NEHRU COLLEGE OF ENGINEERING AND RESEARCH CENTRE
(NAAC Accredited)
(Approved by AICTE, Affiliated to APJ Abdul Kalam Technological University, Kerala)

DEPARTMENT OF MECHATRONICS ENGINEERING

COURSE MATERIALS

MR 402 SOFT COMPUTING TECHNIQUES

VISION OF THE INSTITUTION
To mould true citizens who are millennium leaders and catalysts of change through excellence in education.
MISSION OF THE INSTITUTION
NCERC is committed to transform itself into a center of excellence in Learning and Research in Engineering and Frontier Technology and to impart quality education to mould technically competent citizens with moral integrity, social commitment and ethical values. We intend to facilitate our students to assimilate the latest technological know-how and to imbibe discipline, culture and spiritually, and to mould them in to technological giants, dedicated research scientists and intellectual leaders of the country who can spread the beams of light and happiness among the poor and the underprivileged.

ABOUT DEPARTMENT

- Established in: 2013
- Course offered: B.Tech Mechatronics Engineering
- Approved by AICTE New Delhi and Accredited by NAAC
- Affiliated to the University of Dr. A P J Abdul Kalam Technological University.

DEPARTMENT VISION

To develop professionally ethical and socially responsible Mechatronics engineers to serve the humanity through quality professional education.

DEPARTMENT MISSION

1) The department is committed to impart the right blend of knowledge and quality education to create professionally ethical and socially responsible graduates.

2) The department is committed to impart the awareness to meet the current challenges in technology.

3) Establish state-of-the-art laboratories to promote practical knowledge of mechatronics to meet the needs of the society

PROGRAMME EDUCATIONAL OBJECTIVES

I. Graduates shall have the ability to work in multidisciplinary environment with good professional and commitment.

II. Graduates shall have the ability to solve the complex engineering problems by applying electrical, mechanical, electronics and computer knowledge and engage in lifelong learning in their profession.

III. Graduates shall have the ability to lead and contribute in a team with entrepreneur skills, professional, social and ethical responsibilities.

IV. Graduates shall have ability to acquire scientific and engineering fundamentals necessary for higher studies and research.

PROGRAM OUTCOME (PO'S)

Engineering Graduates will be able to:

PO 1. Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO 2. Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO 3. Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO 4. Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO 5. Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

PO 6. The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO 7. Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO 8. Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO 9. Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO 10. Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO 11. Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO 12. Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAM SPECIFIC OUTCOME(PSO'S)

PSO 1: Design and develop Mechatronics systems to solve the complex engineering problem by integrating electronics, mechanical and control systems.

PSO 2: Apply the engineering knowledge to conduct investigations of complex engineering problem related to instrumentation, control, automation, robotics and provide solutions.

COURSE OUTCOME

After the completion of the course the student will be able to

CO 1	Understand the concepts of Fuzzy sets and fuzzy logic.
CO 2	Acquire knowledge to introduce types of Fuzzy Inference System and difference among them, review of gradient-based optimization techniques steepest descent method and Newton's method.
CO 3	Interpret about derivative-free optimization and supervised learning neural networks.
CO 4	Describe about Unsupervised Learning Neural Networks.

CO 5	Explain about Adaptive Neuro-Fuzzy Inference system and Coactive Neuro Fuzzy modeling.
CO 6	Acquire knowledge in Printed Character Recognition, Inverse Kinemaics, Automobile Fuel Efficiency Prediction and Color Recipe Prediction.

CO VS PO'S AND PSO'S MAPPING

CO	PO1	PO 2	PO3	PO 4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PS0 1	PSO 2
CO 1	3	-	3	-	-	-	-	-	-	-	-	3	2	2
CO 2	3	3	3	-	-	-	-	-	-	-	-	3	2	2
CO 3	3	3	3	3	-	-	-	-	-	-	-	3	2	2
CO 4	3	-	3	3	-	-	-	-	-	-	-	3	2	2
CO 5	3	-	2	-	-	-	-	-	-	-	-	3	2	2
CO 6	3	-	2	-	3	-	-	-	-	-	-	3	2	2

Note: H-Highly correlated=3, M-Medium correlated=2, L-Less correlated=1

SYLLABUS

Course code	Course Name	L-T-P - Credits	Year of Introduction
MR402	**Soft Computing Techniques**	**3-0-0:3**	**2016**
Prerequisite : NIL			

Course Objectives

- To introduce the concepts of fuzzy sets and fuzzy logic
- To make students familiar with neural networks that can learn from available examples

Syllabus

Introduction to Neuro – Fuzzy and Soft Computing – Fuzzy Rules and Fuzzy Reasoning – Extension Principle and Fuzzy Relations - Fuzzy Inference Systems – Fuzzy Models -Derivative-based Optimization – Genetic Algorithms – Radial Basis Function Networks – Adaptive Neuro-Fuzzy Inference Systems – Coactive Neuro Fuzzy Modeling – Framework Neuron Functions for Adaptive Networks – Neuro Fuzzy Spectrum- Printed Character Recognition – Inverse Kinematics Problems – Automobile Fuel Efficiency Prediction – Soft Computing for Color Recipe Prediction.

Expected outcome .

- The students will be familiar with the techniques of soft computing and adaptive neuro fuzzy inferencing systems and will be able to use the techniques to simulate and optimize engineering systems.

Text Book:

1. J.S.R.Jang, C.T.Sun and E.Mizutani, "Neuro-Fuzzy and Soft Computing", PHI, 2004, Pearson Education 2004.
2. S.N.Sivanandam & S.N.Deepa "Principles of Soft Computing" Wiley India Pvt. Ltd., 2007

References:

1. Timothy J.Ross, "Fuzzy Logic with Engineering Applications", McGraw-Hill, 1997.
2. Davis E.Goldberg, "Genetic Algorithms: Search, Optimization and Machine Learning", Addison Wesley, N.Y., 1989.
3. S. Rajasekaran and G.A.V.Pai, "Neural Networks, Fuzzy Logic and Genetic Algorithms", PHI, 2003.
4. R.Eberhart, P.Simpson and R.Dobbins, "Computational Intelligence - PC Tools", AP Professional, Boston, 1996.

Course Plan

Module	Contents	Hours	Sem. Exam Marks
I	Introduction to Neuro – Fuzzy and Soft Computing – Fuzzy Sets – Basic Definition and Terminology – Set-theoretic Operations – Member Function Formulation and Parameterization – Fuzzy Rules and Fuzzy Reasoning – Extension Principle and Fuzzy Relations .	7	15%
II	Fuzzy Inference Systems – Mamdani Fuzzy Models – Sugeno Fuzzy Models – Tsukamoto Fuzzy Models. Derivative-based Optimization – Descent Methods – The Method of Steepest Descent – Classical Newton's Method	7	15%
FIRST INTERNAL EXAMINATION			
III	Step Size Determination – Derivative-free Optimization – Genetic Algorithms – Simulated Annealing – Random Search – Downhill Simplex Search. Supervised Learning Neural Networks – Perceptrons - Adaline – Back propagation Mutilayer Perceptrons	7	15%
IV	Radial Basis Function Networks – Unsupervised Learning Neural Networks – Competitive Learning Networks – Kohonen Self-Organizing Networks – Learning Vector Quantization – Hebbian learning.	7	15%
SECOND INTERNAL EXAMINATION			

V	Adaptive Neuro-Fuzzy Inference Systems – Architecture – Hybrid Learning Algorithm – Learning Methods that Cross-fertilize ANFIS and RBFN – Coactive Neuro Fuzzy Modeling – Framework Neuron Functions for Adaptive Networks – Neuro Fuzzy Spectrum.	7	20%
VI	Printed Character Recognition – Inverse Kinematics Problems – Automobile Fuel Efficiency Prediction – Soft Computing for Color Recipe Prediction.	7	20%
END SEMESTER EXAM			

QUESTION PAPER PATTERN:

QUESTION PAPER PATTERN

Maximum Marks : 100 Exam Duration:3 hours

PART A: FIVE MARK QUESTIONS

8 compulsory questions – 1 question each from first four modules and 2 questions each from last two modules (8 x 5= 40 marks)

PART B: 10 MARK QUESTIONS

6 questions uniformly covering the first four modules. Each question can have maximum of three sub questions, if needed. Student has to answer any 3 questions (3 x10 = 30 marks)

PART C: 15 MARK QUESTIONS

3 questions uniformly covering the last two modules. Each question can have maximum of four sub questions, if needed. Student has to answer any two questions (2 x15 = 30 marks)

QUESTION BANK

MODULE I				
Q:NO:	**QUESTIONS**	**CO**	**KL**	**PAGE NO:**
1	Write about the soft computing constituents and conventional artificial intelligence.	CO1	K3	14
2	Explain the cooperation between a neural character recognizer and a knowledge base.	CO1	K2	15

3	Define support, core, normality, crossover points and fuzzy singleton.	CO1	K1	24
4	Draw an expert system.	CO1	K1	16
5	Illustrate an intelligent system.	CO1	K3	17
6	Define Fuzzy numbers, bandwidth, symmetry, open left and open right.	CO1	K1	27
7	Explain about the characteristics of soft computing.	CO1	K2	18
8	Define fuzzy sets and membership functions.	CO1	K1	20
9	Define the following fuzzy set theoretic operations. Containtment or subset, Union(disjunction), Intersection and Complement.	CO1	K1	28
MODULE II				
1	Draw the block diagram for a fuzzy inference system.	CO2	K1	53
2	Write about the T-norm and T-conorm operators.	CO2	K3	57
3	Explain the Mamdani fuzzy inference system using min and max for T-norm and T-conorm operators respectively with diagram.	CO2	K3	55
4	Write about the Sugeno fuzzy model.	CO2	K3	62
5	Write about the Tsukamoto fuzzy model.	CO2	K3	65
6	Explain about various defuzzification schemes for obtaining a crisp output.	CO2	K2	59

7	Write about optimization.	CO2	K3	66
8	Write about descent method of optimization.	CO2	K3	66
9	Explain about Newton's method of optimization.	CO2	K2	71
MODULE III				
1	Write about initial bracketing.	CO3	K3	73
2	Write about the common characteristics of derivative free optimization methods.	CO3	K3	79
3	Write an algorithm for initial bracketing procedure for searching three points $\theta_{1,}$ $\theta_{2,}$ and $\theta_{3.}$	CO3	K3	74
4	Write a simple genetic algorithm for maximization problems.	CO3	K3	84
5	Draw the flowchart for the random search method.	CO3	K1	92
6	Write down the modified random search method.	CO3	K3	91
7	Write down the step for the random search.	CO3	K3	90
8	Write down the simulated Annealing algorithm.	CO3	K3	87
9	Explain about Adaline.	CO3	K2	103
MODULE IV				
1	Write down the functional equivalence of RBFN and the conditions under which an RBFN and a FIS are functionally equivalent.	CO4	K3	123

2	Explain in detail about Hebbian learning.	CO4	K2	139
3	Explain Kohonen self-organizing network and its training with figure.	CO4	K2	131
4	Explain about four Radial Basis function networks with figure.	CO4	K2	118
5	Explain the network representation of learning vector quantization.	CO4	K2	135
6	Illustrate the working competitive learning network.	CO4	K3	127
7	Explain in detail about Learning Vector Quantization.	CO4	K2	135
8	Explain in detail about competitive learning networks.	CO4	K2	127
9	Write about Interpolation and Approximation RBFNs.	CO4	K3	125
10	Determine the weights after one iteration for Hebbian learning of a single neuron network starting with initial weights w=[1,-1] input as x1=[1,2] , x2=[2,3] , x3=[1,-1] and c=1. (Use bipolar activation function).	CO4	K5	139
11	Explain about unsupervised learning Neural Networks with example.	CO4	K2	127
12	Explain about Radial Basis Function Networks.	CO4	K2	118
MODULE V				
1	Draw equivalent ANFIS architecture for a two-input two-rule Tsukamoto fuzzy model.	CO5	K1	149

2	Illustrate the Sugeno model reasoning mechanism for the following common rule set with two fuzzy if-then rules. Rule1: If x is A_1 and y is B_1, then $f_1=p_1x+q_1y+r_1$, Rule2: If x is A_2 and y is B_2, then $f_2=p_2x+q_2y+r_2$.	CO5	K3	145
3	Draw equivalent ANFIS architecture for a two-input first-order Sugeno fuzzy model with two rules and explain each layer.	CO5	K1	145
4	Draw and explain equivalent ANFIS/CANFIS architecture for a two-input, one output Sugeno fuzzy model	CO5	K1	154
5	Explain nonlinear rule for neuron functions in Adaptive networks.	CO5	K2	163
6	Draw and explain equivalent ANFIS/CANFIS architecture for a two-input, one output Sugeno fuzzy model.	CO5	K1	158
7	Explain about Coactive Neuro Fuzzy Modeling.	CO5	K2	153
8	Illustrate the Neuro-fuzzy spectrum.	CO5	K3	171
9	Write about the learning methods that cross fertilize ANFIS and RBFN methods?	CO5	K3	151
10	Write about hybrid learning algorithm.	CO5	K3	149
		MODULE VI		

1	Write down the main concerns in color recipe prediction.	CO5	K2	183
2	Explain the architecture of CANFIS with five color rules for color recipe prediction.	CO5	K2	188
3	Explain about printed character recognition using ANFIS.	CO5	K2	176
5	Illustrate GA serach control by the modified simplex crossover.	CO5	K3	201
6	Explain in detail about Automobile Fuel Efficiency Prediction.	CO5	K2	180
7	Draw the architecture of color paint manufacturing intelligence.	CO5	K1	192
9	Illustrate the input-output relation in a typical color recipe prediction system.	CO5	K3	183
10	Explain about Printed Character Recognition with example.	CO5	K2	176
11	Explain the genetic strategies used in color paint manufacturing intelligence.	CO5	K2	201

APPENDIX 1

CONTENT BEYOND THE SYLLABUS

Module I

MR402

1.1. Introduction to Neuro-Fuzzy and soft computing.

1. Introduction
2. Soft computing constituents and conventional artificial intelligence
   1. From conventional AI to computational Intelligence
   2. neural Networks
   3. Fuzzy Set Theory
   4. Evolutionary Computation.
3. Neuro-Fuzzy and Soft computing characteristics.

1. Introduction

Soft computing (SC)

It is an innovative approach to construct computationally intelligent systems.

Neuro-fuzzy computing.

It is used to design intelligent systems.

Intelligent Systems

It is a 'humanlike' expertise within a specific domain, adapt themselves and learn to do better in changing environment.

Nero-fuzzy and soft computing

The integration of neural networks and fuzzy inference systems together with derivative-free

1

optimization techniques is called neuro-fuzzy and soft computing.

Neural networks

It recognize patterns and adapt themselves to cope with changing environments.

Fuzzy inference systems

It incorporates human knowledge and perform inferencing and decision making.

2. Soft computing constituents and conventional Artificial intelligence.

Definition:

Soft computing is an emerging approach to computing which parallels the remarkable ability of the human mind to reason and learn in an environment of uncertainty and imprecision.

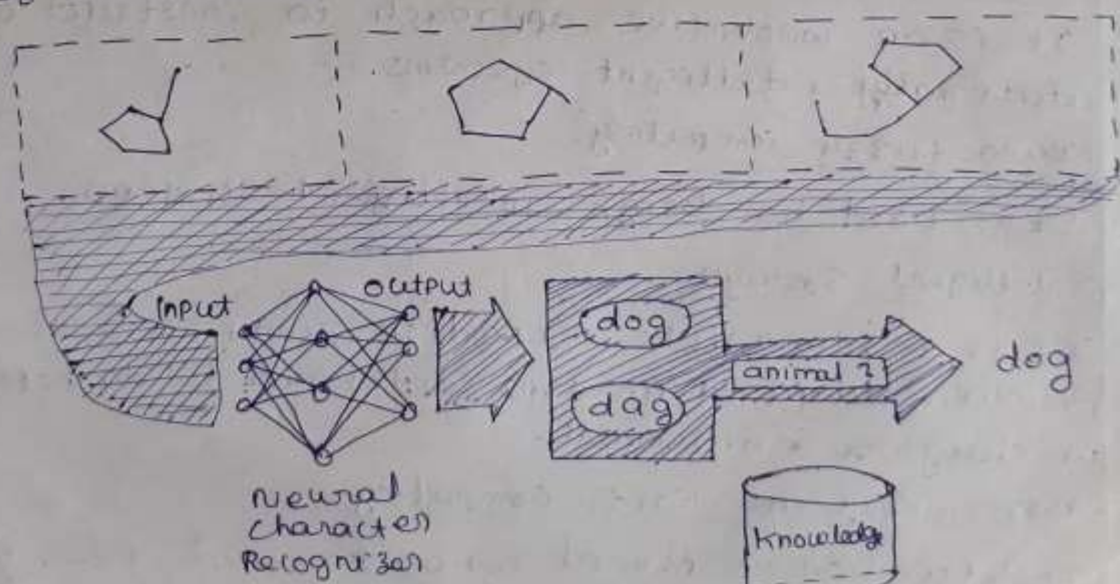


Figure 1.1 A neural character recognizer and a

(2)

knowledge base cooperate in responding to three hand-written characters that form a word "dog". Figure 1.1 illustrates a situation in which a neural character recognizer and a knowledge base are used together to determine the meaning of a hand-written word.

The neural character recognizer generates two possible answers "dog" and "dag", since the middle character could be either an "o" or "a".

If the knowledge base provides an extra piece of information that the given word is related to animals then the answer "dog" is picked up correctly.

Table 1.1 Soft computing constituents (the first three items) and conventional artificial intelligence.

Methodology	Strength
1. Neural network	Learning and adaptation
2. Fuzzy set theory	Knowledge representation via fuzzy if-then rules
3. Genetic algorithm and simulated annealing	Systematic random search
4. Conventional AI	Symbolic manipulation

1. From Conventional AI to Computational Intelligence

a) Conventional AI focuses on symbolic rules.

b) The most successful conventional AI product is

(3)

the knowledge-based system or expert system (ES) as in Figure 1.2.

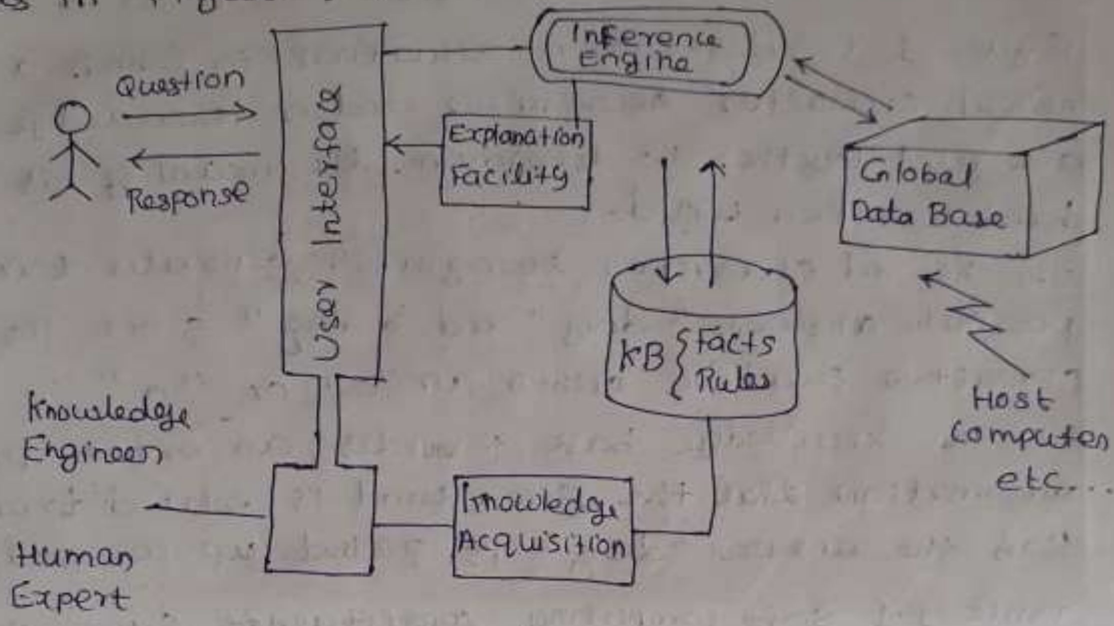


Figure 1.2 An expert system: one of the most successful (conventional) AI products.

Figure 1.3 is a schematic representation of an intelligent system that can sense its environment (perceive) and act on its perception (react).

2. Neural networks

a) The human brain is a source of natural intelligence and a truly remarkable parallel computer.

b) The brain processes incomplete information obtained by perception at a rapid rate.

c) Nerve cells function about 10^6 times slower than electronic circuit gates, but human brains process visual and auditory information much faster than modern computers.

(4)

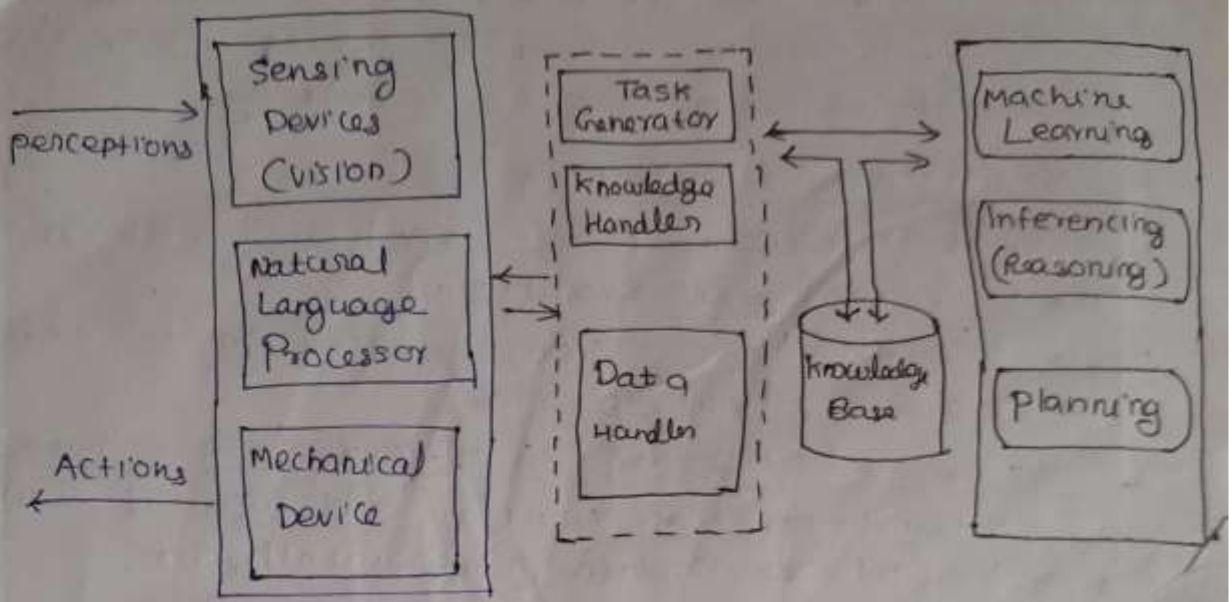


Figure 1.3 An intelligent system.

3. Fuzzy Set Theory

a) The human brain interprets imprecise and incomplete sensory information provided by perceptive organs.

b) Fuzzy set theory provides a systematic calculus to deal with imprecise and incomplete information linguistically, and it performs numerical computation by using linguistic labels specified by membership functions.

c) A selection of fuzzy if-then rules forms the key component of a fuzzy inference system (FIS) that can effectively model human expertise in a specific application.

d) Fuzzy inference system lacks the adaptability to deal with changing external environments.

(5)

e) Incorporation of neural network learning concepts in fuzzy inference system is neuro-fuzzy modeling, a central technique in soft computing.

4. Evolutionary Computation

a) natural intelligence is the product of millions of years of biological evolution.

i. Genetic algorithms (GAs) – based on the evolutionary principle of natural selection.

ii. Immune modeling and Artificial Life – based on the assumption that chemical and physical laws may be able to explain living intelligence.

Artificial life :- Attempts to realise life like behavior by imitating the processes that occur in the development or mechanics of life.

iii. Heuristically informed search techniques are employed in many AI applications.

iv. Simulated annealing and random search are other candidates that explore the search space in a stochastic (random) manner.

3. NEURO - FUZZY AND SOFT COMPUTING CHARACTERISTICS

With neuro-fuzzy modeling as a backbone, the characteristics of soft computing can be summarised as follows :

a) Human expertise soft computing utilises human expertise in the form of fuzzy if-then rules.

b) Biologically inspired computing models.

Inspired by biological neural networks, artificial

6

neural networks are employed extensively in soft computing to deal with perception, pattern recognition, and nonlinear regression and classification problems.

c) New optimization techniques

Soft computing applies innovative optimization methods arising from various sources; they are genetic algorithms, simulated annealing, the random search method, and the downhill Simplex method.

d) Numerical Computation

Unlike symbolic AI, soft computing relies mainly on numerical computation.

e) New application domains

Application domains are mostly computation intensive and include adaptive signal processing, adaptive control, nonlinear system identification, nonlinear regression, and pattern recognition.

f) Model-free learning

Neural networks and adaptive fuzzy inference systems have the ability to construct models using only target system sample data.

g) Intensive computation

Without assuming too much background knowledge of the problem being solved, neuro-fuzzy and soft computing rely heavily on high-speed number-crunching computation to find rules or regularity in data sets.

h) Fault tolerance

The deletion of a neuron in a neural network, or a rule in a fuzzy inference system, does not necessarily destroy the system. Instead, the system continues performing because of its parallel and

7

redundant architecture.

i) Goal driven characteristics

Neuro-fuzzy and soft computing are goal driven. the path leading from the current state to the solution does not really matter as long as we are moving toward the goal in the long run.

j) Real_world applications

soft computing is an integrated approach that can usually utilize specific techniques within subtasks to construct generally satisfactory solutions to real-world problems.

1.2 FUZZY SETS

A classical set is a set with a crisp boundary.

eg: A classical set A of real numbers greater than 6

$$A = \{x \mid x > 6\}.$$

A classical set do not reflect the nature of human concepts and thoughts, which tend to be abstract and imprecise.

A fuzzy set is a set without a crisp boundary. The gradual transition from "belong to a set" to "not belong to a set" is characterised by membership functions.

1.3 BASIC DEFINITION AND TERMINOLOGY

Definition: Fuzzy sets and membership functions.

If X is a collection of objects denoted generically by x, then a fuzzy set A in X is defined as a set of ordered pairs:

$$A = \{(x, \mu_A(x)) \mid x \in X\}$$

where $\mu_A(x)$ is called the membership function for the fuzzy set A.

(8)

The MF maps each element of X to a membership grade (or membership value) between 0 and 1. X is referred to as the universe of discourse, or simply the universe

Example

Fuzzy sets with a discrete nonordered universe

Let X = {San Francisco, Boston, Los Angeles} be the set of cities one may choose to live in. The fuzzy set C = "desirable city to live in" may be described as follows:

C = {(San Francisco, 0.9), (Boston, 0.8), (Los Angeles, 0.6)}.

Apparently the universe of discourse X is discrete and it contains nonordered objects - in this case, three big cities in the United States.

Membership grades are subjective.

Example

Fuzzy sets with a discrete ordered universe

Let X = {0, 1, 2, 3, 4, 5, 6} be the set of numbers of children a family may choose to have. Then the fuzzy set A = "sensible number of children in a family" may be described as follows:

$A = \{(0, 0.1), (1, 0.3), (2, 0.7), (3, 1), (4, 0.7), (5, 0.3), (6, 0.1)\}$.

Here we have a discrete ordered universe X; the MF for the fuzzy set A is shown in Figure 1.4. The membership grades of this fuzzy set are subjective measures.

(9)

(a) MF on a Discrete Universe

95

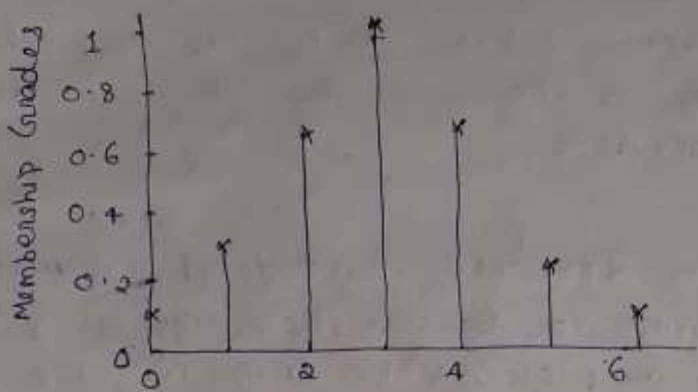


Figure 1.4 A = "sensible number of children in a family".

Example

Fuzzy sets with a continuous universe

Let $X = R^+$ be the set of possible ages for human beings. Then the fuzzy set B = "about 50 years old" may be expressed as

$$B = \{(x, \mu_B(x)) \mid x \in X\}.$$

where $$\mu_B(x) = \frac{1}{1 + \left(\frac{x-50}{10}\right)^4}.$$

(b) MF on a continuous universe

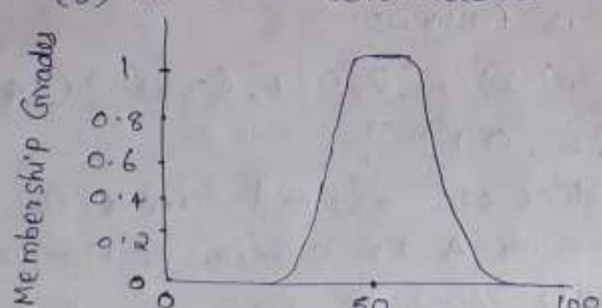


Figure 1.5 B = "about 50 years old".

(10)

The construction of a fuzzy set depends on two things :

i. The identification of a suitable universe of discourse (discussion) and

ii. The specification of an appropriate membership function.

The specification of membership functions is subjective, which means that the membership functions specified for the same concept by different persons may vary considerably.

Denoting a fuzzy set :

A fuzzy set A can be denoted as follows :

$$A = \begin{cases} \sum_{x_i \in X} \mu_A(x_i)/x_i, & \text{if } X \text{ is a collection of discrete objects} \\ \int_X \mu_A(x)/x, & \text{if } X \text{ is a continuous space (usually the real line } R\text{).} \end{cases}$$

The summation and integration signs stand for the union of $(x, \mu_A(x))$ pairs; they do not indicate summation or integration. Similarly, "/" is only a marker and does not imply division.

Example

Linguistic variables and linguistic values.

Suppose that X = "age". Then we can define fuzzy sets "young", "middle aged", and "old" that are characterised by MFs $\mu_{old}(x)$, $\mu_{middle\ aged}(x)$, and $\mu_{old}(x)$, respectively.

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

(11)

value of "young", then we have the expression "age is young", and so forth for the other values. Typical MFs for these linguistic values are displayed in Figure 1.6, where the universe of discourse X is totally covered by the MFs and the transition from one MF to another is smooth and gradual.

A fuzzy set is uniquely specified by its membership function.

Definition : Support

The support of a fuzzy set A is the set of all points x in X such that $\mu_A(x) > 0$:

$$\text{support}(A) = \{x \mid \mu_A(x) > 0\}.$$

Definition : Core

The core of a fuzzy set A is the set of all points x in X such that $\mu_A(x) = 1$:

$$\text{core}(A) = \{x \mid \mu_A(x) = 1\}.$$

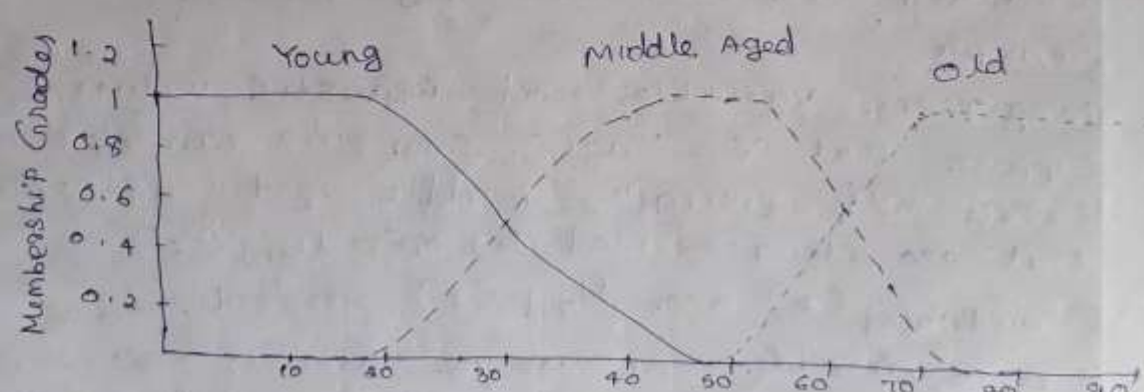


Figure 1.6 Typical MFs of linguistic values "young," "middle aged," and "old."

(12)

Definition : Normality

A fuzzy set A is normal if its core is nonempty. In other words, we can always find a point $x \in X$ such that $\mu_A(x) = 1$.

Definition : Crossover points

A crossover point of a fuzzy set A is a point $x \in X$ at which $\mu_A(x) = 0.5$:

$$\text{crossover}(A) = \{x \mid \mu_A(x) = 0.5\}.$$

Definition: Fuzzy singleton

A fuzzy set whose support is a single point in X with $\mu_A(x) = 1$ is called a fuzzy singleton.

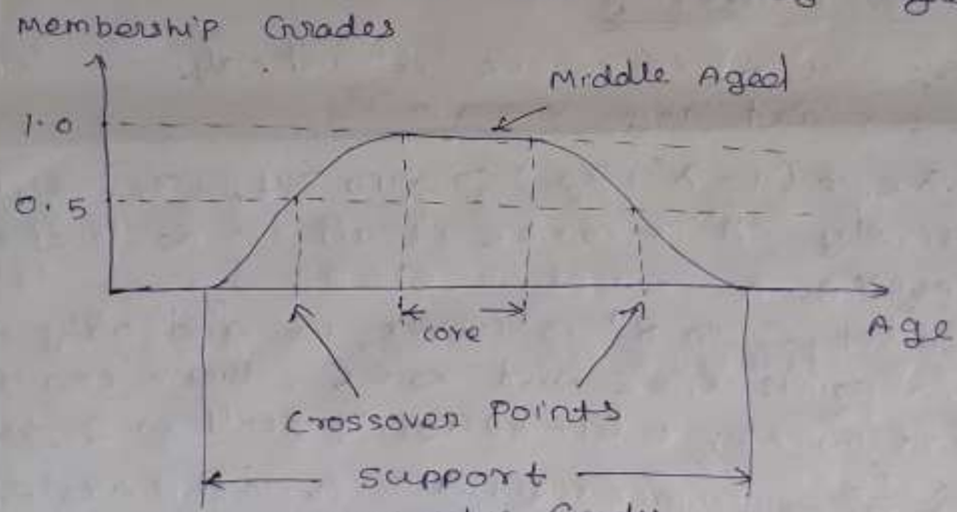


Figure 1.7(a)

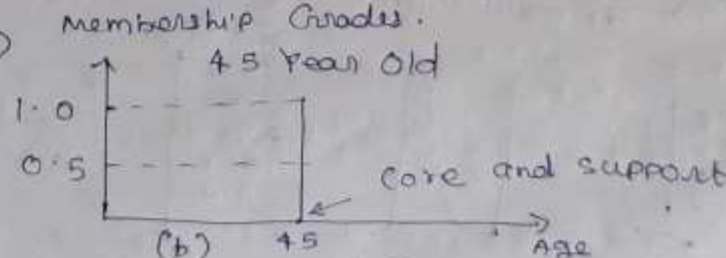


Figure 1.7. Cores, supports, and crossover points of (a) the fuzzy set "middle aged", and (b) the fuzzy singleton "45 year old".

13

Definition: α - cut, Strong α - cut

The α-cut or α-level set of a fuzzy set A is a crisp set defined by

$$A_\alpha = \{x \mid \mu_A(x) \geq \alpha\}.$$

Strong α-cut or strong α-level set are defined similarly:

$$A'_\alpha = \{x \mid \mu_A(x) > \alpha\}.$$

support $(A) = A'_0$

core $(A) = A_1$

where A is a fuzzy set.

Definition: Convexity

A fuzzy set A is convex if and only if for any $x_1, x_2 \in X$ and any $\lambda \in [0, 1]$,

$$\mu_A(\lambda x_1 + (1-\lambda)x_2) \geq \min\{\mu_A(x_1), \mu_A(x_2)\}.$$

Alternatively, A is convex if all its α-level sets are convex.

A crisp set C in R^n is convex if and only if for any two points $x_1 \in C$ and $x_2 \in C$, their convex combination $\lambda x_1 + (1-\lambda)x_2$ is still in C, where $0 \leq \lambda \leq 1$.

(a) Two convex fuzzy set (b) A non convex Fuzzy set

Figure 1.8 (a) Two convex membership function;
(b) a non convex membership function.

(14)

Definition: Fuzzy numbers

A fuzzy number A is a fuzzy set in the real line (R) that satisfies the conditions for normality and convexity.

Definition: Bandwidths of normal and convex fuzzy sets

For a normal and convex fuzzy set, the bandwidth or width is defined as the distance between the two unique crossover points:

$$\text{width}(A) = |x_2 - x_1|,$$

where $\mu_A(x_1) = \mu_A(x_2) = 0.5$.

Definition: Symmetry

A fuzzy set A is symmetric if its MF is symmetric around a certain point $x = c$, namely $\mu_A(c + x) = \mu_A(c - x)$ for all $x \in X$.

Definition: Open left, open right, closed

A fuzzy set A is open left if $\lim_{x \to -\infty} \mu_A(x) = 1$ and $\lim_{x \to +\infty} \mu_A(x) = 0$;

Open right if $\lim_{x \to -\infty} \mu_A(x) = 0$ and

$\lim_{x \to +\infty} \mu_A(x) = 1$ and

Closed if $\lim_{x \to -\infty} \mu_A(x) = \lim_{x \to +\infty} \mu_A(x) = 0$.

Eg: The fuzzy set "young" in Figure 1.6 is open left; "old" is open right; and "middle aged" is closed.

(15)

1.4 SET-THEORETIC OPERATIONS

Definition: 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) if and only if $\mu_A(x) \leq \mu_B(x)$ for all x.

In symbols,

$$A \subseteq B \Longleftrightarrow \mu_A(x) \leq \mu_B(x).$$

A is contained in B

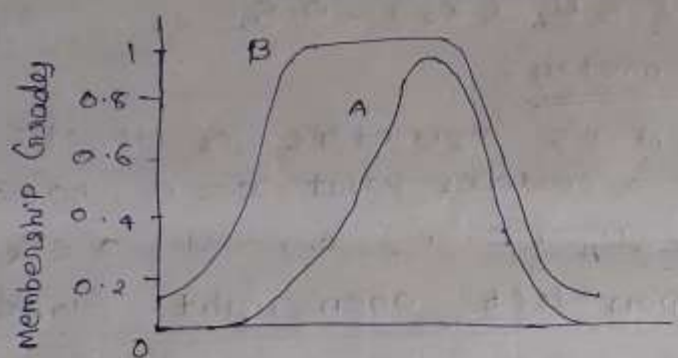


Figure 1.9 The concept of $A \subseteq B$.

Figure 1.9 illustrates the concept of $A \subseteq B$.

Definition: Union (disjunction)

The union of two fuzzy sets A and B is a fuzzy set C, written as $C = A \cup B$ or $C = A$ or B, whose MF is related to those of A and B by

$$\mu_C(x) = \max(\mu_A(x), \mu_B(x)) = \mu_A(x) \vee \mu_B(x)$$

Fuzzy sets have union, intersection and complement operations which were initially defined in Zadeh's seminal (inspirational) paper.

(16)

Table 1.2 Basic identities of classical sets, where A, B, and C are crisp sets; $\bar{A}$, $\bar{B}$, and $\bar{C}$ are their corresponding complements; X is the universe; and ϕ is the empty set.

Law of contradiction	$A \cap \bar{A} = \phi$
Law of the excluded middle	$A \cup \bar{A} = X$
Idempotency	$A \cap A = A$, $A \cup A = A$
Involution	$\bar{\bar{A}} = A$
Commutativity	$A \cap B = B \cap A$, $A \cup B = B \cup A$
Associativity	$(A \cup B) \cup C = A \cup (B \cup C)$ $(A \cap B) \cap C = A \cap (B \cap C)$
Distributivity	$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$ $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$
Absorption	$A \cup (A \cap B) = A$ $A \cap (A \cup B) = A$
Absorption of complement	$A \cup (\bar{A} \cap B) = A \cup B$ $A \cap (\bar{A} \cup B) = A \cap B$
DeMorgan's laws	$\overline{A \cup B} = \bar{A} \cap \bar{B}$ $\overline{A \cap B} = \bar{A} \cup \bar{B}$

Definition: Intersection (conjunction)

The intersection of two fuzzy sets A and B is a fuzzy set C written as $C = A \cap B$ or $C = A$ AND B, whose MF is related to those of A and B by

$$\mu_C(x) = \min(\mu_A(x), \mu_B(x)) = \mu_A(x) \wedge \mu_B(x).$$

As in the case of the union, it is obvious that the intersection of A and B is the "largest" fuzzy set which is

(17)

contained in both A and B. This reduces to the ordinary intersection operation if both A and B are nonfuzzy.

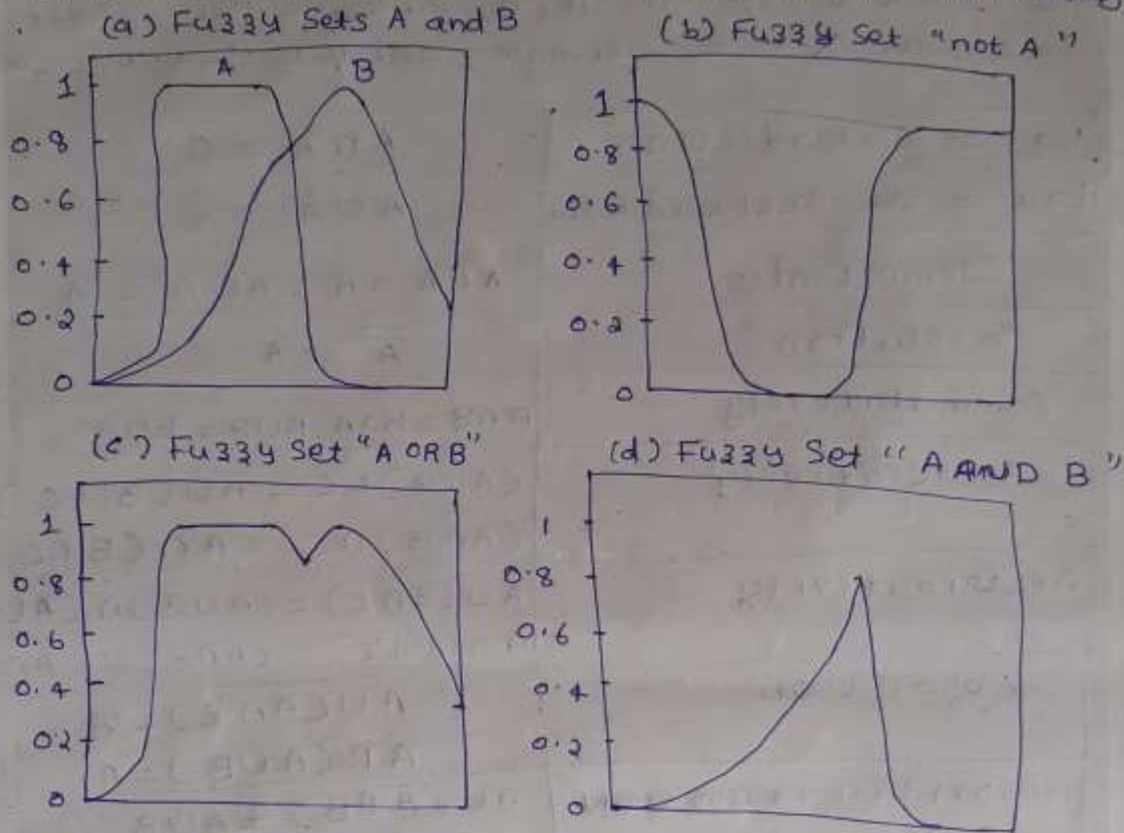


Figure 1.10 Operations on fuzzy sets:
(a) two fuzzy sets A and B; (b) $\bar{A}$; (c) $A \cup B$; (d) $A \cap B$.

<u>Definition</u>: <u>complement</u> (negation)

The complement of fuzzy set A, denoted by $\bar{A}$ ($\sim A$, NOT A) is defined as $\mu_{\bar{A}}(x) = 1 - \mu_A(x)$.

The max, min, and the complement operators are referred to as the <u>classical</u> or standard fuzzy operators for intersection, union, and negation, respectively, on fuzzy sets.

(18)

Definition: Cartesian product and co-product

Let A and B be fuzzy sets in X and Y, respectively. The Cartesian product of A and B, denoted by $A \times B$, is a fuzzy set in the product space $X \times Y$ with the membership function

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

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

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

Both $A \times B$ and $A+B$ are characterized by two-dimensional MFs.

1.5 Member function Formulation and Parameterization

1. MFs of One Dimension
2. MFs of Two Dimension
3. Derivatives of Parameterized MFs.

1. **MFs of One Dimension**

MFs with a single input.

Definition: Triangular MFs

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

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

(19)

OR

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

The parameters $\{a,b,c\}$ (with $a<b<c$) determine the x coordinates of the three corners of the underlying triangular MF.

Figure 1.11 illustrates a triangular MF defined by triangle $(x; 20, 60, 80)$.

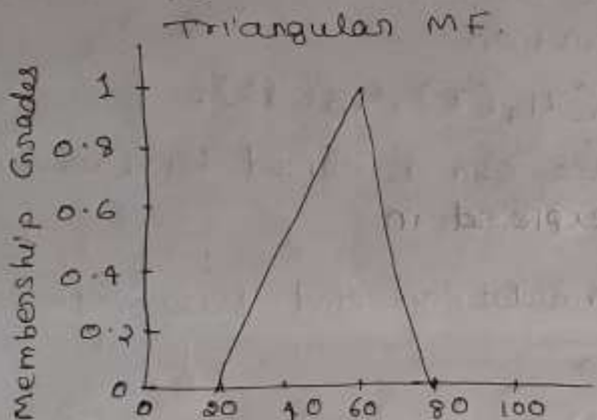


Figure 1.11 triangle $(x; 20, 60, 80)$.

Definition : Trapezoidal MFs

A trapezoidal MF is specified four parameters $\{a,b,c,d\}$ as follows.

$$\text{trapezoid}(x;a,b,c,d) = \begin{cases} 0, & x \le a. \\ \frac{x-a}{b-a}, & a \le x \le b. \\ 1, & b \le x \le c. \\ \frac{d-x}{d-c}, & c \le x \le d. \\ 0, & d \le x. \end{cases}$$

The parameters $\{a,b,c,d\}$ (with $a<b\le c<d$) determine the x coordinates of the four corners

(20)

of the underlying trapezoidal MF.

Figure 1.12 illustrates a trapezoidal MF defined by trapezoid $(x; 10, 20, 60, 95)$.

Trapezoidal MF.

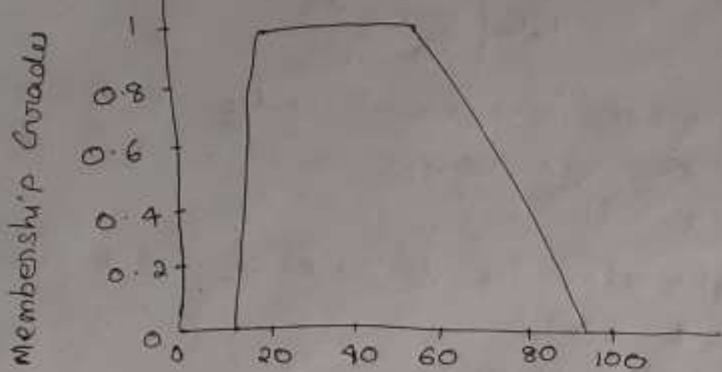


Figure 1.12 Trapezoid $(x; 10, 20, 60, 95)$.

<u>Definition</u>: <u>Gaussian MFs</u>

A Gaussian MF is specified by two parameters $\{c, \sigma\}$:

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

A Gaussian MF is determined completely by c and σ; c represents the MFs center and σ determines the MFs width.

Figure 1.13 plots a Gaussian MF defined by gaussian $(x; 50, 20)$.

Gaussian MF

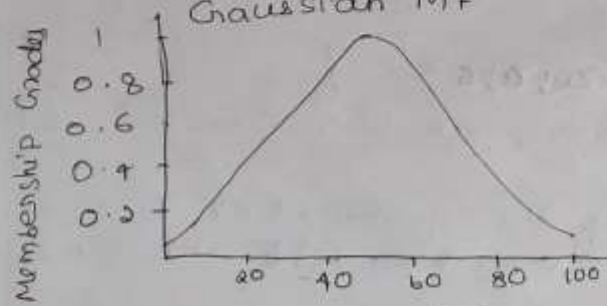


Figure 1.13 gaussial $(x; 50, 20)$:

(21)

Definition: Generalised bell MFs (Cauchy MF).

A generalised bell MF (or bell MF) is specified by three parameters $\{a, b, c\}$:

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

where the parameter b is usually positive. If b is negative, the shape of this MF becomes an upside-down bell.)

Figure 1.14 illustrates a generalised bell MF defined by bell (x; 20, 4, 50).

Generalised Bell MF

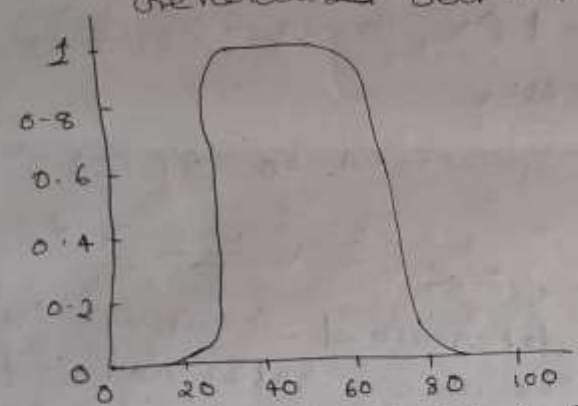


Figure 1.14 - bell (x; 20, 4, 50).

Definition: Sigmoidal MFs

A sigmoidal MF is defined by

$$\text{sig}(x; a, c) = \frac{1}{1 + \exp[-a(x - c)]}$$

where a controls the slope at the crossover point $x = c$.

Depending on the sign of the parameter a, a sigmoidal MF is inherently open right or left and thus is

(22)

appropriate for representing concepts such as "very large" or "very negative".

Sigmoidal functions are employed widely as the activation function of artificial neural networks.

Definition: Left-right MF (L-R MF)

A left-right MF or L-R MF is specified by three parameters $\{\alpha, \beta, c\}$:

$$LR(x; c, \alpha, \beta) = \begin{cases} F_L\left(\frac{c-x}{\alpha}\right), & x \leq c. \\ F_R\left(\frac{x-c}{\beta}\right), & x \geq c, \end{cases}$$

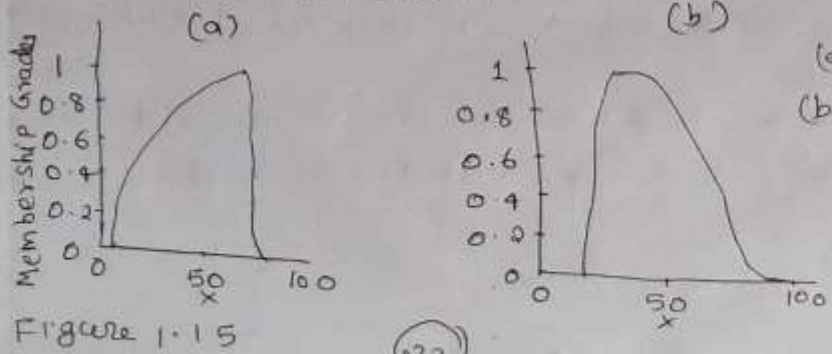
where $F_L(x)$ and $F_R(x)$ are monotonically decreasing functions defined on $[0, \infty)$ with $F_L(0) = F_R(0) = 1$ and $\lim_{x\to\infty} F_L(x) = \lim_{x\to\infty} F_R(x) = 0$.

Eg: Let

$$F_L(x) = \sqrt{\max(0, 1-x^2)},$$

$$F_R(x) = e^{-|x|^3}.$$

Based on the preceding $F_L(x)$ and $F_R(x)$, Figure 1.15 illustrates two L-R MFs specified by $LR(x; 65, 60, 10)$ and $LR(x; 25, 10, 40)$.



(a) $LR(x; 65, 60, 10)$.
(b) $LR(x; 25, 10, 40)$.

Figure 1.15
Two L-R MFs:

23

2. MFs of Two Dimensions

a) MFs with two inputs.

b) Each input in a different universe of discourse.

c) One natural way to extend one-dimensional MFs to two-dimensional ones is via cylindrical extension, defined next.

Definition: Cylindrical extensions of one-dimensional fuzzy sets.

If A is a fuzzy set in X, then its cylindrical extension in X × Y is a fuzzy set c(A) defined by

$$c(A) = \int_{X \times Y} \mu_A(x) / (x, y).$$

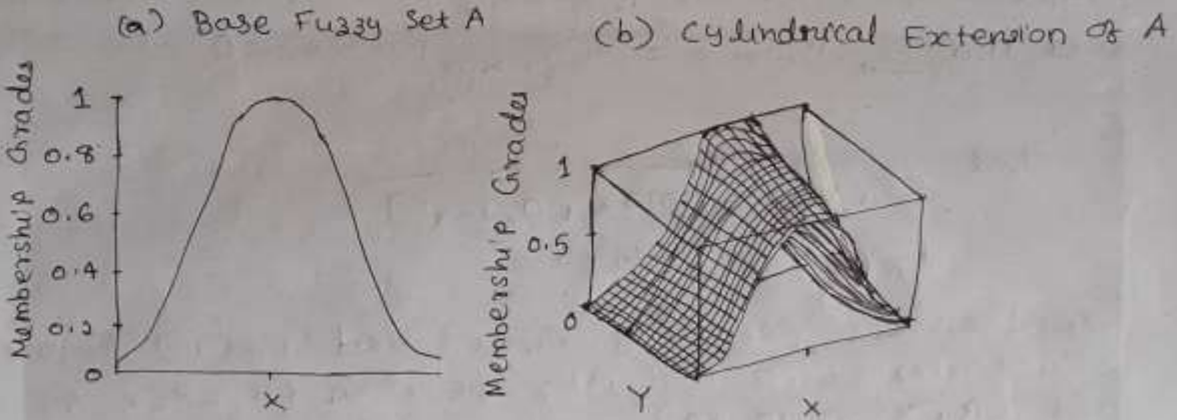


Figure 1.16 (a) Base set A ; (b) its cylindrical extension c(A).

The operation of projection, on the other hand, decreases the dimension of a given (multidimensional) membership function.

(24)

Definition: Projections of fuzzy sets

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

$$R_X = \int_X [\max_y \mu_R(x,y)]/x$$

and

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

respectively.

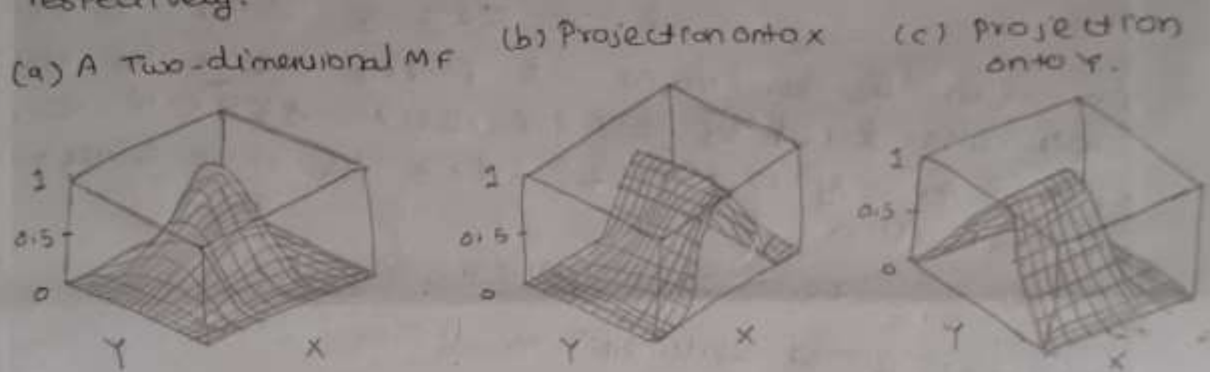


Figure 1.17

(a) Two-dimensional fuzzy set R;

(b) R_X (projection of R onto X); and

(c) R_Y (projection of R onto Y).

Figure 1.17 (a) shows the MF for fuzzy set R;

MFs of two dimensions fall into two categories: composite and noncomposite. 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.

(25)

Example composite and noncomposite MFs.

Suppose that fuzzy set A = "(x, y) is near (3, 4)" is defined by

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

Then this two-dimensional MF is composite, since it can be decomposed into two Gaussian MFs:

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

$$= \text{gaussian}(x; 3, 2)\ \text{gaussian}(y; 4, 1).$$

Note that we can view the fuzzy set A as two statements joined by the connective AND: "x is near 3 AND y is near 4" where the first statement is defined by

$$\mu_{near\,3}(x) = \text{gaussian}(x; 3, 2),$$

and the second statement is defined by

$$\mu_{near\,4}(y) = \text{gaussian}(y; 4, 1).$$

Thus the multiplication of these two MFs is used to interpret the AND operation of these two statements.

On the other hand, if this fuzzy set is defined by

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

then it is noncomposite.

3. Derivatives of Parameterized MFs.

i. To make a fuzzy system adaptive, we need to know the derivatives of an MF with respect to its argument (input) and parameters.

(26)

ii. Derivative information plays a central role in the learning or adaptation of a fuzzy system.

iii. For the Gaussian MF, let

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

Then

$$\frac{\partial y}{\partial x} = -\frac{x-c}{\sigma^2} y.$$

$$\frac{\partial y}{\partial \sigma} = \frac{(x-c)^2}{\sigma^3} y.$$

$$\frac{\partial y}{\partial c} = \frac{x-c}{\sigma^2} y.$$

For the bell MF, let

$$y = bell(x; a, b, c) = \frac{1}{1+\left|\frac{x-c}{a}\right|^{2b}}.$$

Then

$$\frac{\partial y}{\partial x} = \begin{cases} -\frac{2b}{x-c} y(1-y), & \text{if } x \neq c. \\ 0 & , \text{if } x = c. \end{cases}$$

$$\frac{\partial y}{\partial a} = \frac{2b}{a} y(1-y).$$

$$\frac{\partial y}{\partial b} = \begin{cases} -2\ln\left|\frac{x-c}{a}\right| y(1-y), & \text{if } x \neq c. \\ 0, & \text{if } x = c. \end{cases}$$

$$\frac{\partial y}{\partial c} = \begin{cases} \frac{2b}{x-c} y(1-y), & \text{if } x \neq c. \\ 0, & \text{if } x = c. \end{cases}$$

27

1.6 FUZZY RULES AND FUZZY REASONING

Fuzzy rules and fuzzy reasoning are the backbone of fuzzy inference systems.

Fuzzy rules and fuzzy reasoning have been successfully applied to a wide range of areas, such as automatic control, expert systems, pattern recognition, time series prediction, and data classification.

1. Extension principle and fuzzy relations.
2. Fuzzy if - then Rules
3. Fuzzy Reasoning

1. Extension principle and Fuzzy Relations

a. Extension Principle
b. Fuzzy Relations

a. Extension principle

The extension principle is a basic concept of fuzzy set theory that provides a general procedure for extending crisp domains of mathematical expressions to fuzzy domains.

A common point - to - point mapping of a function $f(\cdot)$ to a mapping between fuzzy sets.

suppose that f is a function from X to Y and A is a fuzzy set on X defined as

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

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

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

28

Example

Application of the extension principle to 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 $\vee$ represents max. The following figure illustrates this example.

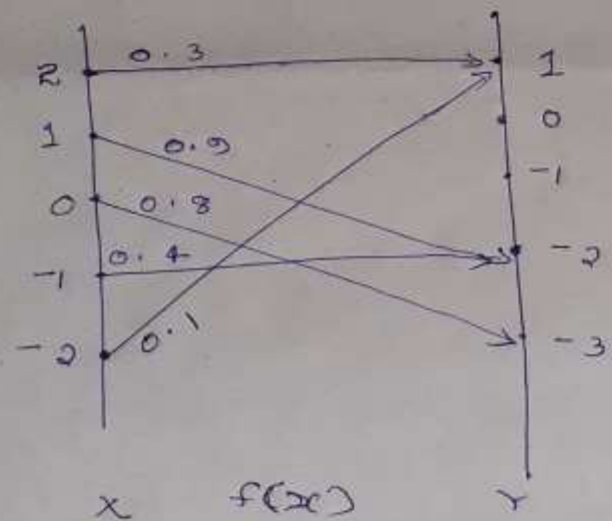


Figure 1.18 Extension principle on fuzzy sets with discrete universes.

28a

where

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

Definition: Extension principle

Suppose that function f is a mapping from an n-dimensional Cartesian product space $X_1 \times X_2 \times \ldots X_n$ to a one-dimensional universe Y such that $y = f(x_1, \ldots, x_n)$, and suppose $A_1, \ldots, A_n$ are n fuzzy sets in $X_1, \ldots X_n$, respectively. Then the extension principle asserts that the fuzzy set B induced by the mapping f is defined by

$$\mu_B(y) = \begin{cases} \displaystyle\max_{(x_1, \ldots, x_n), (x_1, \ldots, x_n) = f^{-1}(y)} [\min_i \mu A_i(x_i)], & \text{if } f^{-1}(y) \neq \phi \\ 0, & \text{if } f^{-1}(y) = \phi. \end{cases}$$

b. **Fuzzy Relations.**

Binary fuzzy relations are fuzzy sets in $X \times Y$ which map each element in $X \times Y$ to a membership grade between 0 and 1.

Definition: Binary fuzzy relation

Let X and Y be two universe of discourse. Then

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

is a binary fuzzy relation in $X \times Y$.

Examples of binary fuzzy relations:

1. Let $X = Y = R^+$ (the positive real line) and R = "y is much greater than x."

(29)

2. x is close to y (x and y are numbers)
3. x depends on y (x and y are events)
4. x and y look alike (x and y are persons, objects, and so on).
5. If x is large, then y is small (x is an observed reading and y is a corresponding action).

Definition : Max - min composition

Let R_1 and R_2 be two fuzzy relations defined on $X \times Y$ and $Y \times Z$, respectively.
The max-min composition of R_1 and R_2 is a fuzzy set defined by

$$R_1 \circ R_2 = \{ [(x, z), \max_y \min(\mu_{R_1}(x, y), \mu_{R_2}(y, z))] \mid x \in X, y \in Y, z \in Z \}.$$

or equivalently,

$$\mu_{R_1 \circ R_2}(x, z) = \max_y \min[\mu_{R_1}(x, y), \mu_{R_2}(y, z)]$$
$$= \vee_y [\mu_{R_1}(x, y) \wedge \mu_{R_2}(y, z)]$$

with the understanding that $\vee$ and $\wedge$ represent max and min respectively.
Max-min composition is also called the max-min product.
Several properties common to binary relations and max-min composition are given next, where R, S, and T are binary relations on $X \times Y$, $Y \times Z$, and $Z \times W$, respectively.

30

Associativity : $R \circ (S \circ T) = (R \circ S) \circ T$

Distributivity over union : $R \circ (S \cup T) = (R \circ S) \cup (R \circ T)$

weak distributivity over intersection : $R \circ (S \cap T) \subseteq (R \circ S) \cap (R \circ T)$

Monotonicity : $S \subseteq T \Rightarrow R \circ S \subseteq R \circ T$

Definition: Max - product composition

Assuming the same notation as used in the definition of max-min composition, we can define max-product composition as follows:

$$\mu_{R_1 \circ R_2}(x, z) = \max_y [\mu_{R_1}(x, y)\, \mu_{R_2}(y, z)].$$

Example: Max-min and max-product composition.

Let

R_1 = "x is relevant to y"

R_2 = "y is relevant to z"

be two fuzzy relations defined on $X \times Y$ and $Y \times Z$, respectively, where $X = \{1, 2, 3\}$, $Y = \{\alpha, \beta, \gamma, \delta\}$, and $Z = \{a, b\}$. Assume that R_1 and R_2 can be expressed as the following relation matrices:

$$R_1 = \begin{bmatrix} 0.1 & 0.3 & 0.5 & 0.7 \\ 0.4 & 0.2 & 0.8 & 0.9 \\ 0.6 & 0.8 & 0.3 & 0.2 \end{bmatrix}$$

$$R_2 = \begin{bmatrix} 0.9 & 0.1 \\ 0.2 & 0.3 \\ 0.5 & 0.6 \\ 0.7 & 0.2 \end{bmatrix}$$

31

Now we want to find $R_1 \circ R_2$, which can be interpreted as a derived fuzzy relation, "x is relevant to z" based on R_1 and R_2. For simplicity suppose that we are only interested in the degree of relevance between 2 ($\in X$) and a ($\in Z$). If we adopt max-min composition, then

$$\mu_{R_1 \circ R_2}(2, a) = \max(0.4 \wedge 0.9, 0.2 \wedge 0.2, 0.8 \wedge 0.5, 0.9 \wedge 0.7)$$
$$= \max(0.4, 0.2, 0.5, 0.7)$$
$$= 0.7 \text{ (by max-min composition)}.$$

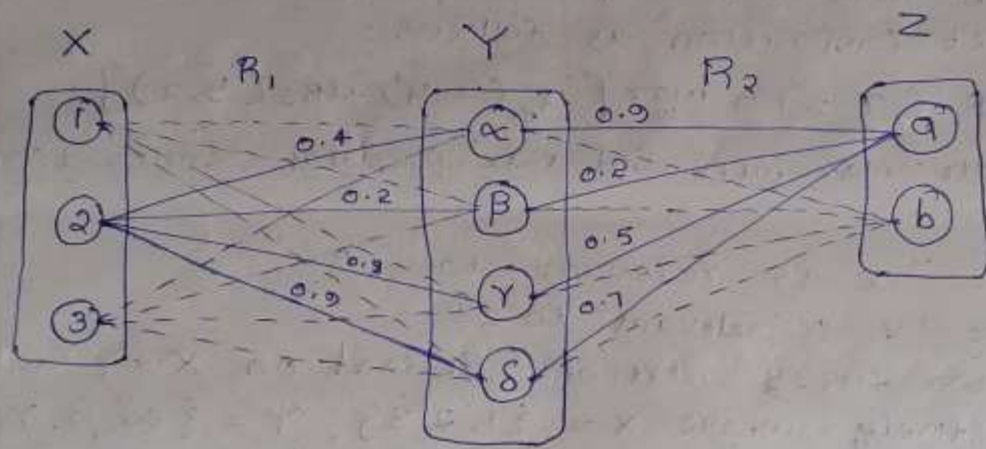


Figure 1.18. composition of fuzzy relations.

On the other hand, if we choose max-product composition instead, we have

$$\mu_{R_1 \circ R_2}(2, a) = \max(0.4 \times 0.9, 0.2 \times 0.2, 0.8 \times 0.5, 0.9 \times 0.7)$$
$$= \max(0.36, 0.04, 0.40, 0.63)$$
$$= 0.63 \text{ (by max-product composition)}.$$

Figure 1.18 illustrates the composition of two fuzzy relations, where the relation between element 2 in X

32

and element a in z is built up via the four possible paths (solid lines) connecting these two elements). The degree of relevance between 2 and a is the maximum of these four paths' strengths, while each path's strength is the minimum (or product) of the strengths of its constituent links.

2. FUZZY IF-THEN RULES

1. Linguistic Variables
2. Fuzzy if - then Rules

1. Linguistic Variables

Principle of incompatibility: "As the complexity of a system increases, our ability to make precise and yet significant statements about its behavior diminishes until a threshold is reached beyond which precision and significance become almost mutually exclusive characteristics".

Definition: Linguistic variables and other related terminology

A linguistic variable is characterised by a quintuple (x, T(x), X, G, M) in which x is the same name of the variable; T(x) is the term set of x - that is, the set of its linguistic values or linguistic terms; X is the universe of discourse; G is a 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.

(33)

Definition: Concentration and dilation of linguistic values

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

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

In particular, the operation of concentration is defined as $CON(A) = A^2$,

while that of dilation is expressed by

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

The negation operator NOT and the connectives AND and OR as

$$NOT(A) = \neg A = \int_X [1 - \mu_A(x)]/x,$$

$$A \text{ AND } B = A \cap B = \int_X [\mu_A(x) \wedge \mu_B(x)]/x,$$

$$A \text{ OR } B = A \cup B = \int_X [\mu_A(x) \vee \mu_B(x)]/x,$$

respectively, where A and B are two linguistic values whose meaning are defined by $\mu_A(\cdot)$ and $\mu_B(\cdot)$.

Definition: Contrast intensification.

The operation of contrast intensification of a linguistic value A is defined by

34

$$INT(A) = \begin{cases} 2A^2, & \text{for } 0 \leq \mu_A(x) \leq 0.5, \\ \neg 2(\neg A)^2, & \text{for } 0.5 \leq \mu_A(x) \leq 1. \end{cases}$$

The contrast intensifier INT increases the values of $\mu_A(x)$ which are above 0.5 and diminishes those which are below this point. Thus, contrast intensification has the effect of reducing the fuzziness of linguistic value A. The inverse operator of contrast intensifier is contrast diminisher DIM.

Definition: Orthogonality

A term set $T = t_1, \dots, t_n$ of a linguistic variable x on the universe X is orthogonal if it fulfills the following property:

$$\sum_{i=1}^{n} \mu_{t_i}(x) = 1, \forall x \in X,$$

where the t_i's are convex and normal fuzzy sets defined on X and these fuzzy sets make up the term set T.

For the MFs in a term set to be intuitively reasonable, the orthogonality requirement has to be followed to some extent.

2. Fuzzy If-then Rules

A fuzzy if-then rule (also known as fuzzy rule, fuzzy implication, or fuzzy conditional statement) assumes 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.

35

often "x is A" is called the antecedent or premise while "y is B" is called the consequence or conclusion.

eg:

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

There are two ways to interpret the fuzzy rule $A \rightarrow B$.

i) A coupled with B

$$R = A \rightarrow B = A \times B = \int_{X \times Y} \mu_A(x) \tilde{*} \mu_B(y) / (x, y)$$

where $\tilde{*}$ is a T-norm operator and $A \rightarrow B$ is used again to represent the fuzzy relation R.

ii) A entails B

It can be written as four different formulas :-

a) Material implication : $R = A \rightarrow B = \sim A \cup B$.

b) propositional calculus:

$$R = A \rightarrow B = \sim A \cup (A \cap B).$$

c) Extended propositional calculus:

$$R = A \rightarrow B = (\sim A \cap \sim B) \cup B.$$

d) Generalisation of modus ponens:

$\mu_R(x, y) = \sup\{c \mid \mu_A(x) \tilde{*} c \leq \mu_B(y)$ and $0 \leq c \leq 1\}$,

where $R = A \rightarrow B$ and $\tilde{*}$ is a T-norm operator.

36

Rule Forms

In general, three forms of rules exist for any linguistic variables.

1. Assignment statement.
   eg. x is not large AND not very small.
2. conditional statement
   eg. IF x is very big THEN y is medium
3. Unconditional statement
   eg. set pressure high

Fuzzy Reasoning

Fuzzy Reasoning, also known as approximate reasoning, is a inference procedure that derives conclusions from a set of fuzzy if-then rules and known facts.

The basic rule of inference in traditional two-value topic is modus ponens, according to which we can infer the truth of a proposition B from the truth of A and the implication A → B. For instance, if A is identified with "the tomato is red" and B with "the tomato is ripe" then if it is true that "the tomato is red" it is also true that "the tomato is ripe". This concept is illustrated as follows:

Premise 1 (fact): x is A

premise 2 (rule): if x is A then y is B

- -

Consequence (conclusion): y is B

37

Premise 1 (fact) : x is A'

Premise 2 (rule) : if x is A then y is B

Consequence (conclusion) : y is B'

where A' is close to A and B' is close to B. When A, B, A', and B' are fuzzy sets of approximate universes, the foregoing inference procedure is called approximate reasoning or fuzzy reasoning; it is also called generalized modus ponens (GMP for short), since it has modus ponens as special case.

4/2/2020

38

Module II

FUZZY INFERENCE SYSTEMS (FIS)

The fuzzy inference system is a popular computing framework based on the concepts of fuzzy set theory, fuzzy if-then rules, and fuzzy reasoning. It is multidisciplinary in nature.

Other names of fuzzy inference system include:

i. Fuzzy-rule-based system.
ii. Fuzzy expert system
iii. Fuzzy model
iv. Fuzzy associative memory,
v. Fuzzy logic controller and
vi. Fuzzy system.

Three conceptual components of FIS:

1. RULE BASE
   Contains a selection of fuzzy rules.
2. Database (or Dictionary)
   Defines the membership functions used in the fuzzy rules.
3. Reasoning mechanism
   performs the inference procedure upon the rules and given facts to derive a reasonable output or conclusion.

A fuzzy inference system with a crisp output is shown in Figure 2.1.

1. The basic fuzzy inference system can take either fuzzy inputs or crisp inputs (fuzzy singletons).

(1)

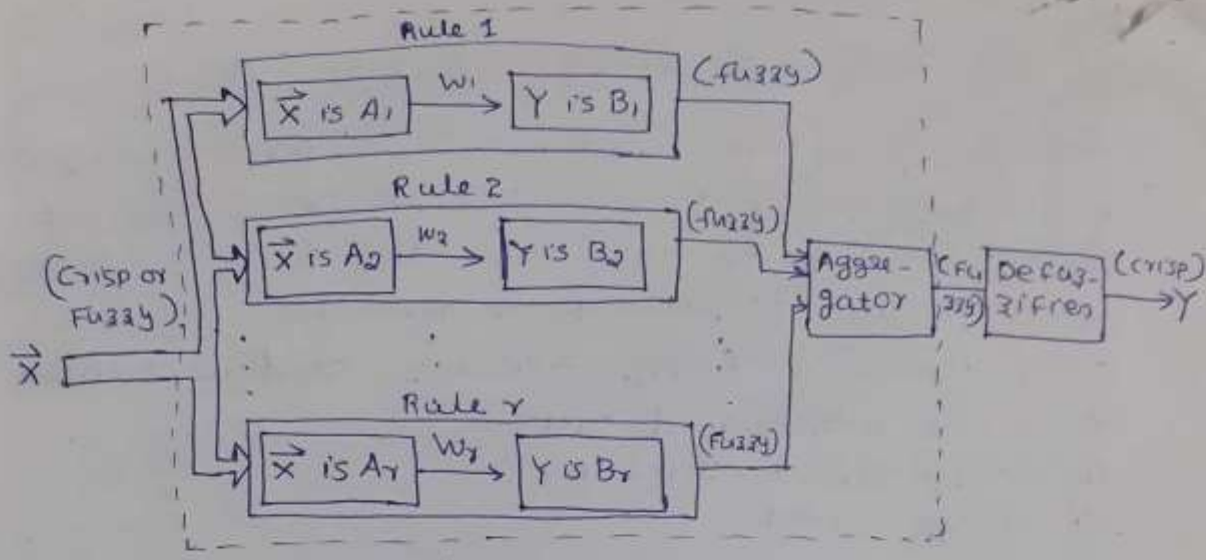


Figure 2.1 Block diagram for a fuzzy inference system.

2. The outputs FIS produces are almost always fuzzy sets.
3. Sometimes it is necessary to have a crisp output, especially in a situation where a fuzzy inference system is used as a controller.
4. A method of defuzzification is used to extract a crisp value that best represents a fuzzy set.
5. A fuzzy inference system with a crisp output is shown in Figure 2.1.
6. Dashed line indicates a basic fuzzy inference system with fuzzy output.
7. An example of a fuzzy inference system without defuzzification block is the two-rule two-input system.

⑤

Definition:

Approximate reasoning (fuzzy reasoning)

Let A, A', and B be fuzzy sets of X, X, and Y, respectively. Assume that the fuzzy implication $A \rightarrow B$ is expressed as a fuzzy relation R on $X \times Y$. Then the fuzzy set B induced by "x is A" and the fuzzy rule "if x is A then y is B" is defined by

$$\mu_{B'}(y) = \max_x \min[\mu_{A'}(x), \mu_R(x,y)]$$
$$= \vee_x [\mu_{A'}(x) \wedge \mu_R(x,y)],$$

or equivalently,

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

Multiple Rules with multiple Antecedents

Multiple fuzzy rules with multiple antecedents are involved in describing a system's behavior.

The interpretation of multiple rules is usually taken as the union of the fuzzy relations corresponding to the fuzzy rules.

Therefor

premise 1 (fact): x is A' and y is B',

premise 2 (rule 1): if x is A_1 and y is B_1 then z is C_1,

premise 3 (rule 2): if x is A_2 and y is B_2 then z is C_2,

consequence (conclusion): z is C'

To verify this inference procedure, let $R_1 = A_1 \times B_1 \rightarrow C_1$ and $R_2 = A_2 \times B_2 \rightarrow C_2$. Since the max-min composition operator $\circ$ is distributive over the $\cup$ operator, it follows

③

that

$$C' = (A' \times B') \circ (R_1 \cup R_2)$$
$$= [(A' \times B') \circ R_1] \cup [(A' \times B') \circ R_2]$$
$$= C_1' \cup C_2',$$

where C_1' and C_2' are the inferred fuzzy sets for rules 1 and 2, respectively.

Three types of fuzzy inference systems.

1. MAMDANI FUZZY MODELS
2. SUGENO FUZZY MODELS
3. TSUKAMOTO FUZZY MODELS

The differences lie among these three fuzzy inference systems lie in the consequents of their fuzzy rules and thus their aggregation and defuzzification procedures differ accordingly.

MAMDANI FUZZY MODELS

1. The mamdani Fuzzy Inference system was proposed in 1975 by Ebhasim Mamdani. Basically, it was anticipated to control a steam engine and boiler combination by synthesizing a set of fuzzy rules obtained from people working on the system.

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

(1)

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. Figure 2.2 is an illustration of how a two-rule Mamdani fuzzy inference system derives the overall output z when subjected to two crisp inputs x and y.

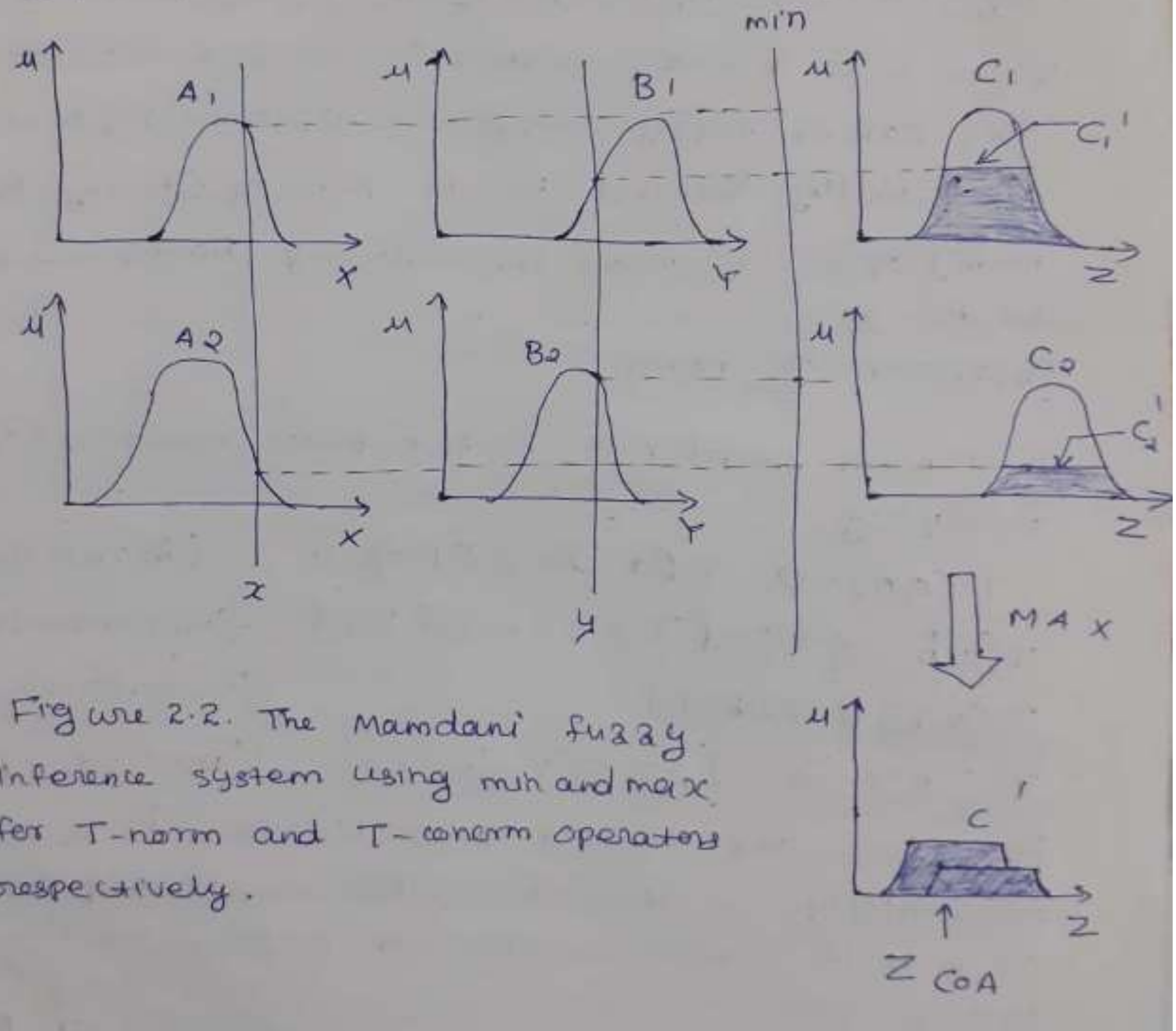


Figure 2.2. The Mamdani fuzzy inference system using min and max for T-norm and T-conorm operators respectively.

5

Fuzzy Intersection

The intersection of two fuzzy sets A and B is specified in general by a function

$T : [0,1] \times [0,1] \rightarrow [0,1]$, which aggregates two membership grades as follows:

$$\mu_{A \cap B}(x) = T(\mu_A(x), \mu_B(x)) = \mu_A(x) \tilde{*} \mu_B(x)$$

where $\tilde{*}$ is a binary operator for the function T. This class of fuzzy intersection operators, which are usually referred to as T-norm (triangular norm) operators, meets the following basic requirements.

Definition T-norm.

A T-norm operator is a two-place function $(T\cdot,\cdot)$ satisfying

$T(0,0) = 0, \ T(a,1) = T(1,a) = a$ (boundary)

$T(a,b) \leq T(c,d)$ if $a \leq c$ and $b \leq d$ (monotonicity)

$T(a,b) = T(b,a)$ (commutativity)

$T(a, T(b,c)) = T(T(a,b), c)$ (associativity)

boundary – the correct generalisation to crisp sets

monotonicity – a decrease in the membership values in A or B cannot produce an increase in the membership value in $A \cap B$.

commutativity – the operator is indifferent to the order of the fuzzy sets to be combined.

6

associativity – to ~~to~~ take the intersection of any number of sets in any order of pairwise groupings.

Eg: Four T-norm operators.

Four of the most frequently used T-norm operators are

1. Minimum: $T_{min}(a,b) = min(a,b) = a \wedge b$.
2. Algebraic product: $T_{ap}(a,b) = ab$.
3. Bounded product: $T_{bp}(a,b) = 0 \vee (a+b-1)$.
4. Drastic product: $T_{dp}(a,b) = \begin{cases} a, & \text{if } b = 1. \\ b, & \text{if } a = 1. \\ 0, & \text{if } a, b < 1. \end{cases}$

Definition T-conorm (S-norm)

A T-conorm (or S-norm) operator is a two-place function $S(\cdot,\cdot)$ satisfying

$S(1,1) = 1$, $S(0,a) = S(a,0) = a$ (boundary)

$S(a,b) \leq S(c,d)$ if $a \leq c$ and $b \leq d$ (monotonicity)

$S(a,b) = S(b,a)$ (commutativity)

$S(a, S(b,c)) = S(S(a,b), c)$ (associativity).

Eg: Four T-conorm operators

1. Maximum: $S(a,b) = max(a,b) = a \vee b$.
2. Algebraic sum: $S(a,b) = a + b - ab$.
3. Bounded sum: $S(a,b) = 1 \wedge (a+b)$.

7

4. Drastic sum: $s(a,b) = \begin{cases} a, & \text{if } b=0. \\ b, & \text{if } a=0 \\ 1, & \text{if } a,b>0. \end{cases}$

Defuzzification

Defuzzification refers to the way a crisp value is extracted from a fuzzy set as a representative value. In general, there are five methods for defuzzifying a fuzzy set A of a universe of discourse Z, as shown in Figure 2.3.

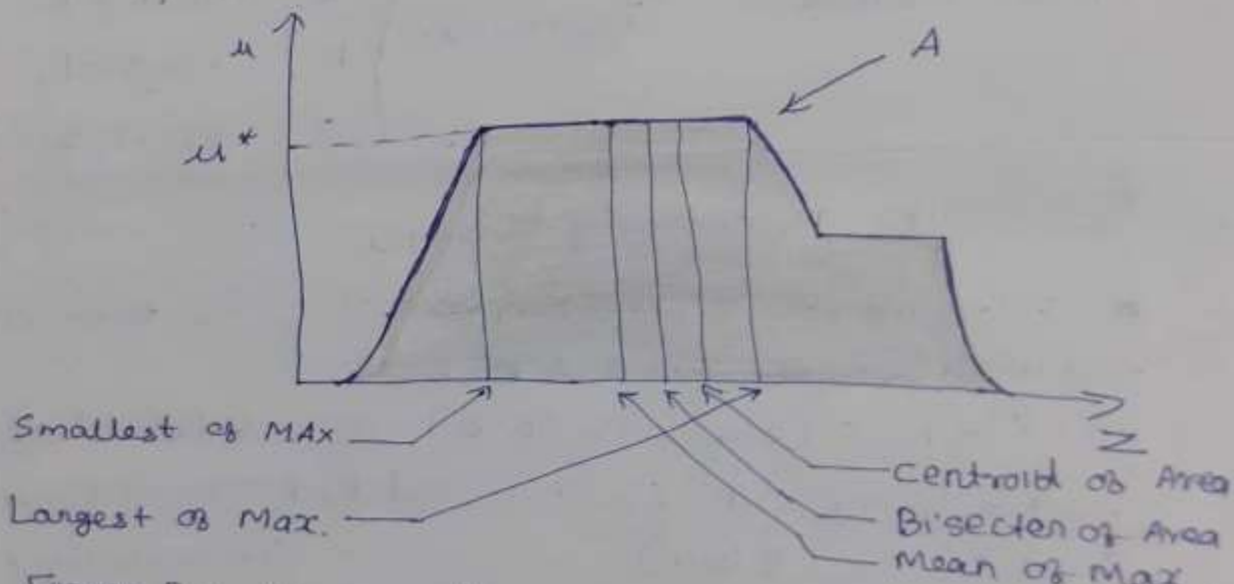


Figure 2.3 Various defuzzification schemes for obtaining a crisp output.

1. Centroid of area z_{COA}:

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

(8)

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

where $\mu_A(z)$ is the aggregated output MF. This is the most widely adopted defuzzification strategy.

2. Bisector of area z_{BOA}:

z_{BOA} satisfies

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

where $\alpha = \min\{z \mid z \in Z\}$ and $\beta = \max\{z \mid z \in Z\}$. That is, 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.

3. Mean of maximum z_{MOM}: z_{MOM} is the average of the maximizing z at which the MF reach a maximum μ^*.

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

where $Z' = \{z \mid \mu_A(z) = \mu^*\}$.

The mean of maximum is the defuzzification ~~stat~~ strategy employed in Mamdani's fuzzy logic controllers.

4. Smallest of maximum z_{SOM}: z_{SOM} is the minimum (in terms of magnitude) of the maximizing z.

9

5. Largest of maximum z_{LOM} :

z_{LOM} is the maximum (in terms of magnitude) of the maximizing z. Because of their obvious bias, z_{SOM} and LOM are not used as often as the other three defuzzification methods.

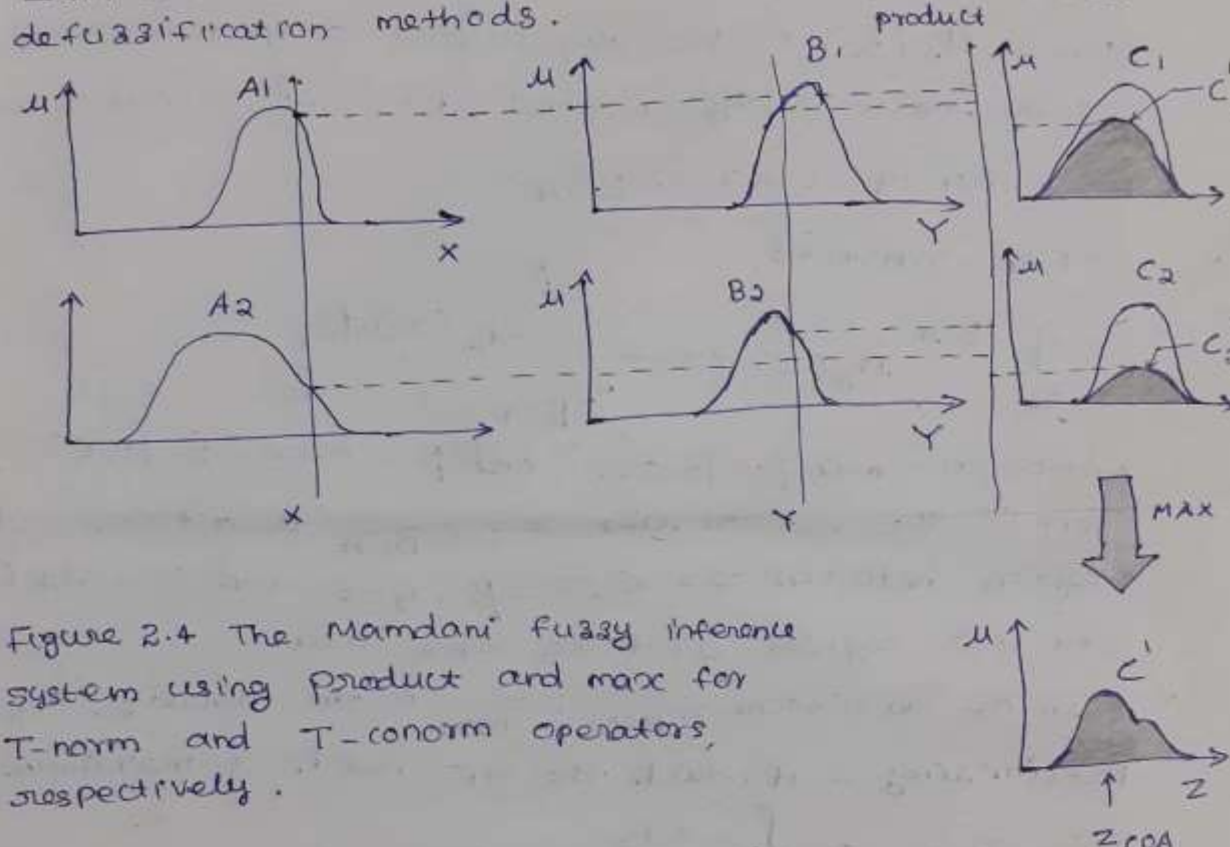


Figure 2.4 The Mamdani fuzzy inference system using product and max for T-norm and T-conorm operators, respectively.

The calculation needed to carry out any of the five defuzzification operations is time-consuming unless special hardware support is available.

Eg: Single-input single-output Mamdani fuzzy model

An example of a single-input single-output Mamdani fuzzy model with three rules can be expressed as

(10)

$$\begin{cases} \text{If } X \text{ is small then } Y \text{ is small.} \\ \text{If } X \text{ is medium then } Y \text{ is medium.} \\ \text{If } X \text{ is large then } Y \text{ is large.} \end{cases}$$

Example: Two-input single-output Mamdani fuzzy model.

An example of a two-input single-output mamdani fuzzy model with four rules can be expressed as

$$\begin{cases} \text{If } X \text{ is small and } Y \text{ is small then } Z \text{ is negative large} \\ \text{If } X \text{ is small and } Y \text{ is large then } Z \text{ is negative small.} \\ \text{If } X \text{ is large and } Y \text{ is small then } Z \text{ is positive small.} \\ \text{If } X \text{ is large and } Y \text{ is large then } Z \text{ is positive large.} \end{cases}$$

SUGENO FUZZY MODEL (TSK fuzzy model)

The Sugeno fuzzy model was proposed by Takagi, Sugeno, and Kang in an effort to develop a systematic approach to generating fuzzy rules from a given input-output data set.

A typical fuzzy rule in a Sugeno fuzzy model has the form

if x is A and y is B then $z = f(x, y)$,

where A and B are fuzzy sets in the antecedent, while $z = f(x, y)$ is a crisp function in the consequent,

Usually $f(x, y)$ is a polynomial in the input variables x, and y, but it can be any function as long

(11)

as it can appropriately describe the output of the model within the fuzzy region specified by the antecedent of the rule.

First-order sugeno fuzzy model

when $f(x,y)$ is a first-order polynomial, the resulting fuzzy inference system is called a first-order sugeno fuzzy model.

zero-order sugeno fuzzy model

When f is a constant, we then have a zero-order sugeno fuzzy model, which can be viewed either as a special case of the Mamdani fuzzy Inference system, in which each rule's consequent ~~we~~ is specified by a fuzzy singleton, or a special case of the Tsukamoto fuzzy model.

The overlap of MFs in the consequent of a Mamdani model does not have a decisive effect on the smoothness; it is the overlap of the antecedent MFs that determines the smoothness of the resulting input-output behavior.

Figure 2.5 shows the fuzzy reasoning procedure for a first-order Sugeno fuzzy model.

Since each rule has a crisp output, the overall output is obtained via weighted average, thus avoiding the time-consuming process of defuzzification required in a Mamdani model. In practice, the weighted average operator is sometimes replaced with the weighted sum operator (that is, $z = w_1 z_1 + w_2 z_2$ in Figure 2.5) to reduce computation

(12)

further, especially in the training of a fuzzy inference system.

Since the only fuzzy part of a sugeno model is in its antecedent, it is easy to demonstrate the distinction between a set of fuzzy rules and nonfuzzy ones.

Example: Two-input single-output sugeno fuzzy model.

An example of a two-input single-output Sugeno fuzzy model with four rules can be expressed as

$$\begin{cases} \text{If } X \text{ is small and } Y \text{ is small then } z = -x+y+1. \\ \text{If } X \text{ is small and } Y \text{ is large then } z = -y+3 \\ \text{If } X \text{ is large and } Y \text{ is small then } z = -x+3 \\ \text{If } X \text{ is large and } Y \text{ is large then } z = x+y+2. \end{cases}$$

Unlike the Mamdani fuzzy model, Sugeno fuzzy model cannot follow the compositional rule of inference strictly in its fuzzy reasoning mechanism.

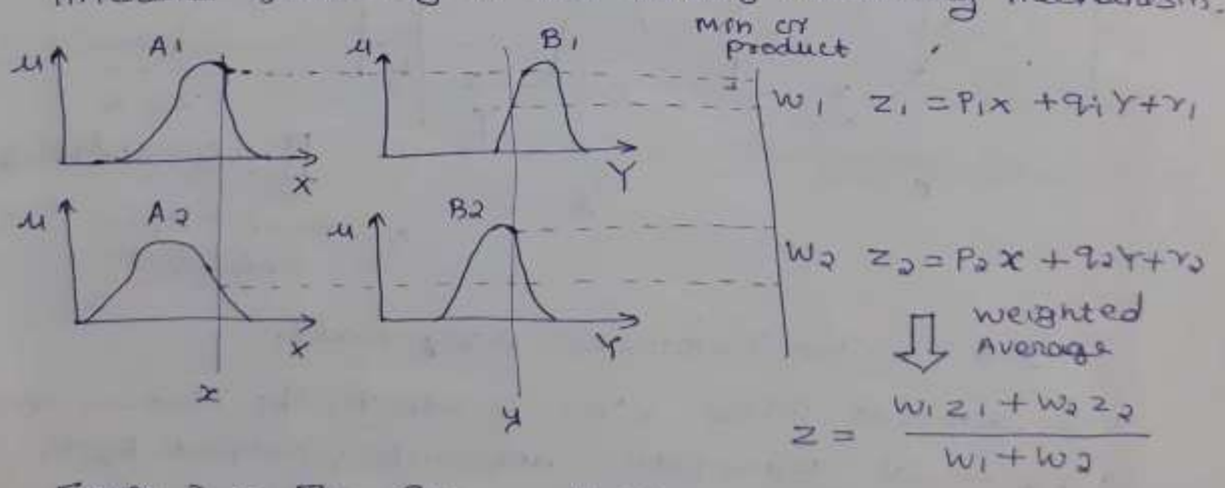


Figure 2.5 The Sugeno fuzzy model.

(13)

TSUKAMOTO FUZZY MODELS

In the Tsukamoto fuzzy models, the consequent of each fuzzy if-then rule is represented by a fuzzy set with a monotonical MF, as shown in Figure 2.6. As a result, the inferred output of each rule is defined as a crisp value induced by the rule's firing strength. The overall output is taken as the weighted average of each rule's firing strength.

Figure 2.6 illustrates the reasoning procedure for a two-input two-rule system.

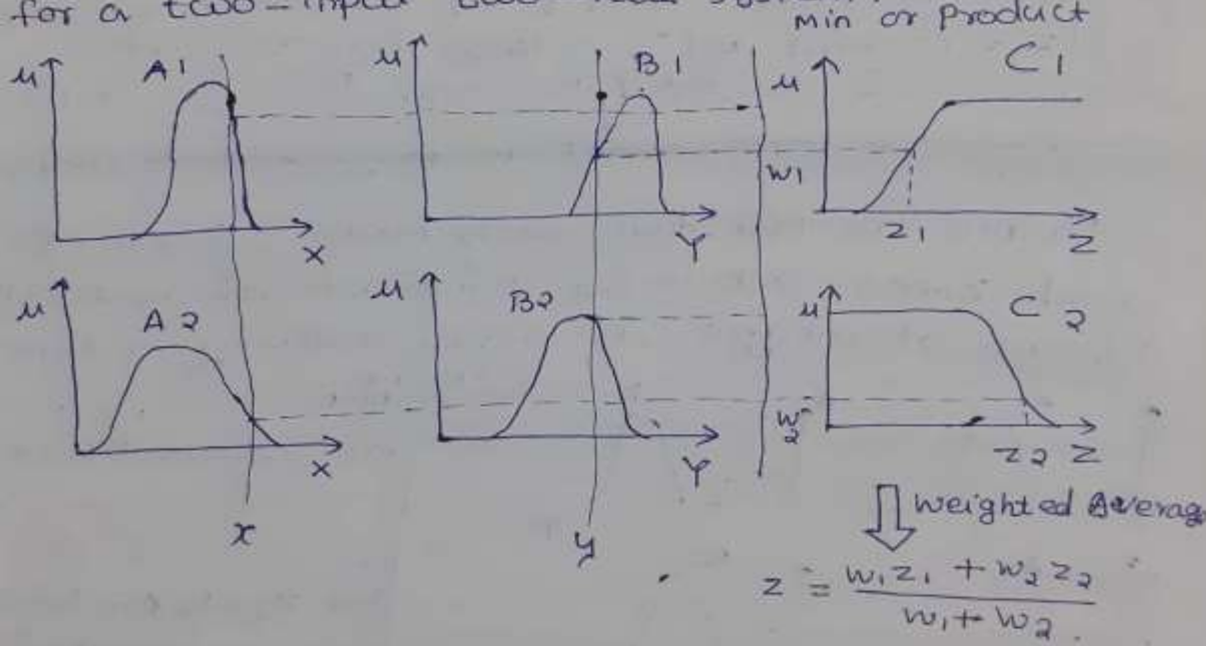


Figure 2.6. The Tsukamoto fuzzy model.

Since each rule infers a crisp output, the Tsukamoto fuzzy model aggregates each rule's output by the method of weighted average and thus avoids the time-consuming process of defuzzification.

(14)

Example: Single-input Tsukamoto fuzzy model.

An example of a single-input Tsukamoto fuzzy model can be expressed as

$$\begin{cases} \text{If } X \text{ is small then } Y \text{ is } C_1 \\ \text{If } X \text{ is medium then } Y \text{ is } C_2 \\ \text{If } X \text{ is large then } Y \text{ is } C_3. \end{cases}$$

Since the reasoning mechanism of the Tsukamoto fuzzy model does not follow strictly the compositional rule of inference, the output is always crisp even when the inputs are fuzzy.

DERIVATIVE-BASED OPTIMIZATION.

1. Gradient-based optimization technique capable of determining search directions according to an objective function's derivative information.

DESCENT METHODS

1. Minimizing a real-valued objective function E defined on an n-dimensional input space $\theta = [\theta_1, \theta_2, \cdots, \theta_n]^T$. Finding a minimum point $\theta = \theta^*$ that minimizes $E(\theta)$ is of primary concern.
2. In iterative descent methods, the next point θ_{next} is determined by a step down from the current point θ_{now} in a direction vector $\underline{d}$:

$$\theta_{next} = \theta_{now} + \eta d,$$

where η (Eta) is some positive step size regulating

(15)

to what extent to proceed in that direction. In neuro-fuzzy literature, the term learning rate is used for the step size η.

That is

3. $$\theta_{k+1} = \theta_k + \eta_k d_k \quad (k = 1, 2, 3, \ldots),$$

where k denotes the current iteration number, and θ_{now} and θ_{next} represent two consecutive elements in a generated sequence of solution candidates $\{\theta_k\}$. The θ_k is intended to converge to a (local) minimum θ^*.

4. The iterative descent methods compute the kth step $\eta_k d_k$ through two procedures:

1. First determining direction d, and
2. Calculate step size η.

The next point θ_{next} should satisfy the following inequality:

$$E(\theta_{next}) = E(\theta_{now} + \eta d) < E(\theta_{now}).$$

5. The principal differences between various descent algorithms lie in the first procedure for determining successive directions.

The second procedure determines optimum step size by line minimization:

$$\eta^* = \arg \min_{\eta > 0} \phi(\eta),$$

where $\phi(\eta) = E(\theta_{now} + \eta d)$.

The search of η^* is accomplished by line search (or one-dimensional search)

16

Descent methods include
1. Gradient-based Methods
2. The method of steepest Descent
3. Newton's Methods.

1. Gradient-based Methods

When the straight downhill direction d is determined on the basis of the gradient (g) of an objective function E, such descent methods are called gradient-based descent methods.

The gradient of a differentiable function $E: R^n \rightarrow R$ at θ is the vector of first derivatives of E, denoted as g. That is,

$$g(\theta) \; (= \nabla E(\theta)) \overset{\text{def}}{=} \left[\frac{\partial E(\theta)}{\partial \theta_1}, \frac{\partial E(\theta)}{\partial \theta_2}, \ldots, \frac{\partial E(\theta)}{\partial \theta_n} \right]^T.$$

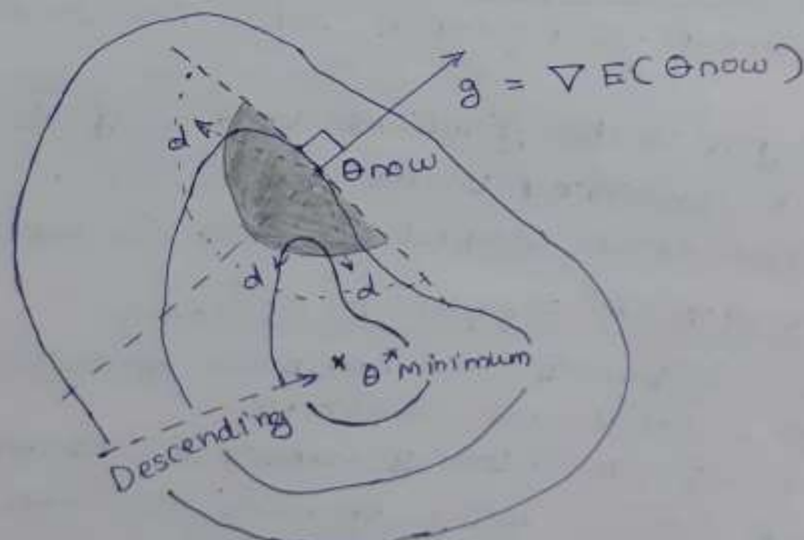


Figure 2.7 Feasible descent directions. Directions from the starting point θ_{now} in the shaded area are possible descent vector candidates. When $d = -g$, d is the

17

steepest descent direction at a local point θ_{now}.
A class of gradient-based descent methods has the following fundamental form, in which feasible descent directions can be determined by deflecting the gradients through multiplication by G (i.e. deflected gradients):

$$\theta_{next} = \theta_{now} - \eta G g.$$

with some positive step size η and some positive definite matrix G.

For minimizing the objective function, the descent procedures are typically repeated until one of the following stopping criteria is satisfied:

1. The objective function value is sufficiently small;
2. The length of the gradient vector g is smaller than a specified value", or
3. The specified computing time is exceeded.

2. THE METHOD OF STEEPEST DESCENT

The most frequently used nonlinear optimization technique due to its simplicity.
when G = I (the identity matrix), the equation
$\theta_{next} = \theta_{now} - \eta G g$ becomes the steepest descent formula: $\theta_{next} = \theta_{now} - \eta g$.

(18)

The negative gradient direction $(-g)$ points to the locally steepest downhill direction. Going in the negative gradient direction may not be a shortcut to reach the minimum point θ^*.

If the steepest descent method employs line minimization in $\eta^* = \arg\min_{\eta > 0} \phi(\eta)$.

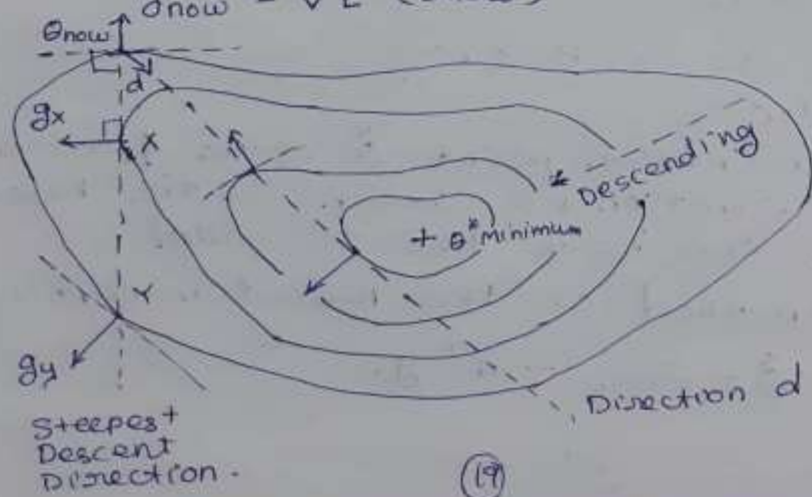
If the minimum point η^* in a direction d is obtained at each iteration then

$$\phi'(\eta) = \frac{dE(\theta_{now} - \eta g_{now})}{d\eta}$$
$$= \nabla^T E(\theta_{now} - \eta g_{now})\, g_{now}$$
$$= g^T_{next}\, g_{now}$$
$$= 0.$$

where g_{next} is the gradient vector at the next point. This indicates that the next gradient vector g_{next} is always orthogonal to the current gradient vector g_{now}.

Figure 2.8

$g_{now} = \nabla E(\theta_{now})$



In the steepest descent direction, θ_{next}, may be either X or Y, or close to them.

(19)

3. Newton's Methods

classical newton's method.

The descent direction d can be determined by using the second derivatives of the objective function E.

If the starting position θ_{now} is sufficiently close to a local minimum, the objective function E is expected to be approximated by a quadratic form:

$$E(\theta) \approx E(\theta_{now}) + g^T(\theta - \theta_{now}) + \frac{1}{2}(\theta - \theta_{now})^T H (\theta - \theta_{now}),$$

where H is the Hessian matrix, consisting of the second partial derivatives of $E(\theta)$.

The preceding equation is the Taylor series expansion of $E(\theta)$ up to the second-order terms.

Differentiating the above equation

$$0 = g + H(\hat{\theta} - \theta_{now})$$

where $\hat{\theta}$ is the minimum point.

If the inverse of H exists, we have a unique solution.

When the minimum point $\hat{\theta}$ of the approximated quadratic function is chosen as the next point θ_{now}, we have the so-called newton's method or the Newton-Raphson method.

$$\hat{\theta} = \theta_{now} - H^{-1} g.$$

(20)

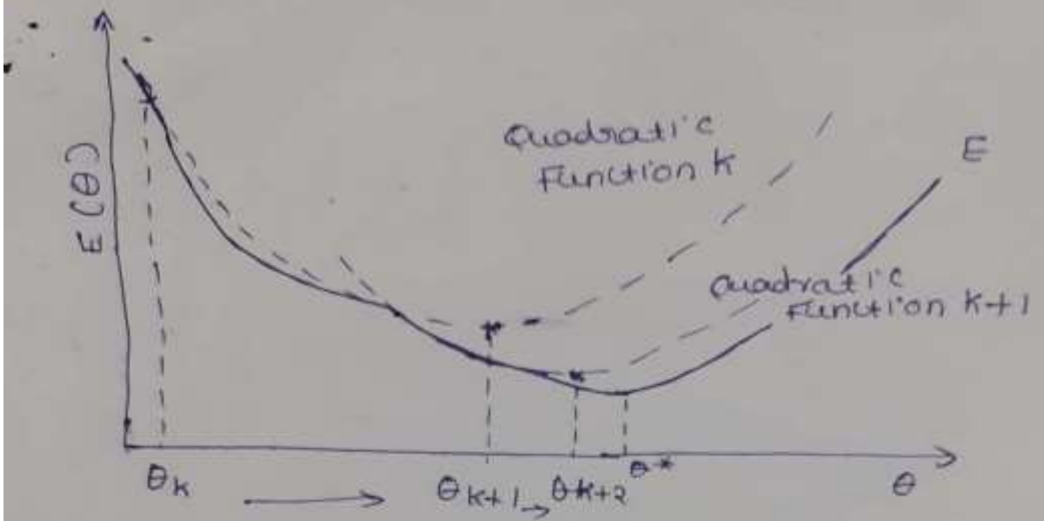


Figure 2.9 Newton's (or Newton-Raphson) method for minimising a general objective function E, which is approximated locally as a quadratic form; this approximate function is minimized exactly.

The step $-H^{-1}g$ is called the Newton step, and its direction is called the Newton direction.

The general gradient-based formula reduces to Newton's method when $G = H^{-1}$ and $\eta = 1$.

If H is positive definite and $E(\theta)$ is quadratic, then Newton's method directly gets to a local minimum in the single Newton step. If $E(\theta)$ is not quadratic, then the minimum may not be reached in a single stride, and Newton's method should be repeatedly employed.

Figure 2.9 illustrates the progress of repeated application of Newton's method to a single-variable objective function.

MR 402

Module III

3.1 STEP SIZE DETERMINATION

The formula of a class of gradient-based descent methods given by

$$\theta_{next} = \theta_{now} + \eta d = \theta_{now} - \eta G g.$$

Where, η is the step size.

The efficiency of the step size determination affects the entire minimization process.

For a general function E,

$$\phi'(\eta) = 0, \quad \text{where } \phi(\eta) = E(\theta_{now} + \eta d).$$

The univariate function $\phi(\eta)$ should be minimized on the line determined by the current point θ_{now} and the direction d. This can be accomplished by line search (or one-dimensional search) methods.

3.1.1 Initial Bracketing

3.1.2 Line searches

3.1.1 Initial Bracketing

The function E is unimodal (single highest value) over the closed interval.

Determining the initial interval in which a relative minimum must lie is of critical importance.

To begin with line searches, some routine must be employed for initially bracketing an assumed minimum into the starting interval.

Initial bracketing is categorized into two schemes:

1

1. A scheme, by function evaluations, for finding the three points to satisfy
$E(\theta_{k-1}) > E(\theta_k) < E(\theta_{k+1})$, $\theta_{k-1} < \theta_k < \theta_{k+1}$.
2. A scheme, by taking the first derivatives, for finding two points to satisfy
$E'(\theta_k) < 0$, $E'(\theta_{k+1}) > 0$, $\theta_k < \theta_{k+1}$.

Algorithm A initial bracketing procedure for searching three points θ_1, θ_2, and θ_3

(1) Given a starting point θ_0 and $h \in R$, let θ_1 be $\theta_0 + h$. Evaluate $E(\theta_1)$.
If $E(\theta_0) \geq E(\theta_1)$, $i \leftarrow 1$,
(i.e., go downhill) go to (2).
Otherwise, $h \leftarrow -h$. (i.e., set backward direction)
(i.e., go uphill) $E(\theta_{-1}) \leftarrow E(\theta_1)$,
$\theta_1 \leftarrow \theta_0 + h$,
$i \leftarrow 0$,
go to (3).

(2) Set the next point by: $h \leftarrow 2h$, $\theta_{i+1} \leftarrow \theta_i + h$.

(3) Evaluate $E(\theta_{i+1})$;
If $E(\theta_i) \geq E(\theta_{i+1})$, $i \leftarrow i+1$,
(i.e., still go downhill) go to 2
Otherwise, Arrange θ_{i-1}, θ_i, and θ_{i+1} in the decreasing order. Then, we obtain the three points: $(\theta_1, \theta_2, \theta_3)$. Stop.

②

3.1.2 Line Searches

The process of determining η^* that minimizes a one-dimensional function $\phi(\eta)$ is achieved by searching on the line for the minimum.

The method of line searches (or one-dimensional searches) is important because higher dimensional problems are ultimately solved by repeating line searches.

Line search include two components

1. sectioning (or bracketing), and
2. polynomial interpolation.

Methods for line search :

a) Newton's method
b) Secant method
c) sectioning methods
d) Polynomial interpolation methods

a) Newton's Method

When $\phi(\eta_k)$, $\phi'(\eta_k)$, and $\phi''(\eta_k)$ are available, the classical Newton method can be applied to solve the equation $\phi'(\eta_k)=0$:

$$\eta_{k+1} = \eta_k - \frac{\phi'(\eta_k)}{\phi''(\eta_k)} \quad \text{— (1)}$$

b) Secant Method

If we use both η_k and η_{k-1} to approximate the second derivative in the above equation (1) and if the first derivatives alone are available

3

then we have an estimated η_{k+1}:

$$\eta_{k+1} = \eta_k - \frac{\phi'(\eta_k)}{\dfrac{\phi'(\eta_k) - \phi'(\eta_{k-1})}{\eta_k - \eta_{k-1}}}$$

This is the so-called method of false position or the secant method.

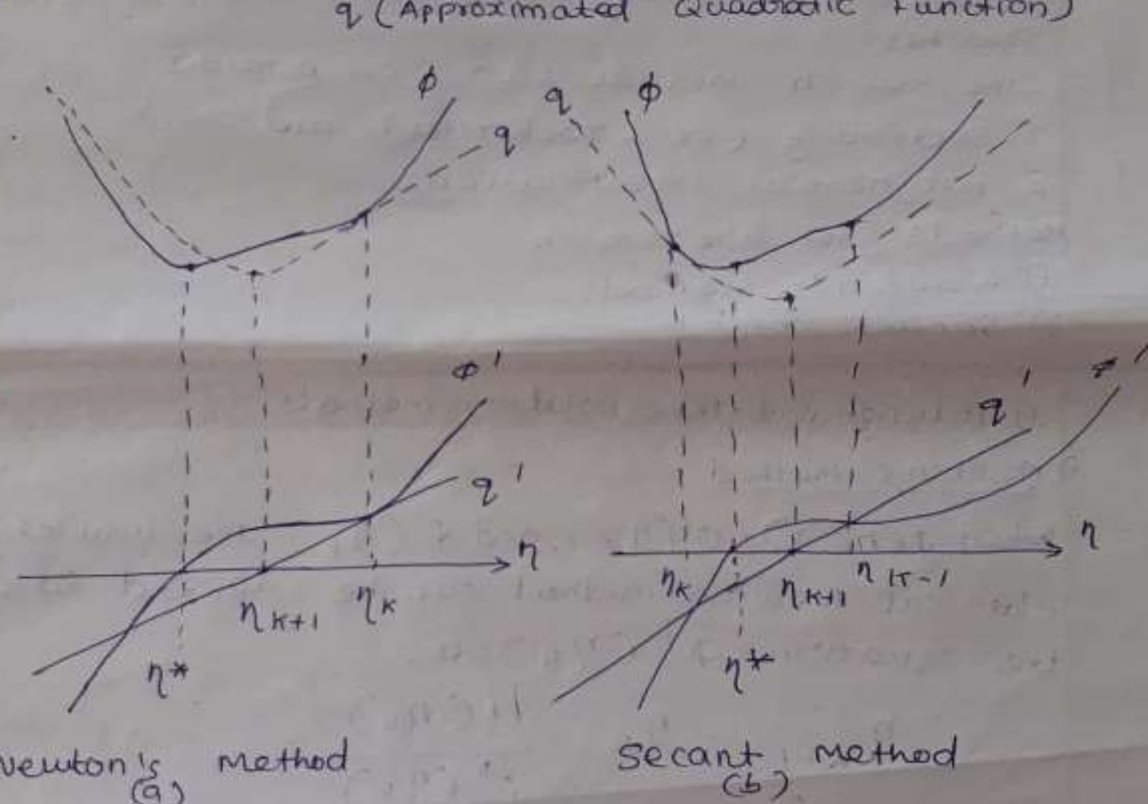


Figure 3.1. Newton's method (left) and the secant method (right) to determine the step size.

c) Sectioning methods

A sectioning algorithm begins with an interval $[a_1, b_1]$ in which the minimum η^* must lie,

(4)

and then reduces the length of the interval at each iteration by evaluating the value of ϕ at a certain number of points. The two endpoints a_1 and b_1 can be found by the initial bracketing.

The bisection method is one of the simplest sectioning methods for solving $\phi'(\eta^*) = 0$, if first derivatives are available.

Algorithm: Bisection method

(1) Given a small value $\varepsilon \in R$ an initial interval with two endpoints a_1 and a_2 such that $a_1 < a_2$ and $\varphi(a_1)\varphi(a_2) < 0$.

$$\eta_{left} \leftarrow a_1$$

$$\eta_{right} \leftarrow a_2$$

(2) Calculate the midpoint η_{mid}; That is

$$\eta_{mid} \leftarrow \frac{(\eta_{right} + \eta_{left})}{2}$$

If $\varphi(\eta_{right})\varphi(\eta_{mid}) < 0$, $\eta_{left} \leftarrow \eta_{mid}$.

otherwise, $\eta_{right} \leftarrow \eta_{mid}$.

(3) Check if $|\eta_{left} - \eta_{right}| < \varepsilon$. If it holds, terminate the algorithm. Otherwise, go to (2).

The bisection method replaces the right or left endpoint by the interval's midpoint based on the function evaluation at the midpoint. The length of the bracketing interval is halved at each iteration.

5

Polynomial Interpolation

Polynomial interpolation methods are based on curve-fitting procedures, which work well when the objective function possesses a certain degree of smoothness.

A quadratic interpolation method constructs a smooth quadratic curve q that passes through three evaluated points, (η_1, ϕ_1), (η_2, ϕ_2), and (η_3, ϕ_3):

$$q(\eta) = \sum_{i=1}^{3} \phi_i \frac{\pi_{j \neq i} (\eta - \eta_j)}{\pi_{j \neq i} (\eta_i - \eta_j)},$$

where $\phi_i \equiv \phi(\eta_i)$, $i = 1, 2, 3$.

The quadratic function has a unique minimum point which can be easily determined by solving $q'(\eta) = 0$.

Hence, the next trial point η_{next} is given by

$$\eta_{next} = \frac{1}{2} \frac{(\eta_2^2 - \eta_3^2)\phi_1 + (\eta_3^2 - \eta_1^2)\phi_2 + (\eta_1^2 - \eta_2^2)\phi_3}{(\eta_2 - \eta_3)\phi_1 + (\eta_3 - \eta_1)\phi_2 + (\eta_1 - \eta_2)\phi_3}.$$

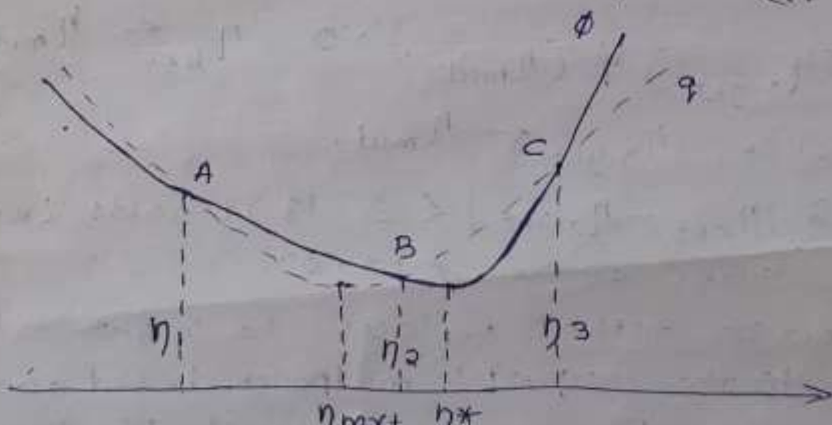


Figure 3.2 Quadratic interpolation.

6

3.2 Derivative-Free Optimization.

Common characteristics of derivative-free optimization methods.

1. Derivative freeness
2. Intuitive guidelines
3. Slowness
4. Flexibility
5. Randomness
6. Analytic opacity
7. Iterative nature

1. Derivative freeness

These methods do not need functional derivative information to search for a set of parameters that minimize a given objective function.
These methods rely exclusively on repeated evaluations of the objective function.

2. Intuitive guidelines

The guidelines followed by the search procedures are based on simple intuitive concepts. Some of these concepts are motivated by so-called nature's wisdom.

3. Slowness

Without using derivatives, these methods are slower than derivative based-optimization methods for continuous optimization problems.

4. Flexibility

By minimizing (or maximizing) a single objective function of this type, we can do structure and parameter identification at the same time.

(7)

5. Randomness

Derivative-free optimization methods use random number generators in determining subsequent search directions.

6. Analytic opacity

It is difficult to do analytic studies of these methods because of their randomness and problem-specific nature.

7. Iterative nature

These techniques are iterative in nature and we need certain stopping criteria to determine when to terminate the optimization process.

The most popular derivative-free optimization methods are:

1. Genetic Algorithms
2. Simulated annealing
3. Random search method, and
4. downhill simplex search.

3.3 GENETIC ALGORITHMS (GAs)

Genetic algorithms are derivative-free stochastic optimization methods based loosely on the concepts of natural selection and evolutionary processes, proposed and investigated by John Holland at the University of Michigan in 1975.

Major components of GAs include encoding schemes, fitness evaluations, parent selection, cross-over operators, and mutation operators.

8

1. Encoding schemes

Encoding schemes provide a way of translating problem-specific knowledge directly into the GA framework, and thus play a key role in determining GAs performance.

Encoding schemes transform points in parameter space into bit string representations.

eg: A point (11, 6, 9) in a three-dimensional parameter space can be represented as a concatenated binary string:

$$\underbrace{1011}_{11}\;\underbrace{0110}_{6}\;\underbrace{1001}_{9}$$

(gene, gene, gene)

2. Fitness evaluation

Mechanism for selecting individuals (strings) for reproduction according to their fitness (objective function value).

The first step after creating a generation is to calculate the fitness value of each member in the population.

we can use the rankings of members in a population as their fitness values.

3. Selection

After evaluation, we have to create a new population from the current generation.

The selection operation determines which parents participate in producing offspring for the next

9

generation, and it is analogous to survival of the fittest in natural selection.

selection probability equal to $f_i / \sum_{k=1}^{k=n} f_k$, where n is the population size.

4. Crossover

Crossover operators are used to generate new chromosomes that we hope will retain good features from the previous generation.

Crossover is usually applied to selected pairs of of parents with a probability equal to a given crossover rate.

Crossover point

100 | 11110 ⇒ 100 | 10010
101 | 10010 ⇒ 101 | 11110

(a)

1 | 0011 | 110 ⇒ 1 | 0110 | 110
1 | 0110 | 010 ⇒ 1 | 0011 | 010

(b)

Figure 3.3 Crossover operators:
(a) One-point crossover; (b) two-point crossover.

One-point crossover is the most basic crossover operator, where a crossover point on the genetic code is selected at random and two

(10)

parent chromosomes are interchanged at this point.

In two-point crossover, two crossover points are selected and the part of the chromosome string between these two points is then swapped to generate two children.

The effect of crossover is similar to that of mating in the natural evolutionary process, in which parents pass segments of their own chromosomes on to their children. Therefore, some children are able to outperform their parents if they get "good" genes or genetic traits from both parents.

5. Mutation

If no amount of gene mixing can produce a satisfactory solution a mutation operator can be used to generate new chromosomes. The most common way of implementing mutation is to flip a bit with a probability equal to a very low given mutation rate.

Mutated Bit.

1 0 0 1 1 1 1 0 ⟹ 1 0 0 1 1 0 1 0

Figure 3.4. Mutation operator.

In the natural evolutionary process, selection, crossover, and mutation all occur in the single act of generating offspring.

Figure 3.5

(11)

A simple genetic algorithm for maximization problem is

Step 1 :

Initialize a population with randomly generated individuals and evaluate the fitness value of each individual.

Step 2 :

(a) Select two members from the population with probabilities proportional to their fitness values

(b) Apply crossover with a probability equal to the crossover rate.

(c) Apply mutation with a probability equal to the mutation rate.

(d) Repeat (a) to (d) until enough members are generated to form the next generation.

Step (3)

Repeat steps 2 and 3 until a stopping criterion is met.

We may choose a policy of always keeping a certain number of best members when each new population is generated; this principle is usually called elitism.

12

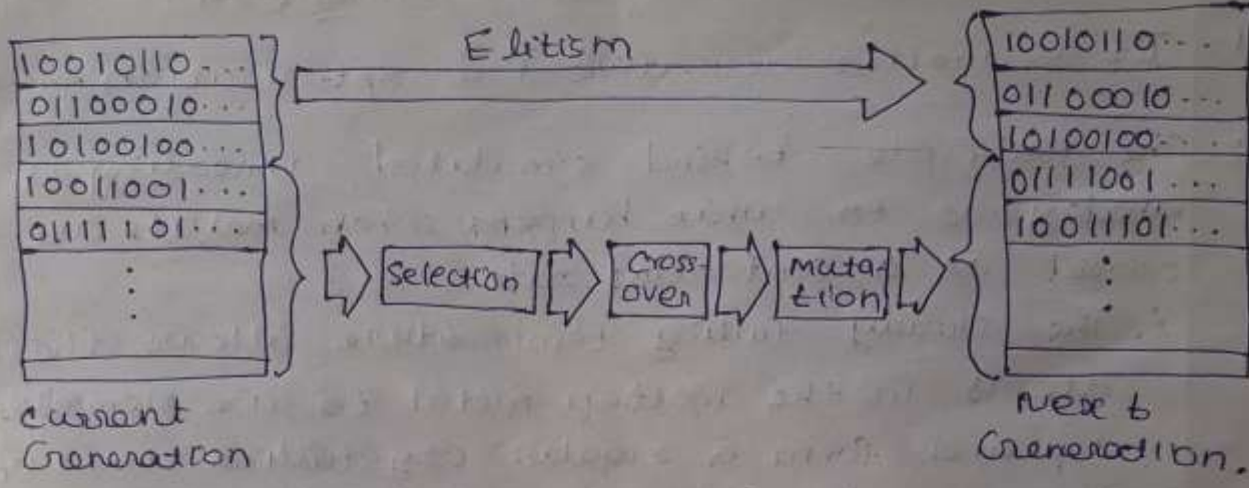


Figure 3.5. Producing the next generation in GAs.

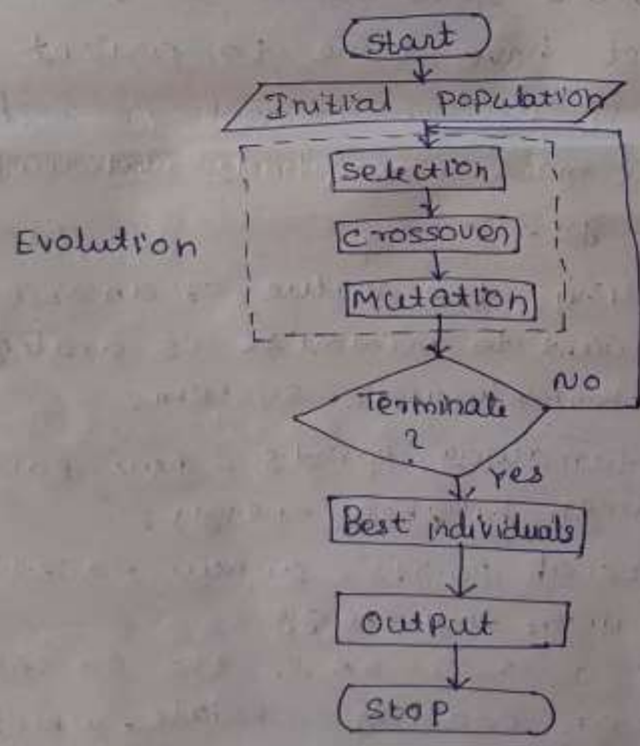


Figure 3.6 Flowchart for genetic algorithm.

(13)

SIMULATED ANNEALING (SA)

1. It is another derivative-free optimization method.
2. The principle behind simulated annealing is analogous to what happens when metals are cooled at a controlled rate.
   i. The slowly falling temperature allows the atoms in the molten metal to line themselves up and form a regular crystalline structure that has high density and low energy.
   ii. If the temperature goes down too quickly, the atoms do not have time to orient themselves into a regular structure and the result is a more amorphous material with higher energy.
3. In simulated annealing, the value of an objective function that we want to minimise is analogous to the energy in a thermodynamic system.

   High temperature – far away points – new point with higher energy.

   Low temperature – Local points – new point with low energy.
4. The most important part of SA is the so-called annealing schedule or cooling schedule, which specifies how rapidly the temperature is lowered from high to low values.

(14)

Algorithm:

The basic steps involved in a general SA method.

Step1 :

choose a start point x and set a high starting temperature T. Set the iteration count k to 1.

Step2 :

Evaluate the objective function :

$$E = f(x).$$

(An objective function $f(\cdot)$ maps an input vector x into a scalar E)

(The task of SA is to sample the input space effectively to find an x that minimizes E.)

Step3 :

Select Δx with probability determined by the generating function $g(\Delta x, T)$

($\Delta x = x_{new} - x$ and T is the temperature)

Set the new point x_{new} equal to $x + \Delta x$.

Step 4 :

Calculate the new value of the objective function :

$$E_{new} = f(x_{new}).$$

Step5 :

Set x to x_{new} and E to E_{new} with probability determined by the acceptance function $h(\Delta E, T)$, where $\Delta E = E_{new} - E$.

(After a new point x_{new} has been evaluated, SA decides whether to accept or reject it based on the value of an acceptance function $h(\cdot,\cdot)$. eg : $h(\Delta E, T) = \frac{1}{1 + \exp(\Delta E/(cT))}$)

15

where c is a system - dependent constant,
T is the temperature and
ΔE is the energy difference between x_{new} and x.
($\Delta E = f(x_{new}) - f(x)$).

Step 6:

Reduce the temperature T according to the annealing schedule. ($T = \eta T$, when η is between 0 & 1).
(An annealing schedule regulates how rapidly the temperature T goes from high to low values, as a function of time or iteration counts).

Step 7:

Increment iteration count K. If k reaches the maximum iteration count, stop the iterating. otherwise, go back to step 3.

Example: Traveling Salesman problem. (TSP)

In a typical traveling salesperson problem (TSP), we are given n cities, and the distance (or cost) between all pairs of these cities is an n×n distance (or cost) matrix D, where the element d_{ij} represents the distance (or cost) of traveling from city i to city j.

The problem is to find a closed tour in which each city, except for the starting one is visited exactly once, such that the total length (cost) is minimised.

For a common traveling salesperson problem we need three more sets for SA.

16

1. Inversion
2. Translation
3. Switching

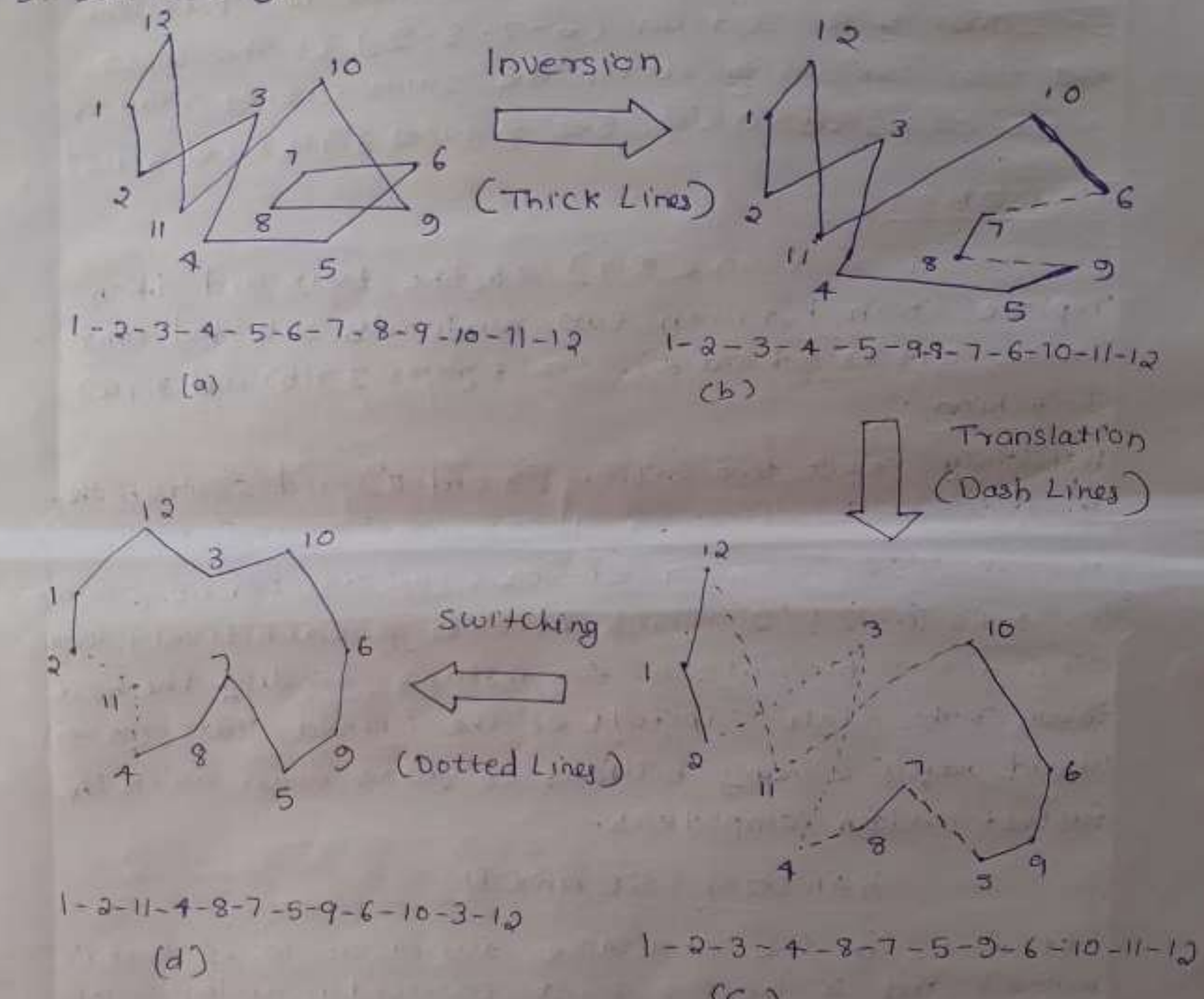


Figure 3.7 Three operations for generating move sets in the traveling salesperson problem.

17

Inversion

Remove two edges from the tour and replace them to make it another legal tour. This is equivalent to removing a section (6-7-8-9) of the tour and then replacing with the same cities running in the opposite order. See figures 3.7(a) and 3.7(b)

Translation

Remove a section (8-7) of the tour and then replace it in between two randomly selected consecutive cities (4 and 5). See Figures 3.7(b) and 3.7(c).

Switching.

Randomly select two cities (3 and 11) and switch them in a tour. See Figures 3.7 (c) and figures 3.7(d).

For a TSP with n cities, the number of possible tours is $(n-1)!/2$, which becomes prohibitively large even for a moderate n. For instance, finding the best tour of the state capitals of the United States ($n=50$) would require many billions of years even with the fastest modern computers.

RANDOM SEARCH

Let $f(x)$ be the objective function to be minimized and x be the point currently under consideration. The original random search method tries to find the optimal x by iterating the following four steps:

Step1: Choose a start point x as the current point.

Step2: Add a random vector dx to the current

(18)

point x in the parameter space and evaluate the objective function at the new point at $x+dx$.

Step3:

If $f(x+dx) < f(x)$, set the current point x equal to $x+dx$.

Step4:

Stop if the maximum number of function evaluation is reached. Otherwise, go back to step 2 to find a new point.

This is a truly random method in the sense that search directions are purely guided by a random number generator.

Modified random search method involves the following six steps:

Step1:

Choose a start point x as the current point. Set initial bias b equal to a zero vecter.

Step2:

Add a bias term b and a random vector dx to the current point x in the input space and evaluate the objective function at the new point at $x+b+dx$.

Step3:

If $f(x+b+dx) < f(x)$, set the current point x equal to $x+b+dx$ and the bias b equal to $0.2b + 0.4dx$; goto step 6. Otherwise, go to the next step.

19

step4:
If $f(x+b-dx) < f(x)$, set the current point x equal to $x+b-dx$ and the bias b equal to $b-0.4dx$, go to step6. Otherwise, go to the next step.

step 5:
Set the bias equal to $0.5b$ and go to step 6.

step6:
stop if the maximum number of function evaluations is reached. Otherwise go back to step 2 to find a new point.

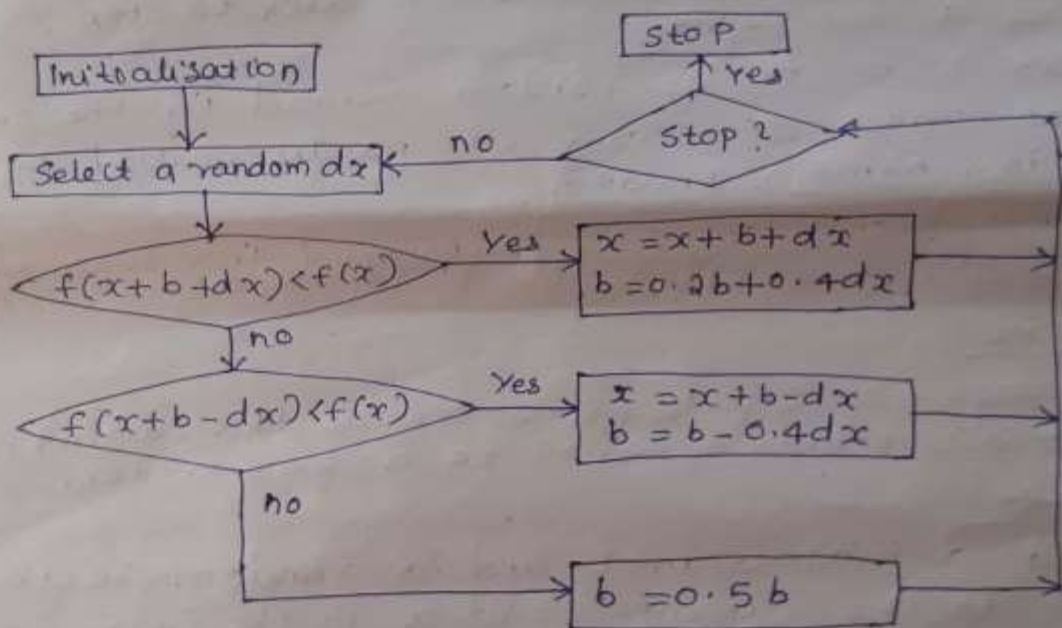


Figure 3.8 Flow Chart for the random search method.

Usually the initial bias is set to a zero vector. Each component of the random vector dx should be a random variable that has a zero mean and a variance proportional to the range of the corresponding parameter.

(20)

DOWNHILL SIMPLEX SEARCH

A simplex is a triangle in two-dimensional space and tetrahedron in three-dimensional space.
An easy way to set up a simplex is to start with an initial starting point P_0 and the other n points can be taken as

1) $$P_i = P_0 + \lambda_i e_i , \; i = 1, \dots n,$$

where e_i's are unit vectors

λ_i is a constant reflecting the guess of the characteristic length scale of the optimisation problem.

2) Let $\bar{P}$ be the average of these $n+1$ points.

3) Reflection:

Define the reflection point P^* and its value y^* as

$$P^* = \bar{P} + \alpha(\bar{P} - P_h),$$
$$y^* = f(P^*),$$

where the reflection coefficient α is a positive constant.

4) Expansion:

Define the expansion point P^{**} and its value y^{**} as

$$P^{**} = \bar{P} + \gamma(P^* - \bar{P}),$$
$$y^{**} = f(P^{**}),$$

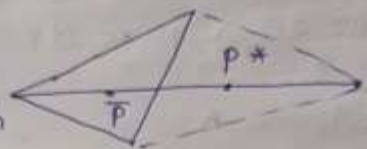
where the expansion coefficient γ is greater than unity.

21

(a) P_h

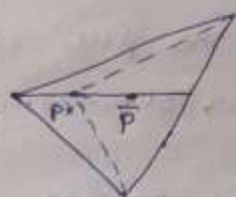
$P^* = \bar{P} + \alpha(\bar{P} - P_h)$

(b) P_h



$P^{**} = \bar{P} + \gamma(P^* - \bar{P})$

(c) P_h



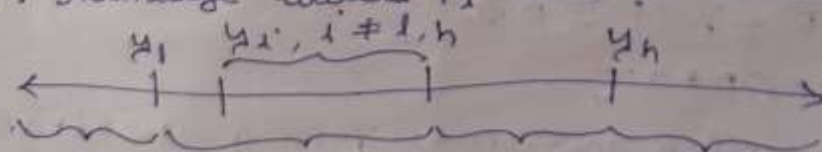
$P^* = \bar{P} + \beta(P_h - \bar{P})$

(d) P_h

$P_i = (P_i + P_1)/2$

P_1

Figure 3.9 Outcomes for a cycle in the downhill simplex search after
(a) reflection away from P_h;
(b) reflection and expansion away from P_h;
(c) contraction along one dimension connecting P_h and $\bar{P}$;
(d) Shrinkage toward P_1 alone all dimensions.



$l \rightarrow$ low
$h \rightarrow$ high

Figure 3.10 Four intervals used in the downhill simplex search.

22

5) Contraction:

Define the contraction point p^{**} and its value y^{**} as

$$p^{**} = \bar{P} + \beta(P_h - \bar{P}),$$

$$y^{**} = f(P^{**}),$$

where the contraction coefficient β lies between 0 and 1.

6) Shrinkage:

Replace each P_i with $(P_i + P_l)/2$. Finish this cycle.

l and h are respectively the indices for the minimum and maximum of y_i.

$$y_l = \min_i (y_i)$$

$$y_h = \max_i (y_i)$$

Depending on the value of y^*

1. If y^* is in interval 1, go to expansion.
2. If y^* is in interval 2, replace P_h with P^* and finish this cycle.
3. If y^* is in interval 3, replace P_h with P^* and go to contraction.
4. If y^* is in interval 4, go to contraction.

Depending on the value of y^{**}

1. If y^{**} is in interval 1, replace P_h with P^{**} and finish this cycle. Otherwise, replace P_h with the original reflection point P^* and finish this cycle.

23

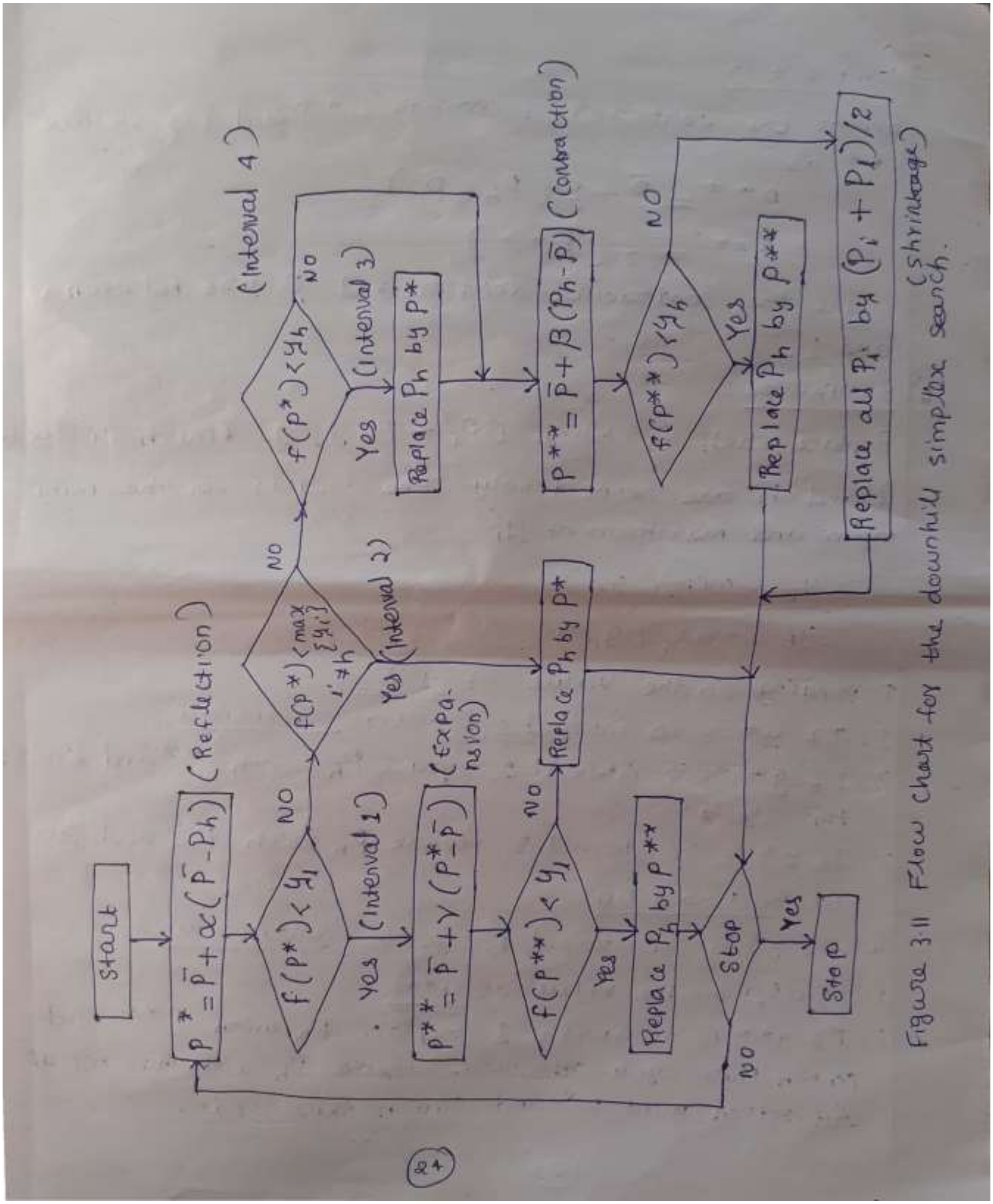


Figure 3.11 Flow chart for the downhill simplex search.

Before starting using this method, we need to determine three constants α, β and γ, which are the coefficients for reflection, contraction, and expansion.

SUPERVISED LEARNING NEURAL NETWORKS

Artificial neural networks, or simply neural networks (NNs) models can be classified according to various criteria such as their

1. learning methods (supervised versus unsupervised)
2. architectures (feedforward versus recurrent),
3. output types (binary versus continuous),
4. node types (uniform versus hybrid),
5. Implementations (software versus hardware),
6. connection weights (adjustable versus hardwired)
7. operations (biologically motivated versus psychologically motivated), and so on.

Modeling problems with desired input-output data sets so the resulting networks must have adjustable parameters that are updated by a supervised learning rule. Such networks are referred to as supervised learning or mapping networks. Since the network shapes the input-output mappings of the networks according to a given training data set.

(25)

PERCEPTRONS

1. Architecture and Learning Rule
2. Exclusive-OR Problem.

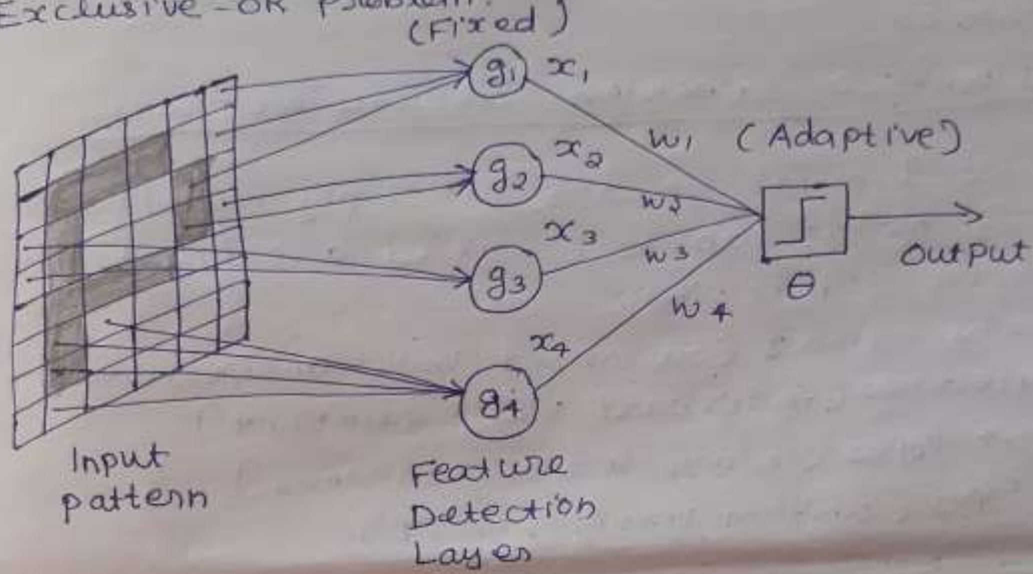


Figure 3.12 The perceptron.

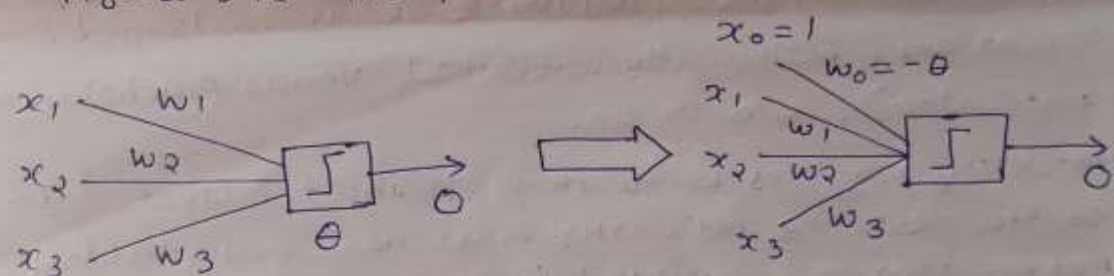


Figure 3.13, Introduction of the bias connection weight; the bias term $w_0 (= -\theta)$ can be viewed as the connection weight between the output unit and a "dummy" incoming signal x_0 that is always equal to 1.

1. Architecture and Learning Rule.

The perceptron represents one of the early attempts

(26)

to build intelligent and self-learning systems using simple components. It was derived from a biological brain neuron model introduced by McCulloch and Pitts in 1943.

Figure 3.12 is a typical perceptron setup for pattern-recognition application, in which visual patterns are represented as matrices of elements between 0 and 1.

First layer:

The first layer of the perceptron acts as a set of "feature detectors" that are hardwired to the input signals to detect specific features.

second layer

The second (output) layer takes the outputs of the "feature detectors" in the first layer and classifies the given input pattern.

Learning

Learning is initiated by making adjustments to the relevant connection strengths (i.e weights w_i) and a threshold value θ.

g_i

Each function g_i in layer 1 is a fixed function that has to be determined a priori; it maps all or a part of the input pattern into a binary value $x_i \in \{-1, 1\}$ or a bipolar value $x_i \in \{0, 1\}$.

x_i

The term x_i is referred to as active or excitatory if its value is 1, inactive if

27

its value is 0, and inhibitory if its value is -1.

output

The output is a linear threshold element with a threshold value θ.

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

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

$$= f\left(\sum_{i=0}^{n} w_i x_i\right), \quad x_0 = 1$$

$f(\cdot)$ is the activation function of the perceptron. Activation function of the is typically either a signum function $sgn(x)$ or step function $step(x)$:

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

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

A single-layer perceptron (Figure 3.12) repeats the following steps until the weights coverage.

1. Select an input vector x from the training data set.
2. If the perception 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 rate.

98

2. Exclusive - OR Problem

i. The simplest and most well-known pattern recognition problem in neural network literature is the exclusive-OR (XOR) problem.

ii. The task is to classify a binary input vector to class 0 if the vector has an even number of 1's, or assign it to class 1.

iii. For a two-input binary XOR Problem, the desired behavior is regulated by a truth table

	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

To construct a straight line to partition the two-dimensional input space into two regions each containing only data points of the same class.

Using a single-layer perceptron to solve this problem requires satisfying the following four inequalities:

$$0 \times w_1 + 0 \times w_2 + w_0 \leq 0 \iff w_0 \leq 0,$$
$$0 \times w_1 + 1 \times w_2 + w_0 > 0 \iff w_0 > -w_2,$$
$$1 \times w_1 + 0 \times w_2 + w_0 > 0 \iff w_0 > -w_1,$$
$$1 \times w_1 + 1 \times w_2 + w_0 \leq 0 \iff w_0 \leq -w_1 - w_2.$$

(29)

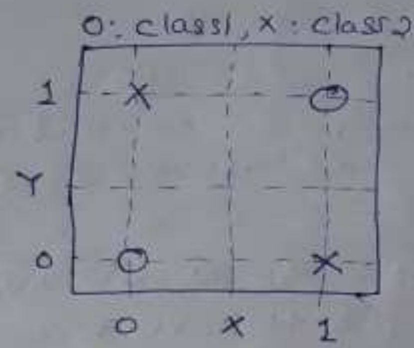


Figure 3.14 XOR problem

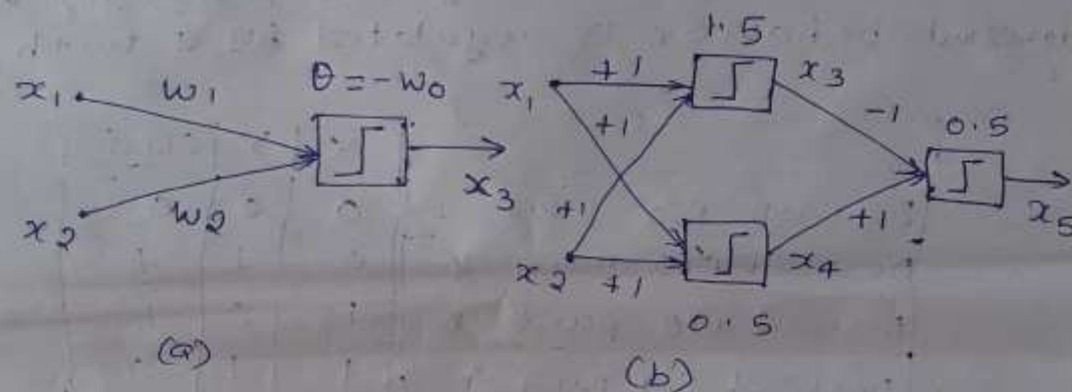


Figure 3.15 Perceptrons for the two-input exclusive-OR problem:
(a.) the single-layer perceptron, and
(b.) the two-layer perceptron.
Both use the step function as the activation function for each node.

The above set of inequalities is self-contradictory when considered as a whole.
It is possible to solve the problem with two-layer perceptron illustrated in Figure 3.15(b) in which the connection weights and thresholds

(30)

are indicated.

ADALINE

The adaptive linear element (or Adaline), suggested by Widrow and Hoff, is the simplest intelligent self-learning system that can adapt itself to achieve a given modeling task.

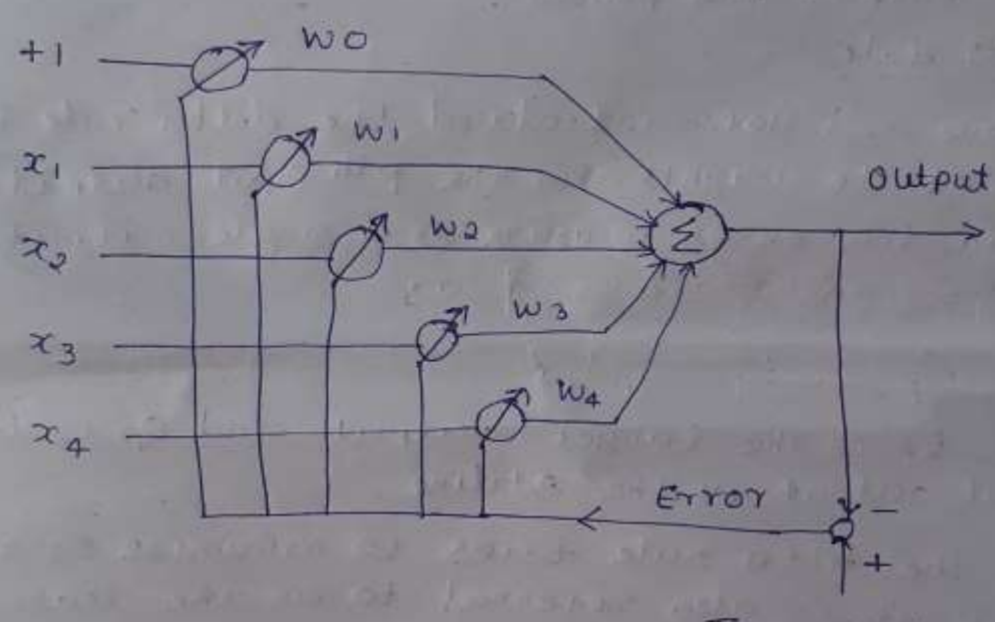


Figure 3.16 Adaline (adaptive linear element).

Figure 3.16 is a schematic diagram for such a network. It has a purely linear output unit; hence the network output o is a weighted linear combination of the inputs plus a constant term:

$$o = \sum_{i=1}^{n} w_i x_i + w_0 .$$

(31)

In a simple physical implementation, the input signals x_i are voltages and the w_i are conductances of controllable resistors; the network's output is the summation of the currents caused by the input voltages. The problem is to find a suitable set of conductances (or weights) such that the input-output behavior of the Adaline is close to a set of desired input-output data points.

Delta Rule

Widrow and Hoff introduced the delta rule for adjusting the weights. For the p^{th} input-output pattern, the error measure of a single-output Adaline can be expressed as

$$E_p = (t_p - o_p)^2,$$

where t_p is the target output and o_p is the actual output of the Adaline.

Since the delta rule tries to minimize squared errors, it is also referred to as the least mean square (LMS) learning procedure or Widrow-Hoff learning rule.

The features of the delta rule are as follows:

1. Simplicity
2. Distributed learning
   Learning is not reliant on central control of the network; it can be performed locally at each node level;
3. On-line (or pattern-by-pattern) learning: weights are updated after presentation of each pattern.

(32)

Training Algorithm for Adaline

The Adaline networks training algorithm is as follows:

step 0: weights and bias are set to some random values but not zero. Set the learning rate parameter, α.

step 1: Perform steps 2-6 when stopping condition is false.

step 2: perform steps 3-5 for each bipolar training pair s:t.

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

$$x_i = s_i.$$

step 4: Calculate the net input to the output unit

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

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

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

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

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

33

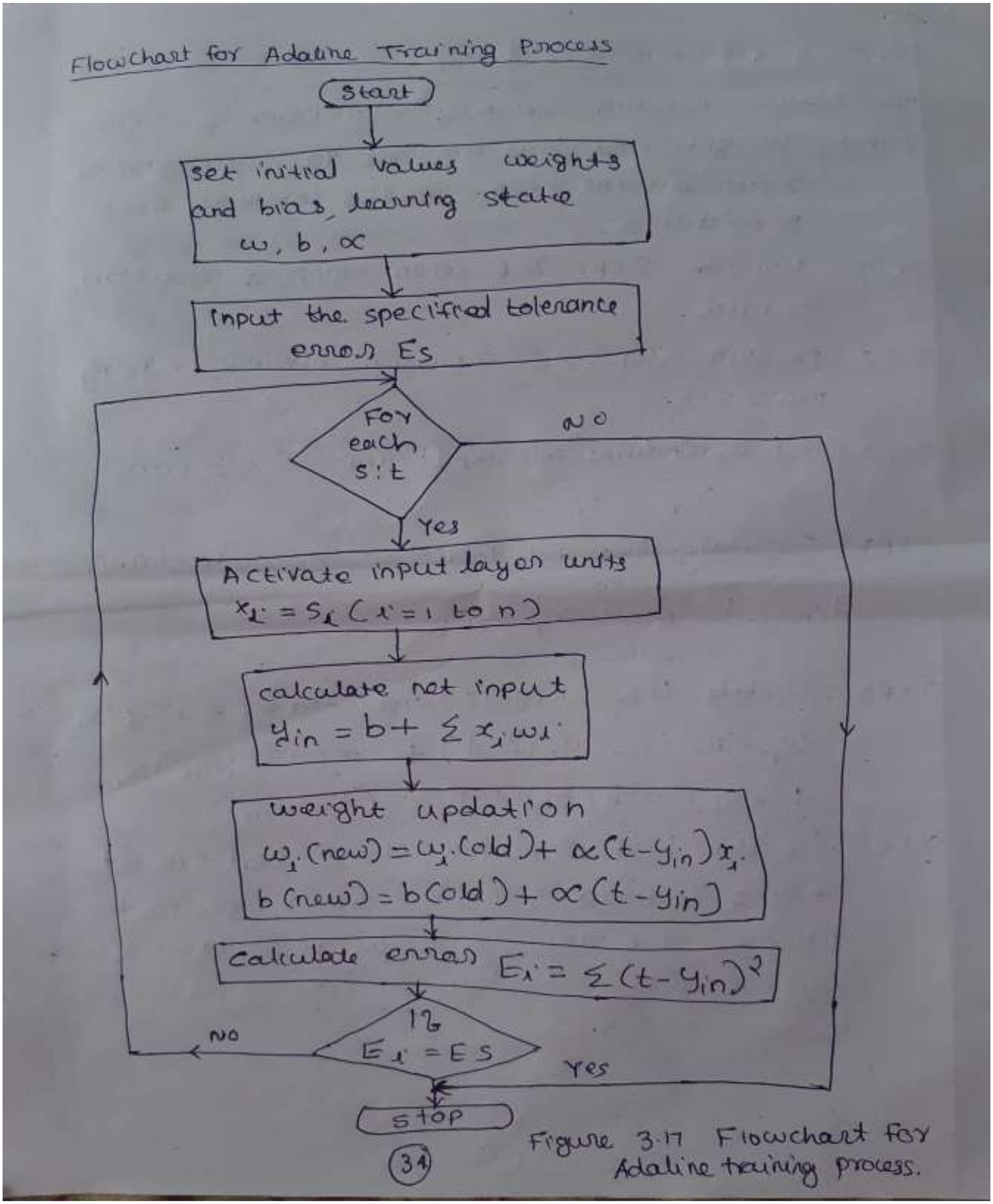


Figure 3.17 Flowchart for Adaline training process.

The flowchart for the training process is shown in figure 3.17. This gives a pictorial representation of the network training. The conditions necessary for weight adjustments have to be checked carefully. The weights and other required parameters are initialized. Then the net input is calculated, output is obtained and compared with the desired output for calculation of error. On the basis of the error factor weights are adjusted.

Testing the Adaline Algorithm

It is essential to perform the testing of a network that has been trained. When training is completed, the Adaline can be used to classify input patterns. A step function is used to test the performance of the network. The testing procedure for the Adaline network is as follows :

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

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

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

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

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

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

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

35

Back Propagation Multilayer perceptrons

Adaptive networks

An adaptive network (Figure 3.18) is a network structure whose overall input-output behavior is determined by a collection of modifiable parameters.

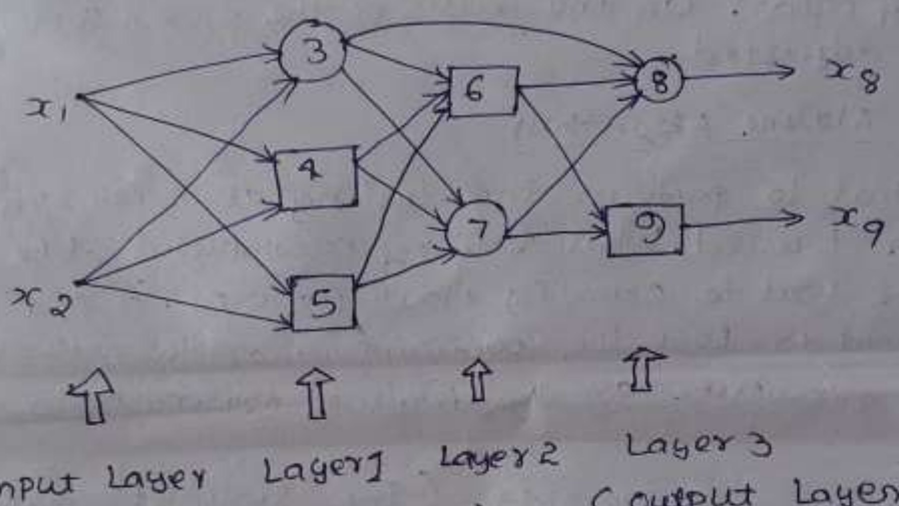


Figure 3.18 A feedforward adaptive network in layered representation.

The configuration of an adaptive network is composed of a set of nodes connected by directed links, where each node performs a static node function on its incoming signals to generate a single node output and each link specifies the direction of signal flow from one node to another.

The parameters of an adaptive network are distributed into its nodes, so each node has a local parameter set.

36

If a node's parameter set is not empty, then its node function depends on the parameter values; we use a square to represent this kind of adaptive node.

If a node has an empty parameter set, then its function is fixed; a circle is used to denote this type of fixed node.

Each adaptive node can be decomposed into a fixed node plus one or several parameter nodes.

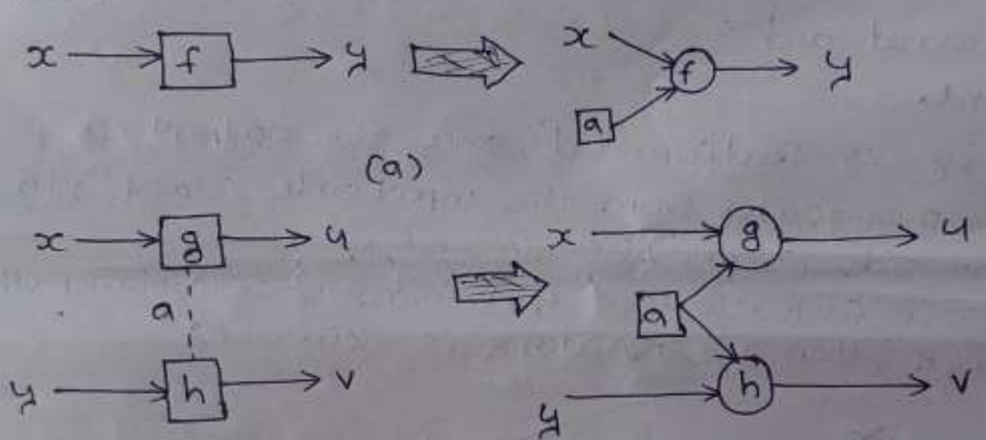


Figure 3.19 Decomposition of adaptive nodes.
(a) a single node; (b) parameter sharing problem.

Figure 3.19(a) shows an adaptive network with only one node, which can be represented as $y = f(x,a)$ where x and y are the input and output, respectively and a is the parameter of the node. An equivalent representation is to move the parameter out of the node and put it into a parameter node.

(37)

The parameter node is useful in solving certain representation problems, such as the parameter sharing problem in Figure 3.19(b), where two adaptive nodes $u = g(x, a)$ and $v = h(y, a)$ share the same parameter a, as denoted by the dotted line linking these two nodes.

Adaptive networks are generally classified into two categories on the basis of the type of connections they have:

1. Feedforward and
2. Recurrent.

Figure 3.18 is feedforward, since the output of each node propagates from the input side (left) to the output side (right) unanimously.

If there is a feedback link that forms a circular path in a network, then the network is recurrent.

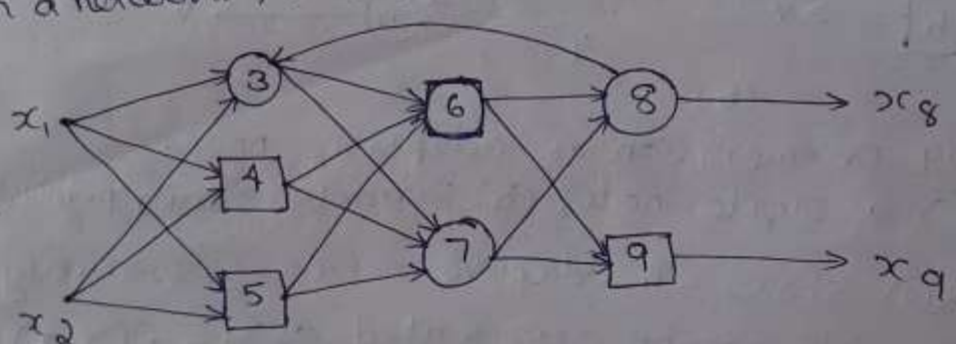


Figure 3.20. A recurrent adaptive network.

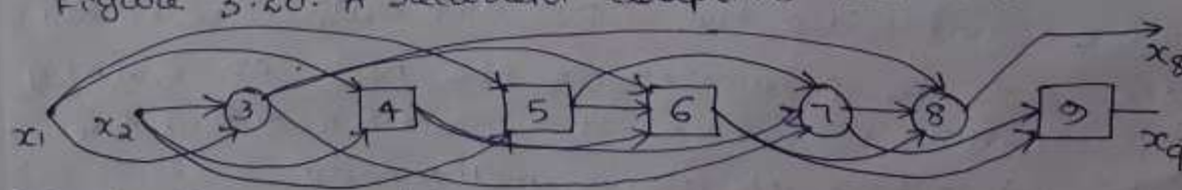


Figure 3.21. A feedforward adaptive network in topological ordering representation.

(38)

Example: An adaptive network with a single linear node

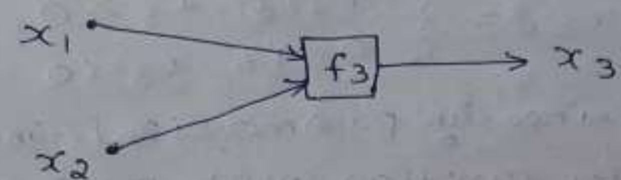


Figure 3.22. A linear single-node adaptive network.

Figure 3.22 is an adaptive network with a single node specified by

$$x_3 = f_3(x_1, x_2 ; a_1, a_2, a_3) = a_1 x_1 + a_2 x_2 + a_3.$$

where x_1 and x_2 are inputs and a_1, a_2, and a_3 are modifiable parameters.

Example: <u>Perceptron network</u>

If we add another node to let the output of the adaptive network in Figure 3.22 have only two values 0 and 1; then the nonlinear network shown in Figure 3.23 is obtained.

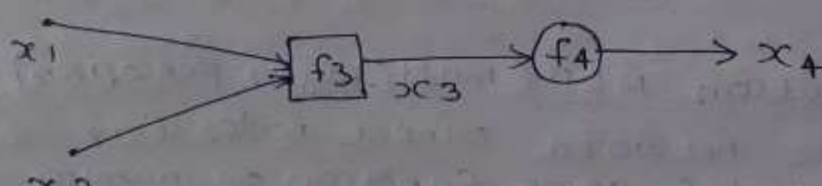


Figure 3.23 A nonlinear single-node - Adaptive network.

The node outputs are expressed as

$$x_3 = f_3(x_1, x_2; a_1, a_2, a_3) = a_1x_1 + a_2x_2 + a_3,$$

and $x_4 = f_4(x_3) = \begin{cases} 1 & \text{if } x_3 \geq 0 \\ 0 & \text{if } x_3 < 0 \end{cases}$.

where f_3 is a linearly parameterized function & f_4 is a step function which maps x_3 to either 0 or 1.

The overall function of this network can be viewed as a linear classifier

Backpropagation

The procedure of finding a gradient vector in a 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.

Backpropagation learning rule

If the gradient vector is used in a simple steepest descent method, the resulting learning paradigm is often referred to as the backpropagation learning rule.

A backpropagation MLP (multilayer perceptrons) is an adaptive network whose nodes (or neurons) perform the same function on incoming signals; the most commonly used activation Figure 2.24 functions in backpropagation MLPs:

(10)

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: $f(x) = x$.

Backpropagation MLPs are by far the most commonly used Neural Network (NN) structures for applications in a wide range of areas, such as pattern recognition, signal processing, data compression, and automatic control.

Backpropagation Learning Rule

The net input $\bar{x}$ of a node is defined as the weighted sum of the incoming signals plus a bias term. For instance, the net input and output of node j in Figure 3.24 are

$$\bar{x}_j = \sum_i w_{ij} x_i + w_j,$$

$$x_j = f(\bar{x}_j) = \frac{1}{1+\exp(-\bar{x}_j)},$$

where x_i is the output of node i located in any one of the previous layers, w_{ij} is the weight associated with the link connecting nodes i and j, and w_j is the bias of node j.

Figure 3.25 shows a two-layer backpropagation MLP with three inputs to the input layer.

(41)

three neurons in the hidden layer, and two output neurons in the output layer.

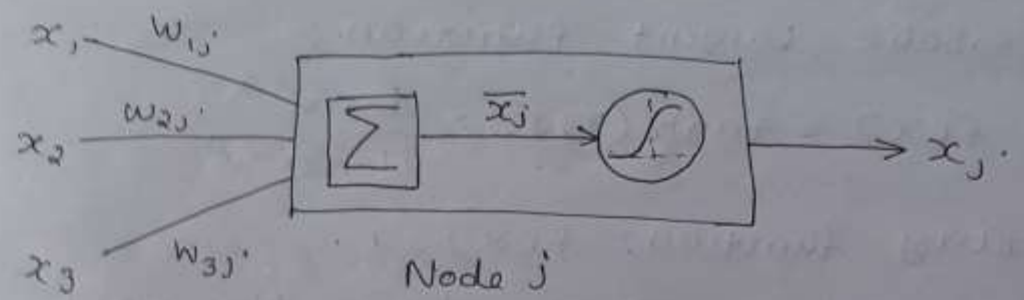


Figure 3.24 Node j of a backpropagation MLP.

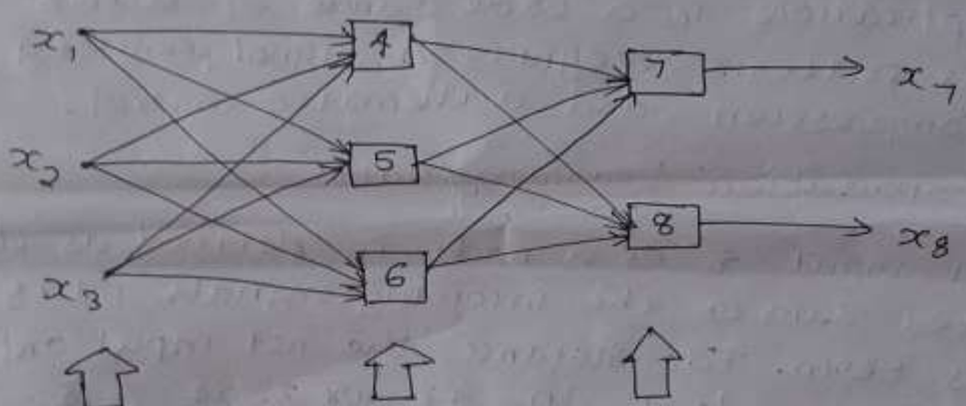


Figure 3.25 A 3-3-2 backpropagation MLP.

The backward error propagation or back-propagation (BP) or the generalised delta rule (GDR) is.

1. A squared error measure for the p-th input-output pair is defined as $\boxed{E_p = \sum_k (d_k - x_k)^2}$,

(4-2)

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 pth data pair is presented.

2. To find the gradient vector, an error term $\bar{\epsilon}_i$ for node i is defined as

$$\bar{\epsilon}_i = \frac{\partial^+ E_p}{\partial \bar{x}_i}$$

By the chain rule, the recursive formula for $\bar{\epsilon}_i$ can be written as

$$\bar{\epsilon}_i = \begin{cases} -2(d_i - x_i)\dfrac{\partial x_i}{\partial \bar{x}_i} = -2(d_i - x_i)x_i(1-x_i) & \text{if node } i \text{ is a output node} \\ \dfrac{\partial x_i}{\partial \bar{x}_i} = \sum_{j,i<j} \dfrac{\partial^+ E_p}{\partial \bar{x}_j}\dfrac{\partial \bar{x}_j}{\partial x_i} = x_i(1-x_i)\sum_{j,i<j} \bar{\epsilon}_j \, w_{ij} & \text{otherwise,} \end{cases}$$

where w_{ij} is the connection weight from node i to j; and w_{ij} is zero if there is no direct connection.

3. The weight update w_{ki} for on-line (pattern-by-pattern) learning is

$$\Delta w_{ki} = -\eta \frac{\partial^+ E_p}{\partial w_{ki}} = -\eta \frac{\partial^+ E_p}{\partial \bar{x}_i}\frac{\partial \bar{x}_i}{\partial w_{ki}} = -\eta \, \bar{\epsilon}_i \, x_k,$$

where η is a learning rate that affects the convergence speed and stability of the weights during learning.

4. For off-line (batch) learning, the connection weight w_{ki} is updated only after presentation of the

(43)

entire data set

$$\Delta w_{ki} = -\eta \frac{\partial^+ E}{\partial w_{ki}} = -\eta \sum_p \frac{\partial^+ E_p}{\partial w_{ki}},$$

or, in vector form,

$$\Delta w = -\eta \frac{\partial^+ E}{\partial w} = -\eta \nabla_w E,$$

where $E = \Sigma_p E_p$.

Methods of Speeding Up MLP Training.

1. One way to speed up off-line training is to use the so-called momentum term

$$\Delta w = -\eta \nabla_w E + \alpha \Delta w_{prev},$$

where w_{prev} is the previous update amount, and the momentum constant α (b/w 0.1 and 1).
The use of momentum terms does not always seem to speed up training; it is more or less application dependent.

2. Another useful technique to speedup is normalized weight updating:

$$\Delta w = -k \frac{\nabla_w E}{\|\nabla_w E\|}$$

This causes the network's weight vector to move the same Euclidean distance k in the weight space with each update, which allows control of the distance k based on the history of error measures.

(44)

3. Other methods for speeding up MLP back-propagation training include
   a) the quick-propagation algorithm,
   b) the delta-bar-delta approach,
   c) the extended Kalman filter method,
   d) second-order optimization, and
   e) the optimal filtering approach.
4. Data scaling on the raw training data and then use the processed data to train the MLP.
5. In output scaling, the range of target values is constrained to remain within the range of the sigmoidal activation function.
6. In input scaling, the range of each input is scaled to the range of the activation function used.

MLP's Approximation Power

The approximation power of back propagation MLPs has been explored by some researchers. Yet there is very little theoretical guidance for determining network size in terms of the number of hidden nodes and hidden layers it should contain.

(45)

MR402 Module IV

RADIAL BASIS FUNCTION NETWORKS

A radial basis function is a real_valued function whose value depends only on the distance between the input and some fixed point, either the origine or center.

Locally tuned and overlapping receptive fields are well-known structures that have been studied in regions of the cerebral cortex, the visual cortex, and others.

Drawing on knowledge of biological receptive fields, Moody and Darken proposed a network structure that employs local receptive fields to perform function mappings. The network structure is called the radial basis function network or RBFN.

ARCHITECTRURE

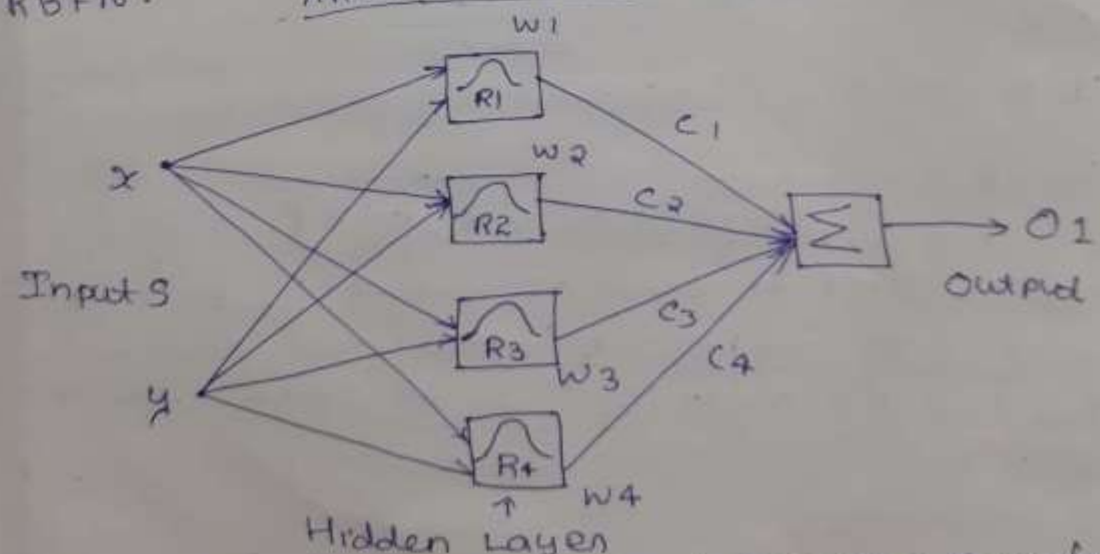


Figure 4.1 Four RBFNs that possess four basis functions - a single-output RBFN that uses weighted sum.

①

1. Figure 4.1 shows a schematic diagram of an RBFN with four receptive field units; the activation level of the ith receptive field unit (or hidden unit) is

$$w_i = R_i(x) = R_i(\|x - u_i\| / \sigma_i), \quad i = 1, 2, \cdots, H,$$

where x is a multidimensional input vector,
u_i is a vector with the same dimension as x,
H is the number of radial basis functions
$R_i(\cdot)$ is the ith radial basis function with a single maximum at the origin.

2. There are no connection weights between the input layer and the hidden layer.

3. Typically, $R_i(\cdot)$ is a Gaussian function.

$$R_i(x) = \exp\left(-\frac{\|x - u_i\|^2}{2\sigma_i^2}\right)$$

Or a logistic function:

$$R_i(x) = \frac{1}{1 + \exp[\|x - u_i\|^2 / \sigma_i^2]}$$

4. The activation level of radial basis function w_i computed by the ith hidden unit is maximum when the input vector x is at the center u_i of that unit.

⑧

The output of an RBFN

The output of an RBFN can be computed in two ways.

1. The simpler method

As shown in Figure 4.1, the final output is the weighted sum of the output value associated with each receptive field:

$$d(x) = \sum_{i=1}^{H} c_i w_i = \sum_{i=1}^{H} c_i R_i(x), \qquad (4.1)$$

where c_i is the output value associated with the ith receptive field, or c_i is the connection weight between the receptive field i and the output unit.

2. Complicated Method

A more complicated method for calculating the overall output is to take the weighted average of the output associated with each receptive field:

$$d(x) = \frac{\sum_{i=1}^{H} c_i w_i}{\sum_{i=1}^{H} w_i} = \frac{\sum_{i=1}^{H} c_i R_i(x)}{\sum_{i=1}^{H} R_i(x)} \qquad (4.2)$$

Weighted average has a higher degree of computational complexity, but it is advantageous in the areas of overlap between two or more receptive fields.

For representation purposes, if the radial basis function $R_i(x)$ is changed in each node of layer 2

(3)

in Figure 4.1 to its normalised counterpart $R_i(x) / \Sigma_i R_i(x)$ then the overall output is specified by the above equation.

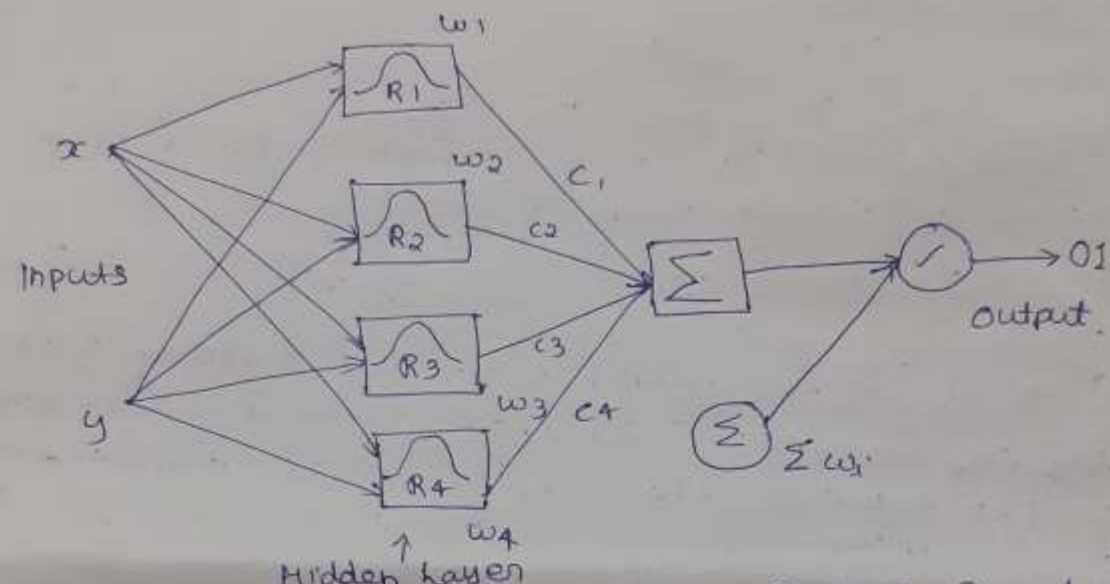


Figure 4.2 Four RBFNs that possess four basis functions. single-output RBFN that uses weighted average.

A more explicit representation is shown in Figure 4.2, where the division of the weighted sum $(\Sigma_i c_i w_i)$ by the activation total $(\Sigma_i w_i)$ is indicated in the division node in the last layer.

Figure 4.3 and 4.4 are the two-output counterparts of the RBFNs in 4.1 and 4.2.

4

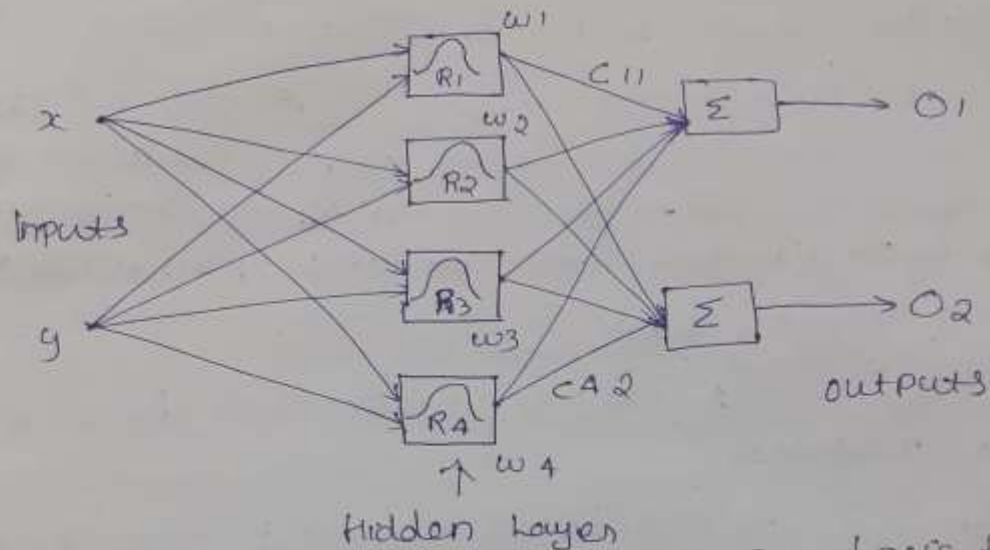


Figure 4.3 Four RBFNs that possess for basis function; two-output RBFN that uses weighted sum;

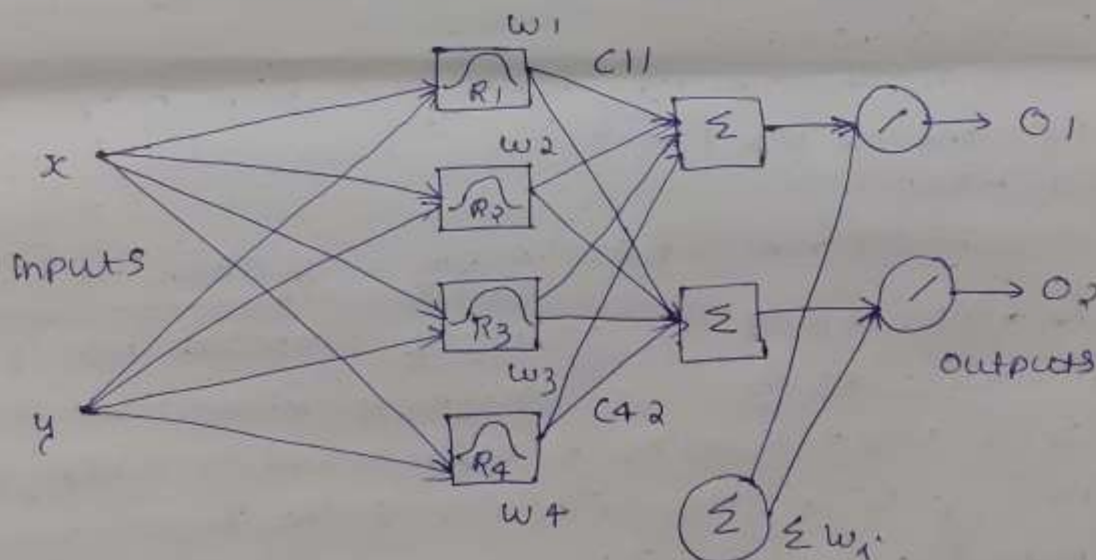


Figure 4.4 Four RBFNs that possess four basis functions; Two-output RBFN that uses weighted average.

5

Extension of Moody - Darken's RBFN

1. Moody - Darken's RBFN may be extended by assigning a linear function to the output function of each receptive field - that is, making c_i a linear combination of the input variables plus a constant:

$$c_i = a_i^T x + b_i,$$

where a_i is a parameter vector and b_i is a scalar parameter.

2. An RBFN's approximation capacity may be further improved with supervised adjustments of the center and shape of the receptive field (or radial basis) functions.

Several learning algorithms have been proposed to identify the parameters (u_i, σ_i and c_i) of an RBFN.

FUNCTIONAL EQUIVALENCE OF RBFN TO FIS

1. An extension of the originally proposed Moody - Darken's RBFN is to assign a linear function as the output function of each receptive field; that is, C_i is a linear function of the input variables instead of a constant:

$$C_i = \vec{a}_i \cdot \vec{x} + b_i, \qquad (4.3)$$

where $\vec{a}_i$ is a parameter vector and b_i is a scalar parameter.

6

2. Using Equation (4.3), the extended RBFN response given by Equation (4.1) or Equation (4.2) is identical to the response produced by the first-order Sugeno fuzzy inference system (FIS).
3. While the RBFN consists of radial basis functions, the FIS comprises a certain number of membership functions.
4. With those radially shaped functions, both FIS and RBFN have a mechanism whereby they can produce a center-weighted response to small receptive fields localizing the primary input excitation.

The conditions under which an RBFN and a FIS are functionally equivalent are:

a) Both the RBFN and the FIS under consideration use the same aggregation method to derive their overall outputs.

b) The number of receptive field units in the RBFN is equal to the number of fuzzy if-then rules in the FIS

c) Each radial basis function of the RBFN is equal to a multidimensional composite MF of the premise part of a fuzzy rule in the FIS

d) Corresponding radial basis function and fuzzy rule should have the same response function. That is they should have the same constant terms or linear equations.

7

Interpolation and Approximation RBFNs

Interpolation RBFNs

Assuming that there is no noise in the training data set, estimate a function d(•) that yields exact desired outputs for all training data.

Estimation of function d(.) that yields exact desired outputs for all training ata is called an "interpolation" problem

when RBFN is used with the same number of basis functions as training patterns then RBFN is called as interpolation RBFN, where each neuron in the hidden layer responds to one particular training input pattern.

Example:

consider a Gaussian basis function centered at u_i with a width parameter σ:

$$w_i = R_i(\|x-u_i\|) = \exp\left[-\frac{(x-u_i)^2}{2\sigma_i^2}\right].$$

Each training input x_i serves as a center for the basis function, R_i. Thus from Equation (4-1) a Gaussian interpolation RBFN:

$$d(x) = \sum_{i=1}^{n} c_i \exp\left[-\frac{(x-x_i)^2}{2\sigma_i^2}\right].$$

Each training input x_i serves as a center

For given σ_i, $i = 1, \ldots, n$, the following n simultaneous linear Equations are there

(8)

for the n unknown weight coefficients, C_i :

$$d_1 = c_1 \exp\left[-\frac{\|x_1-x_1\|^2}{2\sigma_1^2}\right] + \cdots + c_n \exp\left[-\frac{\|x_1-x_n\|^2}{2\sigma_n^2}\right],$$

$$d_2 = c_1 \exp\left[-\frac{\|x_2-x_1\|^2}{2\sigma_1^2}\right] + \cdots + c_n \exp\left[-\frac{\|x_2-x_n\|^2}{2\sigma_n^2}\right].$$

$$\vdots$$

$$d_n = c_1 \exp\left[-\frac{\|x_n-x_1\|^2}{2\sigma_1^2}\right] + \cdots + c_n \exp\left[-\frac{\|x_n-x_n\|^2}{2\sigma_n^2}\right].$$

Writing them in matrix form,

$$\begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_n \end{bmatrix} = \begin{bmatrix} \exp\left[-\frac{\|x_1-x_1\|^2}{2\sigma_1^2}\right] & \cdots & \exp\left[-\frac{\|x_1-x_n\|^2}{2\sigma_n^2}\right] \\ \exp\left[-\frac{\|x_2-x_1\|^2}{2\sigma_1^2}\right] & \cdots & \exp\left[-\frac{\|x_2-x_n\|}{2\sigma_n^2}\right] \\ \vdots & \vdots & \vdots \\ \exp\left[-\frac{\|x_n-x_1\|^2}{2\sigma_1^2}\right] & \cdots & \exp\left[-\frac{\|x_n-x_n\|^2}{2\sigma_n^2}\right] \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix}$$

Rewriting the preceding in a compact form, then there is a unique solution :

$$C = G^{-1} D,$$

where G^{-1} denotes the inverse matrix of G.

Approximation RBFN

When there are fewer basis functions than there are available training samples, an initial guess is required to determine their center positions.

(9)

In approximation RBFN matrix G is not square and the least-squares methods are commonly used to find the matrix C in D = GC.

UNSUPERVISED LEARNING NEURAL NETWORKS

1. When no external teacher or critic's instruction is available, only input vectors can be used for learning. Such an approach is learning without supervision, or unsupervised learning
2. An unsupervised learning system (or agent) evolves to extract features or regularities in presented patterns, without being told what outputs or classes associated with the input patterns are desired.
3. The learning system detects or categorizes persistent features without any feedback from the environment.
4. Unsupervised learning is frequently employed for data clustering, feature extraction, and similarity detection.
5. Unsupervised learning Neural Networks attempt to learn to respond to different input patterns with different parts of the network.

COMPETITIVE LEARNING NETWORKS

1. With no available information regarding the desired outputs, unsupervised learning networks update weights only on the basis of the input patterns.

(10)

2. The competitive learning network is a popular scheme which is a type of unsupervised data clustering or classification.

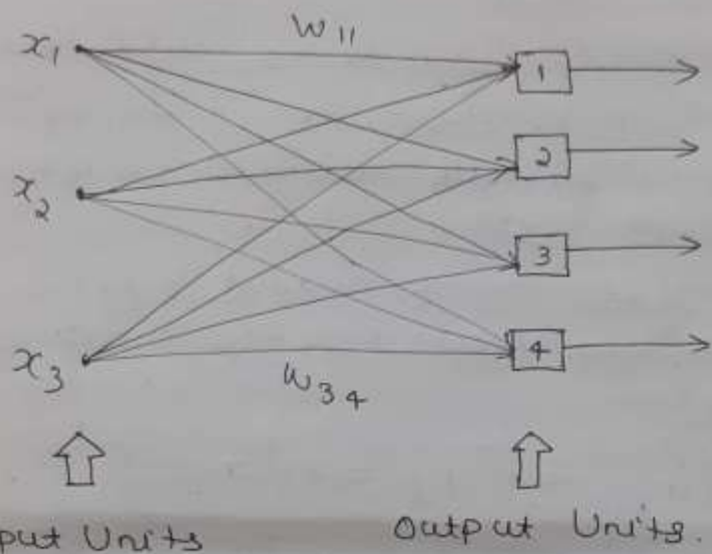


Figure 4.5 Competitive learning network.

3. Figure 4.5 presents an example for competitive learning network.
4. All input units i are connected to all output units j with weight w_{ij}.
5. The number of inputs is the input dimension.
6. The number of outputs is equal to the number of clusters that the data are to be divided into.
7. A cluster center's position is specified by the weight vector connected to the corresponding output unit.

(11)

8. In Figure 4.5, the three-dimensional input data are divided into four clusters, and the cluster centers (denoted as the weights) are updated via the competitive learning rule.

9. The input vector $x = [x_1, x_2, x_3]^T$ and the weight vector $w_j = [w_{1j}, w_{2j}, w_{3j}]^T$ for an output unit j are generally assumed to be normalized to unit length.

10. The activation value a_j of output unit j is calculated by the inner product of the input and weight vectors:

$$a_j = \sum_{i=1}^{3} x_i w_{ij} = x^T w_j = w_j^T x.$$

11. The output unit with the highest activation must be selected for further processing, which is what is implied by competitive.

12. Assuming that output unit k has the maximal activation, the weights leading to this unit are updated according to the competitive or the so-called winner-take-all learning rule:

$$w_k(t+1) = \frac{w_k(t) + \eta(x(t) - w_k(t))}{\|w_k(t) + \eta(x(t) - w_k(t))\|}.$$

13. The preceding weight update formula includes a normalization operation to ensure the updated weight is always of unit length.

(12)

14. Only the weights at the winner output unit k are updated; all other weights remain unchanged.
15. A competitive learning network performs an on-line clustering process on the input patterns when the process is complete, the input data are divided into disjoint ~~clu~~ clusters such that similarities between individuals in the same cluster are larger than those in different clusters.

16 Dissimilarity measure

1. Using the Euclidean distance as a dissimilarity measure is a more general scheme of competitive learning, in which the activation of output unit j is

$$a_j = \left(\sum_{i=1}^{3} (x_i - w_{ij})^2 \right)^{0.5} = \| x - w_j \|$$

2. The weights of the output unit with the smallest activation are updated according to

$$w_k(t+1) = w_k(t) + \eta(x(t) - w_k(t)).$$

17. Limitation of competitive learning

A limitation of competitive learning is that some of the weight vectors that are initialized to random values may be far from any input vector and, subsequently, it never gets updated.

1. Solution: The above limitation of competitive learning can be prevented by initializing the weights to samples from the input data itself,

13

thereby ensuring that all of the weights got updated when all the input patterns are presented.

2. Solution:

An alternative solution for the limitation would be to update the weights of both the winning and losing units, but use a significantly smaller learning rate η for the losers; this is commonly referred to as leaky learning.

18. Stability - plasticity dilemma.

If η decreasing with time, may become too small to update cluster centers when new data of a different probability nature are presented.

KOHONEN SELF-ORGANIZING NETWORKS

1. Kohonen self-organizing networks also known as Kohonen feature maps or topology-preserving maps, are another competition-based network paradigm for data clustering.
2. It impose a neighborhood constraint on the output units, such that a certain topological property in the input data is reflected in the output units' weights.
3. Figure 4.6 presents a relatively simple Kohonen self-organizing network with 2 inputs and 49 outputs.
4. The learning procedure of Kohonen feature maps is similar to that of competitive learning networks.
5. A similarity (dissimilarity) measure is selected

(14)

and the winning unit is considered to be the one with the largest (smallest) activation.

6. For kohonen feature maps, the winning unit's weights and all of the weights in a neighborhood around the winning units get update.

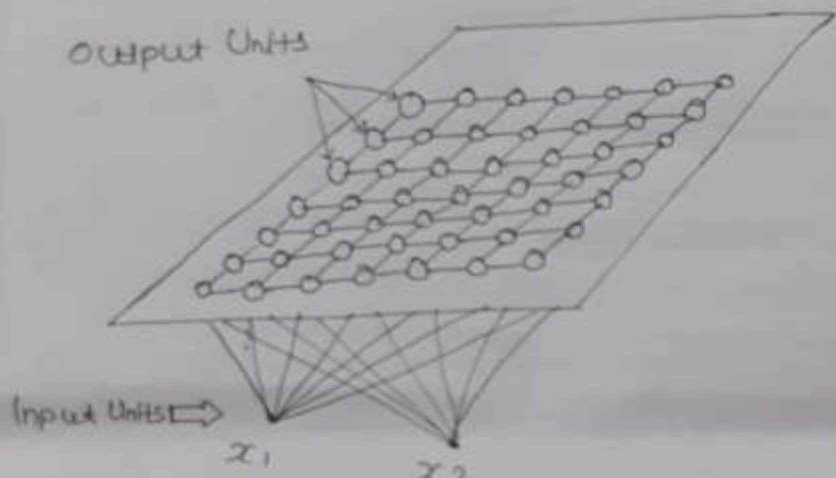


Figure 4.6 (a) A kohonen self-organizing network with 2 input and 49 output units;

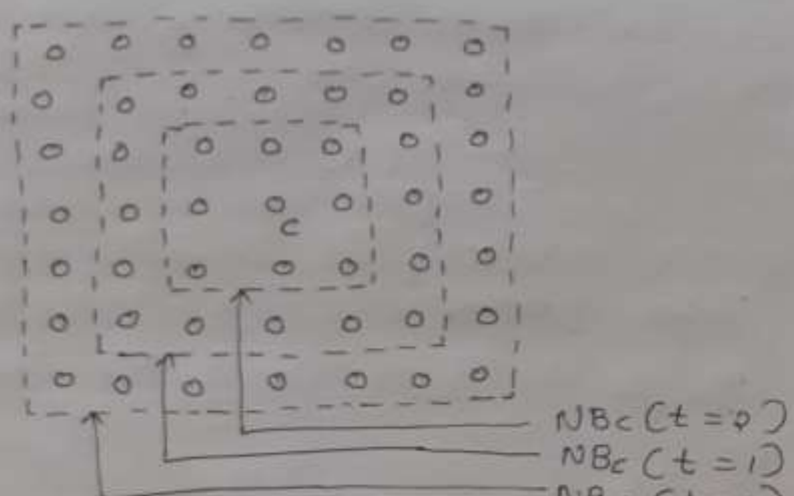


Figure 4.6 (b) the size of a neighbourhood around a winning unit decreases gradually with each iteration.

15

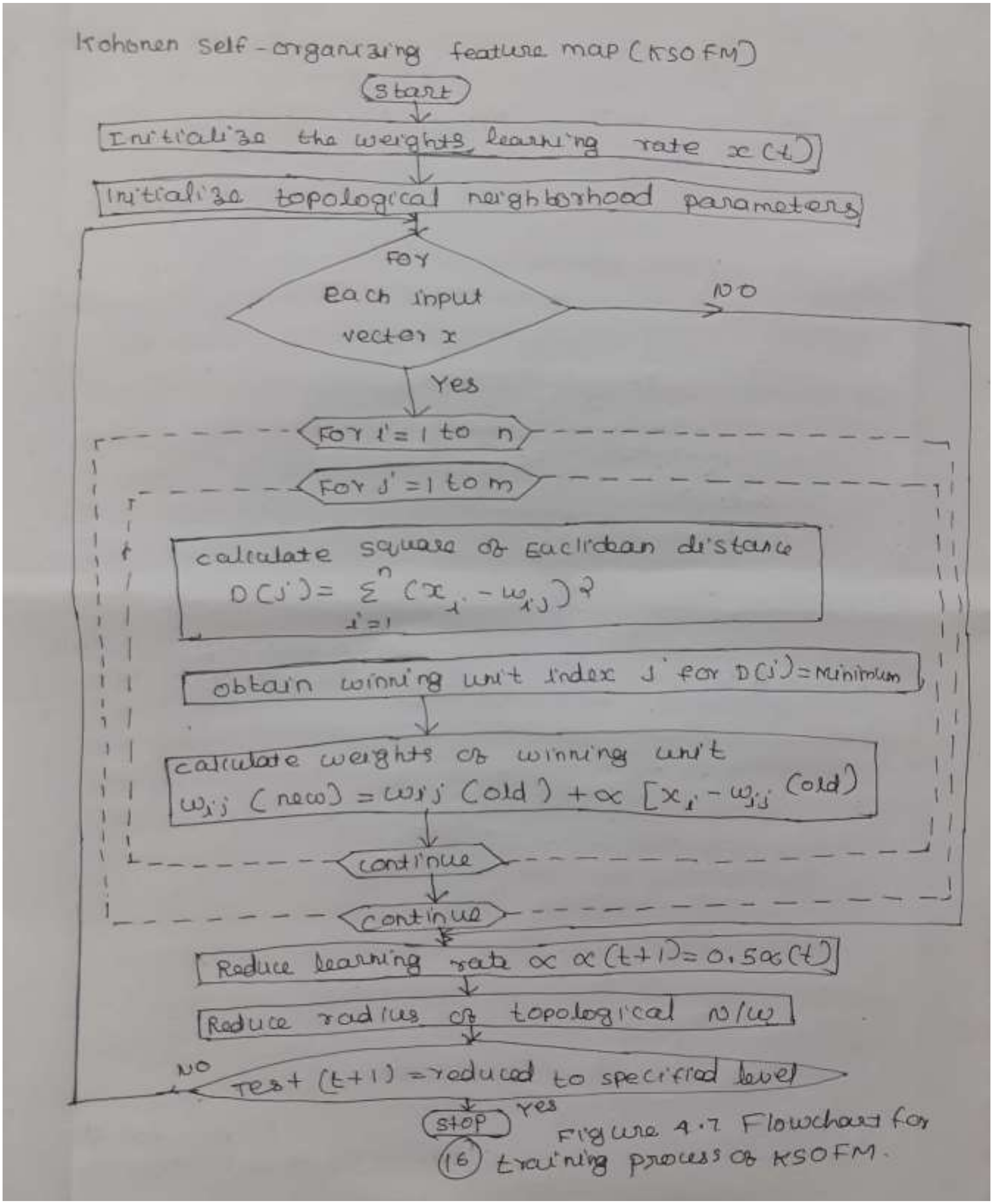


Figure 4.7 Flowchart for training process of KSOFM.

A sequential description of how to train a kohonen self-organizing network is as follows:

Step 1: Select the winning output unit as the one with the largest similarity measure (or smallest dissimilarity measure) between all weight vectors w_i and the input vector x. If the Euclidean distance is chosen as the dissimilarity measure, then

$$\|x - w_c\| = \min_i \|x - w_i\|,$$

where the index refers to the winning unit.

Step 2: Let NB_c denote a set of index corresponding to a neighborhood around winner C. The weights of the winner and its neighboring units are then updated by

$$\Delta w_i = \eta (x - w_i), \; i \in NB_c,$$

where η is a small positive learning rate. Instead of defining the neighborhood of a winning unit, we can use a neighborhood function $\Omega_c(i)$ around a winning unit C.

Eg: the Gaussian function can be used as the neighborhood function:

$$\Omega_c(i) = \exp\left(\frac{-\|p_i - p_c\|^2}{2\sigma^2}\right),$$

where p_i and p_c are the positions of the output units i and c, respectively, and σ reflects the

17

scope of the neighborhood. By using the neighborhood function, the update formula can be rewritten as

$\Delta w_i = \eta \Omega_c (i) (x - w_i)$, where i is the index for all output units.

To achieve a better convergence, the learning rate η and the size of neighborhood (or σ) should be decreased gradually with each iteration.

LEARNING VECTOR QUANTIZATION (LVQ)

1. LVQ is a process of classifying the patterns, wherein each output unit represents a particular class.
2. LVQ is an adaptive data classification method based on training data with desired class information.
3. LVQ employs unsupervised data-clustering techniques to preprocess the data set and obtain cluster centers
4. LVQ's network architecture closely resembles that of a competitive learning network, except that each output unit is associated with a class.
5. Figure 4.8 presents an example for Learning Vector Quantization where the input dimension is 2 and the input space is divided into six clusters.
   a) The first two clusters belong to class 1, while the other four clusters belong to class 2.
6. The LVQ learning algorithm involves two steps.
   step1: An unsupervised learning data clustering method is used to locate several cluster centers without using the class information.

(18)

step 2 : The class information is used to fine-tune the cluster centers to minimize the number of misclassified cases. (supervised)

once the clusters are obtained, their classes must be labeled before moving to the second step of supervised learning.

Labeling can be achieved by voting method.

Voting method - A cluster is labeled class k if it has data points belonging to class k as a majority within the cluster.

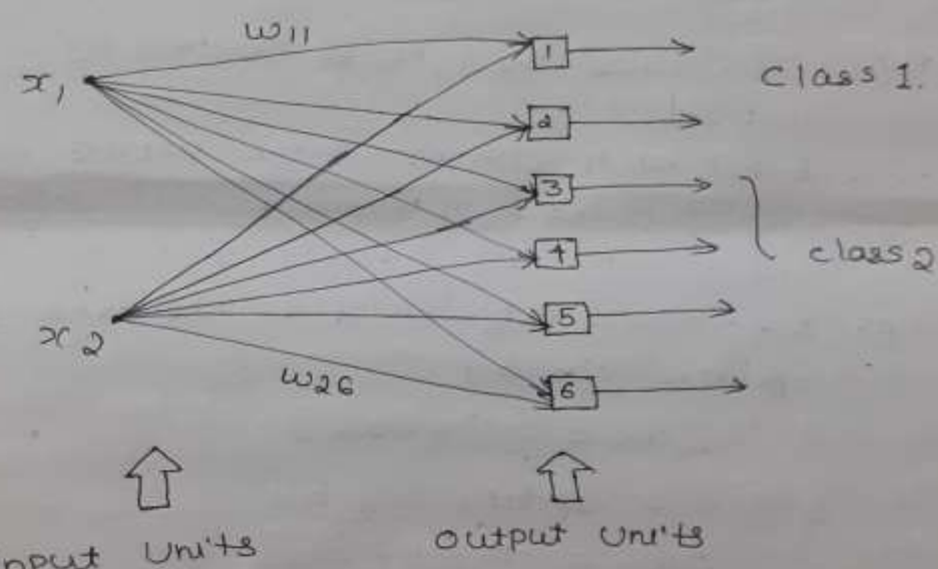


Figure 4.8 Learning vector quantization (LVQ) network representation.

7. The clustering process for LVQ is based on the general assumption that similar input patterns generally belong to the same class.

8. During second step the learning method is straight-forward.

(19)

First, the weight vector (or cluster center) w that is closest to the input vector x must be found. If x and w belong to the same class, we move w toward x; otherwise move w away from the input vector x.

9. After learning, an LVQ network classifies an input vector by assigning it to the same class as the output unit that has the weight vector (cluster center) closest to the input vector.

10. A sequential description of the LVQ method is as follows:

Step1: Initialize the cluster centers by a clustering method.

Step2: Label each cluster by the voting method.

Step3: Randomly select a training input vector x and find k such that $\|x - w_k\|$ is a ~~mo~~ minimum.

Step4: If x and w_k belong to the same class, update w_k by

$$\Delta w_k = \eta(x - w_k).$$

Otherwise, update w_k by

$$\Delta w_k = -\eta(x - w_k).$$

The learning rate η is a positive small constant and should decrease with each iteration.

Step5: If the maximum number of iterations is reached, stop. Otherwise, return to Step3.

<u>Flowchart</u>

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

20

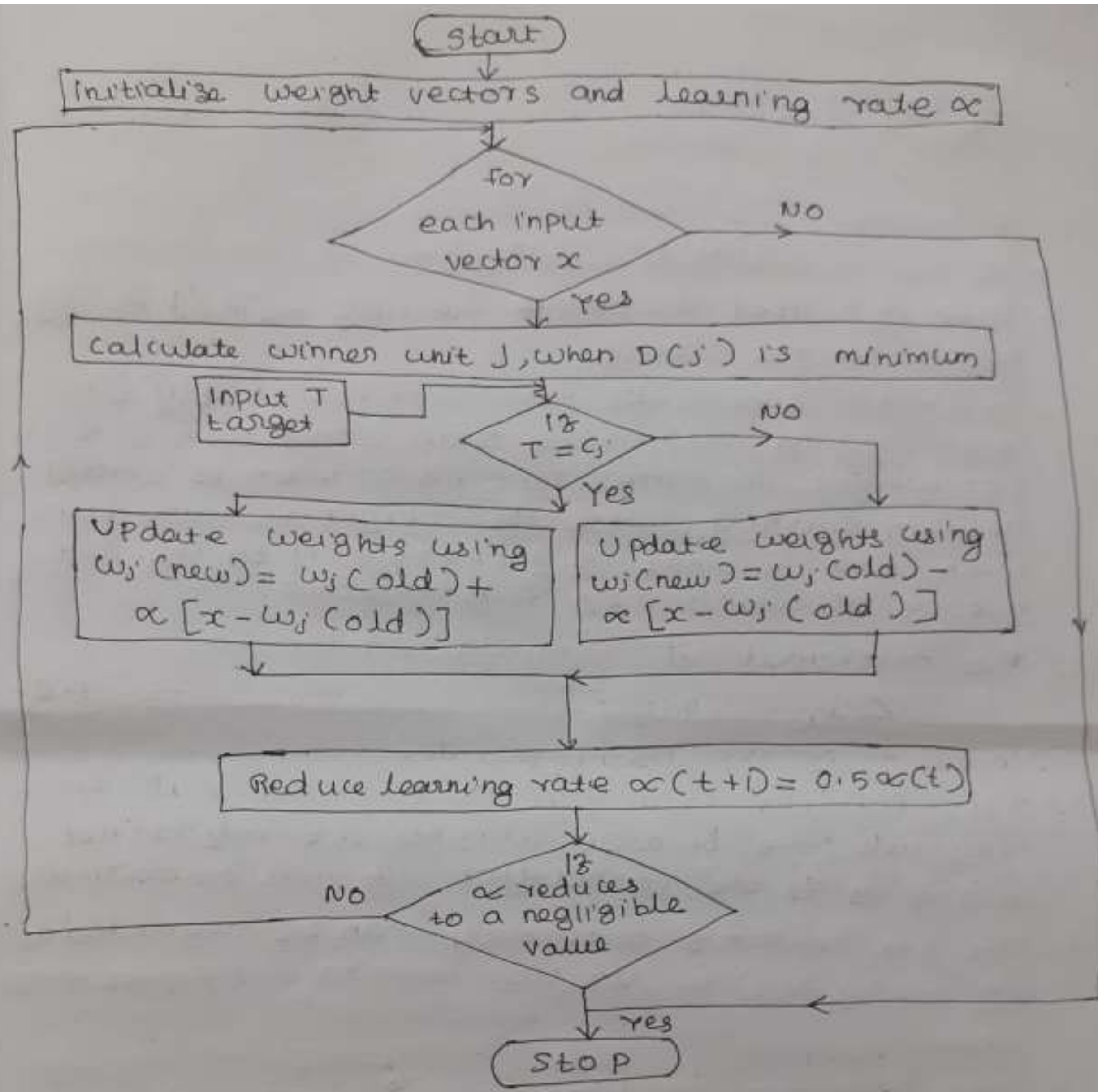


Figure 4.9 Flowchart for LVQ.

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

T = category or class for the training vector x

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

C_j = cluster or class or category associated

(21)

with j'th output unit.

The Euclidean distance of j'th output unit is

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

HEBBIAN LEARNING

1. Hebb described a simple learning method of synaptic weight change.

   When two cells fire simultaneously (i.e., have strong responses), their connection strength (or weight) increases. This phinomenon is called Hebbian learning, where the weight increase between two neurons is proportional to the frequency at which they fire together.

2. The mathematical formula for this is

$$\Delta w_{ij} = \eta y_i y_j , \qquad — 4.6$$

where η is the learning rate.

3. This formula is a type of correlational learning rule ~~since~~ because weights are adjusted according to the correlation of neuron outputs

4. The ith neuron's output y_i can be regarded as an input (x_i) to another neuron j (Figure 4.10)

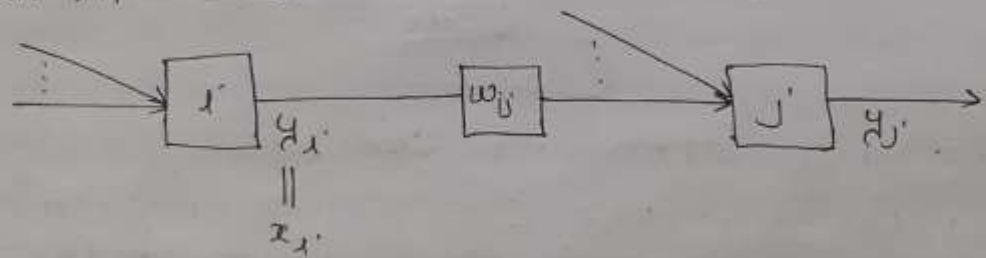


Figure 4.10 A simple network topology for Hebbian learning; weight w_{ij} resides between two neurons i and j.

(22)

So equation (4.6) can be written as

$$\Delta w_{ij} = \eta y_j x_i. \qquad (4.7)$$

5. A weight is assumed to change proportionately to the correlation of the input and output signals. By using a neuron function $f(\cdot)$, y_j is given by $y_j = f(w_j^T x)$.

Thus, Equation 4.7 is equivalent to the following:

$$\Delta w_{ij} = \eta f(w_j^T x) x_i.$$

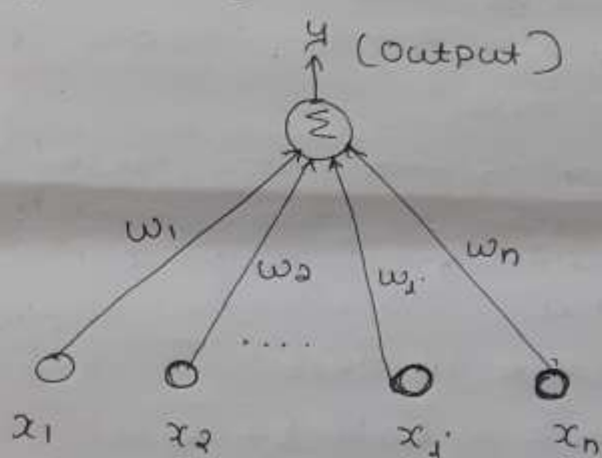


Figure 4.11 One-layer single-output network with Hebbian learning for principal component analysis.

6. Figure 4.11 is a single-layer n-input one-output neural network with identity activation functions.
   a) The output y is equal to $\sum_{i=1}^{n} w_i x_i$, or in matrix form, $y = w^T x = x^T w$, where $x = [x_1, \ldots, x_n]^T$ is the input vector

23

and $\omega = [\omega_1, \dots, \omega_n]^T$ is the weight vector.
The corresponding Hebbian learning rule is

$$\Delta\omega = \eta y x.$$

7. Donald Hebb stated in 1949 that in the brain, the learning is performed by the change in the synaptic gap.
Hebb explained it: "When an axon of cell A is near enough to excite cell B, and repeatedly or permanently takes place in firing it, some growth process or metabolic change takes place in one or both the cells such that A's efficiency, as one of the cells firing B, is increased."

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

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

9. The Hebb rule is more suited for bipolar data than binary data.

10. Flowchart of Training Algorithm
The training algorithm is used for the calculation and adjustment of weights. The flowchart for the training algorithm of Hebb network is given in Figure 4.12.
$s:t$ refers to each training input and target output pair. Till there exist a pair of training

(24)

input and target output, the training process takes place; else, it is stopped.

II. Training Algorithm

The training algorithm of Hebb network is given below:

Step0: First initialize the weights. Basically in this network they may be set to zero, i.e., $w_i = 0$ for $i = 1$ to n where "n" may be the total number of input neurons.

Step1: Steps 2-4 have to be performed for each input training vector and target output pair,

Step2: Input units activations are set. Generally the activation function of input layer is identity function.

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

Step3: Output units activations are set.

$$y = t$$

Step4: weight adjustments and bias adjustments are performed:

$$w_i(new) = w_i(old) + x_i y$$
$$b(new) = b(old) + y$$

The Hebb rule can be used for pattern association, pattern categorization, pattern classification.

25

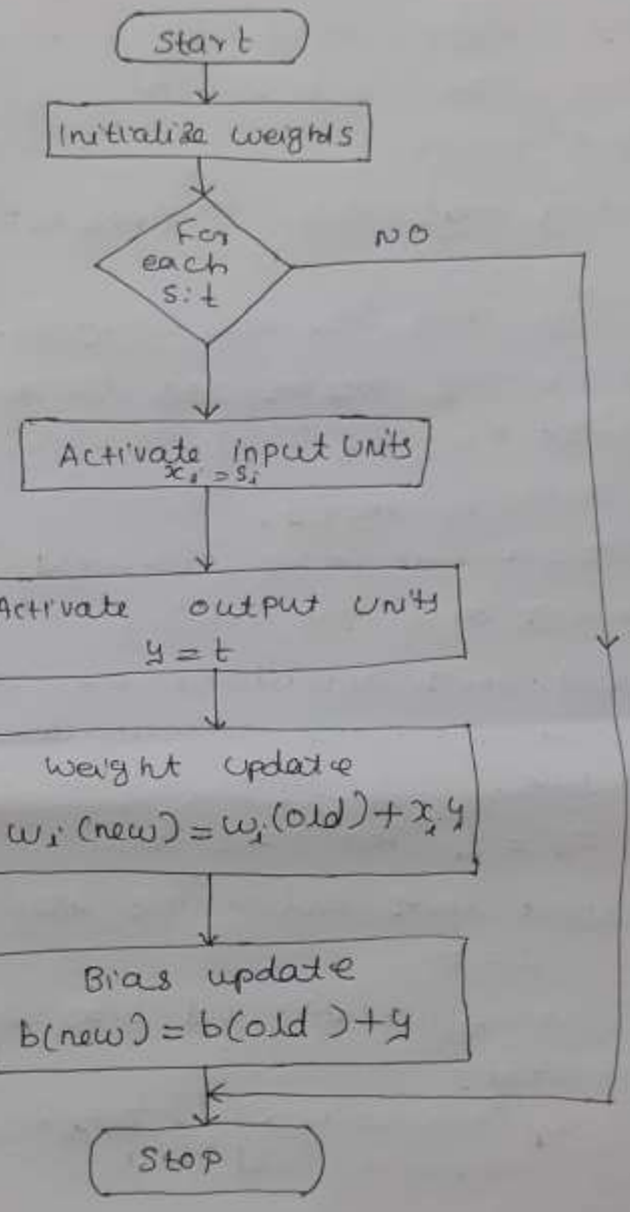


Figure 4.12 Flowchart of Hebb training algorithm.

(26)

MR 402 - Module V

ADAPTIVE NEURO-FUZZY INFERENCE SYSTEMS (ANFIS)

1. Adaptive networks
   a) Functionally, there are almost no constraints on the node functions of an adaptive network except for the requirement of piece wise differentiability.
   b) Structurally, the only limitation on the network configuration is that it should be of the feedforward type.
   c) Adaptive networks can be employed directly in a wide variety of applications of modeling, decision making, signal processing, and control.
2. ANFIS is a class of adaptive networks that are functionally equivalent to fuzzy inference systems.

ANFIS ARCHITECTURE

1. Assume that the fuzzy inference system has two inputs x and y and one output z.
2. For a first-order sugeno fuzzy model, a common set with two fuzzy if-then rules is the following:

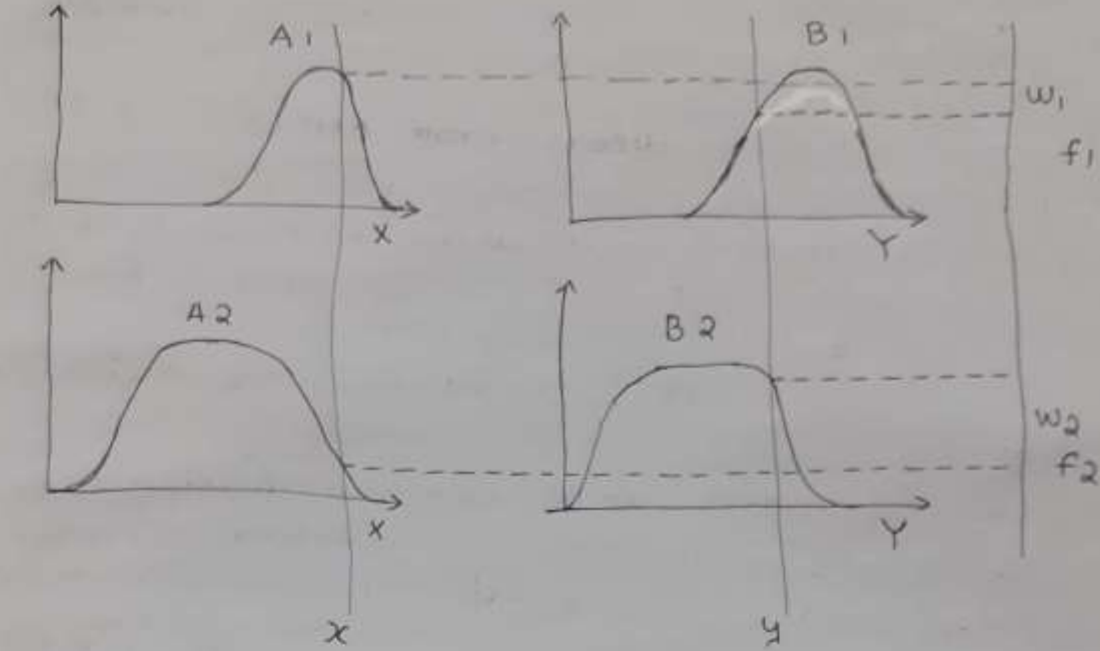
   Rule 1: If x is A_1 and y is B_1, then $f_1 = p_1x + q_1y + r_1$,

   Rule 2: If x is A_2 and y is B_2, then $f_2 = p_2x + q_2y + r_2$.

   a) Figure 5.1 illustrates the reasoning mechanism for this Sugeno model;
   b) The corresponding equivalent ANFIS architecture is shown in Figure 5.2, where nodes of the same layer have similar functions.

1

The output of the ith node in layer l is $O_{l,i}$.



$$f_1 = p_1 x + q_1 y + r_1$$

$$f_2 = p_2 x + q_2 y + r_2$$

$$\Rightarrow \quad f = \frac{w_1 f_1 + w_2 f_2}{w_1 + w_2} = \overline{w}_1 f_1 + \overline{w}_2 f_2$$

Figure 5.1 A two-input first-order Sugeno fuzzy model with two rules;

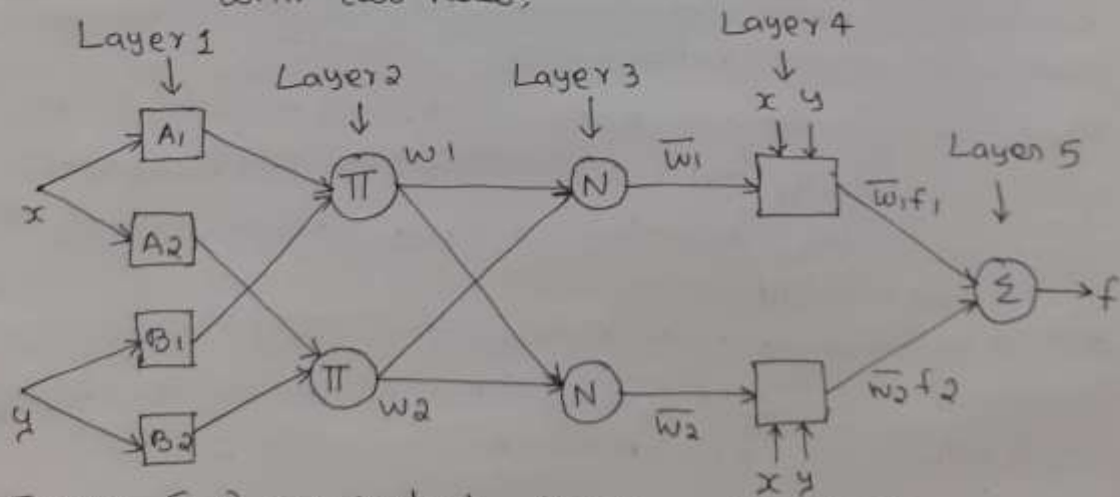


Figure 5.2 equivalent ANFIS architecture.

(6)

Layer 1

Every node i in this layer is an adaptive node with a node function.

$$O_{1,i} = \mu_{A_i}(x), \quad \text{for } i = 1, 2, \text{or}$$

$$O_{1,i} = \mu_{B_{i-2}}(y), \quad \text{for } i = 3, 4.$$

where x (or y) is the input to node i and A_i (or B_{i-2}) is a linguistic label (such as "small" or "large") associated with this node.

$O_{1,i}$ is the membership grade of a fuzzy set A (= A_1, A_2, B_1, or B_2) and it specifies the degree to which the input x (or y) satisfies the quantifier A.

The membership function for A can be any appropriate parameterized membership function such as the generalized bell function:

$$\mu_A(x) = \frac{1}{1 + \left|\frac{x - c_i}{a_i}\right|^{2b}},$$

where $\{a_i, b_i, c_i\}$ is the parameter set.

Layer 2

Every node in this layer is a fixed node labeled Π, whose output is the product of all the incoming signals.

$$O_{2,i} = w_i = \mu_{A_i}(x)\,\mu_{B_i}(y), \quad i = 1, 2.$$

Each node output represents the firing strength of a rule.

3

Layer 3

Every node in this layer is a fixed node labeled N. The ith node calculates the ratio of the ith rule's firing strength to the sum of all rules' firing strengths:

$$O_{3,i} = \overline{w_i} = \frac{w_i}{w_1 + w_2}, \quad i = 1, 2.$$

Outputs of this layer are called normalised firing strengths.

Layer 4

Every node i in this layer is an adaptive node with a node function

$$O_{4,i} = \overline{w_i} \cdot f_i = \overline{w_i}\,(p_i x + q_i y) + r_i),$$

where $\overline{w_i}$ is a normalised firing strength from layer 3 and $\{p_i, q_i, r_i\}$ is the parameter set of this node, parameters in this layer are referred as consequent parameters.

Layer 5

The single node in this layer is a fixed node labeled Σ, which computes the overall output as the summation of all incoming signals:

$$\text{overall output} = O_{5,1} = \sum_i \overline{w_i} f_i = \frac{\sum_i w_i f_i}{\sum_i w_i}$$

3. The structure of this adaptive network is not unique; it is possible to combine layers 3 and 4 to obtain an equivalent network with only four layers. In the same way it is possible to perform the weight normalization at the last layer.

(4)

Figure 5.3 illustrates an ANFIS of this type.

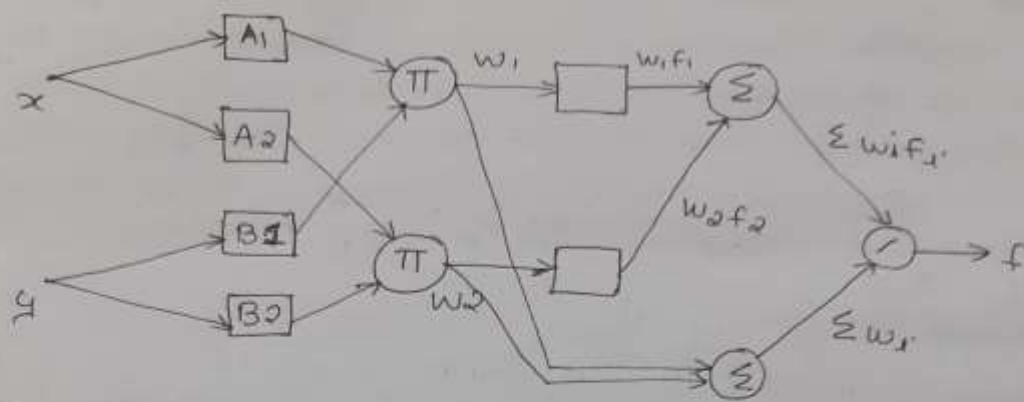
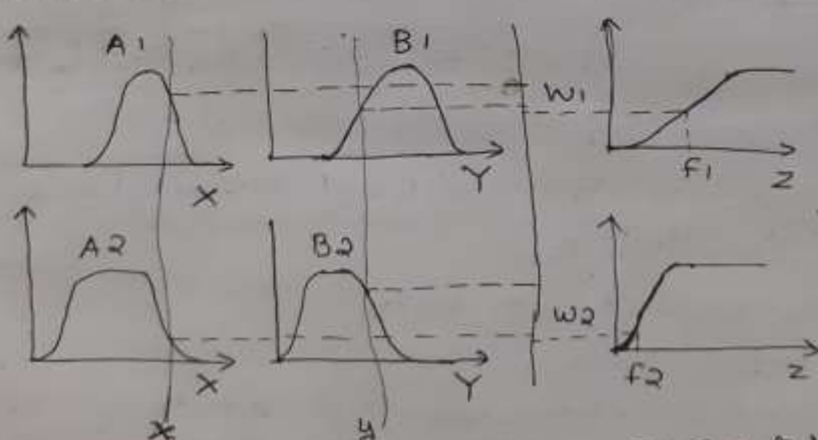


Figure 5.3 ANFIS architecture for the Sugeno fuzzy model, where weight normalization is performed at the very last layer.

4. Tsukamoto fuzzy model and its equivalent ANFIS architecture.

The output of each rule (f_i, $i = 1, 2$) is induced jointly by a consequent membership function and firing strength. Figure 5.4.

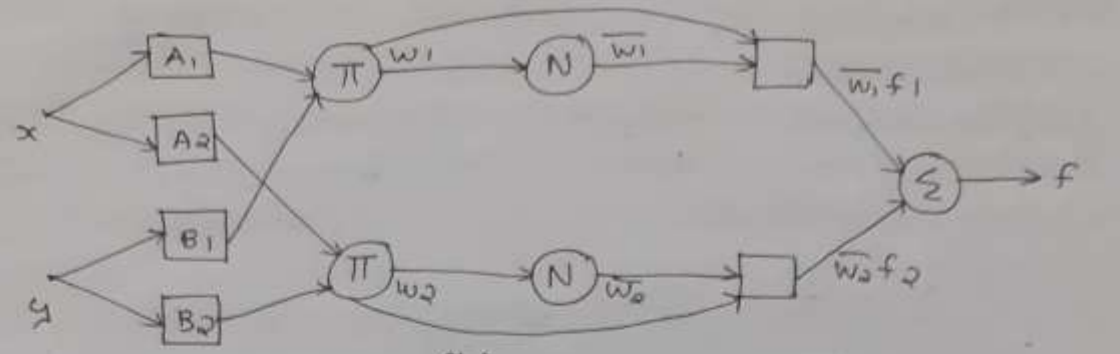
5. Sugeno fuzzy model's ANFIS architecture is transparent and efficient.



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

Figure 5.4 (a)

5



(b)

Figure 5.4 (a) A two-input two-rule Tsukamoto fuzzy model; (b) equivalent ANFIS architecture.

HYBRID LEARNING ALGORITHM

1. From the ANFIS architecture shown in Figure 5.2, observe that when the values of the premise parameters are fixed, the overall output can be expressed as a linear combination of the consequent parameters.
2. The output f in Figure 5.2 can be rewritten as

$$f = \frac{w_1}{w_1+w_2} f_1 + \frac{w_2}{w_1+w_2} f_2$$

$$= \overline{w_1}(p_1x + q_1y + r_1) + \overline{w_2}(p_2x + q_2y + r_2)$$

$$= (\overline{w_1}x)p_1 + (\overline{w_1}y)q_1 + (\overline{w_1})r_1 + (\overline{w_2}x)p_2 + (\overline{w_2}y)q_2 + (\overline{w_2})r_2,$$

which is linear in the consequent parameters p_1, q_1, r_1, p_2, q_2, and r_2.

From this observation, we have

S = set of total parameters,

S_1 = set of premise (nonlinear) parameters,

S_2 = set of consequent (linear) parameters.

(6)

H(·) - represents Identity function
F(·,·) - represents function used in FIS.
Hybrid learning algorithm is a two pass algorithm. It uses 2 passes for its working. The passes are:

1. Forward Pass
2. Backward Pass

Forward pass

In this pass node output goes forward until layer 4 and consequent parameters are identified by the Least Square method.

Backward Pass

The error signals propagate backward and premise parameters are updated by gradient descent method.

Activities in each pass are summarized as: in Table 5.1.

Table 5.1. Two passes in the hybrid learning procedure for ANFIS.

	Forward pass	Backward pass
Premise parameters	Fixed	Gradient descent
Consequent parameters	Least-squares estimator	Fixed
Signals	Node outputs	Error signals

The consequent parameters identified are optimal under the condition that the premise parameters are fixed.
The hybrid approach converges much faster since it reduces the search space dimensions of the original pure backpropagation method.

7

For Tsukamoto ANFIS, the reduction of the search dimensions can be achieved if the membership on the consequent part of each rule is replaced by a piecewise linear approximation with two consequent parameters as shown in Figure 5.5.

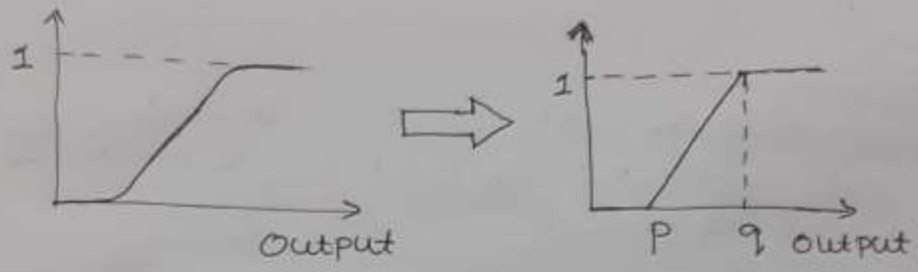


Figure 5.5 Piecewise linear approximation of consequent MFs in Tsukamoto ANFIS.

LEARNING METHODS THAT CROSS-FERTILIZE ANFIS AND RBFN

1. Under certain minor conditions, an RBFN is functionally equivalent to a FIS, and thus adaptive FIS, including ANFIS.
2. An adaptive FIS usually consists of two distinct modifiable parts:
   i. The antecedent part and
   ii. The consequent part.
3. These two parts can be adapted by different optimization methods, one of which is the hybrid learning procedure combining GD (gradient descent) and LSE (least-squares estimator). These learning schemes are equally applicable to RBFNs.
4. The analysis and learning algorithms for RBFNs are also applicable to adaptive FIS (ANFIS/CANFIS).

8

5. A variety of two-phase training algorithms for RBFNs have been reported.

i) A typical scheme is to fix the receptive field (radial basis) functions first and then adjust the weights of the output layer.

ii) There are several schemes proposed to determine the center positions (u_i) of the receptive field functions.

1. Lowe discussed selection of fixed centers based one standard deviations of training data.
2. Moody and Darken discussed unsupervised or self-organized selection of centers u_i by means of vector quantization or clustering techniques.
3. The width parameters σ_i are determined by taking the average distance to the first several nearest neighbors of u_i's.
4. Nowlan employed the so-called soft competition among Gaussian hidden units to locate the centers.
5. Soft-competitive method is based on the maximum likelihood estimator, in contrast to the so-called hard competition such as the k-means winner-take-all algorithm.
6. Once nonlinear parameters are fixed and the receptive fields are frozen, the linear parameters (i.e., the weights of the output layer) can be updated by either the least-squares method or the gradient method.

9

COACTIVE NEURO-FUZZY MODELING

1. ANFIS is an adaptive system with the advantage of being a linguistically interpretable FIS that allows prior knowledge to be embedded in its construction and allows the possibility of understanding the result of learning.
2. Neuro-fuzzy system that enjoys many of the advantages claimed for neural networks (NNs) and the linguistic interpretability of an FIS is the generalised ANFIS "CANFIS", which stands for coactive neuro-fuzzy inference systems, wherein both NNs and FIS play active roles in an effort to reach a specific goal.
3. Neuro-fuzzy models can be characterised by the neuro-fuzzy spectrum, in light of linguistic transparency and input-output mapping precision.

FRAMEWORK

1. CANFIS has multiple outputs.
2. One way to get multiple outputs is to place as many ANFIS models side by side as there are required outputs.
3. MANFIS model is illustrated in Figure 5.6
4. No modifiable parameters are shared by the ANFIS models.
5. Each ANFIS has an independent set of fuzzy rules, which makes it difficult to realize possible certain correlations between outputs.
6. Another way of generating multiple outputs is to

(10)

maintain the same antecedents of fuzzy rules among multiple ANFIS models.

7. Figure 5.6(a) visualizes CANFIS concept.
8. Fuzzy rules are constructed with shared membership values to express correlations between outputs.

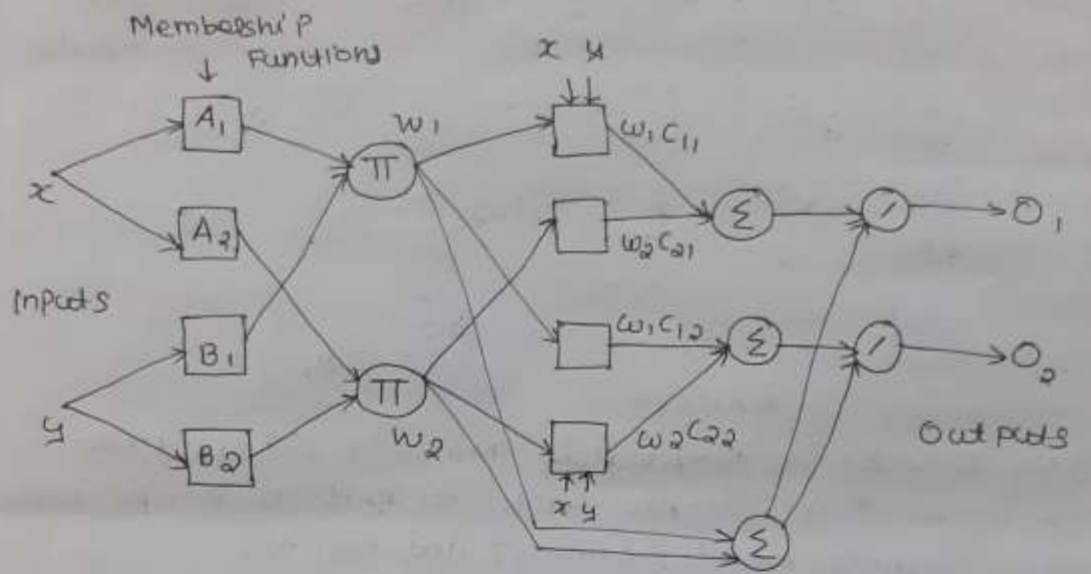
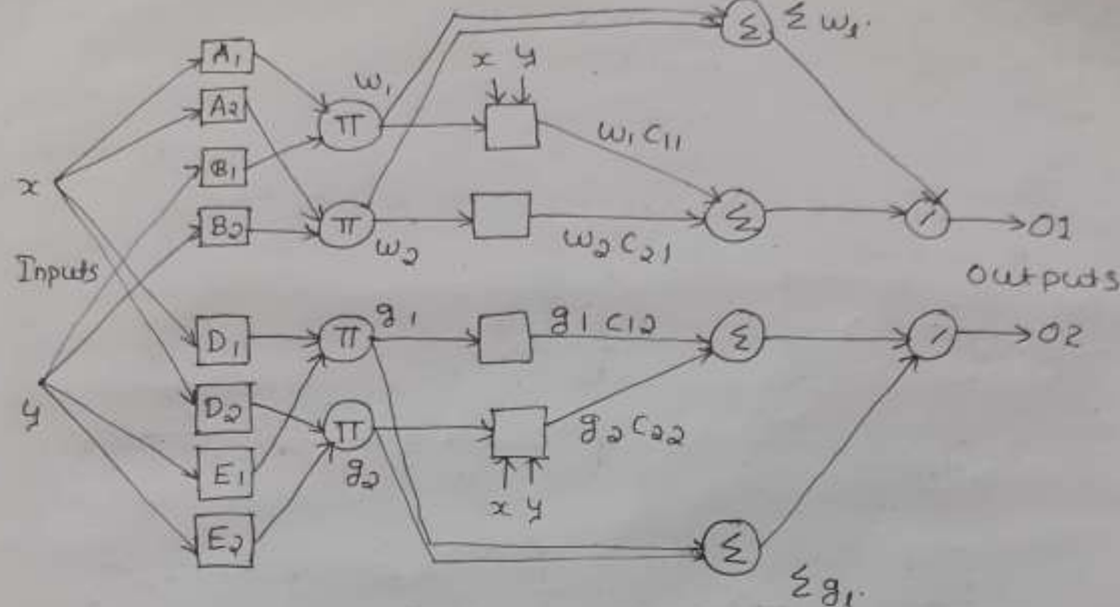


Figure 5.6 (a) Two-output CANFIS architecture with two rules per output.

Architectural Comparisons

1. In both CANFIS and RBFN, locality is considered by Euclidean norms between each local center and the input vector as in Figure 5.7.
2. By comparison, the inner product of each weight vector and input vector is taken in a backpropagation MLP to measure similarity between training patterns.

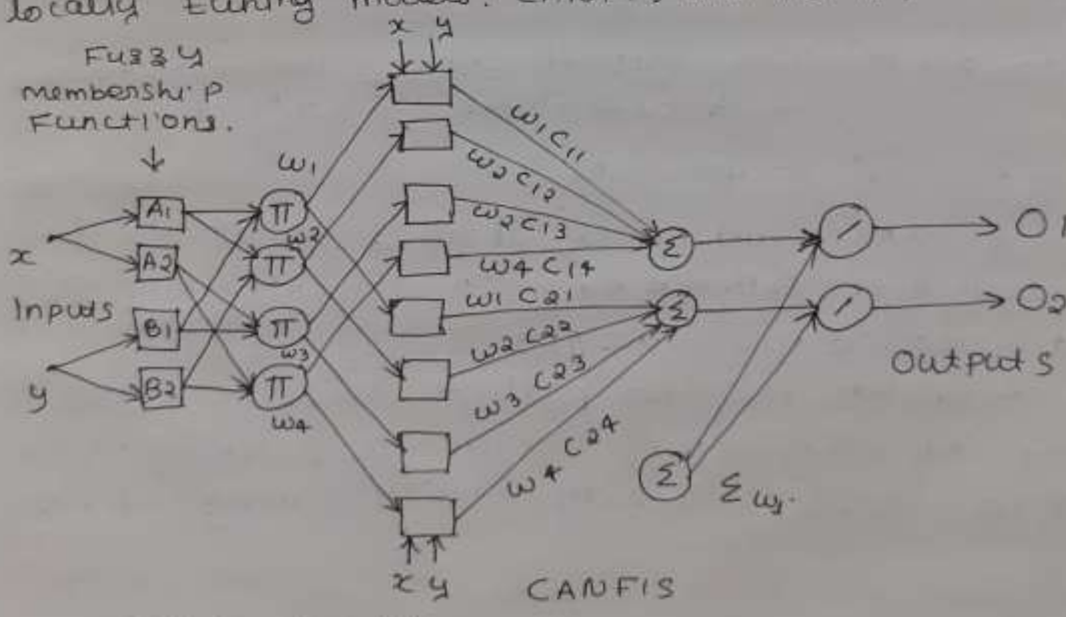
(11)



MANFIS

Figure 5.6 (b) A comparable MANFIS structure.

3. Comparison of a simple backpropagation MLP with locally tuning models: CANFIS and RBFN.



CANFIS

Figure 5.7 (a)

(12)

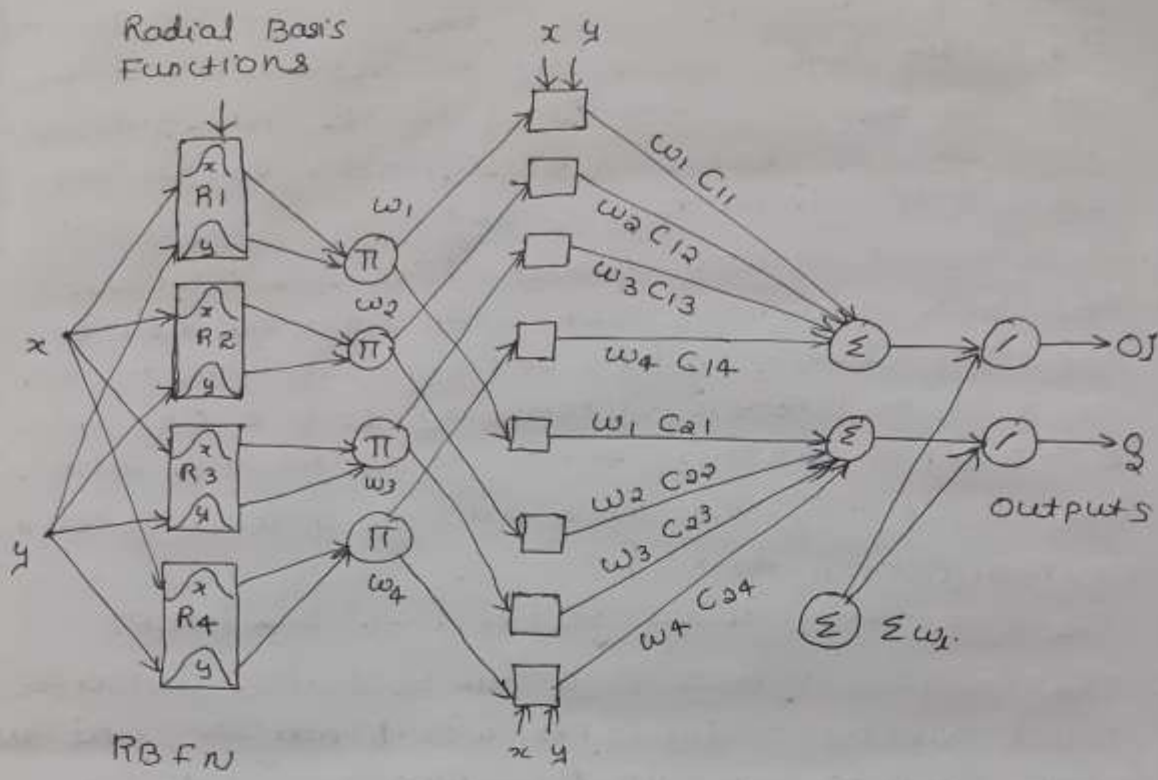


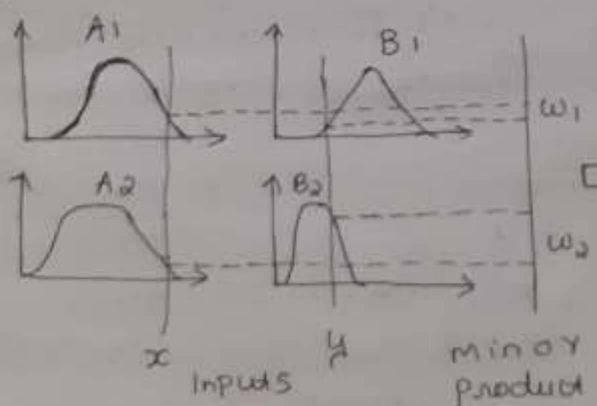
Figure 5.7 (b).

4. A single-output CANFIS can be illustrated in the same schematic diagrams of ANFIS in Figures 5.8 (b) through 5.8 (d).
5. When all three neurons (1, 2, 3) have identity functions in Figure 5.8 (d), the presented CANFIS is equivalent to the Sugeno (TSK) fuzzy inference system in Figure 5.8(a), which accomplishes fuzzy if-then rules (linear rules) such as the following:

   Rule: If x is A_1 and y is B_1, then $C_1 = p_1 x + q_1 y + r_1$.
6. Putting more hidden nodes in the MLP is equivalent to adding more rules to CANFIS.

13

7. The MLP's weights between the output layer and the hidden layer correspond to membership values between the consequent layer and the fuzzy association layer in CANFIS. This comparison emphasizes the inside transparency of CANFIS.
8. CANFIS is locally tuned, like the RBFN.
9. The backpropagation MLP with sigmoidal neuron functions globally updates weight coefficients for every input pattern, attempting to find one specific set of weights common to all training patterns
10. The RBFN may need more data to achieve a certain accuracy than the MLP.
11. The RBFN may learns faster than the MLP.
12. The backpropagation MLP can be better extrapolator than RBFN due to its global nature, and that RBFN fails to estimate the values of functions outside the range of the training data because of the local nature of its hidden receptive fields.
13. An RBFN with normalization may be able to sense beyond the training data set.



$$\text{Output} = \frac{\omega_1 c_1 + \omega_2 c_2}{\omega_1 + \omega_2}$$

$$= \bar{\omega}_1 c_1 + \bar{\omega}_2 c_2$$

$$c_1 = p_1 x + q_1 y + r_1$$

$$c_2 = p_2 x + q_2 y + r_2$$

Figure 5.8(a) Sugeno (TSK) Fuzzy model.

(14)

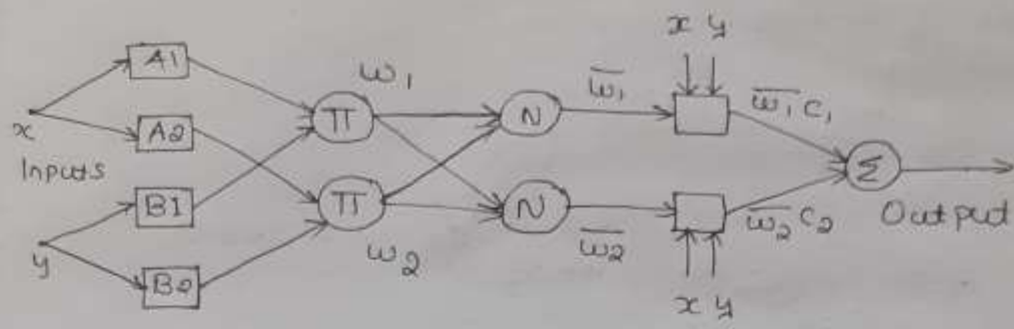


Figure 5.8 (b) ANFIS/CANFIS

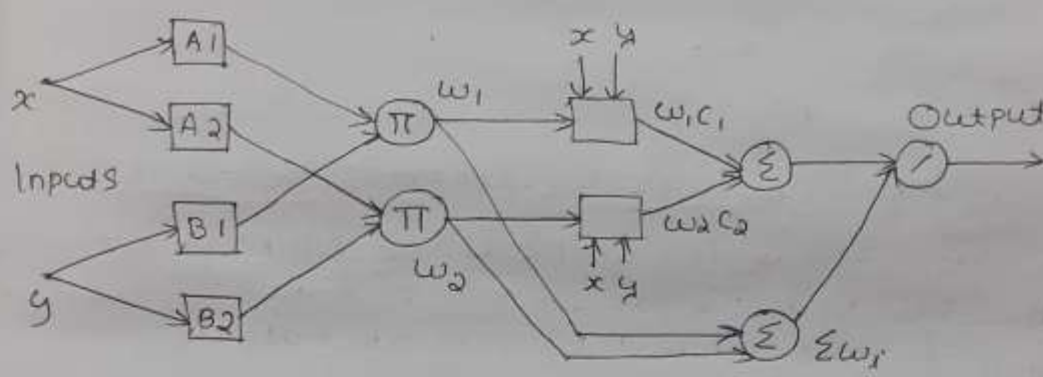


Figure 5.8 (c) ANFIS/CANFIS

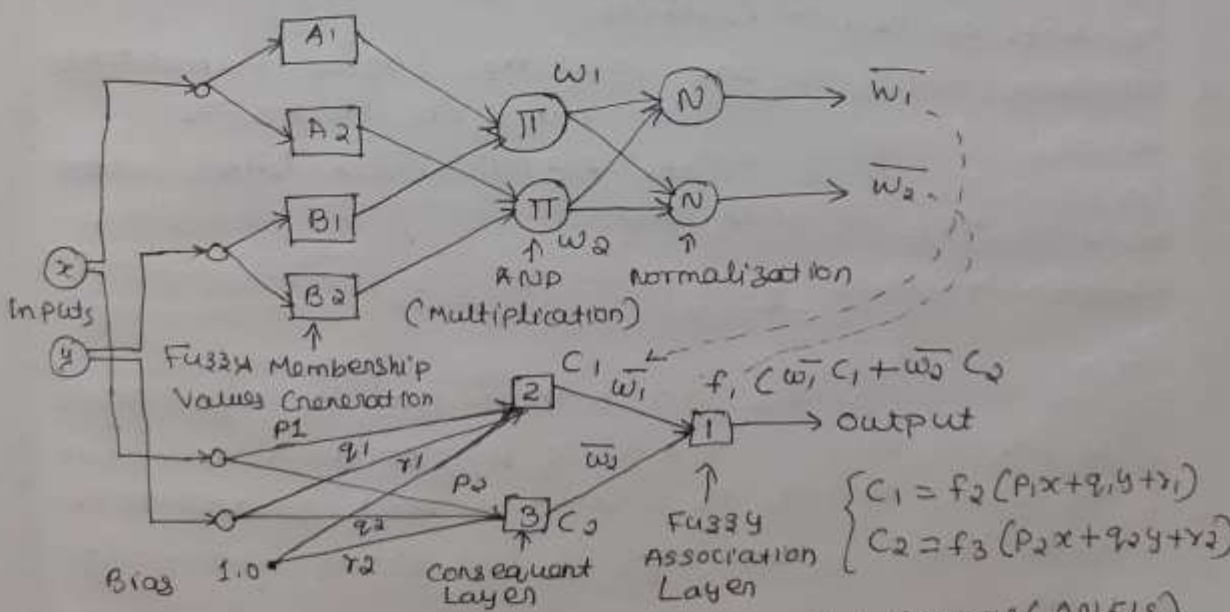


Figure 5.8 (d) CANFIS(ANFIS)

15

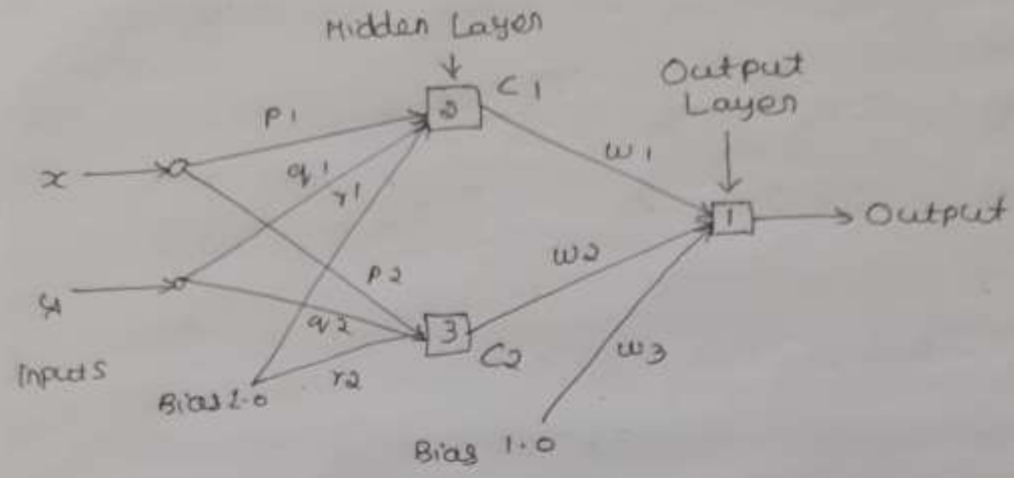


Figure 5.8(e). MLP.

NEURON FUNCTIONS FOR ADAPTIVE NETWORKS

1. Diversity of neuron functions for adaptive networks.
2. Fundamental design aspects for rule formation in CANFIS in light of RBFN and modular network features.
3. In a truly adaptive network there should be no constraints on neuron functions.
4. Various function are used as basis functions for alternative Gaussian functions in RBFNs.
5. Neuron functions in the RBFN hidden layer correspond to MFs in CANFIS.

FUZZY MEMBERSHIP FUNCTIONS VERSUS RECEPTIVE FIELD UNITS

1. Figure 5.9 provides an anatomical view of CANFIS in parallel with a corresponding RBFN with normalization, which divides the output of each neuron in the output layer by the sum of all basis functions outputs.

16

2. The procedure corresponds to the following output function :

$$O_j = \sum_{i=1}^{4} w_i C_{ji} \Big/ \sum_{i=1}^{4} w_i \quad (j=1,2).$$

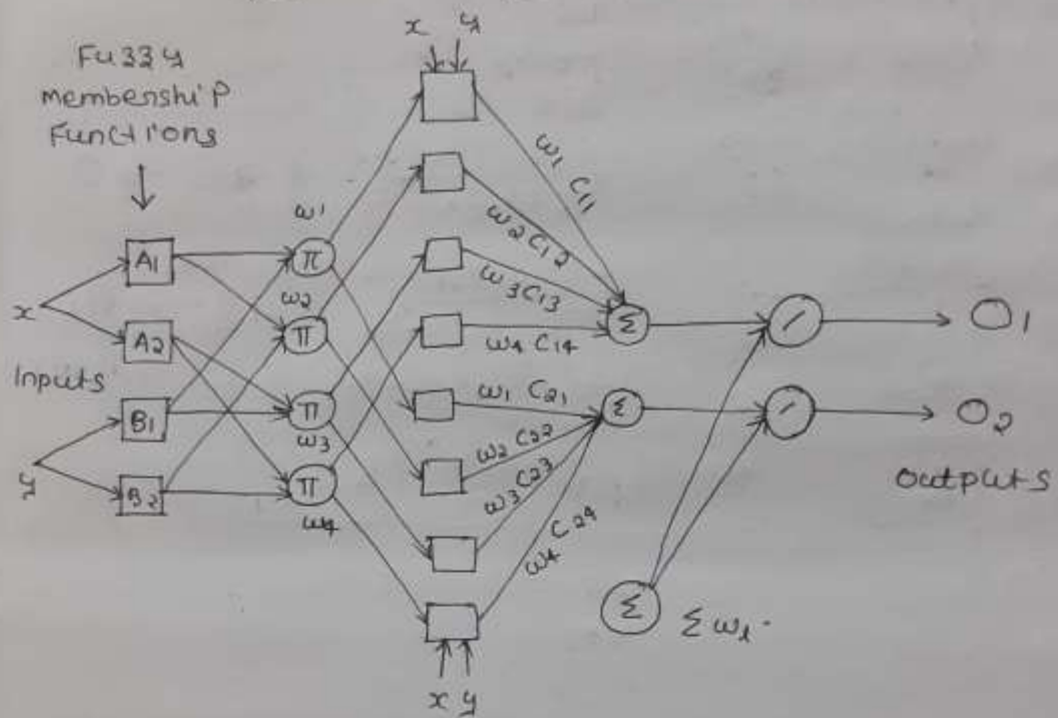


Figure 5.9(a) CANFIS

3. Note that in RBFN the hidden weights C_{ji} are expressed in the form of linear functions rather than just real numbers.

4. CANFIS can construct hyperellipses in higher dimensions through product operations, while RBFNs with Gaussian basis functions can form hyperspheres around their centers because inputs are plugged into the same basis functions, where inputs, x and y, both go to the same basis function, R_i.

17

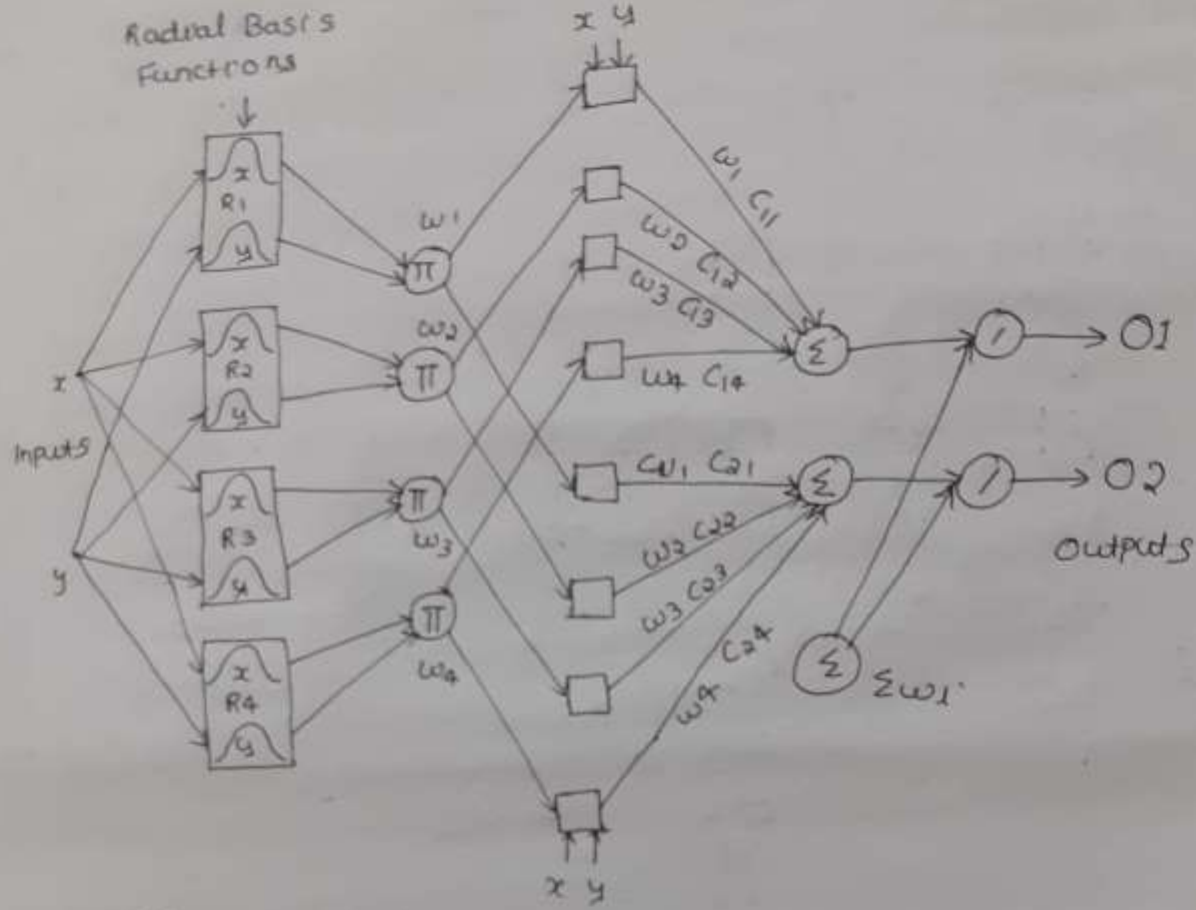


Figure 5.9 (b) RBFN

5. A Neural Network with semi-local activation hidden units (or Gaussian-bar hidden units), which can attain convergence performance comparable to RBFNs. The output response of such an ith unit is given by

$$(w_i =)\ B_i(\| x - u_i \|) = \sum_j P_{ij} \exp\left[- \frac{(x_j - u_{ij})^2}{2\sigma_i^2} \right],$$

where P_{ij} is a positive parameter.

6. By comparison with this summation unit, the Gaussian hidden unit in an RBFN can be regarded as a product unit.

(18)

$$(w_i =) B_i(\|x - u_i\|) = \Pi_j \cdot \exp\left[-\frac{(x_j - u_{ij})}{2\sigma_i^2}\right].$$

Membership Functions in FIS

1. $$\mu_{bell}(x) = \frac{1}{1 + \left|\frac{x-c}{a}\right|^{2b}},$$

2. $$\mu_{modbell}(x) = \max\left\{\frac{2}{1 + \left|\frac{x-c}{a}\right|^{2b}} - 1, 0\right\}$$

where $\{a, b, c\}$ is an adjustable parameter set.

3. Asymmetric functions are common in FIS, such as the following two-sided Gaussian MF:

$$\mu_{tsg}(x) = \begin{cases} g_1(x) & \text{if } x \leq c_1, \\ g_2(x) & \text{if } x \geq c_2, \\ 1 & \text{otherwise}, \end{cases}$$

where $g_i(x)$ is the Gaussian function:

$$g_i(x) = \exp\left[-\frac{(x-c_i)^2}{2a_i^2}\right],$$

and $\{a_i, c_i\}$ are modifiable parameter.

The more sophisticated MFs tend to have more modifiable parameters. Local tuning of parameters within such MFs may be necessary; one of the simplest means of local tuning is to fix center parameters and update shape parameters in the training phase. Another local tuning method uses different learning rates for adjusting those parameters.

(17)

Nonlinear Rule

1. This focus on neuron functions at the consequent layer, such as f_2 and f_3; they form the consequent parts C_1 and C_2 in Figure 5.8(d).
2. In ANFIS f_2 & f_3 are identity functions.
3. When f_2 and f_3 are replaced with non linear functions, we will get nonlinear consequents.
4. The neuron functions in the consequent layer play an important role in rule formations.
5. suppose a neuron function is a sigmoidal function in the consequent layer. Then the nonlinear consequent C_{non}.

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

So a sigmoidal rule.

6. When each rule's consequent is realised by a Neural Network we have neural rules. Figure 5.10(c)
7. In figure 5.10(a)(b) - Note that when the four consequent NNs have sigmoidal output neuron functions with no hidden layers, the four neural rules are reduced to sigmoidal rules. Figure 5.10(a) and Figure 5.10(b) are identical.
8. When two neural consequents, "Neural $Rule_1$" and "Neural $Rule_3$" are combined to form one neural rule (i.e., Local Expert NN_1), and

(20)

"Neural Rule 3" and "Neural Rule 4" are fused into another neural rule (i.e., Local Expert NN_2) – then the network is known as modular network. Figure 5.10(d).

9. CANFIS with neural rules can be equivalent to modular networks.
10. The central idea resides in task decomposition.
11. Another training approach is to train both antecedent and consequent parts concurrently.
12. If backpropagation (steepest descent) apply to CANFIS, the procedure to minimize a sum of squared errors, E, is straight forward;

a) Let O_j and F_j be the jth CANFIS output and the jth neuron function at the final output layer, respectively, as depicted in Figure 5.11.

b) $O_j = F_j(NET_j)$,

where NET_j denotes net input.

c) The procedure for updating the mth rule's consequent, which has a weight coefficient signified by w_{mki}, is as follows:

$$\Delta w_{mki} = -\eta_{wm} \frac{\partial E}{\partial w_{mki}}$$

where η_{wm} is a learning rate.

(21)

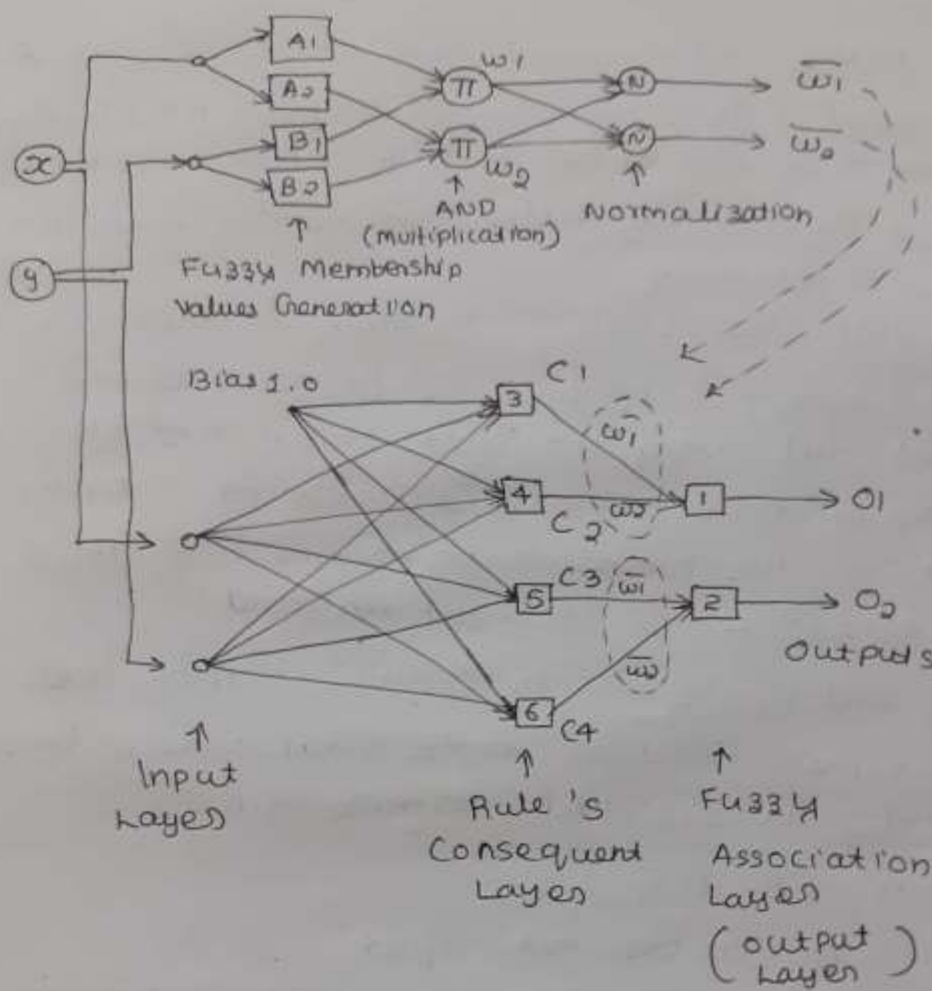


Figure 5.10 (a) CANFIS with two linear/sigmoidal rules per output.

22

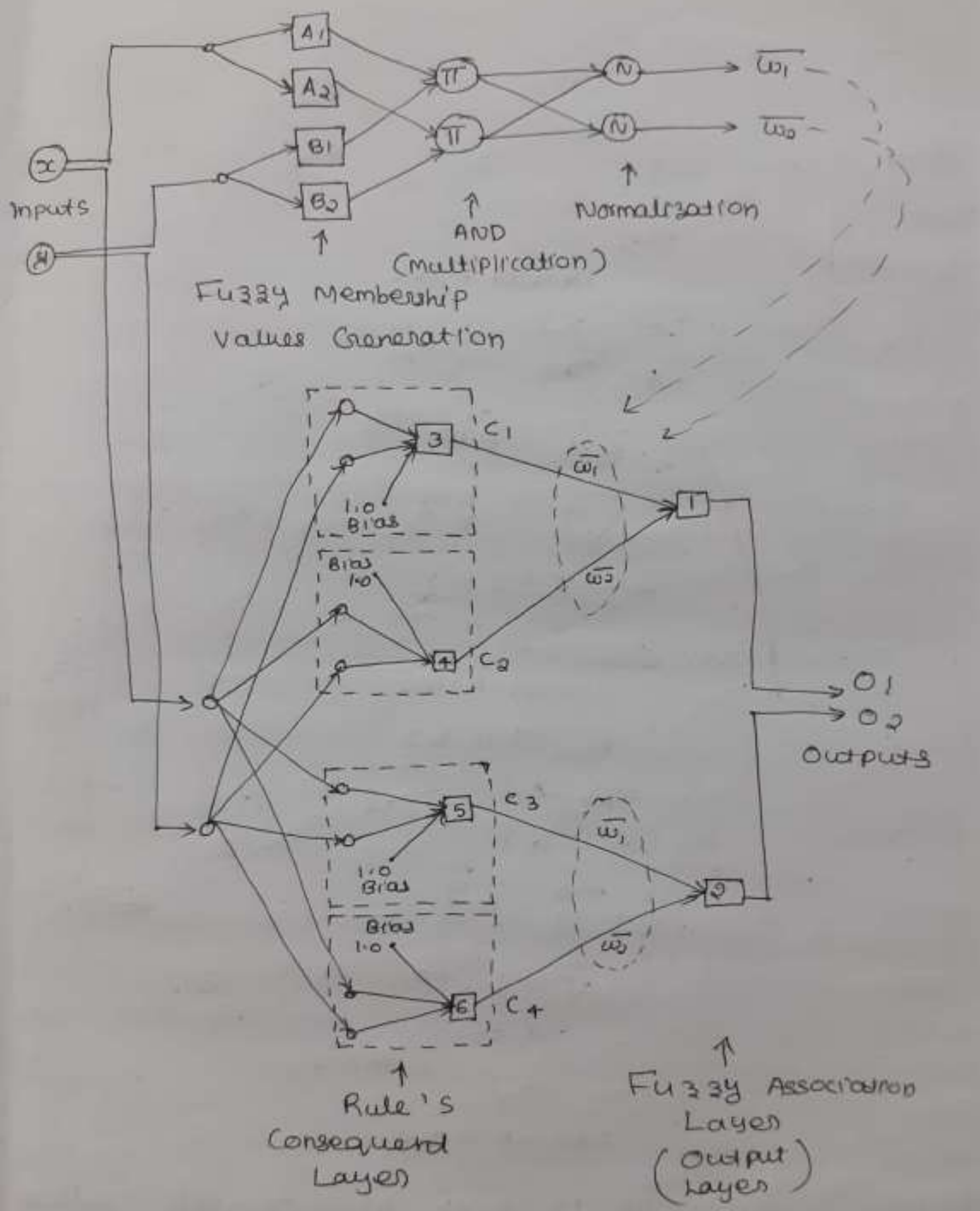


Figure 5.10(b) CANFIS with two linear/sigmoidal rules per output.

(23)

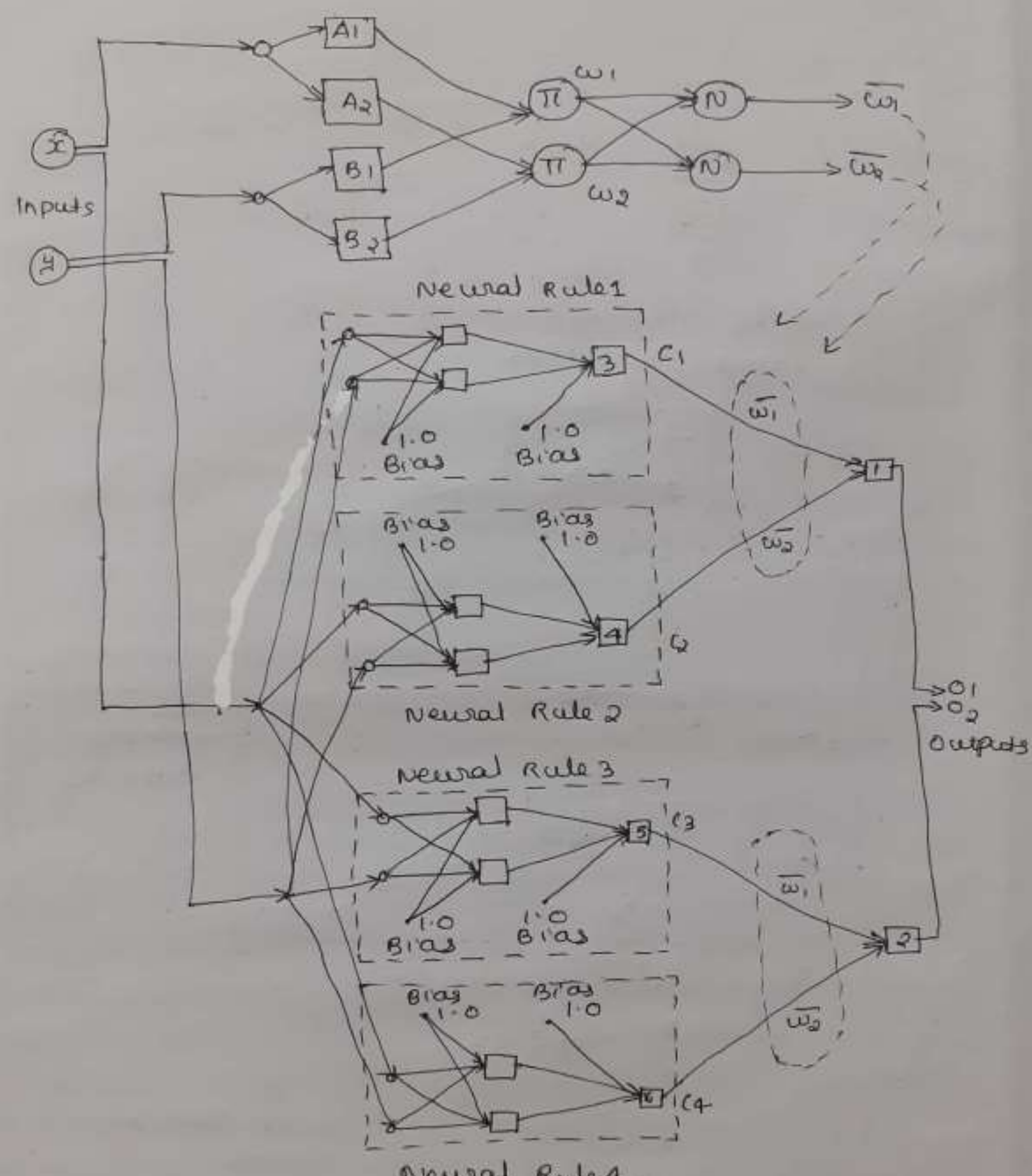


Figure 5.10(c) CANFIS with two neural rules per output.

(24)

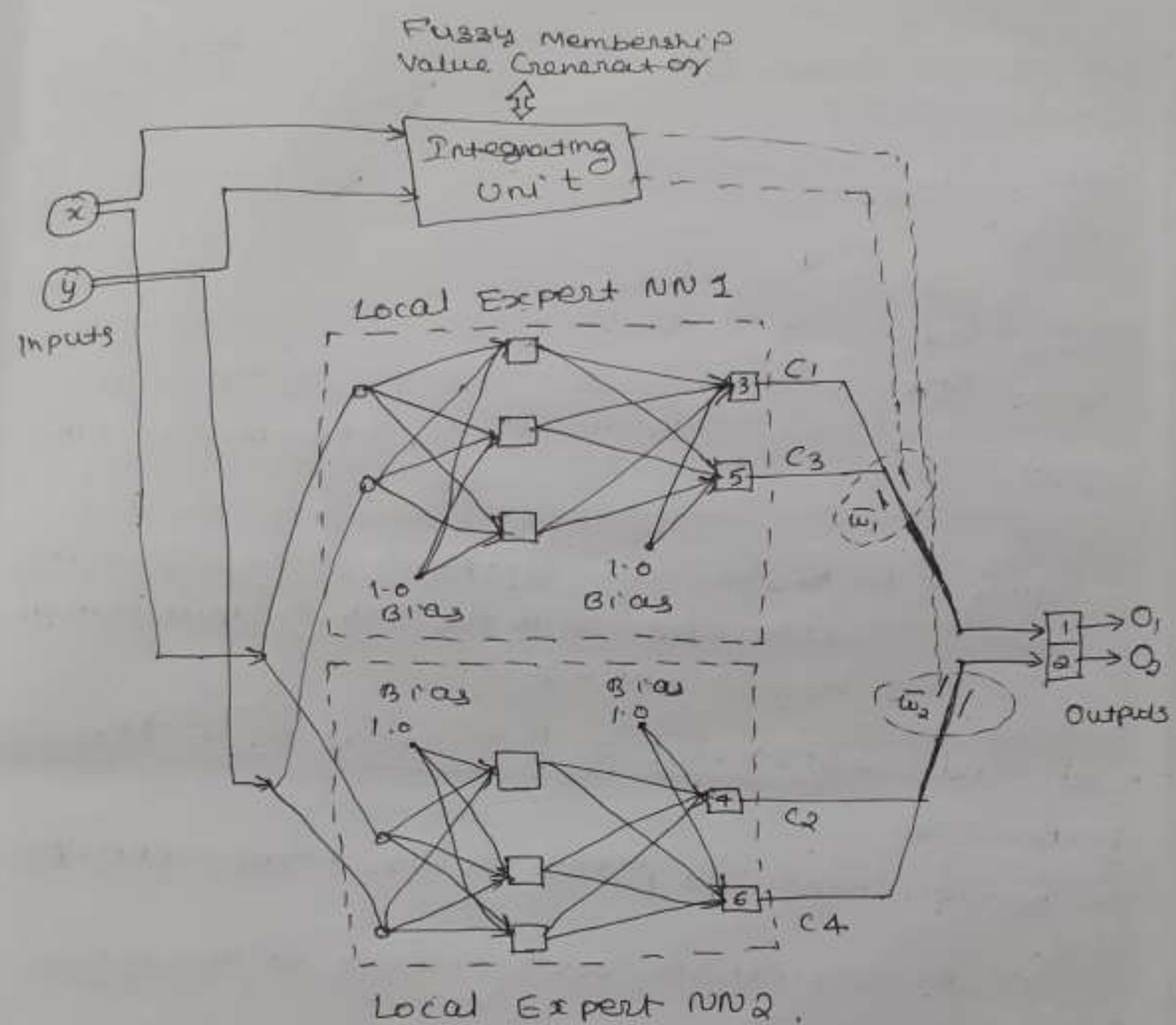


Figure 5.10 (d) Comparable typical modular network.

25

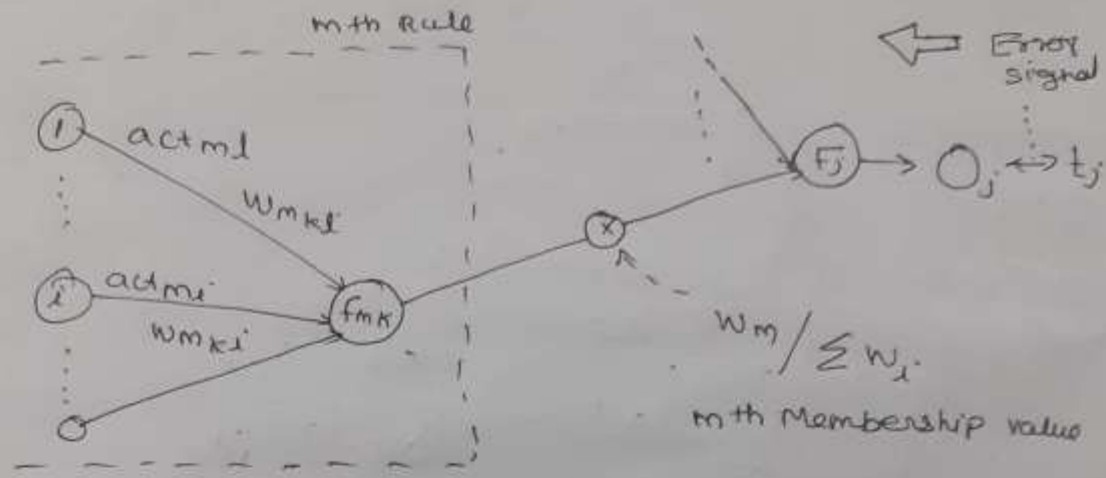


Figure 5.11 Anatomical graph of the mth rule associated with the jth output neuron Fj, in CANFIS.

Modified Sigmoidal and Truncation Filter Functions

1. The neuron functions placed at the final output layer.
2. The fuzzy association layer such as f1 in Figure 5.8(d).
3. Introduces a modified sigmoidal function and a truncation filter function.
4. ANFIS typically possesses the identity function as f_1.
5. 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.

26

6. A modified sigmoidal function, fmod, and a truncation filter function f_{trc} are introduced as neuron functions for the output layer.
i.e:

$$f_{trc}(x) = \begin{cases} MIN & \text{if } x \leq MIN \\ MAX & \text{if } x \geq MAX, \\ x & \text{otherwise} \end{cases}$$

$$f_{mod}(x) = \begin{cases} MIN & \text{if } f(x) \leq MIN \\ MAX & \text{if } f(x) \geq MAX, \\ f(x) & \text{otherwise.} \end{cases}$$

where $f(x)$ is the normal sigmoidal logistic function

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

This improvement keeps neuron outputs within the desired output range, [MIN, MAX]. MIN is set to 0.1 and MAX to 0.9, and outputs that are above MAX or below MIN are then forced to MAX or MIN.
These functions, $f_{trc}(x)$ and $f_{mod}(x)$, are easily implemented and can surely help an NN to learn the boundaries of the output range.

NEURO-FUZZY SPECTRUM

1- Explains the concept of neuro-fuzzy spectrum in terms of the tradeoffs between input-output mapping precision and membership function (MF)

27

interpretability from the fuzzy logic standpoint.

2. Neuro-fuzzy models allow prior knowledge to be embedded via fuzzy rules with appropriate linguistic labels, and they offer the possibility of understanding the resultant models after learning.

a) This observation motivates the concept of neuro-fuzzy spectrum, which is defined on the interpretability-precision plane depicted in Figure 5.12.

3. Ideally, the learning of a neuro-fuzzy model should follow the vertical route to the top in such a way that the mapping precision is being improved while the interpretability maintained.

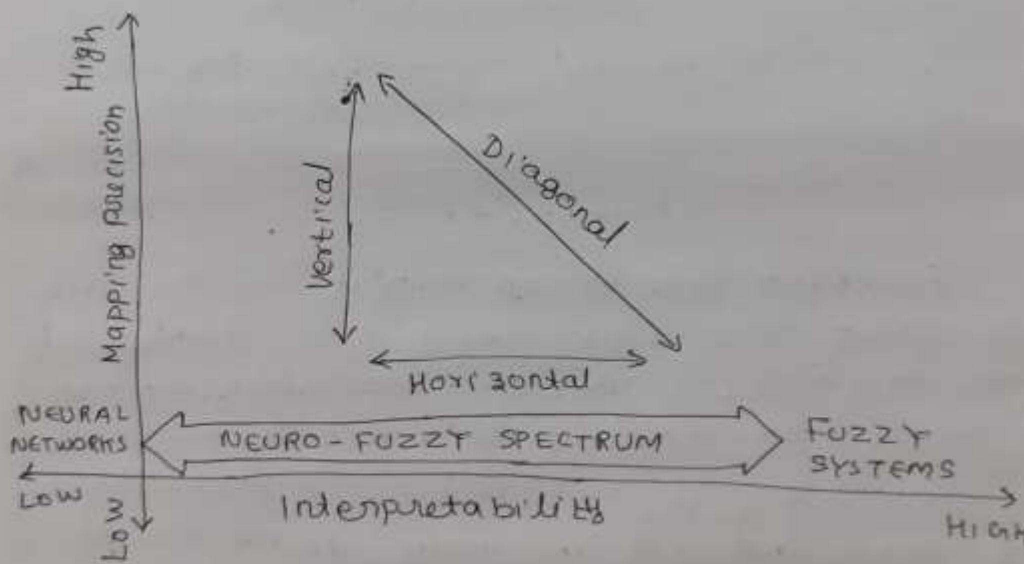


Figure 5.12 Neuro-Fuzzy spectrum. The plane's axes denote neuro-fuzzy spectrum (horizontal route) and input-output mapping precision (vertical route). Ideally, the learning of a neuro-fuzzy model should follow the vertical route to the top, but it often takes the diagonal route of improving

(28)

mapping precision at the expense of interpretability.

4. Dilemma between interpretability and precision.
The learning process often follows the diagonal route of improving mapping precision and deteriorating interpretability at the same time. This situation is refered as the dilemma between interpretability and precision.

5. Adaptive neuro-fuzzy models like ANFIS/CANFIS transit smoothly between the two ends of neuro-fuzzy spectrum: a completely understandable fuzzy inference system and a black-box neural network.

6. Various Approaches to avoid dilemma between precision & Interpretability

i. A given ANFIS/CANFIS structure can be interpreted from different viewpoints, regardless of calling it a fuzzy system.

ii. change MF types or adopt a more sophisticated asymmetric MFs, such as a two-sided bell MFs.

iii. Modify the learning algorithms that maintain MF interpretability.

iv. Alter fuzzy rules structures by setting up nonlinear rules.
CANFIS with neural rules may have less chance to lose MF interpretability than linear rules, because neural rules have more learning power than linear counterparts.

v. put proper constraints on neighboring MFs so that resultant MF interpretability can be retained. The most simplest way is to apply some knowledge to fixing the center positions of MFs.

29

vi. Formulate a new error measure designed to increase interpretability, such as an error measure with a term similar to Shannon's information entropy.

vii. Transform the input space to another space, in which input values can be treated in a linguistically meaningful way.

30

MR402 Module VI

PRINTED CHARACTER RECOGNITION

1. Describes a straightforward design method for a fuzzy inference system to solve pattern recognition.
2. Iterated training is not mandatory for this design method.
3. The method does require some representative, noise-free data points from the recognition system to be modeled.

Exclusive-OR (XOR) problem

1. The Exclusive-OR (XOR) problem to demonstrate the concept behind the design method.
2. To solve a binary XOR problem, need to classify a binary input vector to class 0 if the vector has an even number of 1s; otherwise, it is assigned to class 1.
3. The desired behavior of the two-input XOR problem is described by the following truth table:

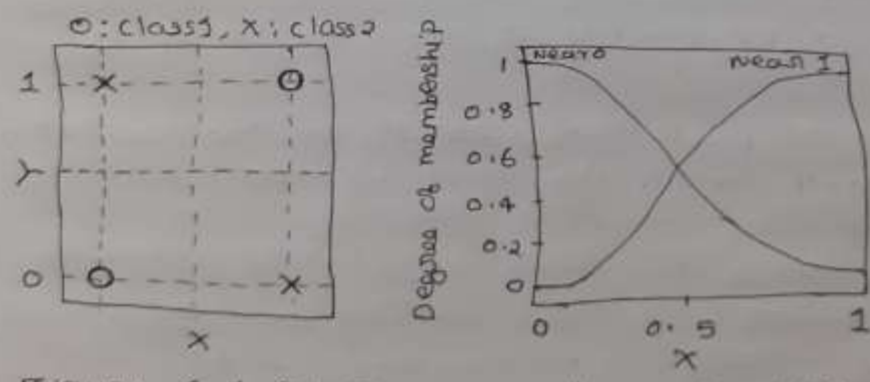


Figure 6.1 (a) Training data for XOR problem (b) Z and S membership functions for "near 0" and "near 1" respectively.

01

	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 pair4	1	1	0

4. From the training data plot in Figure 6 1(a), it is Obvious that the XOR problem is not linearly seperable and cannot be solved by a single-layer perceptron.

5. To use an MLP (multilayer perceptron) with a hidden layer to solve it, we need to train the network.

6. How to train the network

i) By noting that the training data are representative and noise free, it can be used as prototypes for the fuzzy logic design approach based on nearest-neighbor classification or case-based reasoning.

ii) For a given set of prototypes, the underlying rationale for classifying a new data point is simple:

a) Find the prototype nearest to the new data point and assign the point to that prototype class.

b) To do this, we need a similarity measure that quantifies the meaning of near.

c) This is done in terms of membership functions (MFs).

(2)

For eg: the meaning of "near 0" and "near 1"

iii. We still need to know the meaning of closeness between the input data [x, y] and one of the prototypes, say, [0, 1].

a) 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.

b) The most popular fuzzy AND operators are "product" and "min".

iv. Creating a fuzzy rule set for solving the XOR problem is

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.

v. In other words, if input data [x, y] is close to one of the prototypes, it is then assigned to that prototype class.

Now we can move on to a more challenging problem: printed character recognition (PCR)

1. In PCR each of 26 letters is defined as a 7×5 pixel matrix.
2. The challenge is to build a fuzzy inference system that can classify a given set of 35 (= 7×5) pixels to one of the 26 alphabet characters.
3. These 26 prototypes are noise free, and we can employ the concept referred to previously in designing a fuzzy inference system.

(3)

i. 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" in the same way as in Figure 6.1(b)

ii. Set up rules.

Each prototype represents a rule,
So we have 26 rules,
each of them an AND rule with 35 preconditions
Each rule's output is not critical, and we can set it to be an arbitrary constant or MF.

iii. Use the fuzzy inference system.

Distance measure information is embedded in each rule's firing strength - the larger the firing strength of a rule is, the closer a given input is to the prototype of that rule. Therefore, we obtain 26 firing strengths; the alphabet corresponding to the maximal firing strength is then selected as the predicted class.

4. To test the fuzzy inference system, we can assign various noise levels to the input Pattern.

5. The fuzzy PCR system thus obtained performs comparably to a similar system using an MLP (multilayer perceptron).

6. Advantages of PCR

a) It does not required any training.

b) It is a knowledge representation, and each rule in the system represents our insight into the problem we want to solve.

7. In this approach, we did not use any optimization schemes.

(4)

8. we can use derivative-based optimization techniques if the described approach fails to classify noisy characters recognizable by humans correctly.

9. since the described method already gives us a roughly correct fuzzy inference system, the training time required to fine-tune membership functions is likely to be much shorter than that for an MLP starting with random weights.

10. Training a fuzzy inference system for pattern recognition is not exactly the same as the ANFIS.

11. The fuzzy inference system are only interested in the firing strengths, not the final outputs after weighted average defuzzification.

INVERSE KINEMATICS PROBLEMS

1. Use ANFIS to model the inverse kinematics of the two-joint planar robot arm shown in Figure 6.2.

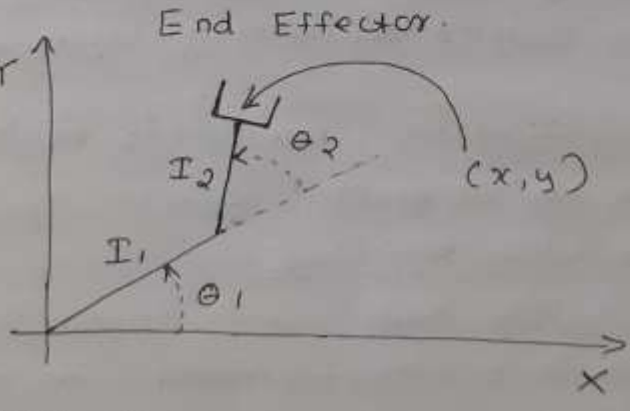


Figure 6.2 Two-joint planar robot arm.

5

2. This problem involves learning to map from an endpoint Cartesian position (x, y) to joint angles (θ_1, θ_2), and it requires that the end effector ("head "hand") be able follow the reference signal without being given the joint angles.
3. The forward kinematics equations from (θ_1, θ_2) to (x, y) are straightforward:

$$\begin{cases} x = l_1 \cos(\theta_1) + l_2 \cos(\theta_1 + \theta_2), \\ y = l_1 \sin(\theta_1) + l_2 \sin(\theta_1 + \theta_2), \end{cases}$$

where l_1 and l_2 are arm lengths; and θ_1 and θ_2 are their respective angles.
The inverse mappings from (x, y) to (θ_1, θ_2) are not as clear.
4. It is possible to find the inverse mappings algebraically, but the solutions are not generally available for a multiple-joint robot arm in 3-D space.
5. Instead of solving the equations directly, we use two ANFIS systems to learning ~~these~~ inverse mappings.
6. When $\sqrt{x^2+y^2}$ is greater than $l_1 + l_2$ or less than $|l_1 - l_2|$, there is no corresponding (θ_1, θ_2). This is called the unreachable workspace.
7. The form of training data pairs is (x, y, θ_1) and (x, y, θ_2), respectively, to train two ANFIS systems.
8. Three MFs are used for each input.

6

9. There were nine rules and 45 parameters for each ANFIS.

AUTOMOBILE MPG PREDICTION

1. Describes the use of ANFIS for nonlinear regression.
2. Address the issue of input selection for finding important input variables and reducing training data dimensions.
3. Automobile MPG (miles per gallon) prediction is used as a case study.
4. An automobile's fuel consumption in terms of MPG is predicted by ANFIS based on several given characteristics, such as number of cylinders weight, model years, and so on.
5. Six input attributes includes profile information about the automobiles:

   No. of cylinders : multi-valued discrete
   Displacement : continuous
   Horse power : continuous
   Weight : continuous
   Acceleration : continuous
   Model year : multi-valued discrete.

6. The attribute to be predicted in terms of the preceding six input attributes is the fuel consumption in MPG.
7. To apply ANFIS to MPG prediction, we need to take care of two problems first.

   i) Data scarcity
   ii) Input space partitioning.

7

i. Data scarcity:

a) For a single-input data-fitting problem of medium complexity, we need 10 data points to come up with a good model.

b) For two input - $10^2 = 100$ data points.

c) For a six-input - $10^6 = 1{,}000{,}000$ data points.

This is prohibitively large for any common modeling problem, data instances may be very less.

Eg: If we have only 392 data instances then $\sqrt[6]{392} = 2.5$ data points for single-input data fitting.

d) This data scarcity dilemma is ubiquitous in multivariate regression.

e) A commonly used solution is to divide the data set into training and test data sets; the training set is used for model building, while the test set is used for model validation.

ii. Input space partitioning:

a) Grid partitioning is the most frequently used input partitioning method.

b) For a problem with six inputs, grid partitioning leads to at least $2^6 = 64$ rules, which results in $(6+1) \times 64 = 448$ linear parameters if we are using first-order Sugeno fuzzy model.

8. Before training a fuzzy inference system, divide the data set into training and test sets.

i. The training set is used to train (or tune)

(8)

a fuzzy model, while the test set is used to determine when training should be terminated to prevent overfitting.

9. ii. The 392 instances are randomly divided into training and test sets of equal size (196).

9. If only two most relevant inputs are selected as predictors, then we can cycle through all the inputs and build $C_2^6 = 15$ fuzzy models.

10. Usually the test error is used as a true measure of the model's performance; therefore, the best model we can achieve occurs when the test error is minimal.

11. The model is expressed as

$$MPG = a_0 + a_1 * cyl + a_2 * disp + a_3 * hp + a_4 * weight + a_5 * accel + a_6 * year,$$

with $a_0, a_1, \ldots, a_6$ being seven modifiable linear parameters.

12. The optimum values of these linear parameters were obtained directly by the least - squares method.

13. The linear model takes all six inputs into consideration, but the error measures are still high since MPG prediction is nonlinear.

14. Input selection technique of choosing the two most relevant inputs can result in a nonlinear mapping with lower error measures.

(9)

SOFT COMPUTING FOR COLOR RECIPE PREDICTION

1. Color recipe prediction is the application of soft computing in the paint industry.
2. Introduces a neuro-fuzzy methodology and another computational intelligence approach that combines a knowledge base (KB) and three principal soft computing components:
   i) fuzzy systems (FSs),
   ii) neural networks (NNs), and
   iii) genetic algorithms (GAs).

COLOR RECIPE PREDICTION

1. In a practical situation, it is necessary to examine the color match in daylight as well as in artificial light.
2. 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 (Figure 6.3).

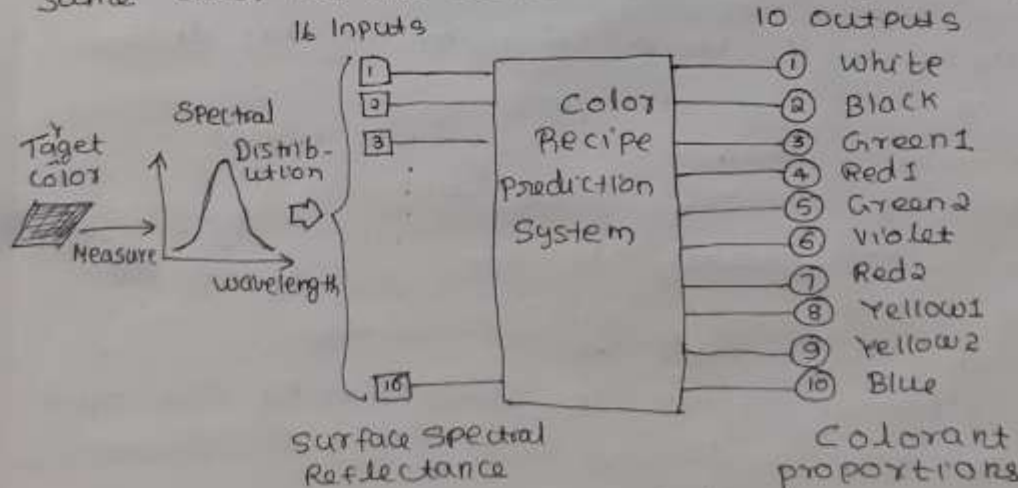


Figure 6.3 Input-output relation in a typical color recipe prediction system.

(10)

3. A succinct description of the main concerns in the recipe prediction is presented in Table 6.1.

Table 6.1 Main concerns in color recipe prediction.

(P1) It is difficult to predict precise colorant concentration. We sometimes need to predict proportions with enough precision to specify levels such as 0.01%, which is the desired minimal colorant proportion level.
(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.
(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. For instance, if the target color is very bright, we have to determine carefully the concentrations of dark-colored pigments.
(P4) It is important to consider human visual sensitivity to color difference, which is closely related to perceptual attributes of color (i.e, lightness, hue and Chroma.
(P5) Some different combinations of colorants may have the same perceptual attributes of color as seen by humans.

(11)

CANFIS MODELING FOR COLOR RECIPE PREDICTION

1. How neuro-fuzzy models can be generalized for application to color recipe prediction, the neuro-fuzzy approaches are expressed within the frameworks of CANFIS (Coactive Neuro-Fuzzy Inference Systems).

1. Fuzzy Partitionings

i. In fuzzy modeling, it is important to determine a reasonable number of membership functions (MFs) to maintain appropriate linguistic meanings.

ii. The color recipe prediction problem has 16 surface spectral reflectance inputs and 10 colorant proportion outputs as depicted in Figure 6.3.

iii. When we pick 16 values ($x_1, \ldots, x_{16}$) from the surface spectral reflectance curve of a given target color.

iv. we have the following 16 fuzzy rules:

Rule 1 : If x_1 (at 400nm) is A_1, then use a rule, C_1.

Rule 2 : If x_2 (at 420nm) is A_2, then use a rule, C_2.

. . .

Rule 16 : If x_{16} (at 700nm) is A_{16}, then use a rule, C_{16}

A_i is a fuzzy linguistic label.

v. The visible color spectrum is 400 nm to 700nm.

vi. without explicit domain knowledge, adaptive learning mechanisms enable ANFIS/CANFIS to build up fuzzy rules automatically.

vii. There is a formula for transforming the surface spectral reflectance of color to perceptual attributes, "lightness", "hue", and "chroma".

viii. These three values must be more suitable for treating color in a linguistically meaning-

(12)

ful way than the 16 spectral values.

2. CANFIS ARCHITECTURES

1. Using hue alone, it is possible to build up fuzzy MFs on the polar coordinates that define five color regions: red, yellow, green, blue, and violet (Figure 6.4).
2. Fuzzy rules in the if-then format serve to determine color selection. For instance,
   Yellow rule: If the target color is "yellow", then use a "yellow" rule, C_y.
3. Each color MF specifies the degree of membership of a color region and assigns the degree value to each color rule (rule's consequent) as the firing strength.
4. In the preceding yellow rule, the firing strength (W_y) is determined by the yellow MF.
5. Consider a case in which each color region has three MFs to express its three degrees of color.
   eg: The rules for the yellow region between green and red area:
   Yellow rule 1: If the target color is "greenish yellow", then use a "greenish yellow" rule, C_{gy},
   Yellow rule 2: If the target color is "very yellow", then use a "very yellow" rule, C_{vy},
   Yellow rule 3: If the target color is "reddish yellow", then use a "reddish yellow" rule, C_{ry}.
6. Instead of increasing MFs, we can construct more sophisticated rules' consequents, such as neural rules.
7. Figure 6.4 illustrates such a CANFIS with five color rules; one color MF is positioned for one color region.
8. The given prediction task is decomposed into five

(13)

color rules or five local color experts, which form rules' consequents.

9. In Figure 6.4, the "green rule" is expressed in a neural rule with 16 spectral reflectance inputs. Each rule can be a linear rule, a sigmoidal rule, or a neural rule.

10. Consider all three perceptual attributes of color - lightness, hue and Chroma to alleviate the problem (P4) in Table 6.1.

11. Set up three MFs for lightness, and Chroma, respectively, and five color MFs for hue.

12. The CANFIS with 45 Fuzzy rules is illustrated in Figure 6.5.

13. The CANFIS architectures have too many adjustable parameters. To accelerate learning, we can employ the modified bell MFs to control the number of firing rules (i.e., local experts).

14. The modified bell MF and the orginal bell-shaped MF are:

$$\mu_{mod}(x) = \max\left\{\frac{2}{1+\left|\frac{x-c}{a}\right|^{2b}} - 1, 0\right\}$$

$$\mu_{original}(x) = \frac{1}{1+\left|\frac{x-c}{a}\right|^{2b}},$$

where $\{a, b, c\}$ is an adjustable parameter set.

The modified bell MF is just the upper half part of the original bell-shaped MF and has a limited base width (support).

(14)

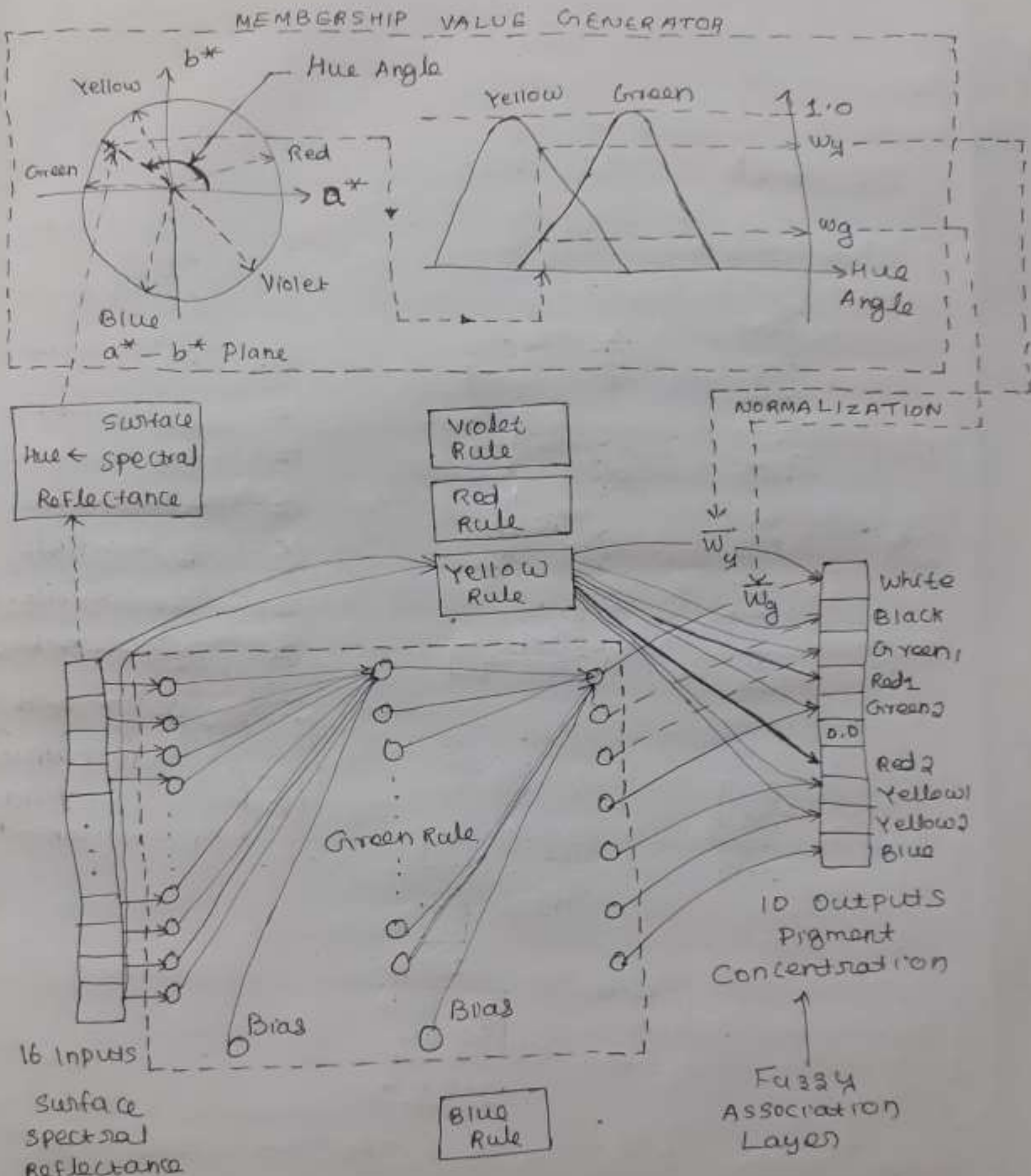


Figure 6-4 CANFIS with five color rules for color recipe prediction.

15

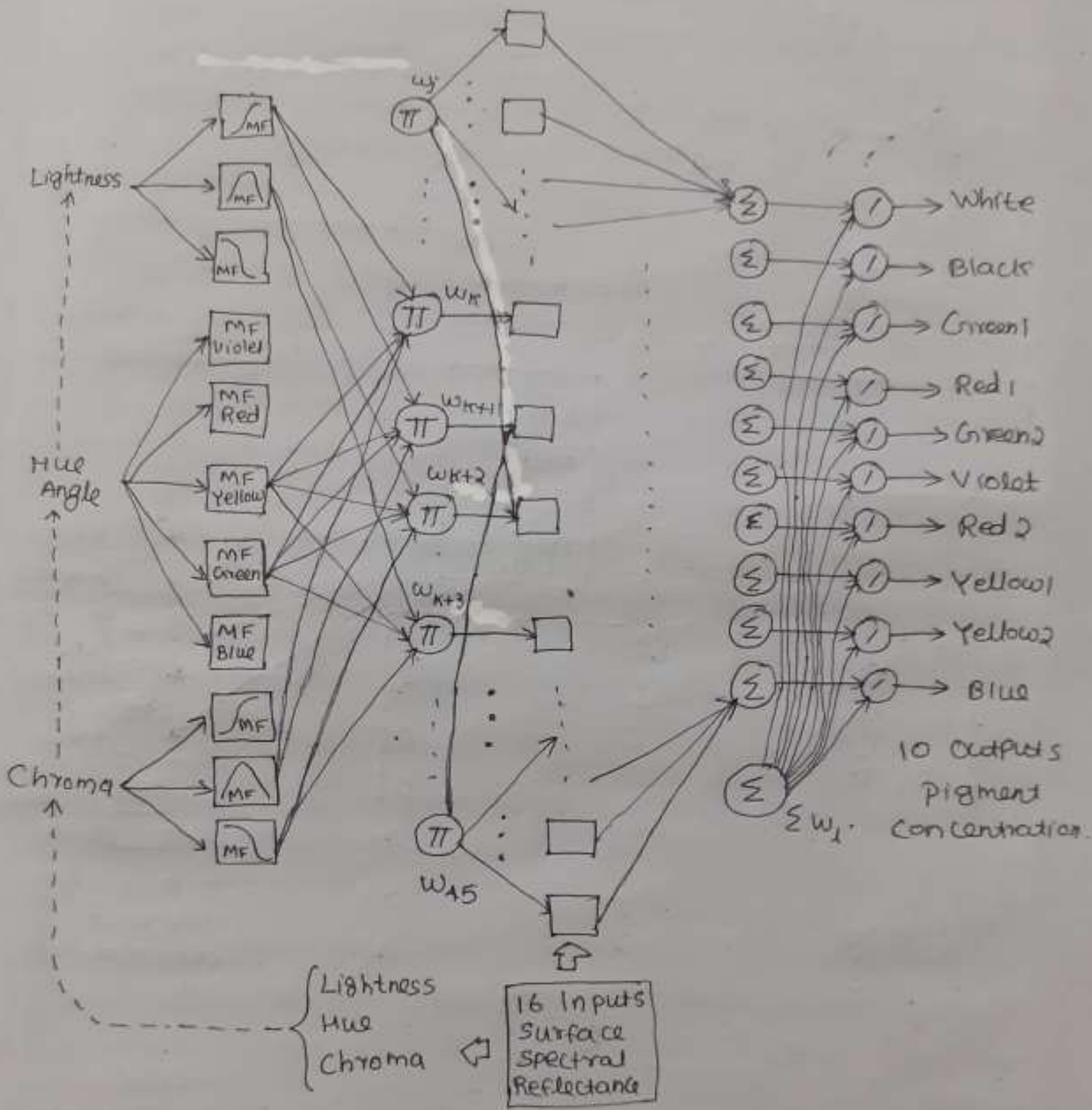


Figure 6.5 CANFIS with 11 MFs (45 fuzzy rules) for color recipe prediction.

16

3. KNOWLEDGE - EMBEDDED STRUCTURES

1. Adaptive fuzzy MFs specify the degree of membership of five color regions (red, yellow, green, blue, violet) according to perceptual attributes of color.
2. Adaptive fuzzy MFs determine what weight should be assigned to each rule's output to produce a final output.
3. Colorist's judgment is applied to the CANFIS architecture; Several Connections to the fuzzy association layer can be pruned.
4. In Figure 6.4, the green rule has no connection line to red units at the fuzzy association layer. That is, a green rule (weighted by a green MF) has no effect on red colorant proportions because of the green-red complementary color relationship.
5. The predicted number of colorants should be about four; this means that 6 of the 10 final outputs should be zero. (To eliminate the problems of (P1) and (P2))

4. CANFIS Simulation.

1. CANFIS with linear or non linear rules with different MF setups.
2. Table 6.2 shows five representative CANFIS descriptions in the simulation-
3. Truncation filter function is used in the output layer of CANFIS.
4. When the rule number was increased, difficulty in determining initial parameter setups was encountered.
5. If five neural rules CANFIS is used as in Figure 6.4 there may be many possible Optimal rule formations.

(17)

TABLE 6.2. Five representative CANFIS models for color recipe prediction.

(a)	CANFIS with 5 sigmoidal rules, as shown in Figure 6.4, with pruned connections 5 bell-shaped MFs are set up for hue angle alone (i.e., 5 rules are for five color regions)
(b)	CANFIS with 15 linear rules with no pruned connections 15 bell-shaped MFs are set up for hue angle alone
(c)	CANFIS with 45 rules, as shown in Figure 6.5, with no pruned connections 3 bell-shaped MFs are set up for lightness 3 bell-shaped MFs are set up for Chroma 5 bell-shaped MFs are set up for hue angle
(d)	CANFIS with 5 neural rules, as shown in Figure 6.4, with no pruned connections 5 modified bell MFs are set up for hue angle alone 5 neural color rules have the same model size (i.e., each neural rule has 22 hidden units)
(e)	CANFIS with five neural rules, as shown in Figure 6.4, with pruned connections 5 modified bell MFs are set up for hue angle alone 5 neural color rules are heuristically optimized independently.

5. COLOR PAINT MANUFACTURING INTELLIGENCE

1. The paint production environment around the prediction system has the color paint manufacturing cycle illustrated in Figure 6.6

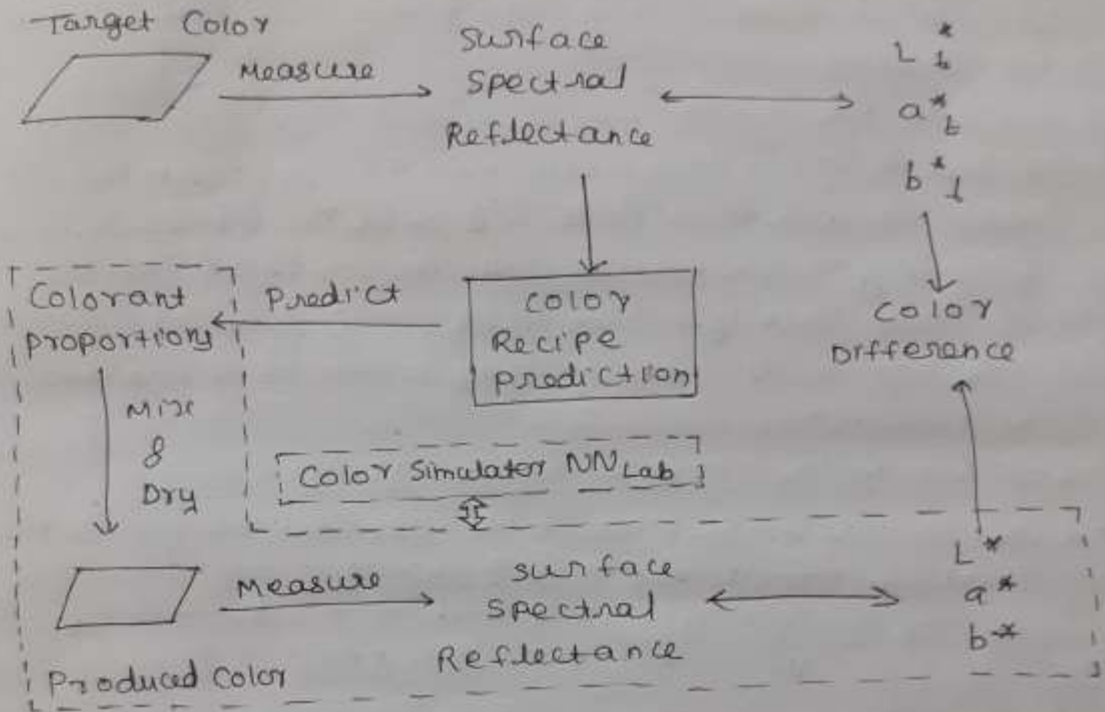


Figure 6.6 Color paint manufacturing process.

2. Basically, the main focus in recipe prediction should be color difference rather than colorant errors.
3. practically, the color difference between pairs of presented colors should be smaller than about 1.0; human eyes cannot distinguish between smaller color differences.
4. As summarized in Table 6.1, there are five major concerns in color recipe prediction.
5. It is important to consider perceived color difference during the prediction process; MLP and NNLab are used

(19)

to cope with the third critical concern (P4) in Table 6.1.

6. A cooperative hybrid system to simulate an entire manufacturing process in an attempt to construct manufacturing intelligence for the color paint industry.

7. Integrate the three major elements of soft computing and problem-specific knowledge. That is, NNs an FS, and a GA with a KB complement each other in obtaining more precise outputs for color recipe prediction through manufacturing simulation based on the entire decision-making process of a professional colorist.

5.1 MANUFACTURING INTELLIGENCE ARCHITECTURE

1. In the initial stage, the first-generation population or starting points for a GA search are set by a fuzzy population generator and a multi-elite generator using results from the CANFIS and NN approaches.

2. In the evolutionary phase, the fusion system tries to improve those encoded proportion members in conjunction with NNs and a KB; that is, two different NNs and a KB are used to make up the fitness function.

3. Genes' colorant concentrations are passed to three functions, which calculate fitness values individually. The three values are combined into the final fitness value. The evolutionary mechanism is illustrated in Figure 6.7.

(20)

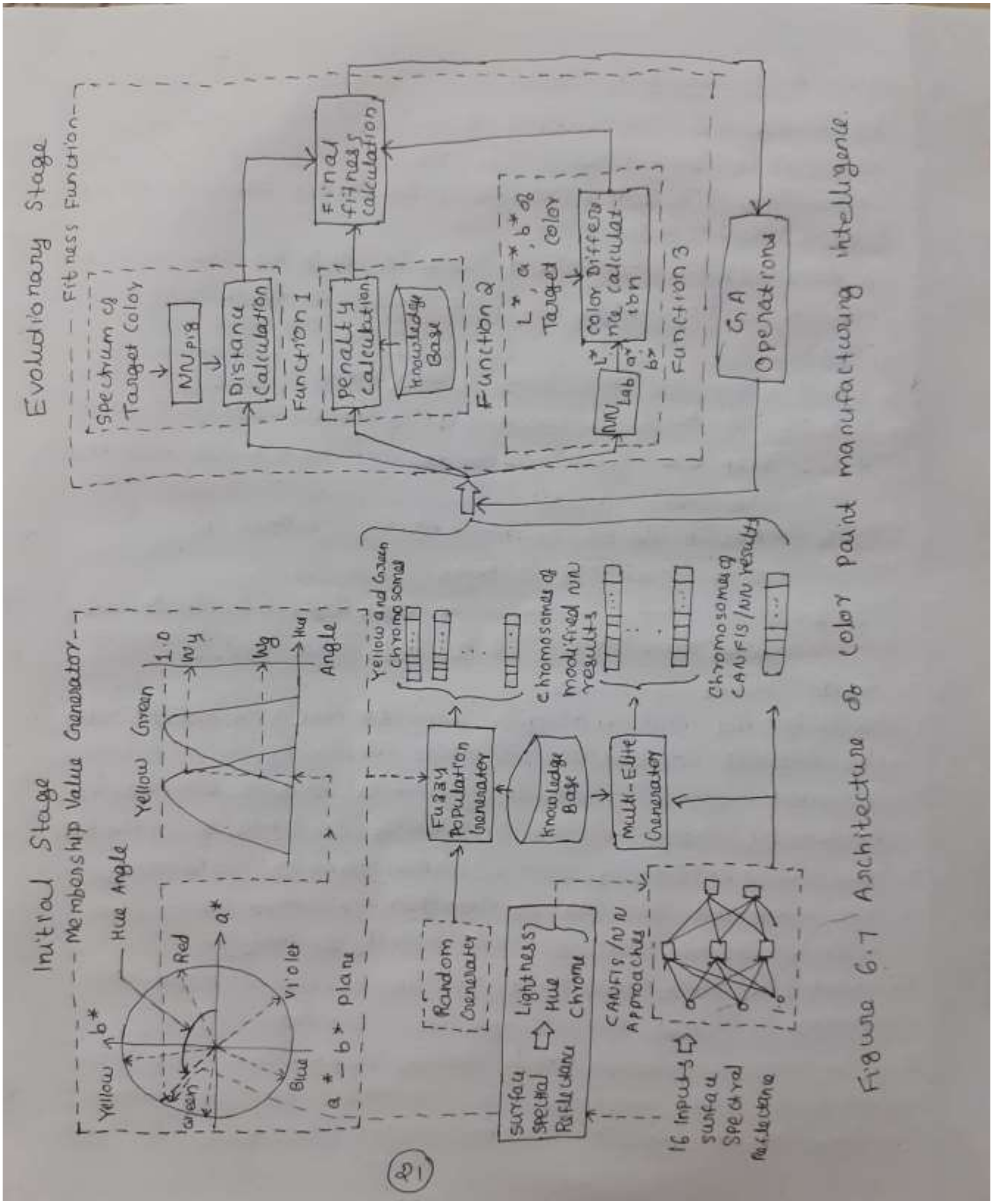


Figure 6.7 Architecture of color paint manufacturing intelligence.

5.2. Knowledge Base

1. Knowledge may be useful in reinforcing some favourable aspects of genetic searches.
2. Performing the color recipe prediction task requires special knowledge.
3. A KB plays an important role in helping the system evolve to recognize specific features of a target color.
4. The KB has the following main rules:

Rule 1: Keep total proportions of colorants around 100%,

Rule 2: Keep the number of necessary colorants around the ideal number,

Rule 3: Avoid use of complementary colorants: e.g., Red and Green,

Rule 4: Avoid use of the same type of colorants at the same time. (e.g., Red_1 and Red_2).

5.3. Multi-elites Generator

1. CANFIS and NN approach results are encoded into the initial population as elite members.
2. A multi-elite generator produces more elites by modifying those results according to rule 4 in the KB.
3. The concentrations of the same type of colorants are summed into one or another of them (e.g., $Red_1 + Red_2 \Rightarrow Red_1$, or $Red_1 + Red_2 \Rightarrow Red_2$).
4. Multiple elite colorant vectors offer several different starting points for GA searches.
5. The number of encoded elites depends on the quality of the CANFIS/NN results; we can take the results of three approaches (NN norm, NN mod,

(22)

CANFIS) and so we have at least three elite members at the initial stage.

5.4. Fuzzy Population Generator.

1. The idea is to generate the initial population according to the fuzzy classification of a target color, which serves to determine color selection.

a) Classify the target color into ~~the~~ one of five color categories (red, yellow, green, blue and violet) on the a*-b* plane, which shows hue and chroma

b) Decide to what extent the desired color belongs to each color category using fuzzy MFs.

c) generate initial color chromosomes by modifying chromosomes generated by a random number generator according to rules in the KB.

2. Eg. when a target color looks greenish yellow, The number of green chromosomes (Num_{Green}) and that of yellow ones (Num_{yellow}) are decided according to the following calculations:

$$Pop_{rest} = Pop_{total} - Pop_{NN},$$

$$Num_{Green} = \frac{M_g \, Pop_{rest}}{M_y + M_g},$$

$$Num_{yellow} = \frac{M_y \, Pop_{rest}}{M_y + M_g},$$

where two membership values, M_y and M_g, signify to what extent the target color belongs to the yellow category and the green one, respectively.

23

Pop_{total} denotes the total population number and Pop_{NN} signifies the number of elite chromosomes from the CANFIS/NN results, including the chromosomes generated by the multi-elite generator.

5.5 Fitness Function

The fitness function consists of three functions: two neural fitness functions (function 1 and function 3) and the KB-based fitness function (function 2).

Function 1

1. Using NN_{pig}, the first function evaluates genes' colorant concentration vectors according to the specified use of colorants.
2. The NN_{pig} (16 X 18 X 21 X 10 neurons) maps surface spectral reflectance to a list of required colorants (Figure 6.8).
3. It gives just ON/OFF values to each output unit to predict which colorants should be used to produce the same color as the target color, where ON means "colorant needed" and OFF means "not needed".
4. Function 1 evaluates each chromosome by calculating a distance in binary space (ON/OFF) after each chromosome's representation has been transformed into the ON/OFF format. Figure 6.8 describes this procedure

Function 2

1. The second function calculates a fitness value based on the KB.
2. The fitness value depends on the extent to which

94

genes' colorant concentration vector obeys the rules in the KB.

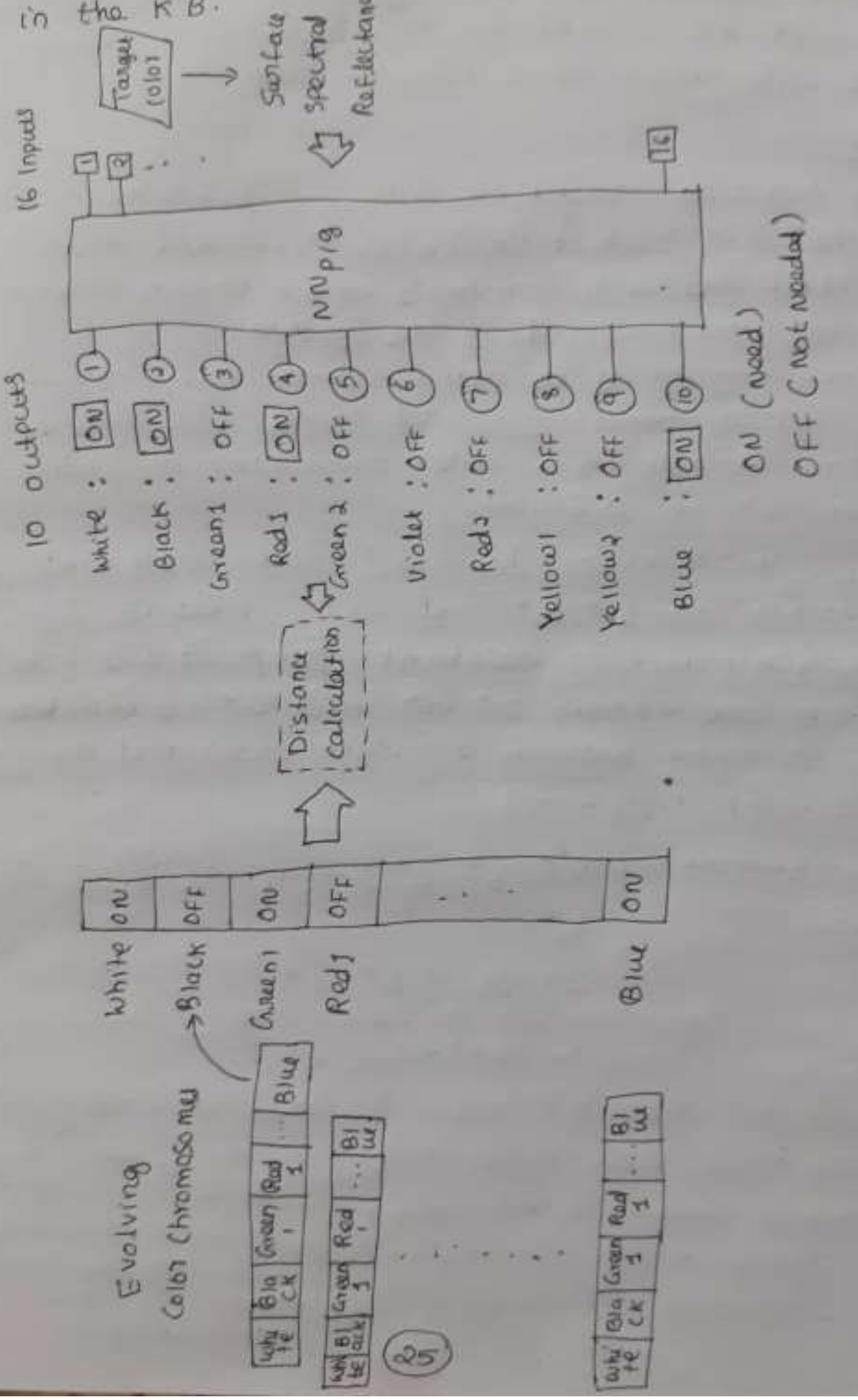


Figure 6.8 A component of the fitness function based on NNp18.

3. To keep the GA search moving in a consistent direction, the KB is used in both the initial stage and in the calculation of fitness values, as in Figure 6-7.

Function 3

1. The third function, based on NN_{Lab}, generates a fitness value with respect to color difference between a target color and each member's color, whose colorant concentrations are predicted by the system.
2. It is time-consuming to manufacture an actual color by mixing genes' specified values, the NN_{Lab} plays a crucial role as a color simulator to predict what color will be produced.
3. The NN_{Lab} (10 x 11 x 14 x 3 neurons) maps colorant concentrations to L^*, a^*, and b^*, that is, by plugging each member's colorant proportions into NN_{Lab}, we can obtain L^*, a^*, and b^* to calculate the color difference between a target color and an individual color (Figure 6.9).
4. Color difference $\sqrt{(L_t^* - L^*)^2 + (a_t^* - a^*)^2 + (b_t^* - b^*)^2}$

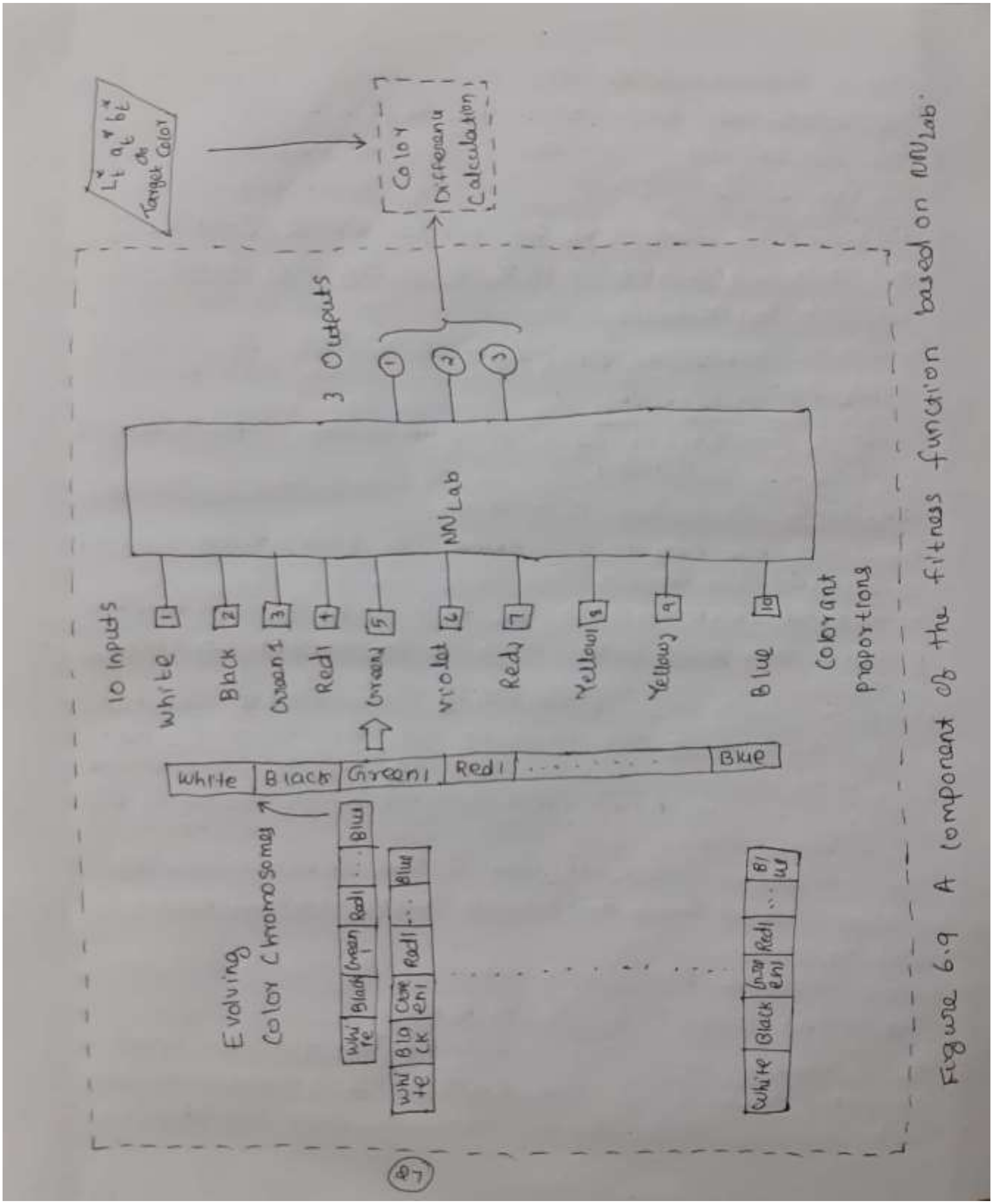
   Lightness L^*

   Hue $\arctan(b^*/a^*)$

   Chroma $\sqrt{(a^*)^2 + (b^*)^2}$

   where L^*, a^*, and b^* are calculated according to surface spectral reflectance and (L_t^*, a_t^*, b_t^*) are the values of a target color.
5. Any color can be uniquely identified by its surface spectral reflectance curve (i.e., its physical color attribute).

26


L*t a*t b*t or Target Color
Color Difference Calculation
3 Outputs
NNLab
10 Inputs
White
Black
Green1
Red1
Green2
Violet
Red2
Yellow1
Yellow2
Blue
Colorant proportions
Evolving Color Chromosomes
Figure 6.9 A component of the fitness function based on NNLab

6. The calculated color difference shows how satisfactorily the predicted color matches the reference color.
7. The use of NN_{Lab} provides a way to take into account human visual sensitivity to color difference.
8. Function 3 determines the fitness value ($fitness_3$) of each cromosome, according to the calculated color difference, E:

$$fitness_3 = \exp(-E).$$

5.6 Genetic Strategies

1. Genetic operations have a significant impact on the quality of solutions.

Modified simplex Crossover

1. modify the selection scheme in performing simplex crossover operations.
2. Modified selection uses the following three procedures:
   i. select one good chromosome with respect to fitness value.
   ii. Pick, with high probability, an elite member (i.e., one of the mutant copies from the initial CANFIS /NN results) as a good chromosome.
   iii. Choose one bad chromosome with respect to fitness value.
3. The procedures share an idea of the downhill simplex method, based on a reflection away from a bad chromosome.
4. This method provide a better GA search direction as illustrated in Figure 6.10.
5. The simplex crossover proposed by Bersini and Seront consists of the following selection and bit-assignment to produce a new child

28

Chromosome Cnew:

a) selection

Choose randomly three chromosomes and arrange them in the decreasing order with respect to their fitness values (name them C_1, C_2, and C_3 in that order.)

b) Bit-assignment

If the i'th bit of C_1 is equal to the i'th bit of C_2, it will be assigned to the i'th bit of Cnew. Otherwise, the inverse of the i'th bit of C_3 will be assigned to the i'th bit of Cnew.

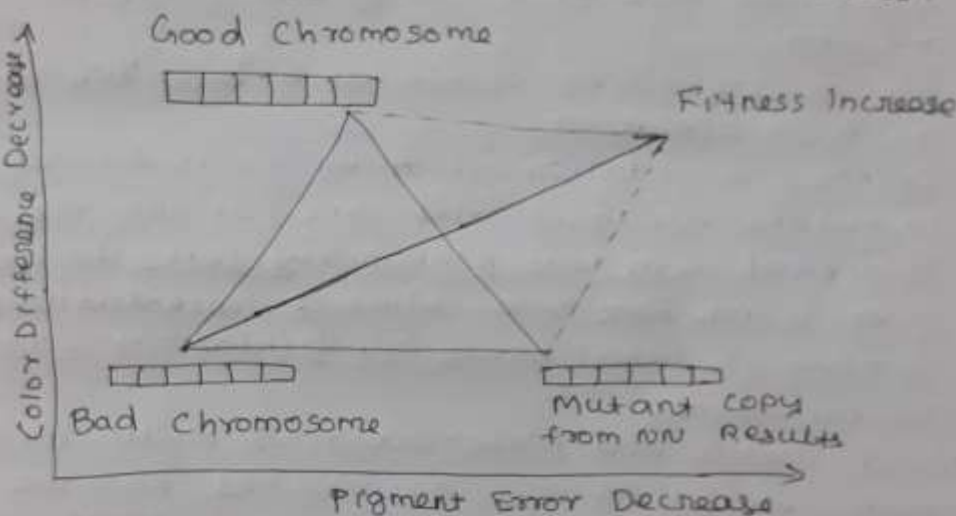


Figure 6.10 GA search control by the modified simplex crossover.

6. procedure 1 lights a direction toward minimizing color difference.
7. In accordance with NN_{Lab}, a chromosome with higher fitness may have smaller color difference.
8. In this GA search, it is desirable to find a direction that minimizes both color difference and colorant errors.
9. The problem is that we cannot calculate colorant errors directly.

(29)

Mutation Strategy

Usual mutation operation is applied to all members with a changeable mutation rate scheme such that a fixed mutation rate (0.01) is adopted with probability of 0.4, and otherwise a mutation rate ranging from 0.09 to 0.69 is decided using a random number.

The following modified operations are also considered:

1. Chromosome Template.
   i. To avoid specifying the use of more colorants than necessary
   ii. Inactivate some genes using the fuzzy population generator.
   iii. Use a chromosome itself as a template to do the mutation operation.
   iv. Before the mutation operation, it is decided whether to mutate an inactivated gene, or not; the mutation is applied with low probability (0.1) to inactivated genes, which have zero values of concentations after decoding the genes' binary representations into colorant concentrations.
2. Local search and preservation of multi-elites.
   i. Multi-elites (i.e., chromosomes from the results of CANFIS / NN approaches) are mutated only at the lower bits of each gene to keep traits similar to the NN results.
   ii. In this way, local search of the NN results is realized.
   iii. The offspring of multi-elites always advances to the next generation; the mutant copies of multi-elites are preserved throughout the entire evolution.
   iv. The manipulation of low-order bits is applied only to multi-elites.

(30)

3. Exchanging mutation

1. After the usual mutation, with low probability, members are subjected to another mutation: exchanging genes that have the same type of colorant information. This mutation is illustrated in Figure 6.11.

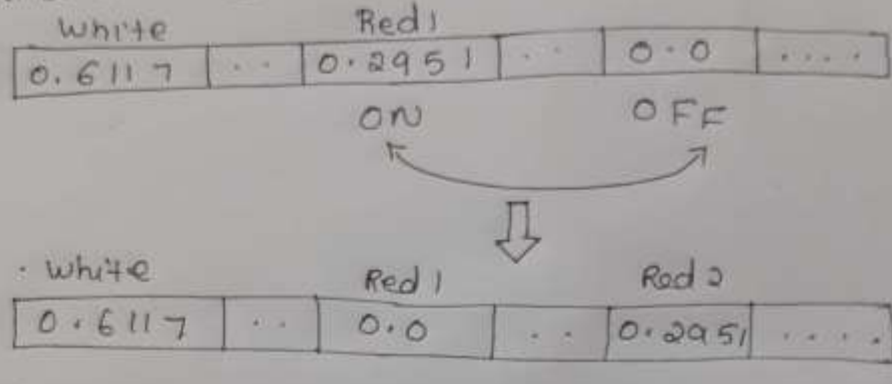


Figure 6.11 Exchanging mutation.

2. Among 10 output colorant proportions, we have three pairs of the same types of colorants, such as Red_1 and Red_2, which have different natures; we must decide which one to use.
3. Exchanging mutation helps us to explore such colorant choices.

(31)

APPENDIX 1

CONTENT BEYOND THE SYLLABUS

1. SOFT COMPUTING

Soft computing is not a mélange. Rather, it is a partnership is which each of the constituent contributes a distinct methodology for addressing problem in its domain. In this perspective, the principal constituent methodologies in soft computing are complementary rather than competitive. In fact, soft computing's main characteristic is its intrinsic capability to create hybrid systems that are based on the integration of constituent technologies. This integration provides complementary reasoning and searching methods that allow us to combine domain knowledge and empirical data to develop flexible computing tools and solve complex problems. Hybrid computing is the combination of hard computing and soft computing which having their inherent advantages and disadvantages. To get the advantages of both these techniques their individuals limitations are reduced for solving a problem more efficiently by Hybrid computing. Hybrid soft computing models have been applied to a large number of classification, prediction, and control problems.

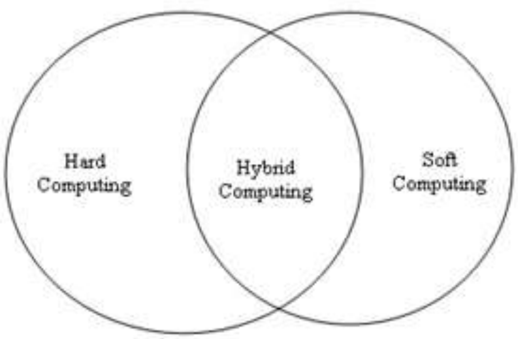


Fig 1

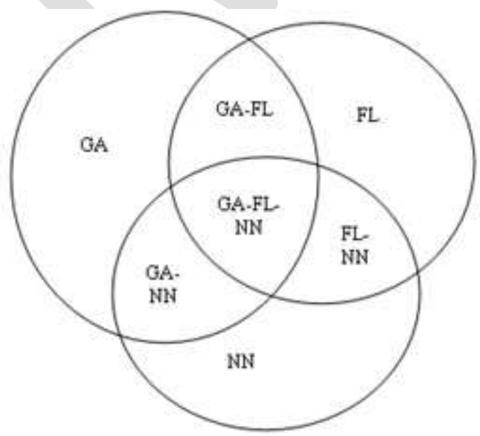


Fig 2

Figures 1 & 2 show schematic diagram of intersections of members of soft computing family & hybrid computing scheme.

2. APPLICATION AREAS OF SOFT COMPUTING

Soft computing techniques have become one of promising tools that can provide practice and reasonable solution. Soft computing techniques are used in different fields shown in fig 4

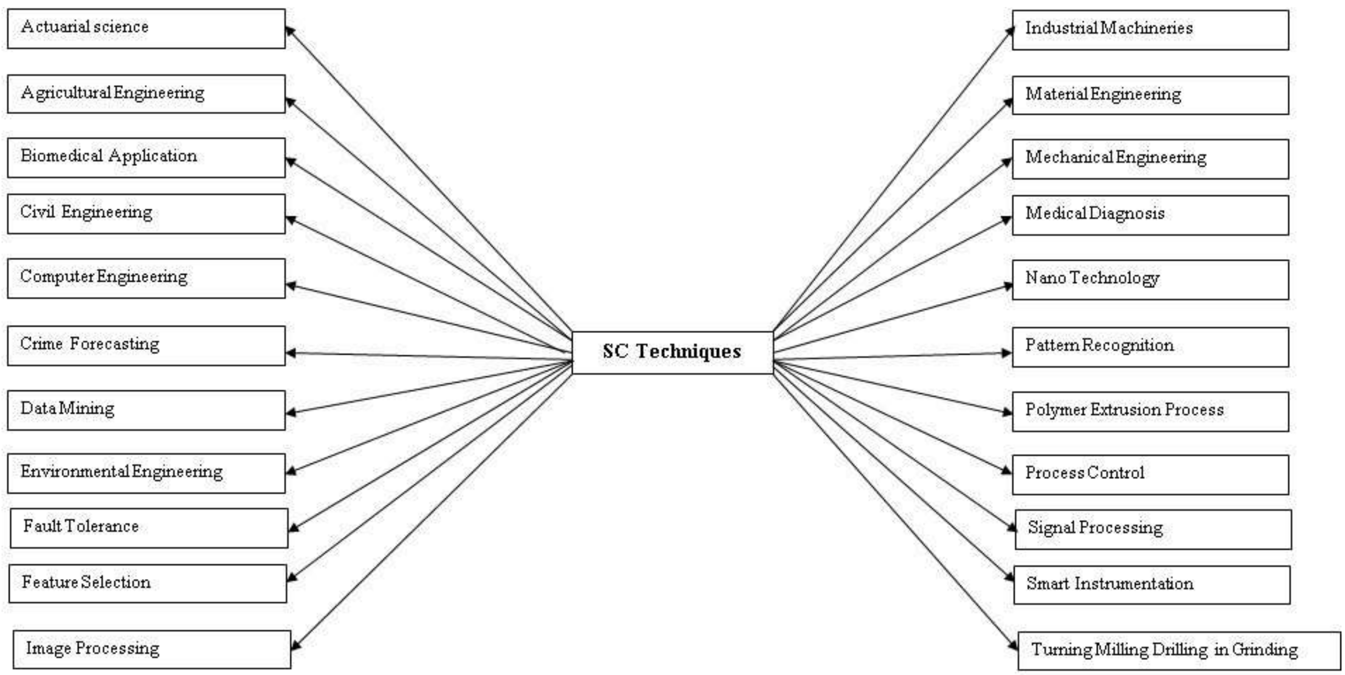


Fig 4

Actuarial Science

Actuarial science is the discipline that applies mathematical and statistical methods to evaluate risk in the insurance and finance industries. Actuarial science includes a number of interrelating subjects, including probability, mathematics, statistics, finance, economics, financial economics, and computer programming. Historically, actuarial science used deterministic models in the construction of tables and premiums.

Agricultural Engineering

Agricultural engineering is the engineering discipline that applies engineering science and technology to agricultural production and processing. Agricultural engineering combines the disciplines of animal biology, plant biology, and mechanical, civil, electrical and chemical engineering principles with knowledge of agricultural principles.

Biomedical Application

Biomedical application is a design concept to medicine and biology. This field seeks to close the gap between engineering and medicine: It combines the design and problem solving skills of engineering with medical and biological sciences to advance healthcare treatment, including diagnosis, monitoring, treatment and therapy.

Civil Engineering

Civil engineering is a professional engineering discipline that deals with the design, construction, and maintenance of the physical and naturally built environment, including works like roads, bridges, canals, dams, and buildings. Civil engineering takes place on all levels: in the public sector from municipal through to national governments, and in the private sector from individual homeowners through to international companies.

Computer Engineering

Computer engineering is a discipline that integrates several fields of electrical engineering and computer science required to develop computer systems. Computer engineers usually have training in electronic engineering, software design, and hardware-software integration instead of only software engineering or electronic engineering. Computer engineers are involved in many hardware and software aspects of computing, from the design of individual microprocessors, personal computers, and supercomputers, to circuit design. This field of engineering not only focuses on how computer systems themselves work, but also how they integrate into the larger picture.

Crime Forecasting

Crime forecast is a planning tool that helps to manage crime in our society in different way. Crime is the breaking of rules or laws for which some governing authority can ultimately prescribe a conviction. Crimes may also result in cautions, rehabilitation or be unenforced. By the help of crime forecast we can reduce crime in our societies.

Data Mining

Data mining is a subfield of computer science which is the computational process of discovering patterns in large data sets involving methods at the intersection of artificial intelligence, machine learning, statistics, and database systems. The overall goal of the data mining process is to extract information from a data set and transform it into an understandable structure for further use.

Environmental Engineering

Environmental engineering is the integration of science and engineering principles to improve the natural environment like air, water, and/or land resources, to provide healthy water, air, and land for human habitation like house or home and for other organisms, and to remediate pollution sites.

Fault-Tolerance

Fault-tolerance is the property that enables a system to continue operating properly in the event of the failure of some of its components. If its operating quality decreases at all, the decrease is proportional to the severity of the failure, as compared to a naïvely-designed system in which even a small failure can cause total breakdown. Fault-tolerance is particularly sought-after in high-availability or life-critical systems.

Feature Selection

In machine learning and statistics, feature selection, also known as variable selection, attribute selection or variable subset selection, is the process of selecting a subset of relevant features for use in model construction. Feature selection techniques are a subset of the more general field of feature extraction. Feature extraction creates new features from functions of the original features, whereas feature selection returns a subset of the features.

Image Processing

In imaging science, image processing is any form of signal processing for which the input is an image, such as a photograph or video frame; the output of image processing may be either an image or a set of characteristics or parameters related to the image. Most image-processing techniques involve treating the image as a two-dimensional signal and applying standard signal-processing techniques to it.

Industrial Machineries

Industries machineries are tool that consists of one or more parts, and uses energy to achieve a particular goal. Machines are usually powered by mechanical, chemical, thermal, or electrical means, and are frequently motorized. This is used in mechanical engineering.

Materials Engineering

Materials engineering is an interdisciplinary field applying the properties of matter to various areas of science and engineering. This scientific field investigates the relationship between the structure of materials at atomic or molecular scales and their macroscopic properties. It incorporates elements of applied physics and chemistry.

Mechanical Engineering

Mechanical engineering is a discipline of engineering that applies the principles of physics and materials science for analysis, design, manufacturing, and maintenance of mechanical systems. It is the branch of engineering that involves the production and usage of heat and mechanical power for the design, production, and operation of machines and tools.

Medical diagnosis

Medical diagnosis refers both to the process of attempting to determine or identify a possible disease and to the opinion reached by this process. From the point of view of statistics the diagnostic procedure involves classification tests.

Nano Technology

Nanotechnology is the manipulation of matter on an atomic and molecular scale. Generally, nanotechnology works with materials, devices, and other structures with at least one dimension sized from 1 to 100 nanometers. Nanotechnology entails the application of fields of science as diverse as surface science, organic chemistry, molecular biology, semiconductor physics, micro fabrication, etc.

Pattern Recognition

Pattern recognition generally aim to provide a reasonable answer for all possible inputs and to perform "most likely" matching of the inputs, taking into account their statistical variation. Pattern recognition is studied in many fields, including psychology, psychiatry, and ethology, cognitive science, and traffic flow and computer science.

Polymer Extrusion Process

A polymer is a chemical compound or mixture of compounds consisting of repeating structural units created through a process of polymerization. Polymers are studied in the fields of biophysics and macromolecular science, and polymer science.

Extrusion is a process used to create objects of a fixed, cross-sectional profile. A material is pushed or drawn through a die of the desired cross-section. The two main advantages of this process over other manufacturing processes are its ability to create very complex cross-sections and work materials that are brittle, because the material only encounters compressive and shear stresses.

Process Control

Process control is a statistics and engineering discipline that deals with architectures, mechanisms and algorithms for maintaining the output of a specific process within a desired range. It is extensively used in industry and enables mass production of continuous processes such as oil refining, paper manufacturing, chemicals, power plants and many other industries. Process control enables automation, with which a small staff of operating personnel can operate a complex process from a central control room.

Signal Processing

Signal processing is an area of systems engineering, electrical engineering and applied mathematics that deals with operations on or analysis of signals, or measurements of time-varying or spatially varying physical quantities. Types of signals are sound, images, and sensor data, for example biological data such as electrocardiograms, control system signals, telecommunication transmission signals, and many others.

Smart Instrumentation

As intelligent devices become ubiquitous the challenge is to connect sensors and actuators through smart systems. A major issue is the growing number of communication protocols, with no single standard.

The challenge is to intelligently connect smart instrumentation so that devices can communicate across multiple protocols. At the same time, increases in the volume and importance of data means that privacy, security and robustness of systems is paramount.

Turning Milling Drilling in Grinding

Turning is a machining process in which a cutting tool, typically a non-rotary tool bit, describes a helical tool path by moving more or less linearly while the work piece rotates.

Milling is the machining process of using rotary cutters to remove material from a work piece advancing in a direction at an angle with the axis of the tool. It covers a wide variety of different operations and machines, on scales from small individual parts to large, heavy-duty gang milling operations.

Drilling is a cutting process that uses a drill bit to cut or enlarge a hole of circular cross-section in solid materials. The drill bit is a rotary cutting tool, often multipoint. The bit is pressed against the work piece and rotated at rates from hundreds to thousands of revolutions per minute. This forces the cutting edge against the work piece, cutting off chips from what will become the hole being drilled.

Grinding is used to finish work pieces that must show high surface quality and high accuracy of shape and dimension.

3. SHORT DESCRIPTION OF SOFT COMPUTING IN DIFFERENT AREAS

Sl.no.	Field of applications	Soft computing components	References
1	Aircraft and air traffic	NN, FL, EC	[2], [3]
2	Communication networks	FL, NN, EC	[4], [5], [6], [7]

3	Control and Monitoring	EC, FL, NN	[8], [9], [10], [11],[12],[13]
4	Cooling and Heating	FL, NN, EC	[14], [15], [16], [17], [18], [19]
5	Data communications	FL, NN	[20], [21]
6	Data Security	ANN, FL	[22]
7	Induction Motor Drives	FL, NN	[23], [24]
8	Inverters and Converters	FL, NN	[25], [26]
9	Manufacturing Technologies	FL, NN	[27], [28]
10	Mobile Robots	FL, NN	[29]
11	Multi-Agent Robots	EC, FL	[30]
12	Network Optimization	GA	[31]
13	Power Control	EC	[32]
14	Radio Planning	ANN	[33]
15	Resource Allocation	ANN	[34]
16	Satellite Imaging	ANN, FL, EA	[35]
17	Scheduling	ANN	[36]
18	Spacecraft	NN, FL,	[37], [38]
19	Steel Process Industry	FL, NN	[39]
20	Switched Reluctance Motor Drives	FL	[40]