = (2 + 7) – (1 + 2) + (6 + 2) – (8 + 1) = 5

Put the calculation in table for simplicity

Individuals	String representation	Fitness
Offspring 1	87123532	15
Offspring 2	65416601	17
Offspring 3	65921232	– 2
Offspring 4	23413585	– 5
Offspring 5	87921201	11
Offspring 6	23926601	5

Average fitness : 6.833 Best : 17 Worst : – 5

So that, the overall fitness is improved, since the average is better and worst is improved.

$$\text{Average fitness} = (15 + 17 + (-5) + (-2) + 11 + 5)/6$$
$$= 6.8333$$

3.5.1 Mutation Rate (P_m)

- When individuals are not subjected to crossover, they remain unmodified. The mutation operator is used to change some elements in selected individuals with a probability P_m,

leading to additional genetic diversity to help the search process escape from local optimal traps.

- Small P_m values are commonly adopted in genetic algorithms.

3.6 Bit-wise Operator

- Encoding is the process of representing the solution in the form of a string that conveys the necessary information. It is a process of representing individual genes. The process can be performed using bits, numbers, trees, array or any other objects.
- Just as in a chromosome, each gene controls a particular characteristic of the individual; similarly, each bit in the string represents a characteristic of the solution.
- When choosing an encoding method rely on the following key ideas
  1. Use a data structure as close as possible to the natural representation.
  2. Write appropriate genetic operators as needed.
  3. If possible, ensure that all genotypes correspond to feasible solutions.
  4. If possible, ensure that genetic operators preserve feasibility.

3.7 Fitness Function

- Take Fitness is an important concept in genetic algorithms. The fitness of a chromosome determines how likely it is that it will reproduce. Fitness is usually measured in terms of how well the chromosome solves some goal problem.
- Fitness can also be subjective (aesthetic). E.g., if the genetic algorithm is to be used to sort numbers, then the fitness of a chromosome will be determined by how close to a correct sorting it produces.
- A fitness function quantifies the optimality of a solution (chromosome) so that particular solution may be ranked against all the other solutions. A fitness value is assigned to each solution depending on how close it actually is to solving the problem.
- Ideal fitness function correlates closely to goal plus quickly computable.
- Example : In TSP, f(x) is sum of distances between the cities in solution. The lesser the value, the fitter the solution is.
- The performance of the individual strings is measured by a fitness function. A fitness function is a problem specific user defined heuristic. After each iteration, the members are given a performance measure derived from the fitness function and the "fittest" members of the population will propagate the next iteration.

Fitness = F_i, Hit = λ_i, Survival = ϕ_i, Death = δ_i

$$F_i = 2\lambda_i - \delta_i + \Sigma\phi_i$$

- The fitness function is defined over the genetic representation and measures the *quality* of the represented solution. The fitness function is always problem dependent.
- For instance, in the knapsack problem we want to maximize the total value of objects that we can put in a knapsack of some fixed capacity. A representation of a solution might be an array of bits, where each bit represents a different object and the value of the bit (0 or 1) represents whether or not the object is in the knapsack.
- Not every such representation is valid, as the size of objects may exceed the capacity of the knapsack. The *fitness* of the solution is the sum of values of all objects in the knapsack if the representation is valid or 0 otherwise. In some problems, it is hard or even impossible to define the fitness expression; in these cases, interactive genetic algorithms are used.

3.8 Two Marks Questions with Answers

Q.1 What is genetic algorithm ?

Ans. : A genetic algorithm is a search technique used in computing to find true or approximate solutions to optimization and search problems.

Q.2 What do you mean genetic programming ?

Ans. : Genetic programming is genetic algorithm wherein the population contains programs rather than bit strings.

Q.3 Explain the "Darwinian theory of survival".

Ans. : More individuals are produced each generation that can survive. Phenotypic variation exists among individuals and the variation is heritable. Those individuals with heritable traits better suited to the environment will survive.

Q.4 State the importance of genetic algorithm.

Ans. : Genetic algorithm is that the problem solving strategy involves using "the strings' fitness to direct the search; therefore they do not require any problem-specific knowledge of the search space, and they can operate well on search spaces that have gaps, jumps or noise". As each individual string within a population directs the search, the genetic algorithm searches, in parallel, numerous points on the problem state space with numerous search directions.

Q.5 Compare and contract genetic algorithm with traditional algorithm.

Ans. :

Genetic algorithm	Traditional algorithm
GA generates a population of points at each iteration. The best point in the population approaches an optimal solution.	It generates a single point at each iteration. The sequence of points approaches an optimal solution.
Selects the next population by computation which uses random number generators.	Selects the next point in the sequence by a deterministic computation.
Convergence in each iteration in problem independent.	Improvement in each iteration is problem specific.
Rules are probabilistic.	Rules are fully deterministic.

Q.6 What two requirements should a problem satisfy it by a genetic algorithm ?

Ans. : GA can only be applied to problems that satisfy the following requirements :

i) The fitness function can be well define.

ii) Solutions should be decomposable into steps (building blocks) which could be then encoded as chromosomes.

Q.7 What is use of crossover operator ?

Ans. : A crossover operator is used to recombine two strings to get a better string. In crossover operation, recombination process creates different individuals in the successive generations by combining material from two individuals of the previous generation.

Q.8 Explain two point crossover.

Ans. : Two crossover points are selected, binary string from the beginning of the chromosome to the first crossover point is copied from the first parent, the part from the first to the second crossover point is copied from the other parent and the rest is copied from the first parent again.

Q.9 List the basic components used in all genetic algorithms.

Ans. : The basic components common to almost all genetic algorithms are :

1. Fitness function for optimization
   a. A population of chromosomes
   b. Selection of which chromosomes will reproduce
   c. Crossover to produce next generation of chromosomes
   d. Random mutation of chromosomes in new generation

Q.10 What is role of fitness function in GA ?

Ans. : A fitness function is a type of objective functions which summarizes the goodness of a solution with a single figure of merit. A fitness function quantifies the optimality of a solution (chromosome) so that particular solution may be ranked against all the other solutions. A fitness value is assigned to each solution depending on how close it actually is to solving the problem.

Q.11 Define the term encoding.

Ans. : Encoding is the process of representing the solution in the form of a string that conveys the necessary information. It is a process of representing individual genes. The process can be performed using bits, numbers, trees, array or any other objects.

Q.12 Explain use of tree encoding.

Ans. : In tree encoding every chromosome is a tree of some objects, such as functions or commands in programming language. It is used in genetic programming.

Q.13 Reproduction is sometime called as selection operator. Why ?

Ans. : Reproduction selects good strings in a population and forms a mating pool. This is one of the reasons for the reproduction operation to be sometimes known as the selection operator.

Q.14 What are the various methods of selecting chromosomes for parents to crossover ?

Ans. : Various methods are :

1. Roulette-Wheel selection
2. Boltzmann selection
3. Tournament selection
4. Rank selection
5. Steady state selection

Q.15 How is a population with increasing fitness generated ?

Ans. : If two parents have superior fitness, there is a good chance that a combination of their genes will produce an offspring with even higher fitness.

Q.16 Define mutation rate.

Ans. : Mutation rate is the probability of mutation which is used to calculate number of bits to be muted.

❑❑❑

UNIT IV

4 Neuro Fuzzy Modeling

Syllabus

ANFIS architecture - hybrid learning - ANFIS as universal approximator - Coactive Neuro fuzzy modeling - Framework - Neuron functions for adaptive networks - Neuro fuzzy spectrum - Analysis of Adaptive Learning Capability

Contents

4.1 ANFIS Architecture

- ANFIS stands for Adaptive-Network-based Fuzzy Inference System. The ANFIS is a data driven procedure representing a neural network approach for the solution of function approximation problems. Data driven procedures for the synthesis of ANFIS networks are typically based on clustering a training set of numerical samples of the unknown function to be approximated.
- ANFIS networks have been successfully applied to classification tasks, rule-based process control, pattern recognition and similar problems.
- Fig. 4.1.1 shows basic structure of the adaptive network based fuzzy inference system.

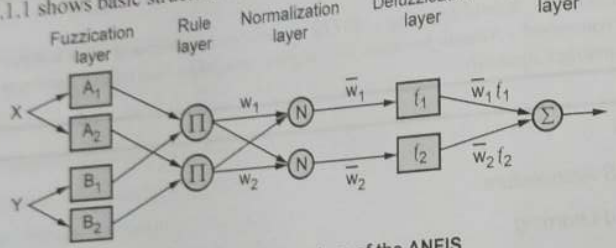


Fig. 4.1.1 Basic structure of the ANFIS

- **Layer 1 :** This was the fuzzification layer. It consisted of defined membership functions of the input variables. The output was a degree of membership value that was calculated based on a Gaussian membership function.

$$\mu_{A_i}(x) = \exp\left[-\left(\frac{x - c_i}{2\delta_i^2}\right)^2\right]$$

where c_i, δ_i are parameters of a membership function.

Layer 2 : This layer executed fuzzy AND of the previous part of the fuzzy rules.

$$w_i = \mu_{A_i}(x) * \mu_{B_i}(x)$$

Layer 3 : This was the normalized layer. This layer normalized the membership functions.

$$\bar{w}_i = \frac{w_i}{\sum_i w_i}$$

Layer 4 : This was the defuzzification layer. This layer executed the consequent part of the fuzzy rules.

$$\bar{w}_i f_i = \bar{w}_i(p_i x + q_i y + r_i)$$

where p_i, q_i, r_i are linear parameters.

Layer 5 : This output and combination layer was calculated by summing up the outputs of previous layers.

$$\sum_i \bar{w}_i f_i = \frac{\sum_i w_i f_i}{\sum_i w_i}$$

- ANFIS serve as a basis for constructing a set of fuzzy if-then rules with appropriate membership functions to generate the stipulated input-output pairs.
- In the ANFIS structure, it is observed that given the values of premise parameters, the final output can be expressed as a linear combination of the consequent parameters.

4.1.1 Advantages and Limitations of ANFIS

Advantages :

1. Faster convergence than typical feed forward neural network.
2. It uses small size training set.
3. Compact model.
4. ANFIS is capable of handling complex and nonlinear problems.

Limitations :

1. Surface collisions around the points.
2. Coefficient signs not always consistent with underlying monotonic relations.
3. The learning procedure of ANFIS does not provide the means to apply constraints that restrict the kind of modifications applied to the membership functions.

4.2 Hybrid Learning

- The ANFIS can be trained by a hybrid learning algorithm. In the forward pass the algorithm uses least-squares method to identify the consequent parameters on the layer 4. In the backward pass the errors are propagated backward and the premise parameters are updated by gradient descent.
- We can apply the gradient method to identify the parameters in an adaptive network, the method is generally slow and likely to become trapped in local minima. Hybrid learning rule combines the gradient method and the least square estimate to identify parameters.
- We assume that, the adaptive network under consideration has only one output.

$$\text{Output} = F(\vec{I}, S)$$

where $\vec{I}$ is the set of input variables and S is the set of parameters and F is the function implemented by the ANFIS.

- If there exists a function H such that the composite function H o F is the linear in some of the elements of S, then the elements can be identified by the least square method.
- If the parameter set S can be decomposed into two sets $S = S_1 \oplus S_2$ where $\oplus$ represents direct sum, such that H o F is linear in the elements of S_2, then upon applying H to output, we have

$$H(\text{output}) = H \circ F(\vec{I}, S)$$

which is linear in the elements of S_2.

- Now given values of elements of S_1, we can plug P training data and obtain a matrix equation :

$$AX = B$$

where X is an unknown vector whose elements are parameters in S_2.

- Let $|S_2| = M$, then the dimensions of A, X and B are P X M, M x 1 and P x 1 respectively.
- Since number of training data pairs (P) is usually greater than number of linear parameters (M), this is an over determined problem.

4.3 ANFIS as Universal Approximator

- When the number of rules is not restricted, a zero-order Sugeno model has unlimited approximation power for matching well any nonlinear function arbitrarily on a compact set. This can be proved using the Stone-Weierstrass theorem.
- Let domain D be a compact space of N dimensions, and let F be a set of continuous real-valued functions on D satisfying the following criteria :

1) **Identity function :** The constant $f(x) = 1$ is in F.
2) **Separability :** For any two points $x_1 \neq x_2$ in D, there is a f in F such that $f(x_1) \neq f(x_2)$.
3) **Algebraic closure :** If f and g are any two functions in F, then fg and af + bg are in F for any two real numbers a and b.

Stone-Weierstrass theorem - II :

- If F is dense on C(D) for set of continuous real-valued function on D. For any $\in > 0$ and any function g in C(D), there is a function f in F such that $|g(x) - f(x)| < \in$ for all $x \in D$.

- The ANFIS satisfies all these requirements. In applications of fuzzy inference systems, the domain is almost always compact.
- It is possible, applying this theorem to prove the universal approximation power of the zero-order Sugeno model.

1. **Indentity function :** The constant $f(x) = 1$ is in F.
   - The first hypothesis requires that our fuzzy inference system be able to compute the identity function $f(x) = 1$.
   - An obvious solution is to set the consequence part of each rule equal to one. A fuzzy inference system with only one rule is able to compute the identity function.
2. **Separability :** For any two points $x_1 \neq x_2$ in D, there is a f in F such that $f(x_1) \neq f(x_2)$.
   - The second hypothesis requires that our fuzzy inference system be able to compute functions that have different values for different points. This is achievable by any fuzzy inference system with appropriate parameters.

Algebraic Closure - Addition I

- **Algebraic closure addition :** If f and g are any two functions in F, then af + bg are in F for any two real numbers a and b.
- Suppose that we have two fuzzy inference system S and $\hat{S}$; each of them has two rules.
- The final output of each system is specified as

$$S : z = \frac{w_1 f_1 + w_2 f_2}{w_1 + w_2}$$

$$\hat{S} : \hat{z} = \frac{\hat{w}_1 f_1 + \hat{w}_2 f_2}{\hat{w}_1 + \hat{w}_2}$$

Sum of z and $\hat{z}$ is :

$$az + b\hat{z} = a\frac{w_1 f_1 + w_2 f_2}{w_1 + w_2} + b\frac{\hat{w}_1 f_1 + \hat{w}_2 f_2}{\hat{w}_1 + \hat{w}_2}$$

$$= \frac{w_1\hat{w}_1 (af_1 + bf_1) + w_1\hat{w}_2 (af_1 + bf_2) + w_2\hat{w}_1 (af_2 + bf_1) + w_2\hat{w}_2 (af_2 + bf_2)}{w_1\hat{w}_1 w_1\hat{w}_2 + w_2\hat{w}_1 + w_2\hat{w}_2}$$

- Therefore, it is possible to construct a four-rule inference system that computes $az + b\hat{z}$

Algebraic Closure - Multiplication I :

- If f and g are any two functions in F, then f_g are in F.

Product of z and $\hat{z}$ is :

$$z\hat{z} = \frac{w_1\hat{w}_1 f_1\hat{f}_1 + w_1\hat{w}_2 f_1\hat{f}_2 + w_2\hat{w}_1 f_2\hat{f}_1 + w_2\hat{w}_2 f_2\hat{f}_2}{w_1\hat{w}_1 + w_1\hat{w}_2 + w_2\hat{w}_1 + w_2\hat{w}_2}$$

- Therefore, it is possible to construct a four-rule inference system that computes $z\hat{z}$.

4.4 Coactive Neuro Fuzzy Modeling

- CANFIS stands for Coactive Neuro-Fuzzy Inference System, has extended basic ideas of its prodecessor ANFIS. In this ANFIS concept has been extended to any number of input-/output pairs. In addition, CANFIS yields advantages from non linear fuzzy rules. This CANFIS realizes the Sugeno type fuzzy inferencing accomplishing fuzzy if then rules.
- CANFIS enables to obtain more than one outputs and has the advantage of nonlinear rule formations. The CANFIS model integrates fuzzy inputs with a modular neural network to quickly solve poorly defined problems.
- Fig. 4.4.1 shows architecture of CANFIS.

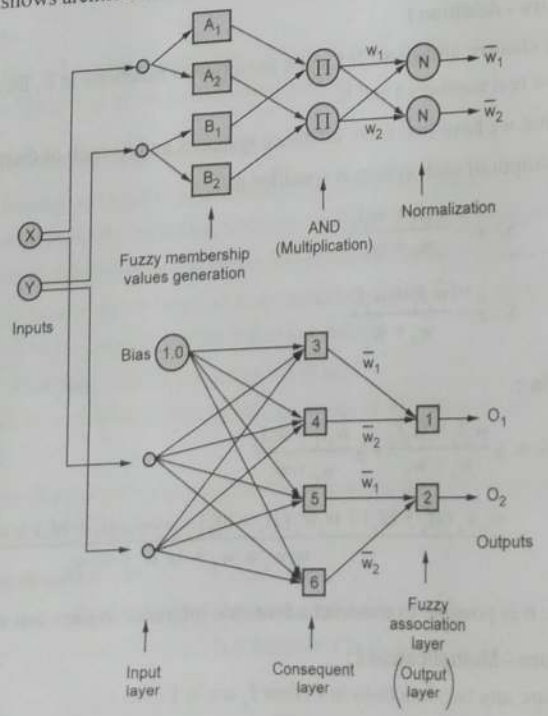


Fig. 4.4.1 Architecture of CANFIS

- In CANFIS the antecedents are the same, but the consequents are different according the number of outputs required. Fuzzy rules are constructed with shared membership values to express correlations between outputs.
- In MANFIS no modifiable parameters are shared by the juxtaposed ANFIS models. Each ANFIS has an independent set of fuzzy rules, which makes it difficult to realize possible correlations between outputs. Also the adjustable parameters increases with the increase in the number of outputs.

4.4.1 Framework

- CANFIS has extended the basic ideas of its predecessor ANFIS : The ANFIS concept has been extended to any number of input-output pairs. In addition, CANFIS yields advantages from nonlinear fuzzy rules.
- To get multiple outputs from ANFIS, multiple ANFIS models are used side by side as per requirement of output. In multiple ANFIS (MANFIS), each ANFIS has an independent set of fuzzy rules, which makes it difficult to realize possible certain correlations between outputs. Output increases as number of adjustable parameters increases.
- Another way of generating multiple outputs is to maintain the same antecedents of fuzzy rules among multiple ANFIS models.
- In both CANFIS and RBFN, locality is considered by Euclidean norms between each local center and the input vector. By comparison both the methods, the inner product of each weight vector and input vector is taken in a backpropagation MLP to measure similarity between training patterns.
- CANFIS powerful capability stems from pattern dependent weights between the consequent layer and the fuzzy association layer. Membership values correspond to those dynamically changeable weights that depend on input pattern.
- CANFIS is locally tuned. RBFN may need more data to achieve certain accuracy than the MLP, although the RBFN may learn faster than MLP. The backpropagation MLP can be a better extrapolator than RBFN due to its global nature.
- RBFN fails to estimate the values of function outside the range of the training data because of the local nature of its hidden receptive fields.
- Neuron functions for adaptive networks : Various functions are used as basis functions for alternative Gaussian functions in RBFNs. Since neuron functions in the RBFN hidden layer correspond to MFs in CANFIS.

4.5 Neuron Functions for Adaptive Networks

- An adaptive neural network is a system that processes information and makes adjustments to the network when necessary. Such networks can be found in computer systems or in organic life forms. They are used to interpret large amounts of complex information and are the basis of modern artificial intelligence technology.

4.5.1 Nonlinear Rule

- Suppose we have a sigmoidal function as a neuron function in the consequent layer. Then we have a nonlinear consequent given as

$$C_{non} = \frac{1}{1 + \exp[-(p_1 x + q_1 y + r_1)]}$$

- When each rule's consequent is realized by an NN, then it is called as neural rule. When four consequent NNs have sigmoidal output neuron functions with no hidden layers, the four neural rules are reduced to sigmoidal rules.

4.5.2 Truncation Filter Function and Modified Sigmoidal Functions

- Sometimes we require very small desired output from NN. The NN is required to learn extreme values close to the rim of the output range.
- When normal sigmoidal logistic functions are introduced at the output layer of an NN, it is known that the NN fails to learn such extreme values. To improve the performance of NN, replace the square error function with an entropic function.
- Alternative way is to introduce a modified sigmoidal function (f_{mod}) and a truncation filter function (f_{trc}) as neuron functions for the output layer.

$$f_{trc}(x) = \begin{cases} MIN & \text{if } x \le MIN \\ MAX & \text{if } x \ge MAX \\ X & \text{otherwise} \end{cases}$$

$$f_{mod} = \begin{cases} MIN & \text{if } x \le MIN \\ MAX & \text{if } x \ge MAX \\ X & \text{otherwise} \end{cases}$$

Where f(x) is the normal sigmoidal logistic function.

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

4.6 Neuro Fuzzy Spectrum

- Neuro-fuzzy system is a fuzzy system that uses a learning algorithm derived from or inspired by neural network theory to determine its parameters (fuzzy sets and fuzzy rules) by processing data samples.
- Neuro-fuzzy models allow prior knowledge to be embedded via fuzzy rules with appropriate linguistic labels. Fig. 4.6.1 shows neuro fuzzy spectrum.

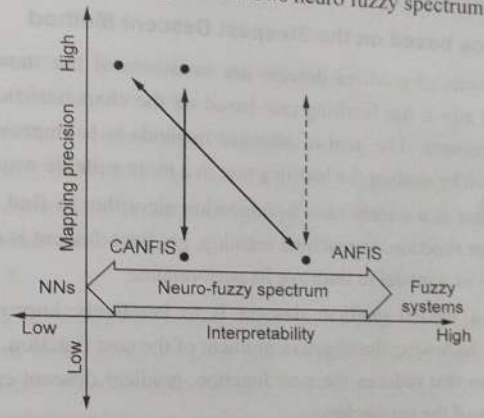


Fig. 4.6.1 Neuro fuzzy spectrum

- The learning process is often follows the diagonal route of improving mapping precision and deteriorating interpretability at the same time, this situation is called as dilemma between **interpretability and precision.**
- Adaptive neuro-fuzzy models like ANFIS / CANFIS transit smoothly between the two ends of neuro-fuzzy spectrum.
- The hybrid learning rule generally interpretable, but this is not always guaranteed.
- Various approaches to alleviating the dilemma :
  1. Apply another way of interpretation to ANFIS / CANFIS, many other interpretable NN models have structures similar to RBFN or modular networks.
  2. Modify the learning algorithms that maintain MF interpretability.
  3. Alter the fuzzy rules structures by setting up nonlinear rules.
  4. Put proper constraints on neighbouring MFs so that resultant MF interpretability can be retained. The simplest way is to apply some knowledge to fixing the center positions of MFs.

5. Formulate a new error measure designed to increase interpretability, such as an error measure with a term similar to Shannon's information entropy.
6. Transform the input space to another space, in which input values can be treated in a linguistically meaningful way.

4.7 Analysis of Adaptive Learning Capability

4.7.1 Convergence based on the Steepest Descent Method

- Adaptive methods of gradient descent are variations of the standard gradient descent algorithm that adjust the learning rate based on the characteristics of the data and the optimization process. The goal of adaptive methods is to improve the convergence of the optimization by scaling the learning rate in a more suitable way for the task at hand.
- Gradient descent is a widely used optimization algorithm to find parameter values that minimize a cost function. In machine learning, gradient descent is often used to optimize the parameters of a model to improve its performance.
- The main idea behind gradient descent is to iteratively improve the values of the parameters by following the negative gradient of the cost function. By taking small steps in the direction that reduces the cost function, gradient descent can efficiently find the optimal values of the parameters.

4.8 Two Marks Questions with Answers

Q.1 What is neuro - fuzzy system ?

Ans. : A neuro-fuzzy system is a neural network which is functionally equivalent to a fuzzy inference model. It can be trained to develop IF-THEN fuzzy rules and determine membership functions for input and output variables of the system.

Q.2 What is fuzzy neuron ?

Ans. : Fuzzy neuron is a basic element of fuzzy BP network. Here input vector and weight vector are represented by triangular LR type fuzzy number. A network of fuzzy neurons is different from a traditional network because the function of each neuron is identified and its semantics is defined. The function of such networks is the modeling of inference rules for classification.

Q.3 State fuzzy inference.

Ans. : Fuzzy inference can be defined as a process of mapping from a given input to an output, using the theory of fuzzy sets.

Q.4 Why to use fuzzy logic in neural network ?

Ans. : Some reasons to use fuzzy logic in neural networks :

1. Fuzzy logic is largely used to define the weights, from fuzzy sets, in neural networks.
2. When crisp values are not possible to apply, then fuzzy values are used.
3. We have already studied that training and learning help neural networks perform better in unexpected situations. At that time fuzzy values would be more applicable than crisp values.
4. When we use fuzzy logic in neural networks then the values must not be crisp and the processing can be done in parallel.

Q.5 Explain dilemma between interpretability and precision.

Ans. : The learning process is often follows the diagonal route of improving mapping precision and deteriorating interpretability at the same time, this situation is called as dilemma between interpretability and precision.

Q.6 What is ANFIS ?

Ans. : ANFIS stands for Adaptive-Network-based Fuzzy Inference System. The ANFIS is a data driven procedure representing a neural network approach for the solution of function approximation problems. Data driven procedures for the synthesis of ANFIS networks are typically based on clustering a training set of numerical samples of the unknown function to be approximated.

❑❑❑

Notes

UNIT V

5 Applications

Syllabus

Modeling a two input sine function - Printed Character Recognition - Fuzzy filtered neural networks - Plasma Spectrum Analysis - Hand written neural recognition - Soft Computing for Color Recipe Prediction.

Contents

5.1 Modeling a Two Input Sine Function

- Sine function is a trigonometric function that is equal to the ratio of perpendicular and hypotenuse of a right triangle.
- The trigonometric ratios can also be considered as functions of a variable which is the measure of an angle. This angle measure can either be given in degrees or radians. Here, we will use radians. The graph of a sine function $y = \sin(x)$ is looks like this :

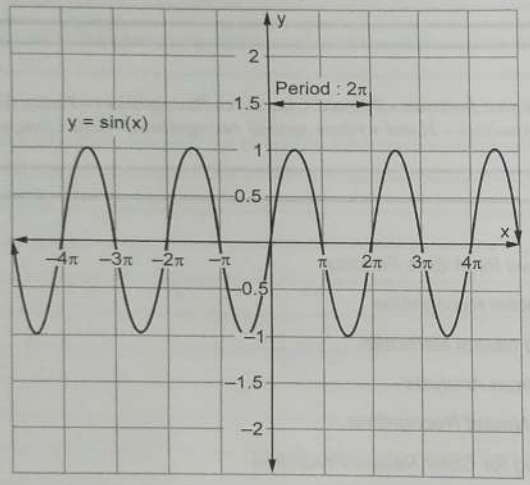


Fig. 5.1.1

- A sine wave refers to the graphical representation of the general function. The sine function and sine waves are used to model periodic phenomena and processes that follow predictable cyclical patterns.
- Most financial/economic data can be modeled by varying the amplitude and periodicity of the general sine function. The amplitude dictates the magnitude of the swings, whereas the periodicity dictates how often the swings happen.

5.2 Printed Character Recognition

- Optical Character Recognition (OCR) is one of the most popular areas of research in pattern recognition because of its immense application potential. OCR uses two methods : template matching and feature classification.

- In the template matching approach, recognition is based on the correlation of a test character with a set of stored templates. In the feature classification method, features are extracted from a standard character image to generate a feature vector.
- A decision tree is formed based on the presence or absence of some of the element in the feature vector. When an unknown character pattern is encountered, this tree is traversed from node to node till a unique decision is reached. The template matching techniques are more sensitive to font and size variations of the characters than the feature classification methods.
- Multilayer Perception (MLP) is often used for the recognition of characters after they are trained with a set of standard patterns by supervised learning.
- Printed character recognition system is based on the concept of nearest neighbour classification or case based reasoning. Iterative training is not required for this design method.
- Let us consider the binary XOR problem. Here we need to classify a binary input vector to class 0 of the vector has an even number of 1s; otherwise it is assigned to class 1. Truth table of two input XOR is as follows :

	X	Y	Class
Desired i/o pair 1	0	0	0
Desired i/o pair 2	0	1	1
Desired i/o pair 3	1	0	1
Desired i/o pair 4	1	1	0

- But XOR problem is not linearly separable and cannot be solved by a single layer perceptron. To use an MLP with a hidden layer to solve it, we need to train the network. Training data is representative and noise free. Here we used as prototypes for the fuzzy logic design approach.
- Find the prototype nearest to the new data point and assign the point to that prototype class. To solve this, we need a similarity measure that quantifies the meaning of near. for this purpose, membership function is used.
- If we take "[x, y] is near [0, 1]}" to mean that "x is near 0 AND y is near 1" then all we need to do is assign an appropriate operator to AND. The most popular fuzzy AND operators are product and min.

- Creating fuzzy rule set for solving the XOR problem is as follows :

  Rule 1 : IF x is near 0 AND y is near 0 THEN output = 0

  Rule 2 : IF x is near 0 AND y is near 1 THEN output = 1

  Rule 3 : IF x is near 1 AND y is near 0 THEN output = 1

  Rule 4 : IF x is near 1 AND y is near 1 THEN output = 0

- If the input data [x, y] is close to one of the prototypes, it is then assigned to that prototype class.
- In printed character recognition, each of 26 letters is defined as a 7×5 pixel matrix. The challenge is to build a fuzzy interface system that can classify a given set of 35 pixels to one of the 26 alphaset characters. These 26 prototypes are noise free.
- Construct MFs for each of the 35 inputs. In the prototypes, each pixel is either 0 or 1, so we can set up MFs for "near 0" and "near 1".
- Set up rules : Each prototype represents a rule, so we have 26 rules, each of them an AND rule with 35 preconditions. Each rules output is not critical and set it for an arbitrary constant of MF.
- Use the fuzzy inference system : Note that the output is a categorical variable for which no numerical order is assumed.

5.3 Fuzzy Filtered Neural Networks

- Fuzzy filtering is task of partitioning a large number of physical data channels into much fewer fuzzy channels. These channels adaptable during a training process, are employed for both noise filtering and feature detection.
- Fuzzy channel defines a range of input signal intensity characterized by an appropriate membership function. The position and shape of membership function are adjusted during a learning process so that the system error is minimized.

Advantages of fuzzy filtering :

1. The feature detected are insensitive to variations in samples and effect of noise is negligible.
2. Duplicate features are automatically combined.
3. This model provides a meaningful interpretation of the detected features.
4. It reduces the complexity of the architecture.

- Fuzzy filtering is implemented by using multilayer-feedforward adaptive network. Fig. 5.3.1 shows fuzzy filtered neural network.

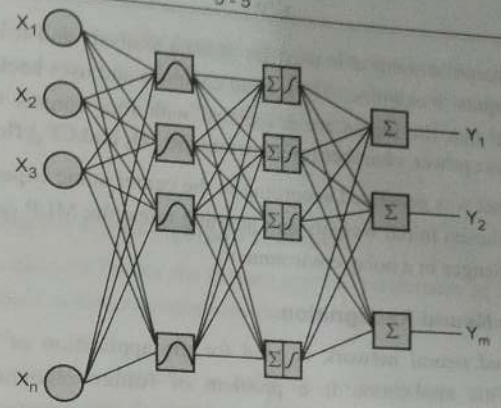


Fig. 5.3.1 Fuzzy filtered neural network

- Layer 1 is input layer. Each node in layer 2 is associated with a generalized bell membership function :

$$\mu_A(x_i) = \frac{1}{1 + \left| \frac{x_i - c_i}{a_i} \right|^{2b_i}}$$

where x_i = Position of a physical channel

A = Linguistic term associated with node function

$\{a_i, b_i, x_i\}$ = Parameter set

- The node output is a normalized weighted sum :

$$\frac{\sum_i \mu_A(x_i)\, f(x_i)}{\sum_i \mu_A(x_i)}$$

where $f(x_i)$ is the intensity of input channel x_i.

- So the fuzzy filter behaves as a bandpass filter with the added capability of learning.

5.4 Plasma Spectrum Analysis

- In VLSI manufacturing, it is necessary to determine the endpoints of an etching process and to detect contamination in a chemical chamber. The normal method of monitoring a chemical process should be to use the full range of optical signals generated by an optical emission spectrometer to determine the actual status of the chemical reaction.

- Multilayer perceptron method is used for plasma analysis and it is three layer network with linear outputs. It examines 731 optical channels and uses backpropagation to adjust the weights so that the inputs are associated with four control variables of an oxide etching process (power, chamber pressure and gas (H_2 and CF_3) flows.
- Training data set was generated according to the center cubic experimental design. With the carefully chosen initial weights and learning rates, the MLP can effectively identify small signal changes in a noisy environment.

5.5 Hand Written Neural Recognition

- A fuzzy filtered neural network is used for the application of handwritten numeral recognition. This application is a problem of feature recognition. Typical neural networks are not as efficient in feature recognition as the quantity of the features they can compute, decreases, as the number of parameters increase.
- Fig. 5.5.1 shows some example of hand written numerical.

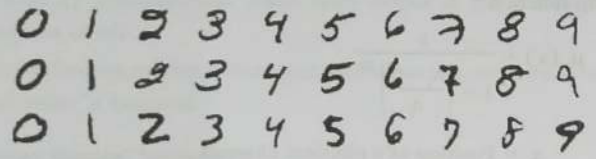


Fig. 5.5.1 Example of hand written numerical

- All the datasets can be grouped together in different sets (fuzzy sets) having similar characteristic. For example, if we deal with an application involving one of the features as color of an object, different fuzzy sets can be formed based on the different colors. Each of these fuzzy sets will have their own membership function depending on their color and other parameters.
- A training data set of images of handwritten numerals from different people was used in order to train the multilayer feedforward adaptive neural network. Some part of the training set is shown above. A three layer feedforward neural network is used and the weights of the layers in the networks are adjusted by backpropagation algorithm.
- After training for 250 epochs the recognition rate of the model was approximately 85 %.
- The images are pre-processed to fit a 32×32 matrix and left aligned. These images are then given as input to the fuzzy filters. A fuzzy filter is a robust input-output model which is invariably unrelated to fluctuations in the input signal frequency or redundant values of the provided input.

- Since a fuzzy filter is used for feature filtering, the system does not have considerable false positives making it a reliable system. All important features are filtered and used for training along with saving the system from the problem of overfitting. A one dimensional fuzzy filtered increased the recognition rate of the model to an accuracy of 90 %.

5.6 Soft Computing for Color Recipe Prediction

- Color recipe prediction relates the surface spectral reflectance of a target color to a list of several required colorant proportions that are needed to produce the same color as the reference color.
- To overcome practical obstacles in color recipe prediction, simple back-propagation Multilayer Perceptron (MLP) approach is used.
- Two types of simple MLPs, NN_{norm} and NN_{mod} have been applied as a touchstone to the recipe prediction to fathom the intrinsic difficulty of the task, NN_{norm} has normal sigmodal functions and NN_{mod} has modified sigmoidal functions in the output layer.
- Both NN_{norm} and NN_{mod} have the same model size, mapping surface spectral reflectance of a target color.
- Main concerns in color recipe prediction :
  1. **P1 :** It is difficult to predict colorant concentrations. Sometimes it needs to predict proportions with enough precision to specify levels such as 0.01%, which is the desired minimal proportion level.
  2. **P2 :** It is necessary to specify use of a limited number of colorants to use for acceptable cost performance requirements. At the same time, in the choice of colorants, we need to avoid the use of complementary colorants and of the same types of colorants.
  3. **P3 :** The magnitude of mean squared error of colorant vectors may not correspond exactly to that of color difference. The question is which colorant has the most significant impact on the entire color.
  4. **P4 :** It is important to consider human visual sensitivity to color difference, which is closely related to perceptual attributes of color.
  5. **P5 :** Some different combinations of colorants may have the same perceptual attributes of color as seen by humans.

Fig. 5.6.1 shows input output relation in a typical color recipe prediction system.

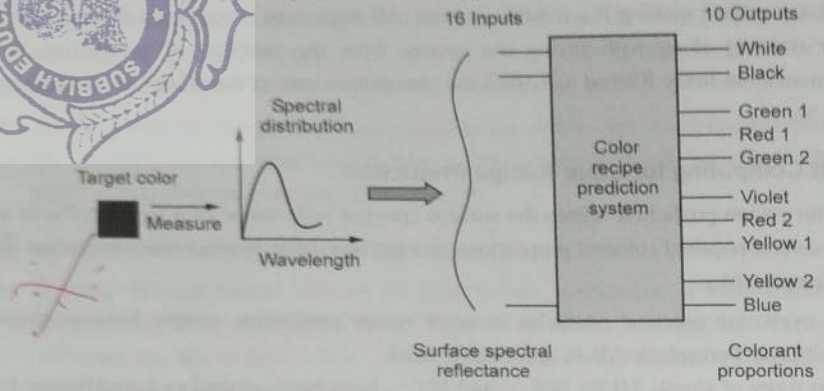


Fig. 5.6.1 Input output relation in a typical color recipe prediction system

- Following is the data set for desired number of colorants required to produce color in data sets.

Parameter	Two colorants desired	Three colorants desired	Four colorants desired	Desired average # of colorants to use
1446 training data	4	60	1382	3.95
302 test data	0	13	289	3.96

- Initially we need to specify which colorants to use. The above table shows the desired number of colorants in data sets. The average number of colorants required to produce any color is fewer than five; this means that 6 of the 10 final outputs should be zero.

5.7 Two Marks Questions with Answers

Q.1 What is task of fuzzy filtering ?

Ans. : Fuzzy filtering is task of partitioning a large number of physical data channels into much fewer fuzzy channels. These channels adaptable during a training process, are employed for both noise filtering and feature detection.

Q.2 What is fuzzy channel ?

Ans. : Fuzzy channel defines a range of input signal intensity characterized by an appropriate membership function. The position and shape of membership function are adjusted during a learning process so that the system error is minimized.

Q.3 Explain advantages of fuzzy filtering.

Ans. :

1. The feature detected are insensitive to variations in samples and effect of noise is negligible.
2. Duplicate features are automatically combined.
3. This model provides a meaningful interpretation of the detected features.
4. It reduces the complexity of the architecture.

Q.4 What is functions of input layer in neural network ?

Ans. : Input layer takes input from outside sources and then these values are multiplied by the interconnecting weights of each neuron.

Q.5 Explain any one approaches of Neuro-Fuzzy models.

Ans. : One approach is to use the neural nets to define the parameters to be used for the fuzzy sets. Once the parameters are learned by the neural network, they no longer exist.

Q.6 Describe Neuro-Fuzzy model.

Ans. : The Neuro-Fuzzy models are a combination of neural networks as well as the fuzzy logic concept. These approaches are combined in order to obtain a better mimic of the human brain.

❑❑❑