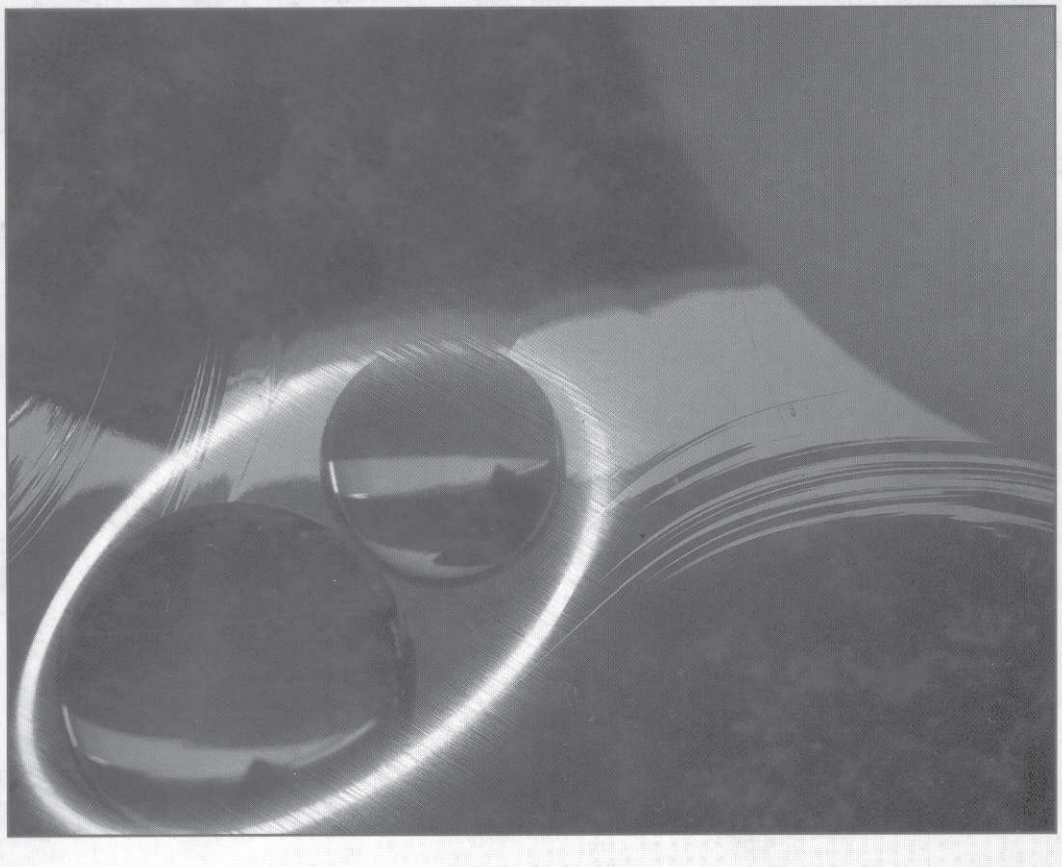
INTRODUCTION TO APPLIED FUZZY ELECTRONICS



AHMAD M. IBRAHIM

INTRODUCTION TO APPLIED FUZZY ELECTRONICS

Ahmad M. Ibrahim

PRENTICE HALL
Upper Saddle River, New Jersey   Columbus, Ohio

Library of Congress Cataloging-in-Publication Data

Ibrahim, Ahmad M.
Introduction to applied fuzzy electronics / by Ahmad M. Ibrahim.
p. cm.
Includes bibliographical references and index.
ISBN 0-13-206400-6 (alk. paper)
1. Digital electronics 2. Fuzzy logic 3. Neural networks
(Computer science) I. Title.
TK7868.D5A37 1997 96-9011
621.39'5—dc20 CIP

Cover Photo: Bruce Peterson/H. Armstrong Roberts
Editor: Charles E. Stewart, Jr.
Production Editor: Sheryl Glicker Langner
Design Coordinator: Jill E. Bonar
Text Designer: Ruttle Graphics, Inc.
Cover Designer: Russ Maselli
Production Manager: Patricia A. Tonneman
Marketing Manager: Debbie Yarnell
Production Supervision: Ruttle Graphics, Inc.
Illustrations: Diphrent Strokes, Inc.
This book was set in Times Roman by Ruttle Graphics, Inc. and was printed and bound by Book Press. The cover was printed by Phoenix Color Corp.

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

ISBN: 0-13-206400-6

Prentice-Hall International (UK) Limited, *London*
Prentice-Hall of Australia Pty. Limited, *Sydney*
Prentice-Hall Canada, Inc., *Toronto*
Prentice-Hall Hispanoamericana, S.A., *Mexico*
Prentice-Hall of India Private Limited, *New Delhi*
Prentice-Hall of Japan, Inc., *Tokyo*
Simon & Schuster Asia Pte. Ltd., *Singapore*
Editora Prentice-Hall do Brasil, Ltda., *Rio de Janeiro*

Contents

1 Introduction

During the sixties, Professor Lotfi Zadeh, of the University of California at Berkeley [1], put forward the proposition that vagueness is an aspect of uncertainty that is different from randomness. He proposed a mathematical way of looking at the intrinsic vagueness of the natural human language; he called his approach "fuzzy logic" [2–6]. The objective of fuzzy logic has been to make computers "think" like humans [7] and remove the barrier between us and the full utilization of computer capabilities.

Computer and humans have complementary strengths and weaknesses. For example, it is not a difficult task for us to identify people's faces in a photograph or understand a statement such as "long distance." These are challenging tasks for a computer; however, a computer has no difficulty in finding the average of a hundred 6-digit numbers in milliseconds.

Computers do not "understand" vague human concepts; data has to be presented to a computer in simple binary logic. Most of human data, however, is not binary. For example, we do not divide people into two well-defined groups such as "good" and "bad." In our thinking a person may still be good in spite of being imperfect. Within the "good" group you may find, whatever criteria you use, those who are really good and those who are bordering on being bad, and a whole range in between. Extending this observation to other things such as temperature, pressure, size, etc., one may conclude that all things admit degrees in their description; these degrees are not random but rather deterministic, based on several factors such as the nature of the situation, experience, etc. This idea is the essence of the so-called fuzzy sets as opposed to classical, or crisp, sets. In classical set theory, an object either belongs to a set or not; a person is either good or bad; temperature is either hot or cold; a glass is either full or empty. Classical set theory was the mathematical background for computer logic. In fuzzy set theory, degrees of belonging to a set are introduced; a person can be, for example, 80% in the "good" set and 40% in the "bad" set, based on our experience and judgment. Fuzzy set theory is the mathematical background that is needed to capture the way people think.

Controversy has surrounded the concept of fuzziness since its inception [7]. Some maintain that probability theory can handle any kind of uncertainty; others think that fuzziness is probability in disguise, or that probability is the only sensible way to describe any kind of uncertainty. Some found that the term "fuzzy" suits the innumerates only; others voiced even stronger opinions. A very readable review of fuzziness critique was given by Laviolette and Seaman [8] from a statistician's point of view. Lindley [9] voiced a challenge that "anything that can be done by alternatives to probability can be done better by probability." Although fuzzy logic is not an alternative to probability, Lindley assumed it is; his challenge is directed toward fuzzy logic. The fact remains, however, that fuzzy logic has led to applications that probability, although it has been with us for a long time, did not address. Fuzzy logic cannot simply be discounted or discredited because it is not perfect or because probability can probably do what fuzzy logic has done.

Supporters of fuzziness, "fuzzy-ists," put forward arguments to show the usefulness of concept and to explain the distinction between fuzziness and randomness. Klir [10], for example, addressed rigorously the question of whether or not probability theory is the only sensible description of uncertainty. The conclusion was that probability theory is capable of conceptualizing only one type of uncertainty and hence concepts of fuzziness are required. Some supporters of fuzziness went as far as adding an aura of mystique to fuzzy set theory [11].

Whether fuzziness can be replaced by probability or not, whether it is the start of a new philosophy or not, it has contributed to engineering and it is here to stay.

In simple terms [12] one can say that the uncertainty of probability, as commonly used in engineering, relates to the randomness of the occurence of a phenomenon. For example, "role a die and get two." The ambiguity or vagueness of a word or concept does not relate to randomness but to fuzziness. The uncertainty in an expression like "comfortable temperature" or "small number" may be termed fuzziness.

Randomness and fuzziness are two different aspects of uncertainty. The passage of time or an increase of information can clarify the uncertainty of randomness. After rolling a die, for example, the number you get is certain. The uncertainty associated with fuzziness is an essential characteristic of words and concepts. Passage of time does not make the statement "small number" become any clearer.

It is interesting to observe that the uncertainly in a quantum-mechanical sense does not belong to either of the previous two categories of uncertainty. It is not due to vagueness intrinsic to the natural human language nor is there an experiment or a test to remove it. Quantum physics uses the notion of uncertainty in a fundamentally different way. Its most fundamental principle, Heisenberg's uncertainty principle, states that certain physical characteristics of a system cannot be simultaneously known with an unlimited precision. For example, the better we can measure the position of an electron the less precisely we know the value of its velocity. Attempts to localize an electron lead to smearing or fuzzification in its velocity.

By the seventies, fuzzy logic gained increasing importance after Ebrahim Mamdani and Seto Assilian, at Queen Mary College in London, demonstrated the practical potential of fuzzy logic [13]. Their pioneering work has led to an exponential growth of fuzzy logic literature and opened the door for various applications such as industrial process control and consumer products. Fuzzy logic uses linguistic modeling of a system, as opposed to mathematical modeling, in determining the control rules, using the experience of a human operator. A difficulty occurred in formally determining the stability of a fuzzy control system. Kiszka et al. [14] pioneered energistic stability studies of fuzzy dynamic systems.

Further advances were made when Togai and Watanabe [15] reported their first fuzzy chip. Yamazaki and Sugeno [16] and Yamakawa [17, 18] developed a microprocessor-based fuzzy controller.

Another era started by combining neural networks capabilities with those of fuzzy logic. Kosko [19] presented a new approach for adaptive fuzzy systems using neural networks.

By 1990, fuzzy logic reached the consumer market [20–22]. Fuzzy logic was used in vacuum cleaners, cameras, microwave ovens, and several other consumer products.

The fuzzy logic tide has reached the Internet. You can use the Internet to search for the most up-to-date information about fuzzy products or join a discussion group on the topic. A good start could be through obtaining the monthly updated information FAQ (frequently asked questions) by sending an e-mail to:

ai + query @cs.cmu.edu

with the message

send fuzzy FAQ

You will receive information about the comp.ai.fuzzy news group (started in 1993), fuzzy BBS, FTP repositories, companies supplying fuzzy tools, etc.

Fuzzy logic has been in vogue for some time now. Some people think it has the answer for everything not only in electronics and systems design but even far beyond that [11]. Thus, combined with neural networks, one may reach the erroneous conclusion that we have reached the ultimate in electronics. Fuzzy logic has proved to be a powerful tool, but remember that the field of electronics is always evolving; fuzzy logic marked the start of an era, but we should not think that it is the zenith of electronics. Remember these famous last words:

"The telegraph is the ultimate in fast communication."—Engineer, 1850s

"With the vacuum tube, we've reached the zenith in communication potential." —Engineer, 1920s

"Transistors are the final step in search for speedy reliable means of communications."—Engineer, 1950s

"Integrated circuits are IT! They can't possibly go beyond this revolutionary new concept."—Engineer, 1960s

It is always important to understand the fundamental concepts, old and new, and be equipped with sufficient background to get the maximum out of electronics, this dynamic field, and prepare for further advances.

This is what this book hopes to achieve: to give the fundamental concepts of fuzzy logic and its applications and provide a platform so that the reader is prepared to use fuzzy logic and to advance further in this area.

To start, a mathematical foundation linking classical sets and Boolean logic is presented in chapter two. This relation is not usually emphasized in digital electronics courses. The way becomes paved for introducing the concepts of fuzzy sets and logic, which are introduced in chapters three and four. In chapter five, fuzzy control systems are introduced and contrasted with PID systems. Design procedures are explained in an easy format. Electronic neural networks are reviewed in chapter six to provide the reader with a platform to pursue further advances in fuzzy logic, a quick view of which is given in chapter seven.

After reading this book, I expect that you will be able to

- Use fuzzy sets and fuzzy algebra.
- Use the building blocks of fuzzy circuits.
- Design fuzzy control systems.
- Design simple neural networks.
- Pursue advances in fuzzy-neural systems.
- Sail through fuzzy logic literature, including the Internet resources, with ease.

Chapter 1 Questions

1–1. What was the original objective of developing fuzzy logic? Has that objective been achieved?

1–2. What is a crisp set?

1–3. What is binary logic?

1–4. Discuss the merits and demerits of using the tag “fuzzy logic” to describe Zadeh’s new approach.

1–5. Distinguish qualitatively between fuzziness and probability as commonly used in engineering.

1–6. Name a few consumer products in which fuzzy electronics were used.

1–7. What is Heisenberg’s uncertainty principle?

1–8. Obtain, using the Internet, a list of companies supplying fuzzy tools.

1–9. Are fuzzy-neural networks the ultimate in electronics?

***1–10.** Max Black envisioned some concepts about vagueness in his paper: “Vagueness—An Exercise in Logical Anlysis, *Philosophy of Science* (1937): 427–455.

a) Summarize an example he gave to illustrate the vagueness intrinsic to language.

b) Explain his distinction between vagueness and ambiguity, and between vagueness and generalization.

c) Give his answer to the question, “Is vagueness subjective?”

d) Sketch the consistency profiles presented to describe a typical vague symbol, a very precise symbol, and a very vague symbol. Comment on these graphs.

e) What was the objective of his “experiment in vagueness”?

References

1. T.S. Perry, “Lotfi A. Zadeh,” *IEEE Spectrum* 6 (1995): 32–35.
2. L.A. Zadeh, “Fuzzy sets,” *Informat. Control* 8 (1965): 338–353.
3. L.A. Zadeh, “Fuzzy Algorithm,” *Informat. Control* 12 (1968): 94–102.
4. L.A. Zadeh, “Toward a Theory of Fuzzy Systems,” in *Aspects of Network and System Theory*, ed. R.E. Kalman and N. DeClaris, (New York: Holt, Rinehart and Winston, 1971), 469–490.
5. L.A. Zadeh, “Outline of a New Approach to the Analysis of Complex Systems and Decision Processes,” *IEEE Trans. Syst. Man. Cybern.* SMC-3 (1973): 28–44.
6. L.A. Zadeh, “The Concepts of a Linguistic Variable and Its Application to Approximate Reasoning I, II, III,” *Informat. Sci.* 8 (1975): 199–251, 301–357; *9* (1975): 43–80.
7. L.A. Zadeh, “Making Computers Think Like People,” *IEEE Spectrum* (August 1984): 26–32.
8. M. Laviolette and J.W. Seaman, Jr., “The Efficacy of Fuzzy Representations of Uncertainty,” *IEEE Trans. Fuzzy Systems,* 2 (1994): 4–15.
9. D.V. Lindley, “The Probability Approach to Treatment of Uncertainty in Artificial Intelligence and Expert Systems”, *Stat. Sci.* 2 (1987): 17–24.
10. G.J. Klir, “Is There More to Uncertainty Than Some Probability Theorists Might Have Us Believe?”, *Int. J. General Systems* 15 (1989): 347–378.
11. B. Kosko, *Fuzzy Thinking*, (New York: Hyperion, 1993).
12. T. Terano, K. Asai, and M. Sugeno, *Fuzzy Systems Theory and Its Applications* (Toronto: Academic Press, Inc., 1992).
13. E.H. Mamdani and S. Assilian, “An Experiment in Linguistic Synthesis with Fuzzy Logic Controller,” *Int. J. Man Mach. Studies* 7 (1975): 1–13.

14. J.B. Kiszka, M.M. Gupta, and P.N. Nikiforuk, "Energistic Stability of Fuzzy Dynamic Systems," *IEEE Trans. Syst. Man Cybern.* SMC-15 (1985): 783–792.
15. M. Togai and H. Watanabe, "Expert System on a Chip: An Engine for Real-time Approximate Reasoning," *IEEE Expert System. Mag.* 1 (1986): 55–62.
16. T. Yamazaki and M. Sugeno, "A Microprocessor Based Fuzzy Controller for Industrial Purposes," in *Industrial Applications of Fuzzy Control*, ed. M. Sugeno, (Amsterdam: North Holland, 1985), 231–240.
17. T. Yamakawa, "High Speed Fuzzy Controller Hardware System," in *Proc. 2nd Fuzzy System Symp.* (Japan, 1986), 122–130.
18. T. Yamakawa, "Fuzzy Microprocessor—Rule Chip and Defuzzifier Chip," in *Int. Workshop on Fuzzy System Applications* (Iizuka, Japan, 1988), 51–52.
19. B. Kosko, *Neural Networks and Fuzzy Systems* (Toronto: Prentice Hall, 1992).
20. A. Sangalli, "Fuzzy Logic Goes to the Market," *New Scientist* 8 (February 1992): 36–39.
21. D.G. Schwartz and G.J. Klir, "Fuzzy Logic Flowers in Japan," *IEEE Spectrum* (July 1992): 32–35.
22. K. Self, "Designing with Fuzzy Logic," *IEEE Spectrum* (November 1990): 42–45.
23. S.M. Sze, *Physics of Semiconductor Devices*, Comp. H. C. Spencer (Toronto: Wiley-Interscience, 1969), 5.

2 Sets, Boolean Logic and Algebra

2.1 Definition of a Set

A set is a collection of objects. The individual objects are referred to as elements or members of the set. The expression "is an element of" is written as $\in$, while "is not an element of" is written as $\notin$.

EXAMPLE: $x \in A$: x is an element of set A.
$x \notin A$: x is not an element of set A.

Usually sets are denoted by capital letters and elements by lower-case letters.

2.2 Description of a Set

There are several possible ways to describe or define a particular set:

1. A set may be defined by specifying the properties of its members. For example, one may specify set A as the collection of all positive integers that are less than or equal to 10. This may be written as

$$A = \{x \mid x \text{ is a positive integer} \leq 10\}.$$

2. A set may be described by listing all of its members. For example, one may say set A is the collection of the following numbers: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10. This is written as

   $$A = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}.$$

3. A formula may be used to describe a set. For example,

   $$A = \{x_{i+1} = x_i + 1, i = 1, \ldots, 5, \text{ where } x_1 = 1\}.$$

   This formula states that A is a set that is composed of five elements. The first element is equal to 1, and each subsequent element is obtained by adding 1 to the previous element.

4. A set can be defined using an operation on some set; such as intersection (logical AND) or union (logical OR), see section 2.3. For example,

   $$A = \{x \mid x \text{ is an element that belongs to B AND C}\},$$

   where B, C are sets

5. A set may be defined using the so-called membership function. The membership function assumes a value of 1 for elements that are members of the set, and assumes a value of 0 for elements that are not members of the set. In shorthand form,

   $$\mu_A(x) = 1 \quad \text{if } x \in A$$

   for all x values considered.

   $$\mu_A(x) = 0 \quad \text{if } x \notin A$$

   For example, let the numbers under consideration (the universe as far as this example is concerned) be 1, 2, 3, 4, 5, 6, 7, 8, 9, and 10. The set of even numbers may be written as

   $$\{(1, 0), (2, 1), (3, 0), (4, 1), (5, 0), (6, 1), (7, 0), (8, 1), (9, 0), (10, 1)\}$$

   where each element in the universe is written associated with its membership function, 0 for non-members, and 1 for members. The set can simply be written as $\{2, 4, 6, 8, 10\}$ where elements with $\mu_A(x) = 0$ are not included and the elements listed are understood to have

   $$\mu_A(x) = 1.$$

   The number of elements of a set is called the cardinality of the set and is denoted by #A or |A|.

EXAMPLE: Let $A = \{2, 4, 6, 8, 10\}$; then $\#A = 5$.

#A is either a natural number (for sets with elements that can be enumerated in the usual sense), or an infinity (for infinite sets). (Different infinities in turn, may have different cardinalities. For instance, there are infinitely many more real numbers than integer numbers. These matters, however, will not be discussed in this book).

The power set of A is the set whose elements are the subsets of A; it is denoted by *P*A.

EXAMPLE: Let A = {1, 2, 3}, then

$$PA = \{\Phi, \{1\}, \{2\}, \{3\}, \{2, 3\}, \{1, 3\}, \{1, 2\}, \{1, 2, 3\}\}$$

where Φ is an empty set (see section 2.3).

EXAMPLE: Let A = $\{x\}$, then $PA = \{\Phi, \{x\}\}$.

2.3 Operations on Sets

Before we discuss some of the basic operations on sets, let us consider first some concepts associated with sets, namely, the empty set, subsets, and the universal set.

Suppose you have a telephone index—it has pages marked with letters from A to Z. You organize the names so that the first letter of a name corresponds with the letter designation of the page. A listing under a given letter may be subdivided according to the area code, for example. You may not know anyone with names starting with, say, K, X, Y, etc.; these pages will be empty. Now, the collection of names under the same letter is an example of a set; subdivision of pages gives subsets. Pages with no names listed are examples of empty sets. The whole collection of names in the index are all the names you may consider phoning; it is the universal set, or the universe, as far as this example is concerned. Thus the empty set is a set that has no elements (it is usually denoted by the symbol Φ) and has a cardinality of zero. The universal set is a set that contains all sets under consideration (it usually is denoted by the symbols I or U. Set B is a subset of A if each element x in B is also an element in A; this is written as B ⊂ A.

It is interesting to note that all empty sets are equal. This reminds me of the fellow who asked the waiter for coffee without milk, and the waiter said, "Sorry, we ran out of milk; do you mind a coffee without cream?"

2.3.1 The Intersection of Sets

Taking the intersection of some sets A, B, C, . . . results in a new set containing only those elements that occur in each of the sets A, B, C,. . . The intersection of sets A and B is denotd by AB or A ∩ B. In shorthand (mathematical language), $A \cap B = \{x \in A \cap B \mid x \in A \text{ and } x \in B\}$.

EXAMPLE: Let A = {1, 2, 3, 4, 5} and B = {2, 4, 6, 8}; then $A \cap B$ = {2, 4}.

EXAMPLE: Let A = {1} and B = {0}; then $A \cap B = \Phi$.

Intersection operation has some properties that are reminiscent of those of multiplication of numbers, with the universal set I playing the role of unity and the empty set Φ playing the role of zero.

EXAMPLE:
* A B = B A (commutative law)
* A(B C) = (A B)C (associative law)
* A $\Phi = \Phi$ and A I = A

Then there are properties that are unique, i.e., dissimilar to multiplication, such as

If $B \subset A$, then B A = B, and A A = A.

EXAMPLE: Let A = {1, 2, 3, 4, 5, 6} and B = {2, 3, 4}; then,
$A \cap B$ = {2, 3, 4} and
$A \cap A$ = {1, 2, 3, 4, 5, 6}.

2.3.2 The Union of Sets

Taking the union of some sets A, B, C, . . . results in a new set containing the elements that occur in at least one of the sets A, B, C, . . . The union of sets A and B is denoted by A + B or $A \cup B$. In shorthand (mathematical language), $A \cup B = \{x \in A \cup B \mid x \in A \text{ or } x \in B\}$.

EXAMPLE: Let A = {1, 2, 3, 4, 5} and B = {2, 4, 6, 8}; then $A \cup B$ = {1, 2, 3, 4, 5, 6, 8}.

EXAMPLE: Let A = {1}, B = {0} and I = {0, 1}; then $A \cup B$ = I.

Union operation has some properties that are reminiscent of those of addition of numbers.

EXAMPLE:
* A + B = B + A, A(B + C) = A B + A C,
* A + (B + C) = (A + B) + C,
* A + Φ = A, and A + I = I

Then there are unique properties, such as
If $B \subset A$, then $B + A = A$, and $A + A = A$, since no new elements are introduced or excluded by these operations in this case.

Also, $A + I = I$, since I is the maximum the result of addition can reach, *i.e.* to include all the elements that are available.

EXAMPLE: Let $A = \{1, 2, 3, 4, 5, 6\}$ and $B = \{2, 3, 4\}$; then

$$A + B = \{1, 2, 3, 4, 5, 6\}$$

and

$$A + A = \{1, 2, 3, 4, 5, 6\}.$$

2.3.3 The Complement of a Set

Taking the absolute complement of a set A results in a new set containing all the elements of the universal set excluding those of set A. The complement of A is denoted by $\overline{A}$. The union of A and $\overline{A}$ gives the universal set, while their intersection gives the empty set, since, by definition, they do not have any common elements. The membership function of set A, $\mu_A(x)$, and that of $\overline{A}$, $\mu_{\overline{A}}(x)$, are related by

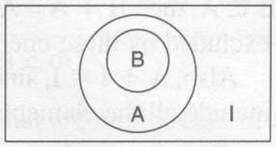
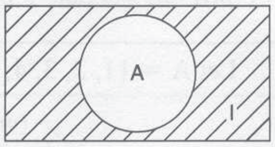
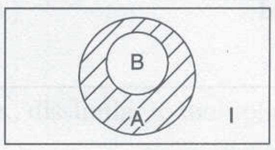
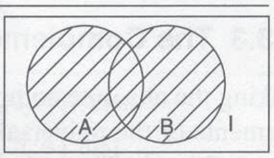
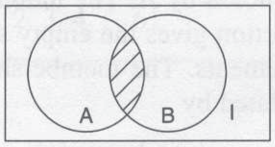
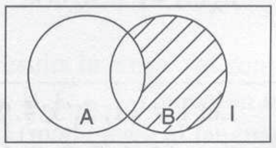
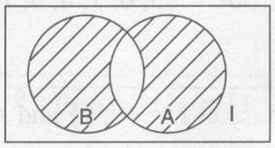
$$\mu_{\overline{A}}(x) = 1 - \mu_A(x).$$

EXAMPLE: Let $I = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$ and $A = \{1, 2, 3, 4\}$; then
$\overline{A} = \{5, 6, 7, 8, 9, 10\}$.

EXAMPLE: Let $I = \{0, 1\}$ and $A = \{1\}$; then
$\overline{A} = \{0\}$.

One can define a complement of set A relative to another set B, rather than I. This is denoted by $\overline{A}_B$.

The so-called Venn diagrams are often used to illustrate various operations on sets. Some examples are shown in Table 2.1. For a smooth transition to fuzzy set theory, one may use an alternative format as shown in Figure 2.1. The elements x in a set are listed along the horizontal axis in some order and the membership function $\mu(x)$ evaluated at each element is listed along the vertical axis. For the classical (or crisp) sets, $\mu(x) = 1$ or 0 as stated before.

Operation	Symbol	Venn diagram
B is a subset of A	$B \subset A$	
Absolute complement of A	$\overline{A}$	
Relative complement of B with respect to A ($B \subset A$)	$A - B$	
Union of A and B	$A \cup B$	
Intersection of A and B	$A \cap B$	
Substraction of B from A ($B \not\subset A$)	$A - B = A - AB$	
Symmetric difference of A and B	$A \Delta B$	

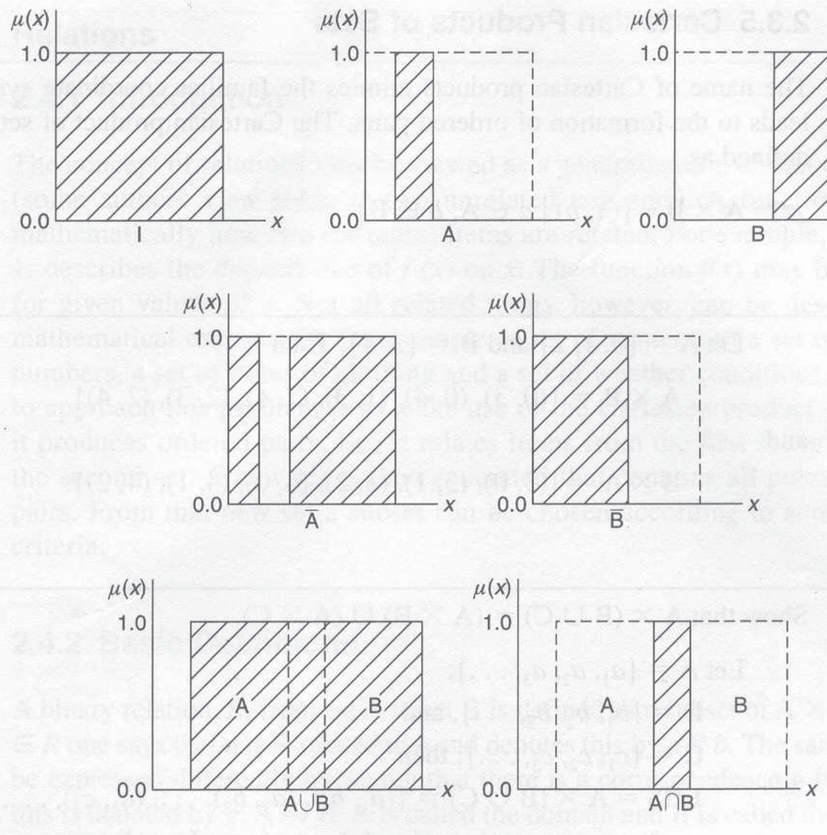


Figure 2.1 An illustration of profile format Venn diagrams.

2.3.4 Summary of Algebra of Sets

1. If $A \subset B$ and $B \subset A$, then $A = B$.
2. If $A \subset B$ and $B \subset C$, then $A \subset C$.
3. $\Phi \subset A$, $A \subset I$.
4. $A + B = B + A$, $A\,B = B\,A$.
5. $A + (B + C) = (A + B) + C$, $A(B\,C) = (A\,B)C$.
6. $A + A = A$, $A\,A = A$.
7. $A(B + C) = A\,B + A\,C$, $A + B\,C = (A + B)(A + C)$.
8. $A + \Phi = A$, $A\,\Phi = \Phi$.
9. $A + I = I$, $A\,I = A$.
10. $A + \bar{A} = I$, $A\,\bar{A} = \Phi$.
11. $\bar{\Phi} = I$, $\bar{I} = \Phi$.
12. $\bar{\bar{A}} = A$.
13. $(\overline{A + B}) = \bar{A}\,\bar{B}$ and $(\overline{AB}) = \bar{A} + \bar{B}$ (DeMorgan's Laws).

2.3.5 Cartesian Products of Sets

The name of Cartesian products mimics the familiar coordinate system since it leads to the formation of ordered pairs. The Cartesian product of sets A and B is defined as

$$\mathrm{A} \times \mathrm{B} = \{(a, b) \mid a \in \mathrm{A}, b \in \mathrm{B}\}.$$

EXAMPLE: Let $\mathrm{A} = \{0, 1, 2\}$ and $\mathrm{B} = \{3, 4\}$. Then

$$\mathrm{A} \times \mathrm{B} = \{(0, 3), (0, 4), (1, 3), (1, 4), (2, 3), (2, 4)\}$$

and

$$\mathrm{B} \times \mathrm{A} = \{(3, 0), (3, 1), (3, 2), (4, 0), (4, 1), (4, 2)\}.$$

EXAMPLE: Show that $\mathrm{A} \times (\mathrm{B} \cup \mathrm{C}) = (\mathrm{A} \times \mathrm{B}) \cup (\mathrm{A} \times \mathrm{C})$.

$$\begin{aligned}\text{Let } \mathrm{A} &= \{a_1, a_2, a_3, \ldots\},\\ \mathrm{B} &= \{b_1, b_2, b_3, \ldots\}, \text{ and}\\ \mathrm{C} &= \{c_1, c_2, c_3, \ldots\}; \text{ then}\\ \mathrm{LHS} &= \mathrm{A} \times (\mathrm{B} \cup \mathrm{C}) = \{(a_1, b_1), (a_1, b_2), \ldots, (a_1, c_2), \ldots,\\ &\qquad (a_2, b_1), \ldots, (a_2, c_1), \ldots\}\\ \mathrm{RHS} &= (\mathrm{A} \times \mathrm{B}) \cup (\mathrm{A} \times \mathrm{C}) = \{(a_1, b_1), (a_1, b_2), \ldots, (a_2, b_1), \ldots,\\ &\qquad (a_1, c_1), (a_1, c_2), \ldots, (a_2, c_1), \ldots\}\end{aligned}$$

$\therefore$ LHS = = RHS.

Selected ordered pairs from the set A × B form a subset R, referred to as a binary relation from A to B. If A = B, R is said to be a binary relation in A.

EXAMPLE: Let $\mathrm{A} = \{0, 1, 2\}$ and $\mathrm{B} = \{3, 4\}$. Suppose we are only interested in those ordered pairs whose second coordinate is a multiple of two; then we select from A × B those pairs that satisfy that condition:

$$\mathrm{R} = \{(0, 4), (1, 4), (2, 4)\} \subset \mathrm{A} \times \mathrm{B}.$$

2.4 Relations

2.4.1 Introduction

The concept of relations may be viewed as a generalization of that of functions (some authors view them as two unrelated concepts). A function describes mathematically how two (or more) items are related. For example, $f(x) = x^2 + 1$, describes the dependence of $f(x)$ on x. The function $f(x)$ may be evaluated for given values of x. Not all related items, however, can be described by a mathematical expression. For example, a set of names and a set of telephone numbers, a set of items of clothing and a set of weather conditions, etc. A way to approach this problem is to make use of the Cartesian product of two sets; it produces ordered pairs, i.e., it relates items from the first set to items from the second set. A new set is thus generated that contains all potential related pairs. From that new set a subset can be chosen according to some specified criteria.

2.4.2 Basic Definitions

A binary relation, R, from set A to set B is defined as a subset of A $\times$ B. If $(a, b) \in R$ one says that a is R-related to b and denotes this by $a\ R\ b$. The same thing can be expressed differently by saying that there is a correspondence g from A to B; this is denoted by g: A $\rightarrow$ B. A is called the domain and B is called the codomain. A relation on A is any correspondence from A to A. There are four types of such a relation R:

1. Reflexive, if $a\ R\ a$ for every $a \in$ A.
2. Symmetric, if $a\ R\ b$ implies $b\ R\ a$.
3. Anti-symmetric, if $a\ R\ b$ and $b\ R\ a$ implies $a = b$.
4. Transitive, if $a\ R\ b$ and $b\ R\ c$ implies $a\ R\ c$.

EXAMPLE: Let A be a set of weather conditions such that A = {snow, rain, sunshine}, and let B be a set of items you may use such that B = {shoes, boots, coat, umbrella}. The Cartesian product A $\times$ B produces all possible combinations of weather conditions and items you may use.

A $\times$ B = {(snow, shoes), (snow, boots) (snow, coat), (snow, umbrella), (rain, shoes), (rain, boots), (rain, coat), (rain, umbrella), (sunshine, shoes), (sunshine, boots), (sunshine, coat), (sunshine, umbrella)}

A relation R from A to B can be defined by selecting ordered pairs from the set A $\times$ B that satisfy a particular relationship. For example,

R = {(snow, boots) (snow, coat), (rain, boots), (rain, umbrella), (sunshine, shoes)}

EXAMPLE: Let A be a set of traffic light conditions, A = {Red, Yellow, Green}, and let B be a set of actions a driver may take, B = {stop, low speed, high speed}. The operation A $\times$ B gives the set of all possible combinations from A to B. From that set one chooses a subset that satisfies traffic law. Thus one may choose

R = {(Red, stop), (Yellow, low speed), (Green, high speed)}.

It is interesting to observe that in the previous two examples, a mathematical function was not required (it probably does not even exist) to define a relation. Only a linguistic description is needed, e.g., if it is raining, use an umbrella, or if it is red, then stop.

EXAMPLE: Let A = {1, 2, 3} and B = {4, 5, 6}, and suppose a correspondence from A to B is defined as "a is half of b" where $a \in$ A, $b \in$ B, and $(a, b) \in R$.

The Cartesian product A $\times$ B is given by

$$\begin{aligned} A \times B &= \{1, 2, 3\} \times \{4, 5, 6\} \\ &= \{(1, 4), (1, 5), (1, 6), (2, 4), (2, 5), (2, 6), (3, 4), (3, 5), (3, 6)\}. \end{aligned}$$

From that set we select a subset R according to the criteria given; Thus

$$R = \{(2, 4), (3, 6)\}.$$

One can write $2R\,4$, and $3R\,6$, while $1\not R\,4$, $1\not R\,5$, $1\not R\,6$, $2\not R\,5$, $2\not R\,6$, $3\not R\,4$, $3\not R\,5$, where $\not R$ means not related to.

If the correspondence were defined differently, we would have obtained a different set. For example, suppose we specified that a is an odd number and b is an even number for all $(a, b) \in R$. Then, we selected the pairs (a, b) from the set A $\times$ B that satisfy the specification given. This leads to

$$R = \{(1, 4), (1, 6), (3, 4), (3, 6)\}.$$

2.4.3 Pictorial Representation

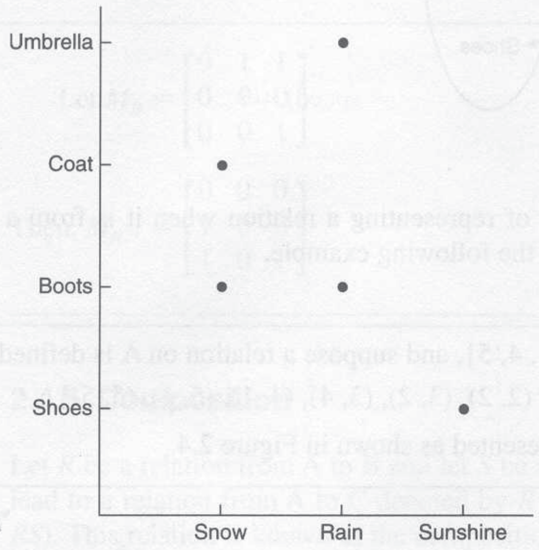
A relation from A to B can be pictured using Cartesian axes, a table, or a listing of the elements that shows the correspondence between elements by arrows.

EXAMPLE: Let A = {snow, rain, sunshine} and let B = {shoes, boots, coat, umbrella}. Suppose

$R = \{$(snow, boots), (snow, coat), (rain, boots), (rain, umbrella), (sunshine, shoes)$\}$.

Then R may be represented pictorially using one of the following:

(1) Using Cartesian axes

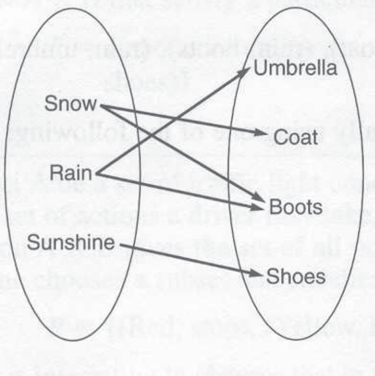


(2) Using an array

	snow	*rain*	*sunshine*
umbrella	0	1	0
coat	1	0	0
boots	1	1	0
shoes	0	0	1

This can be written as a matrix: $\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

(3) Using an arrow diagram



There is another way of representing a relation when it is from a finite set into itself, as illustrated in the following example.

EXAMPLE: Let A = {1, 2, 3, 4, 5}, and suppose a relation on A is defined as

$$R = \{(1, 2), (2, 2), (3, 2), (3, 4), (4, 1), (5, 1), (5, 5)\}.$$

This can be represented as shown in Figure 2.4.

2.4.4 Inverses

If a relation R takes us from a domain A to a codomain B, the inverse of R, denoted by R^{-1}, takes us from B back to A. The inverse of R is defined by

$$R^{-1} = \{(b, a) \mid (a, b) \in R\}.$$

In other words, $bR^{-1}a$ iff aRb.

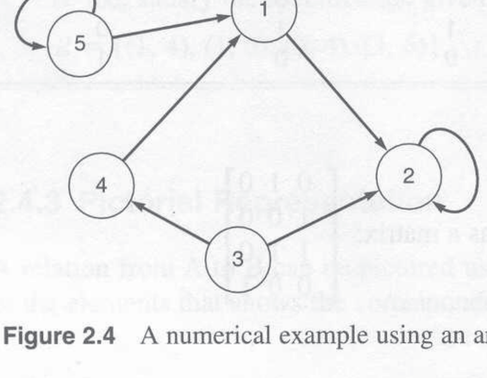


Figure 2.4 A numerical example using an arrow diagram.

If A = B and $R = R^{-1}$, then R is said to be involuntary, or self-inverse, or symmetric.

Furthermore, if M_R is the matrix of a relation R, then the matrix $M_{R^{-1}}$ of R^{-1} is the transpose of the matrix of R:

$$M_{R^{-1}} = M_R^{\mathrm{T}}$$

The transpose of a matrix, M, is the matrix obtained by writing the rows of M, in order, as columns.

EXAMPLE:

$$\text{Let } M_R = \begin{bmatrix} 0 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\text{Then, } M_{R^{-1}} = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

2.4.5 Composition

Let R be a relation from A to B and let S be a relation from B to C. Then R and S lead to a relation from A to C denoted by $R \circ S$ (or sometimes written simply as RS). This relation is known as the composition of R and S. It is defined by

$$R \circ S = \{(a, b) \in A \times B \mid \text{there exists } b \text{ in B such that } (a, b) \in R \text{ and } (b, c) \in S\}.$$

$R \circ S$ can be determined easily using an arrow diagram. Also, one can determine the matrix of $R \circ S$ by multiplying the matrices of R and S, i.e.,

$$M_{R \circ S} = M_R M_S.$$

EXAMPLE: Let A = {1, 2, 3}, B = $\{a, b, c\}$, and C = $\{\alpha, \beta, \gamma\}$.
Let $R = \{(1, a), (1, b), (2, b), (2, c), (3, c)\}$ and
$S = \{(a, \alpha), (b, \gamma), (c, \alpha)\}$.
Then, the diagram is as shown in Figure 2.5.

There is an arrow from 1 to a, followed by an arrow from a to α, meaning that there is a path or a connection between element $1 \in A$ and element $\alpha \in C$ and hence, $1RS\alpha$. Similarly, paths exist

- from 1 to b, then from b to γ, meaning $1\,RS\gamma$
- from 2 to b, then from b to γ, meaning $2\,RS\gamma$

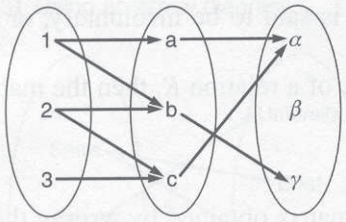


Figure 2.5 An arrow diagram to illustrate the composition relation R and S

- from 2 to c, then from c to α, meaning $2\ RS\alpha$
- from 3 to c, then from c to α, meaning $3\ RS\alpha$

Accordingly, we can write

$$R \circ S = \{(1, \alpha), (1, \gamma), (2, \gamma), (2, \alpha), (3, \alpha)\}$$

Similar results could be reached if we multiply the matrix of R, M_R, and the matrix of S, M_S, to obtain $M_{R \circ S}$.

$$M_R = \begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array}\begin{array}{c} \begin{array}{ccc} a & b & c \end{array} \\ \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} \end{array} \text{ and } M_S = \begin{array}{c} \\ a \\ b \\ c \end{array}\begin{array}{c} \begin{array}{ccc} \alpha & \beta & \gamma \end{array} \\ \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \end{array}$$

$$M_{R \circ S} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array}\begin{array}{c} \begin{array}{ccc} \alpha & \beta & \gamma \end{array} \\ \begin{bmatrix} 1 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \end{array}$$

The nonzero entries indicate which elements are related by $R \circ S$.

2.5 Boolean Algebra

2.5.1 An Overview

In electronics applications, two-element Boolean algebra is particularly important. The universe is {0, 1}; each set has one element, either 0 or 1 (a set with one element is referred to as a singleton). The binary operations can be viewed as a special case of set operations; they are summarized in Table 2.2, and simple circuit models are shown in Figure 2.6.

The AND function of binary logic can be modeled by the circuit shown in Figure 2.6a. Each switch A and B may assume one of two values, ON or OFF, which model the logic states one and zero, respectively. The result of ANDing is modeled by the status of the light. There are four possible combinations of the setting of the two switches, with four possible outcomes that affect the status of the light. These are summarized in Table 2.3a. Such a table is usually referred to as a truth table. The models of the OR and complement are given in Figure 2.6b and c; the truth tables in Table 2.3b and c.

Instead of a mechanical switch, one may use an electric switch such as diode or a transistor. Circuits that perform the AND, OR, and complement operations are available as integrated circuit modules. Instead of drawing the circuit every time, logic symbols are usually used. They are shown in Figure 2.7.

Table 2.2 Binary operations

Operation	*Symbol*	*Corresponding Set Operation*
OR	$+ , \vee$	Union
AND	$\cdot , \wedge$	Intersection
Complement (Negation)	$\neg$, `	Complement

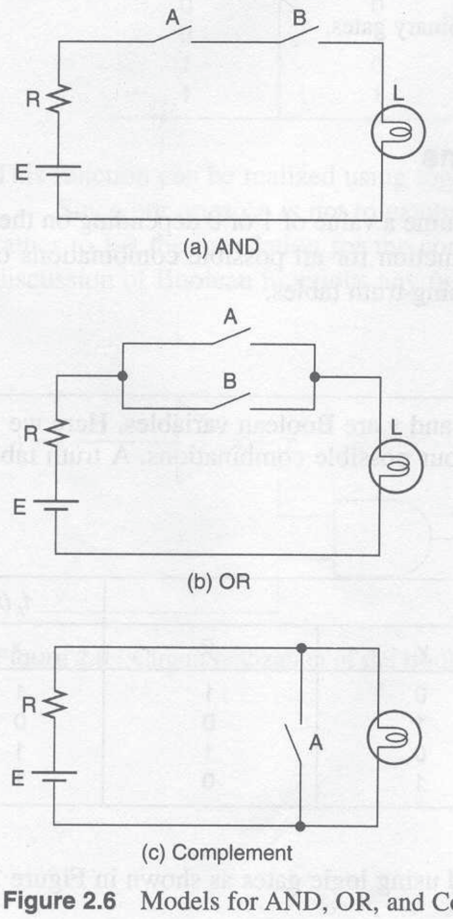


Figure 2.6 Models for AND, OR, and Complement operations.

Table 2.3 Summary of the operation of circuits of Figure 2.6 (OFF ≡ 0, ON ≡ 1; Light ≡ 1, No light ≡ 0)

(a) AND			*(b) OR*			*(c) Complement*	
A	B	L	A	B	L	A	L
0	0	0	0	0	0		
0	1	0	0	1	1	0	1
1	0	0	1	0	1		
1	1	1	1	1	1	1	0

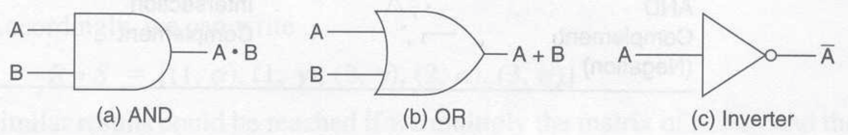


Figure 2.7 Logic symbols of binary gates.

2.5.2 Boolean Functions

A Boolean functions may assume a value of 1 or 0 depending on the status of each variable. The value of the function for all possible combinations of the Boolean variables can be expressed using truth tables.

EXAMPLE: Let $f(x, y) = x + \overline{y}$, where x and y are Boolean variables. Here we have two variables, x and y, resulting in four possible combinations. A truth table can be constructed as follows.

Variables			$f(x, y)$
x	y	$\overline{y}$	$x + \overline{y}$
0	0	1	1
0	1	0	0
1	0	1	1
1	1	0	1

This function can be realized using logic gates as shown in Figure 2.8.

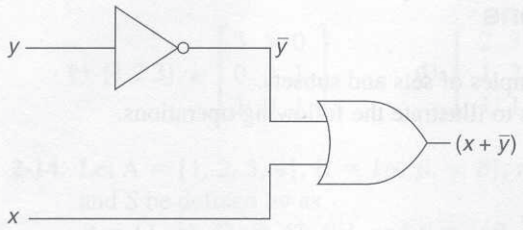


Figure 2.8 Circuit realization of the Boolean function $f(x, y) = x + \bar{y}$.

EXAMPLE: Let $f(x, y) = x \cdot \bar{y}$; then the truth table will be

x	y	$\bar{y}$	$f(x, y)$
0	0	1	0
0	1	0	0
1	0	1	1
1	1	0	0

This function can be realized using logic gates as shown in Figure 2.9.

Since our purpose is not to examine the Boolean logic circuits in detail, but rather to lay the foundation for the concepts of fuzzy logic, we will not take the discussion of Boolean functions any further for now.

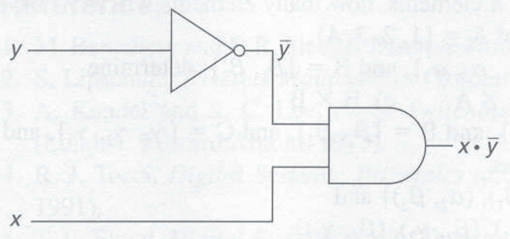


Figure 2.9 Circuit realization of the Boolean function $f(x, y) = x \cdot \bar{y}$.

Chapter 2 Questions

2-1. Give practical examples of sets and subsets.

2-2. Use Venn diagrams to illustrate the following operations.

a) $A \cup B \cup C$

b) $A \cap B \cap C$

c) $A \cap B \cup C$

2-3. Which of the following sets are equal (if any)?
$A = \{1, 2, 3, 4\}$, $B = \{2, 1, 4, 3\}$, $C = \{4, 3, 2, 1\}$, $D = \{1, 2, 4, 6\}$

2-4. Given that $I = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$, $A = \{1, 2, 3, 4, 5\}$, $B = \{4, 5, 6, 7, 8\}$, and $C = \{2, 3, 7, 8, 9\}$ determine

a) $A \cap B$

b) $A \cap C$

c) $\underline{A \cap B}$

d) $\underline{A \cup B}$

e) $A \cap B$

2-5. Given that $I = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$, $A = \{1, 2, 3, 4\}$, $B = \{2, 4, 5, 6\}$, and $C = \{1, 3, 5, 7, 9\}$ determine

a) $A \cap (B \cup C)$

b) $A \cup (B \cap C)$

c) $A \cup \bar{A}$

2-6. Simplify the following expressions using Venn diagrams.

a) $A \cup (A \cap B)$

b) $A \cap (A \cup B)$

c) $(A \cap B) \cup (A \cap \bar{B})$

d) $(A \cap \bar{B}) \cup (\bar{A} \cap B) \cup (A \cap B)$

2-7. Prove the following identities.

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

b) $(I \cap A) \cup (B \cap A) = A$

2-8. If A is a set that has n elements, how many elements will $P(A)$ have?

2-9. Find the power set of $A = \{1, 2, 3, 4\}$.

2-10. Given that $A = \{\alpha_1, \alpha_2, \alpha_3\}$, and $B = \{\beta_1, \beta_2\}$; determine

a) $A \times B$ **b)** $B \times A$ **c)** $B \times B$

2-11. Let $A = \{\alpha_1, \alpha_2, \alpha_3\}$, and $B = \{\beta_1, \beta_2\}$, and $C = \{\gamma_1, \gamma_2, \gamma_3\}$, and let the relations R and S be defined as
$R = \{(\alpha_1, \beta_1), (\alpha_2, \beta_1), (\alpha_2, \beta_2)\}$ and
$S = \{(\beta_1, \gamma_1), (\beta_1, \gamma_2), (\beta_2, \gamma_2), (\beta_2, \gamma_3)\}$.

a) Construct an arrow diagram to show the relation R and the relation S.

b) Determine the composite relation $R \circ S$ using the arrow diagram.

2-12. Give an algorithm for multiplying two matrices.

2-13. Evaluate

a) $[1\ 2\ 3] * \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$ **b)** $\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} * [1\ 2\ 3]$

c) $[1\ 2\ 3] * \begin{bmatrix} 3 & 5 & 0 \\ 0 & 3 & 1 \\ 1 & 0 & 1 \end{bmatrix}$ **d)** $\begin{bmatrix} 2 & 3 & 4 \\ 1 & 2 & 3 \\ 1 & 1 & 2 \end{bmatrix} * \begin{bmatrix} 1 & 3 & 0 \\ 1 & 2 & 1 \\ 0 & 0 & 2 \end{bmatrix}$

2-14. Let A = $\{1, 2, 3, 4\}$, B = $\{\alpha, \beta, \gamma, \delta\}$, and C = $\{a, b, c\}$, and let the relations R and S be defined by as
$R = \{1, \alpha), (2, \delta), (3, \beta)\}$, and $S = \{(\beta, a), (\beta, c), (\gamma, b), (\delta, c)\}$.
Determine $M_{R \circ S}$.

2-15. The exclusive OR (XOR) function is defined by $f(x, y) = \bar{x}y + x\bar{y}$. Construct its truth table and show its circuit realization using AND, OR, and inverter gates.

2-16. Construct the truth table for each of the following Boolean functions.
a) $f(x, y) = x + \bar{y}$
b) $f(x, y) = \bar{x} + y$
c) $f(x, y) = \bar{x}y + xy$
d) $f(x, y) = (\overline{x + y})$

2-17. Simplify the following Boolean functions.
a) $f(x, y) = \bar{x}(x + y)$
b) $f(x, y) = xy\,(x + \bar{y})$
c) $f(x, y) = xy\,(x\bar{y} + y)$

2-18. Construct circuit realizations of the following Boolean functions using AND, OR, and inverter gates.
a) $f(x, y) = \bar{x} + y$
b) $f(x, y, z) = xy + \overline{zy}$
c) $f(x, y) = \overline{\overline{x} \cdot \overline{y}}$
d) $f(x, y) = \overline{\overline{x} + \overline{y}}$
e) $f(x, y, z) = \overline{xyz} + \overline{x}y\overline{z} + \overline{x}y\overline{z}$

References

1. M. Benedicty and F. R. Sledge, *Discrete Mathematical Structures* (Toronto: HBJ, 1987).
2. S. Lipschutz, *Discrete Mathematics* (Toronto: McGraw-Hill, 1976).
3. A. Kandel and S. C. Lee, *Fuzzy Switching and Automata: Theory and Applications* (London: Edward Arnold, 1979).
4. R. J. Tocci, *Digital Systems: Principles and Applications* (New Jersey: Prentice Hall, 1991).
5. T. L. Floyd, *Digital Fundamentals* (Toronto: Merrill, 1990).
6. A. Kaufmann, *Introduction to the Theory of Fuzzy Subsets* (New York: Academic Press, 1975).
7. Y.-F. Lin and S.-Y. Lin, *Set Theory: An Intuitive Approach* (Boston: Houghton Mifflin Company, 1974).
8. P. W. Zehan and R. L. Johnson, *Elements of Set Theory* (Boston: Allyn and Bacon, Inc., 1972).

3 Fuzzy Sets

3.1 Basic Concepts

3.1.1 An Overview

Earlier, the classical or crisp set was defined as a collection of elements. An element either belongs to the set or not. The membership grade (or characteristic function), $\mu_A(x)$, can be either ONE or ZERO; no other values are allowed. This concept has led to the two-valued Boolean logic, which is very natural for electronics systems based on switches, since a switch may assume one of only two possible values. Such a system, however, is not natural to human beings' perceptions. We do not (or should not!) take either of two extreme positions and consider nothing in between. For example, when a universe of students is divided into "successful" and "not successful," usually students have varied degrees of belonging to either of these groups. In the successful group is a whole range from 60% to 100%, the marginally successful to the very successful. Among the non-successful there is also a range, starting with those who qualify for a make-up examination and ending with those who will be expelled. In other words, membership to either group is not just either ONE or ZERO.

In mathematical terms, a fuzzy set has a membership function that allows various degrees of membership for the elements of a given set. The membership function may be defined in terms of discrete values, or more commonly (in electronics applications), by a graph.

If X is a collection of objects denoted by x, then a fuzzy subset A, denoted ~ A, in X is a set of ordered pairs such that

$$\sim\mathrm{A} = \{x, \mu_{\mathrm{A}}(x) \mid x \in \mathrm{X}\}$$

where $\mu_{\mathrm{A}}(x)$ is the membership function of x in A. Elements with a zero membership are usually not listed. A fuzzy set will be written simply as A, i.e., without symbol ~, if it is quite clear that we are talking about a fuzzy set.

EXAMPLE: Consider the statement "a comfortable house for a three-person family," where the available houses have up to eight rooms.

The above statement can be described by a fuzzy set A.

$$\mathrm{A} = \{(1, 0.3), (2, 0.6), (3, 1.0), (4, 0.8), (5, 0.3)\}$$

A house with three rooms has the optimum comfort; a house with fewer or more rooms is less comfortable, although it may still be considered. Accordingly, a house with three rooms has a membership of $\mu_{\mathrm{A}}(x) = 1$, while houses with 1, 2, 4, and 5 rooms have memberships of 0.3, 0.6, 0.8, and 0.3, respectively. Houses with 6, 7, and 8 rooms have a membership of zero.

Different membership functions may be defined, leading to a different set. For exampe, A could have been

$$\mathrm{A} = \{(2, 0.4), (3, 0.8), (4, 1), (5, 0.2)\}.$$

Obviously, the membership function $\mu_{\mathrm{A}}(x)$ reflects how one may interpret the vague statement "a comfortable house for a three-person family."

In the fuzzy-sets literature, one finds various methods of describing a fuzzy set. All methods, however, lead somehow to a clear definition of an ambiguous statement. In the previous example, the degree of comfort, from a single point of view, was well-defined before writing the fuzzy set. Once an ambiguous statement is defined somehow, it is no longer ambiguous, at least for the definer! There is nothing fuzzy about fuzzy sets. The name "fuzzy" may then be viewed as just a name selected to distinguish the class of sets where $\mu_{\mathrm{A}}(x)$ can assume values other than one or zero. This is exactly like "imaginary" numbers. They are used for quite real applications. The name imaginary is just used to denote a different class of numbers.

EXAMPLE: If you remove one grain of sand at a time from a heap of sand it will eventually vanish. Which grain is responsible for the disappearance of the heap?

An answer from a philosophical point of view leads to a long, long discussion. Most probably no definite answer will be reached. From an engineering point of view, you will be required to define first what constitutes a heap of sand. Then, the quantity (Heap − n), where n is the number of grains removed, will have a membership to a fuzzy set based on *your* definition.

In engineering it appears as if the concern is not with the truth in a philosophical sense, but rather with a working model. We are satisfied with a model that explains our observations and enables us to produce new devices. We seek improvement to the model, or even a new one, as our observations are enhanced. There is, in general, room for more than one model.

It is ironic that the above distinction between engineering and philosophy may be considered a philosophical point of view—no escape from philosophy it seems! This situation provides another example of a fuzzy set defined by the observer.

Difficulties occur if you are not defining the membership function for yourself, which is usually the case. Suppose it was the real-estate agent, not you, who is the one to assign values for $\mu_A(x)$ to describe "a comfortable house for a three-person family." How would the task be accomplished? What will be the meaning of $\mu_A(x)$ in this case? One may be tempted to think of $\mu_A(x)$ in probabilistic terms.

Zadeh distinguishes between probability and possibility. The following example is one of the famous examples Zadeh put forward to illustrate how possibility is different from probability [8].

EXAMPLE: The possibility, $\pi(n)$, that Hans ate n eggs for breakfast is

n	0	1	2	3	4	5 +
Possibility	1	1	1	0.8	0.6	0.4

Possibilities were assigned to six events. They range from 0 (impossible) to 1 (entirely possible). The values assigned for $\pi(n)$ were suggested to reflect the degree of ease the observer will feel about the event. Giles [9] suggested a more general approach is to interpret $1 - \pi(n)$ as the "amount of surprise" the observer expects to experience on being told that n is the case, 1 denoting total surprise, and 0 denoting no surprise at all. The observer will not be surprised at all to hear that Hans ate up to two eggs, somewhat surprised if Hans ate three or four eggs, and very much surprised if Hans ate five or more eggs.

Possibility is distinguished from probability here by the facts that more than one event has a possibility of one and that the possibilities of all events may add up to a value higher than one.

Another way of writing a fuzzy set is given in the following example.

EXAMPLE: Let A = "integers close to ten"; then one may write

$$A = 0.1/7 + 0.5/8 + 0.8/9 + 1/10 + 0.8/11 + 0.5/12 + 0/13.$$

Sometimes it is more convenient rather to give the graph that represents the membership function (as in Figure 3.1).

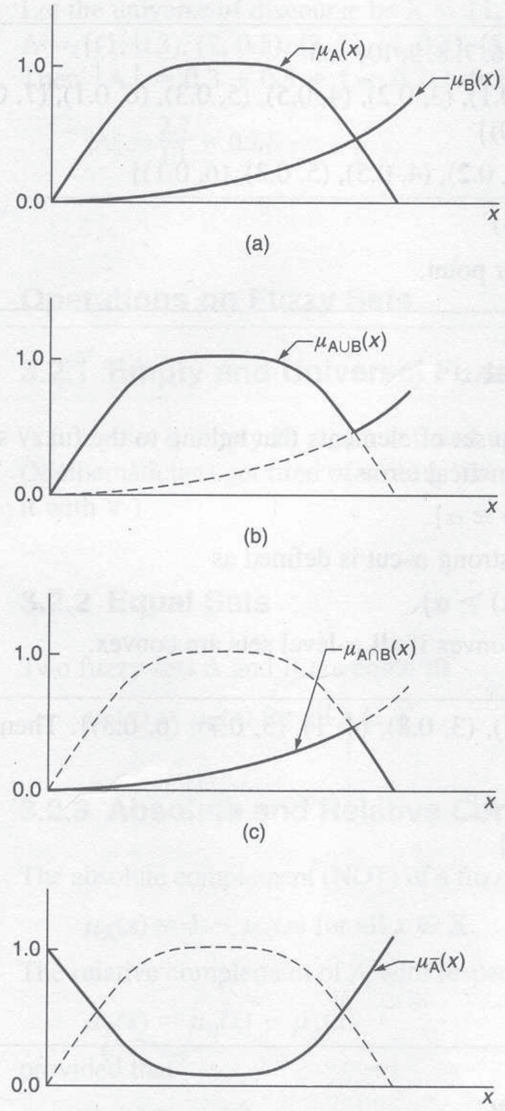


Figure 3.1 An illustration of fuzzy AND, OR, and NOT using generalized profile Venn diagrams. a) membership functions $\mu_A(x)$ and $\mu_B(x)$. b) results of $\mu_{A\cup B}(x)$, using $\mu_{A\cup B}(x) = \max\,[\mu_A(x), \mu_B(x)]$. c) results of $\mu_{A\cap B}(x)$, using $\mu_{A\cap B}(x) = \min\,[\mu_A(x), \mu_B(x)]$. d) results of $\mu_{\bar{A}}(x)$, using $\mu_{\bar{A}}(x) = 1 - \mu_A(x)$.

3.1.2 The Support of a Fuzzy Set

The support of a fuzzy set A is S(A), which is a crisp set of all $x \in X$ such that $\mu_A(x) > 0$.

The element x in X at which $\mu_A(x) = 0.5$ is called the crossover point. A fuzzy set whose support is a single element in X with $\mu_A(x) = 1$ is called a fuzzy singleton.

EXAMPLE: Let X = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10} and

$$A = \{(1, 0), (2, 0.1), (3, 0.2), (4, 0.5), (5, 0.3), (6, 0.1), (7, 0), (8, 0), (9, 0), (10, 0)\}$$
$$= \{(2, 0.1), (3, 0.2), (4, 0.5), (5, 0.3), (6, 0.1)\}$$

Then S(A) = {2, 3, 4, 5, 6}

$x = 4$ is the crossover point.

3.1.3 The α-Level Set

The α-level set is the crisp set of elements that belong to the fuzzy set A at least to the degree α. In mathematical terms,

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

The strong α-level set or strong α-cut is defined as

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

A fuzzy set is said to be convex if all α-level sets are convex.

EXAMPLE: Let A = {(1, 0.2), (2, 0.5), (3, 0.8), (4, 1), (5, 0.7), (6, 0.3)}. Then all possible α-level sets are:

$$A_{0.2} = \{1, 2, 3, 4, 5, 6\}$$
$$A_{0.3} = \{2, 3, 4, 5, 6\}$$
$$A_{0.5} = \{2, 3, 4, 5\}$$
$$A_{0.7} = \{3, 4, 5\}$$
$$A_{0.8} = \{3, 4\}$$
$$A_1 = \{4\}$$

3.1.4 The Cardinality

The cardinality of a fuzzy set A is defined as

$$|A| = \sum_{x \in A} \mu_A(x),$$

and the relative cardinality as

$$\|\text{A}\| = \frac{|\text{A}|}{|\text{X}|},$$

where $|\text{X}|$ is the cardinality of the universe of discourse.

EXAMPLE: Let the universe of discourse be X = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10} and A = {(1, 0.3), (2, 0.5), (3, 1), (4, 0.7), (5, 0.2)}.
Then $|\text{A}| = 0.3 + 0.5 + 1 + 0.7 + 0.2 = 2.7$ and $|\text{X}| = 10$, so

$$\|\text{A}\| = \frac{2.7}{10} = 0.27.$$

3.2 Operations on Fuzzy Sets

3.2.1 Empty and Universal Fuzzy Sets

A fuzzy set is empty iff $\mu_\Phi(x) = 0$ and universal iff $\mu_X(x) = 1$ for all $x \in \text{X}$. (Mathematicians get tired of writing "for all" over and over again, so they replace it with ∀.)

3.2.2 Equal Sets

Two fuzzy sets A and B are equal iff

$$\mu_\text{A}(x) = \mu_\text{B}(x) \text{ for all } x \in \text{X}.$$

3.2.3 Absolute and Relative Complements

The absolute complement (NOT) of a fuzzy set A is denoted by $\bar{\text{A}}$ and is defined by

$$\mu_{\bar{\text{A}}}(x) = 1 - \mu_\text{A}(x) \text{ for all } x \in \text{X}.$$

The relative complement of A with respect to B, denoted by B − A, is defined by

$$\underset{A-B}{\mu_{\text{B}}}(x) = \mu_\text{B}(x) - \mu_\text{A}(x)$$

provided that

$$\mu_\text{B}(x) \geq \mu_\text{A}(x).$$

EXAMPLE: Let A = {(0, 0.3), (1, 0.4), (2, 0.6), (3, 0.7)} and
B = {(0, 0.4), (1, 0.6), (2, 0.8), (3, 0.8)}.

Then

$$\bar{A} = \{[0, (1-0.3)], [1, (1-0.4)], [2, (1-0.6)], [3, (1-0.7)]\}$$
$$= \{(0, 0.7), (1, 0.6), (2, 0.4), (3, 0.3)\}.$$

and

$$B - A = \{(0, 0.1), (1, 0.2), (2, 0.2), (3, 0.1)\}.$$

3.2.4 The Union of Fuzzy Sets

The union of two fuzzy sets A and B is a fuzzy set C given by

$$C = A \cup B \text{ (or } C = A + B)$$

where

$$\mu_C(x) = \mu_A(x) \vee \mu_B(x)$$
$$= \max\,[\mu_A(x), \mu_B(x)];\ x \in X.$$

EXAMPLE: Let $A = \{(4, 0.1), (6, 0.3), (8, 0.6), (10, 1)\}$ and
$B = \{(0, 0.3), (2, 0.6), (4, 1), (6, 1), (8, 0.6), (10, 0.3)\}$.

Then

$$A \cup B = \max[0, 0.3]/0 + \max[0, 0.6]/2 + \max[0.1, 1]/4$$
$$+ \max[0.3, 1]/6 + \max[0.6, 0.6]/8 + \max[1, 0.3]/10.$$
$$= 0.3/0 + 0.6/2 + 1/4 + 1/6 + 0.6/8 + 1/10$$
$$= \{(0, 0.3), (2, 0.6), (4, 1), (6, 1), (8, 0.6), (10, 1)\}$$

3.2.5 The Intersection of Fuzzy Sets

The intersection of two fuzzy sets A and B is a fuzzy set C given by

$$C = A \cap B \text{ (or } C = A - B)$$

where

$$\mu_C(x) = \mu_A(x) \wedge \mu_B(x) = \min[\mu_A(x), \mu_B(x)];\ x \in X.$$

EXAMPLE: Let $A = \{(4, 0.1), (6, 0.2), (8, 0.4), (10, 0.5)\}$ and
$B = \{(0, 0.3), (2, 0.5), (4, 0.7), (5, 0.8), (8, 0.7)\}$.

Then

$$A \cap B = \min[0, 0.3]/0 + \min[0, 0.5]/2 + \min[0.1, 0.7]/4$$

$$+ \min[0, 0.8]/5 + \min[0.2, 0]/6$$
$$+ \min[0.4, 0.7]/8 + \min[0.5, 0]/10.$$
$$= 0/0 + 0/2 + 0.1/4 + 0/5 + 0/6 + 0.4/8 + 0/10$$
$$= \{(4, 0.1), (8, 0.4)\}$$

EXAMPLE: Let the membership functions $\mu_A(x)$ and $\mu_B(x)$ be defined by the graph shown in Figure 3.1a. Then, using the definitions of union (OR), intersection (AND), and complement (NOT), one can obtain the results shown in Figures 3.1b, c, and d.

EXAMPLE: Show that De Morgan's Laws hold for fuzzy sets, that is,

$$\overline{A \cup B} = \bar{A} \cap \bar{B} \qquad (1)$$
$$\overline{A \cap B} = \bar{A} \cup \bar{B} \qquad (2)$$

To show that eq. (1) is true is to show that

$$1 - \max[\mu_A(x), \mu_B(x)] = \min[(1 - \mu_A(x)), (1 - \mu_B(x))].$$

Now, if $\mu_A(x) > \mu_B(x)$, then both sides of the above equation are equal to $(1 - \mu_A(x)$. If $\mu_A(x) < \mu_B(x)$, then both sides become $1 - \mu_B(x)$. Similarly, it can be shown that equation (2) is an identity.

3.2.6 Models of Fuzzy AND, OR, Inverter

Similar to the case of binary operations, fuzzy operations can be illustrated by simple circuits. Instead of using switches to represent inputs, we use fuses (a wire that burns if the current through it exceeds a certain limit, leading to an open circuit between its terminals). Figure 3.2 shows circuits that illustrate fuzzy AND, OR, and Inverter operations.

Figure 3.2a shows a simple circuit that illustrates the concept of fuzzy AND. Two fuses with different ratings are arranged in series in the circuit. The fuse with the lowest rating determines the maximum current that can go through the circuit and hence the brightness of the light, i.e.,

Brightness α min[A, B].

Figure 3.2b illustrates the operation of a fuzzy OR. The fuse with the highest rating determines the brightness of the light, i.e.,

Brightness α max[A, B].

Figure 3.2c illustrates the operation of a fuzzy inverter. If the current in the circuit is unity and the current in the parallel resister is I, then the current through the lamp is (1 − I), i.e.,

Brightness α (1 − I).

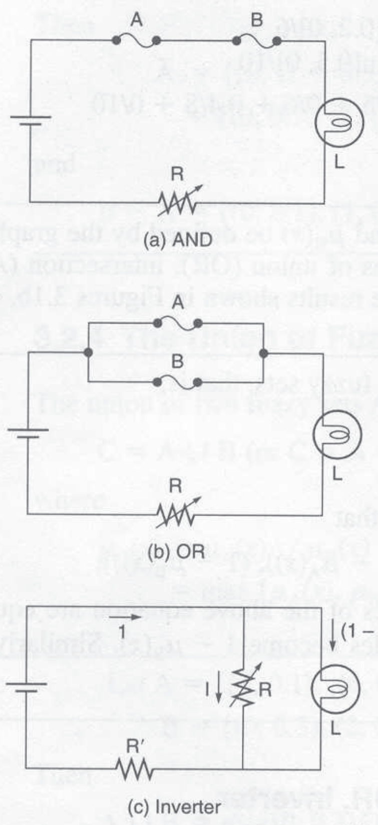


Figure 3.2 Models for fuzzy AND, OR, and Inverter operations.

3.3 Algebraic Operations

3.3.1 Cartesian Multiplication

The Cartesian product of two fuzzy sets A and B is a fuzzy set C denoted by A × B and defined as

$$\mathrm{C} = \mathrm{A} \times \mathrm{B} = \{\mu_{\mathrm{C}}(x)/(a, b) \mid a \in \mathrm{A}, b \in \mathrm{B}, \mu_{\mathrm{C}}(c) = \min(\mu_{\mathrm{A}}(a), \mu_{\mathrm{B}}(b))\}$$

EXAMPLE: Let A = 0.4/3 + 1/5 + 0.6/7 and

B = 1/5 + 0.6/6.

Then

$$\begin{aligned}\mathrm{A} \times \mathrm{B} &= \min(0.4, 1)/(3, 5) + \min(0.4, 0.6)/(3, 6)\\ &\quad + \min(1, 1)/(5, 5) + \min(1, 0.6)/(5, 6)\end{aligned}$$

$$+ \min(0.6, 1)/(7, 5) + \min(0.6, 0.6)/(7, 6).$$
$$= 0.4/(3, 5) + 0.4/(3, 6) + 1/(5, 5) + 0.6/(5, 6)$$
$$+ 0.6/(7, 5) + 0.6/(7, 6)$$

3.3.2 Algebraic Product

The product of two fuzzy sets A and B is denoted by AB and is defined by

$$AB = \{\mu_A(a)\ \mu_B(b) / x \mid x \in A, x \in B\}.$$

EXAMPLE: Let $A = 0.8/3 + 0.9/5 + 0.6/5$ and

$B = 0.7/3 + 0.8/4 + 0.3/5.$

Then

$AB = 0.56/3 + 0.18/5.$

It follows from the definition of the algebraic product that A^α, where α is any positive number, is

$$A^\alpha = \{(\mu_A(x))^\alpha/x \mid x \in A\}.$$

EXAMPLE: Let $A = 0.7/3 + 0.8/5 + 0.3/6 + 1/7.$

Then

$A^3 = 0.34/3 +0.51/5 + 0.027/6 + 1/7.$

The operations concentration and dilation are special cases of A^α. The concentration of A is defined as

$$\mathrm{CON}(A) = A^2$$

while the dilation of A is defined as

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

EXAMPLE: Let $A = 0.7/5 + 0.5/7 + 1/8.$

Then

$\mathrm{CON}(A) = 0.49/5 + 0.25/7 + 1/8$ and

$\mathrm{DIL}(A) = 0.84/5 + 0.7/7 + 1/8.$

These two operations are useful in the representation of linguistic hedges.

EXAMPLE: If the meaning of the term "few" is defined by

$$\text{few} = 1/1 + 1/2 + 0.8/3 + 0.6/4,$$

then

$$\text{CON(few)} = 1/1 + 1/2 + 0.64/3 + 0.36/4 \text{ and}$$
$$\text{DIL(few)} = 1/1 + 1/2 + 0.89/3 + 0.78/4.$$

The CON operation distributes over the union, intersection, and product. Since

$$(A + B)^2 = A^2 + AB + BA + B^2, \text{ and}$$
$$AB + BA \subset A^2 + B^2,$$

then it follows that $(A + B)^2 = A^2 + B^2$, which leads to

$$\text{CON}(A + B) = \text{CON}(A) + \text{CON}(B).$$

Similarly,

$$\text{CON}(A \cap B) = \text{CON}(A) + \text{CON}(B),$$

and

$$\text{CON}(AB) = \text{CON}(A)\ \text{CON}(B).$$

The operation of concentration can be composed with itself. Thus

$$\text{CON}^2(A) = A^4,$$

or, more generally,

$$\text{CON}^{\alpha}(A) = A^{\alpha 2}$$

where α is any integer ≥ 2.

A pictorial representation of the effect of the CON operation is shown in Figure 3.3.
The CON operation reduces the value of $\mu_A(x)$ for every x except where $\mu_A(x) = 1$.
It is interesting to observe that as $\alpha \to \infty$, $\mu_A(x)$ becomes two-valued, 1 or 0.
The CON operation makes a set less fuzzy and leads to crisp sets in the limit.
Similarly, the dilation operation can be generalized as

$$\text{DIL}^{1/\alpha}(A) = A^{0.5/\alpha}$$

where $\alpha \leq 2$.

In the limit as $\alpha \to \infty$ the DIL opeation leads to the universe of discourse.

Closely related to the CON operation is the so-called "contrast intensification." This operation increases the values of $\mu_A(x)$ that are above 0.5 and reduces those that are below 0.5. The result of applying a contrast intensifier INT to a fuzzy set A is denoted by INT(A), thus

$$\mu_{\text{INT(A)}}(x) \geq \mu_A(x) \qquad \text{for } \mu_A(x) \geq 0.5$$
$$\mu_{\text{INT(A)}}(x) \leq \mu_A(x) \qquad \text{for } \mu_A(x) \leq 0.5.$$

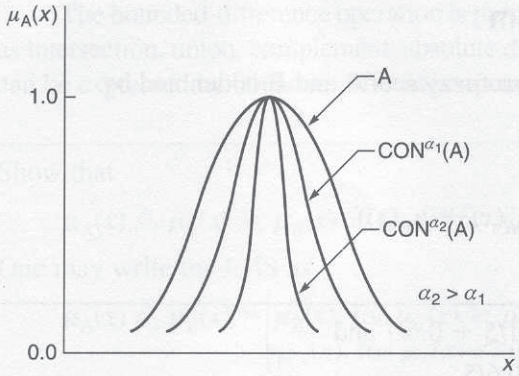


Figure 3.3 Effects of CON operation on the membership function.

This definition leads to a possible expression for such an operator as

$$\mu_{\mathrm{INT(A)}}(x) \geq 2\mu_{A}{}^{2}(x) \qquad \text{for } 0 \leq \mu_A(x) \leq 0.5$$

$$\mu_{\mathrm{INT(A)}}(x) \leq 1 - 2\,[1 - 2\mu_A(x)]^2 \qquad \text{for } 0.5 \leq \mu_A(x) \leq 1.$$

The intensification operation distributes over the union, intersection, and product:

$$\mathrm{INT}(A \cup B) = \mathrm{INT}(A) \cup \mathrm{INT}(B)$$

$$\mathrm{INT}(A \cap B) = \mathrm{INT}(A) \cap \mathrm{INT}(B)$$

$$\mathrm{INT}(AB) = \mathrm{INT}(A)\,\mathrm{INT}(B)$$

3.3.3 Algebraic Sum

The algebraic sum of fuzzy sets A and B is a fuzzy set C,

$$C = A + B,$$

where

$$\mu_C(x) = \mu_A(x) + \mu_B(x) - \mu_A(x)\,\mu_B(x).$$

Do not confuse the algebraic sum with the OR operation.

EXAMPLE: Let $A = 0.5/3 + 1/5 + 0.6/7$ and

$B = 1/3 + 0.6/5.$

Then C is composed of three members, 3, 5, and 7, with membership functions $\mu_C(x_1)$, $\mu_C(x_2)$, and $\mu_C(x_3)$, respectively.

$$\mu_C(x_1) = 0.5 + 1 - 1(0.5) = 1$$

$$\mu_C(x_2) = 1 + 0.6 + 1(0.6) = 1$$

$$\mu_C(x_3) = 0.6 + 0 - 0(0.6) = 0.6$$

Thus, $C = A + B = 1/3 + 1/5 + 0.6/7.$

3.3.4 Bounded Sum

The bounded sum of two fuzzy sets A and B is denoted by

$$C = A \oplus B$$

where

$$\mu_C(x) = \min(1, \mu_A(x) + \mu_B(x)).$$

EXAMPLE: Let $A = 0.5/4 + 1/5 + 0.6/7$ and
$B = 0.3/4 + 0.6/5.$

Then the bounded sum C is composed of three members, 4, 5, and 7, with membership functions $\mu_C(x_1)$, $\mu_C(x_2)$, and $\mu_C(x_3)$, respectively.

$$\mu_C(x_1) = \min(1, 0.5 + 0.3) = 0.8$$
$$\mu_C(x_2) = \min(1, 1 + 0.6) = 1$$
$$\mu_C(x_3) = \min(1, 0.6 + 0) = 0.6$$

Thus, $C = A \oplus B = 0.8/4 + 1/5 + 0.6/7.$

3.3.5 Bounded Difference

The bounded difference of two fuzzy sets A and B is denoted by

$$C = A \ominus B$$

where

$$\mu_C(x) = \min(1, (\mu_A(x) - \mu_B(x)).$$

EXAMPLE: Let $A = 0.5/3 + 0.3/5 + 0.6/7$ and
$B = 0.3/3 + 0.2/5.$

Then the bounded difference is composed of three members, 3, 5, and 7, with membership functions $\mu_C(x_1)$, $\mu_C(x_2)$, and $\mu_C(x_3)$, respectively.

$$\mu_C(x_1) = \min(1, 0.5 - 0.3) = 0.2$$
$$\mu_C(x_2) = \min(1, 0.3 - 0.2) = 0.1$$
$$\mu_C(x_3) = \min(1, 0.6 - 0) = 0.6.$$

Thus, $C = A \ominus B = 0.2/3 + 0.1/5 + 0.6/7.$

The bounded-difference operation is particularly useful because operations such as intersection, union, complement, absolute difference, implication, and equivalence can be expressed by bounded-difference operations (and arithmetic sums) [7].

EXAMPLE: Show that

$$\mu_A(x) \wedge \mu_B(x) = \mu_B(x) \ominus (\mu_B(x) \ominus \mu_A(x)).$$

One may write the LHS as

$$\mu_A(x) \wedge \mu_B(x) = \begin{cases} \mu_B(x), \text{ for } \mu_A(x) \geq \mu_B(x), \\ \mu_A(x), \text{ for } \mu_A(x) < \mu_B(x). \end{cases}$$

$$\text{since } \mu_B(x) \ominus \mu_A(x) = \begin{cases} \mu_B(x) - \mu_A(x), \text{ for } \mu_B(x) \geq \mu_A(x), \\ 0, \qquad\qquad\quad \text{ for } \mu_B(x) < \mu_A(x), \end{cases} \tag{1}$$

then one can examine the RHS under the two conditions $\mu_B(x) > \mu_A(x)$ and $\mu_B(x) < \mu_A(x)$.

- If $\mu_B(x) > \mu_A(x)$, then

  $\mu_B(x) \ominus \mu_A(x) = \mu_B(x) - \mu_A(x)$, and hence

  $$\mu_B(x) \ominus (\mu_B(x) \ominus \mu_A(x)) = \mu_B(x) - (\mu_B(x) - \mu_A(x)) = \mu_A(x).$$
- If $\mu_B(x) < \mu_A(x)$, then $\mu_B(x) \ominus \mu_A(x) = 0$, and hence

  $$\mu_B(x) \ominus (\mu_B(x) \ominus \mu_A(x) = \mu_B(x) \ominus (0) = \mu_B(x).$$

The results may be summarized as

$$\mu_B(x) \ominus (\mu_B(x) \ominus \mu_A(x)) = \mu_B(x), \text{ for } \mu_A(x) \geq \mu_B(x), \quad \mu_A(x), \text{ for } \mu_A(x) < \mu_B(x). \tag{2}$$

From (1) and (2) one may conclude that

$$\mu_A(x) \wedge \mu_B(x) = \mu_B(x) \ominus (\mu_B(x) \ominus \mu_A(x).$$

It can also be shown that

- for union $A \vee B$,

  $\mu_A(x) \vee \mu_B(x) = \mu_A(x) + (\mu_B(x) \ominus \mu_A(x))$.
- for absolute difference, $|A - B|$,

  $|\mu_A(x) - \mu_B(x)| = (\mu_A(x) \ominus \mu_B(x)) + (\mu_B(x) \ominus \mu_A(x))$.
- for implication, $A \rightarrow B$,

  $\mu_A(x) \rightarrow \mu_B(x) = 1 \ominus (x \ominus y)$.
- for equivalence, $A \rightleftarrows B$

  $\mu_A(x) \rightleftarrows \mu_B(x) = 1 \ominus (\mu_A(x) \ominus \mu_B(x) + (\mu_B(x) \ominus \mu_A(x))$.

3.3.6 Convex Combination

The convex combination of n fuzzy sets $A_1, A_2, \ldots, A_n$, is a weighted combination of these sets that leads to a fuzzy set A, where

$$\mu_A(x) = \omega_1(x)\, \mu_{A_1}(x) + \ldots + \omega_n(x)\, \mu_{A_n}(x)$$

with the weights being such that $\omega_1(x) + \ldots + \omega_n(x) = 1$.

EXAMPLE: Let $A_1 = 0.7/2 + 1/4 + 0.7/5$ and
$A_2 = 0.3/2 + 1/3 + 0.4/5$.

Suppose $\omega_1 = 0.8$ and $\omega_2 = 0.2$. Then

$$A = (0.8 \times 0.7 + 0.2 \times 0.3)/2 + 0.8/4 + 0.2/3 + (0.8 \times 0.7 + 0.2 \times 0.4)/5$$
$$= 0.62/2 + 0.8/4 + 0.2/3 + 0.64/5.$$

3.3.7 Fuzzification

The process of fuzzification transforms a set (fuzzy or crisp) to an approximating set that is more fuzzy. This process is a generalization of the dilation operation.

The essence of the fuzzification process is point fuzzification. Point fuzzification transforms a singleton set $1/u$ in U to a fuzzy set u that varies around u. The symbol $\sim$ is used to denote a fuzzifier. For example, ~ 50 represents the fuzzy set of real numbers that are approximately equal to 50 (selected, of course, from the universe of discourse). In order to express the dependence of u on $\sim u$, $\sim u$ is written as $u = K(u)$. The fuzzy set $K(u)$ is referred to as the kernel of fuzzification.

EXAMPLE: Let the universe of discourse be

$$U = 1 + 2 + 3 + 4.$$

Let A be a singleton set given by 1/2. Then one may define K(2) as

$$K(2) = 0.5/1 + 1/2 + 0.7/3 + 0.2/4,$$

and hence,

$$\sim A = \{0.5/1, 1/2, 0.7/3, 0.2/4\}$$

The choice of how to fuzzify, i.e., the description of $K(u)$ depends on the meaning of the sets and your criteria.

Now, consider a general case; let

$$A = \mu_1/x_1 + \ldots + \mu_n/x_n.$$

Since every member and its membership grade is independent of other members and their membership grades, then one may assume that the process of fuzzification is linear, and superposition may be applied, i.e., fuzzifying each member alone.

The process of fuzzification of μ_i/x_i gives rise to two special cases: one where μ_i is kept constant and x_i is fuzzified (support fuzzification), and the other where μ_i is fuzzified and x_i is kept constant (grade fuzzification). We will examine each case in a little more detail.

1. Support fuzzification (s-fuzzification)
   Support fuzzification of A is denoted by SF(A; K) and defined as
   $$A = SF(A; K) = \mu_1\, K(x_1) + \ldots + \mu_n\, K(x_n),$$
   where $\mu_i\, K(x_i)$ is a fuzzy set that is the product of a scalar constant μ_i and a fuzzy set $K(x_i)$.

EXAMPLE: Let the universe of discourse be

$U = 1 + 2 + 3 + 4$, and let

$A = 0.8/1 + 0.5/2.$

Assume

$K(1) = 1/1 + 0.3/2$ and

$K(2) = 1/2 + 0.3/1 + 0.2/3.$

Then

$$SF(A; K) = 0.8(1/1 + 0.3/2) + 0.5(1/2 + 0.3/1 + 0.2/3)$$
$$= 0.8/1 + 0.24/2 + 0.5/2 + 0.15/1 + 0.1/3.$$

Since the union is a maximizing operation, then

$$SF(A; K) = 0.8/1 + 0.5/2 + 0.1/3.$$

An important difference between the s-fuzzification and dilation operations is that a dilation of a crisp set yields the same crisp set, which is not the case, in general, with s-fuzzification.

2. Grade fuzzification (g-fuzzification)
   Grade fuzzification of A is denoted by GF(A; K) and defined as
   $$A = GF(A; K) = K(\mu_1)/x_1 + \ldots + K(\mu_n)/x_n$$
   where $K(\mu_i)$ denotes point fuzzification of μ_i.

EXAMPLE: Let the universe of discourse be

$U = 1 + 2 + 3 + 4$, and let

$A = 0.8/1 + 0.5/2.$

Then

$$GF(A; K) = A = 0.8/1 + 0.5/2.$$

If we define

$K(0.8) = 1/0.8 + 0.7/0.6 + 0.3/0.5$ and
$K(0.5) = 1/0.5 + 0.6/0.4 + 0.5/0.3,$

then

$$A = (1/0.8 + 0.7/0.6 + 0.3/0.5)/1 + (1/0.5 + 0.6/0.4 + 0.5/0.3)/2$$

3.4 Fuzzy Relations

3.4.1 Fundamental Concepts

The concept of relations discussed in section 2.4 may be extended to fuzzy sets. A fuzzy relation from a fuzzy set A to a fuzzy set B is

$$R = A \times B = \{(a, b) \mid a \in A, b \in B\}.$$

When A = B, R is known as a fuzzy relation on A.

Given the finite sets $A = \{a_1, a_2, \ldots, a_m\}$ and $B = \{b_1, b_2, \ldots, b_n\}$, a fuzzy relation A × B can be expressed by an $m \times n$ matrix:

$$M_R = \begin{bmatrix} \mu_R(a_1, b_1) & \mu_R(a_1, b_2) & \ldots & \mu_R(a_1, b_n) \\ \mu_R(a_2, b_1) & \mu_R(a_2, b_2) & \ldots & \mu_R(a_2, b_n) \\ & \cdot & & \\ & \cdot & & \\ \mu_R(a_m, b_1) & \mu_R(a_m, b_2) & & \mu_R(a_m, b_n) \end{bmatrix}$$

This matrix is referred to as a fuzzy matrix. The elements of the fuzzy matrix have values within the interval [0, 1], since μ_R has values within that range.

EXAMPLE: Let $A = \{a_1, a_2, a_3\}$ and $B = \{b_1, b_2, b_3, b_4\}$. Let R be a relation from A to B, given by

$$R = 0.1/(a_1, b_3) + 0.8/(a_1, b_4) + 0.8/(a_2, b_2) + 0.1/(a_3, b_1) + 0.8/(a_3, b_2) + 1/(a_3, b_3) + 0.8/(a_3, b_4).$$

The corresponding fuzzy matrix M_R is a 3 × 4 matrix with entries $\mu_R(a, b)$ that correspond with cell (a, b) such that

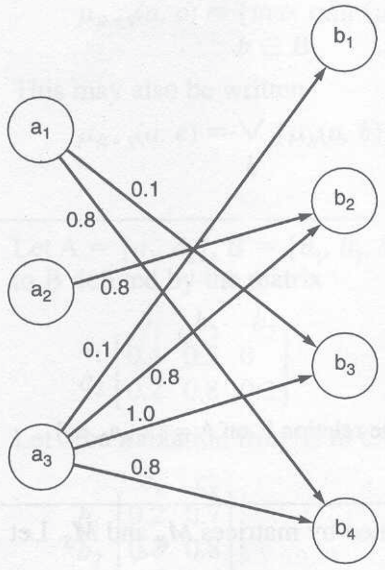


Figure 3.4 A fuzzy arrow graph for the relation R from A to B.

$$M_R = \begin{array}{c|cccc} & b_1 & b_2 & b_3 & b_4 \\ a_1 & 0 & 0 & 0.1 & 0.8 \\ a_2 & 0 & 0.8 & 0 & 0 \\ a_3 & 0.1 & 0.8 & 1 & 0.8 \end{array}$$

The corresponding fuzzy graph is shown in Figure 3.4.

EXAMPLE: Let a fuzzy relation R on A $= \{a_1, a_2, a_3\}$ be

$$R = 0.2/(a_1, a_1) + 0.5/(a_1, a_2) + 0.4/(a_1, a_3) + 1/(a_2, a_2) + 0.3/(a_2, a_3) + 1/(a_3, a_2) + 0.7/(a_3, a_3).$$

Then the fuzzy matrix for R is a 3×3 matrix:

$$M_R = \begin{array}{c} \\ a_1 \\ a_2 \\ a_3 \end{array} \begin{array}{c} \begin{array}{ccc} a_1 & a_2 & a_3 \end{array} \\ \begin{bmatrix} 0.2 & 0.5 & 0.4 \\ 0.0 & 1 & 0.3 \\ 0.0 & 1 & 0.7 \end{bmatrix} \end{array}$$

The corresponding fuzzy graph is as shown in Figure 3.5.

Since the fuzzy relation from A to B is a fuzzy set in A $\times$ B, the operations for fuzzy sets can be used.

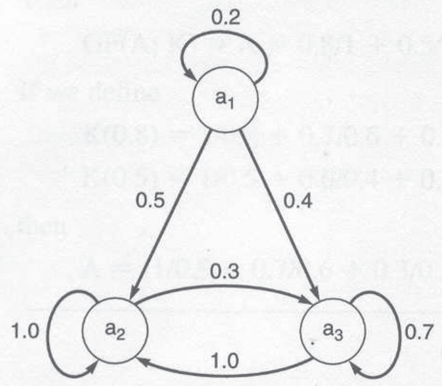


Figure 3.5 A fuzzy arrow graph for the relation R on A = $\{a_1, a_2, a_3\}$.

EXAMPLE: Let R and S be binary relations defined by matrices M_R and M_S. Let

$$M_R = \begin{bmatrix} 0.1 & 0.6 & 0.3 \\ 0 & 0.1 & 1 \\ 0.1 & 0.7 & 0.8 \end{bmatrix} \text{ and}$$

$$M_S = \begin{bmatrix} 0.7 & 0.8 & 1 \\ 0.2 & 0 & 0 \\ 0.3 & 0 & 0.1 \end{bmatrix}$$

Then, $R \cap S$ will be described by $M_{R \cap S}$ given by

$$\begin{bmatrix} \min(0.1, 0.7) & \min(0.6, 0.8) & \min(0.3, 1) \\ \min(0, 0.2) & \min(0.1, 0) & \min(1, 0) \\ \min(0.1, 0.3) & \min(0.7, 0) & \min(0.8, 0.1) \end{bmatrix}$$

$$= \begin{bmatrix} 0.1 & 0.6 & 0.3 \\ 0 & 0 & 0 \\ 0.1 & 0 & 0.1 \end{bmatrix}$$

3.4.2 Composition of Fuzzy Relations

Consider two fuzzy relations R on A × B and S on B × C. Contrary to crisp relations, fuzzy R and S may be composed in various ways, since μ_R and μ_S can assume values in the range [0, 1], not just 0 or 1. Their composition is denoted by $R \circ S$ and is a fuzzy relation on A × C. The best known and the most frequently used is the so-called "max-min" composition. It is defined by

$$\mu_{R \circ S}(a, c) = \{\max_{b \in B} \min (\mu_R(a, b), \mu_S(a, b)) \mid (a, c)\}.$$

This may also be written

$$\mu_{R \circ S}(a, c) = \bigvee_{b} \{\mu_R(a, b) \wedge \mu_S(a, b)\}$$

EXAMPLE: Let A = $\{a_1, a_2\}$, B = $\{b_1, b_2, b_3\}$, and C = $\{c_1, c_2\}$. Let R be a relation from A to B defined by the matrix

$$\begin{array}{c|ccc} & b_1 & b_2 & b_3 \\ a_1 & 0.4 & 0.5 & 0 \\ a_2 & 0.2 & 0.8 & 0.2 \end{array}$$

Let S be a relation from B to C defined by the matrix

$$\begin{array}{c|cc} & c_1 & c_2 \\ b_1 & 0.2 & 0.7 \\ b_2 & 0.3 & 0.8 \\ b_3 & 1 & 0 \end{array}$$

Then the max-min composition of R and S is

$$R \circ S = \begin{array}{c} \\ a_1 \\ a_2 \end{array}\!\!\begin{array}{ccc} b_1 & b_2 & b_3 \\ \end{array} \begin{bmatrix} 0.4 & 0.5 & 0 \\ 0.2 & 0.8 & 0.2 \end{bmatrix} \circ \begin{array}{c} b_1 \\ b_2 \\ b_3 \end{array}\begin{bmatrix} 0.2 & 0.7 \\ 0.3 & 0.8 \\ 1 & 0 \end{bmatrix} = \begin{array}{c} a_1 \\ a_2 \end{array}\begin{bmatrix} m & n \\ p & q \end{bmatrix}$$

where

$$\begin{aligned} m &= \max\{\min(0.4, 0.2), \min(0.5, 0.3), \min(0, 1)\} \\ &= \max[0.2, 0.3, 0] = 0.3 \\ n &= \max[\min(0.4, 0.7), \min(0.5, 0.8), \min(0, 0)] \\ &= \max[0.4, 0.5, 0] = 0.5 \\ p &= \max[\min(0.2, 0.2), \min(0.8, 0.3), \min(0.2, 1)] \\ &= \max[0.2, 0.3, 0.2] = 0.3 \\ q &= \max[\min(0.2, 0.7), \min(0.8, 0.8), \min(0.2, 0)] \\ &= \max[0.2, 0.8, 0] = 0.8 \end{aligned}$$

Other kinds of compositions for fuzzy relations can be defined. For example,

- min-max composition (sometimes written as $R \,\square\, S$). It is the dual composition relation of max-min composition.
- max-product composition defined by
  $R \underset{b}{\cdot} S = \{[(a, c), \max(\mu_R(a, b) \cdot \mu_S(a, c))], a \in A, b \in B, c \in C\}.$
- max-av composition defined by
  $R \underset{\text{av}}{\circ} S = \{[(a, c), \tfrac{1}{2} \max (\mu_R(a, b) + \mu_S(a, c)], a \in A, b \in B, c \in C\}.$

3.4.3 Various Types of Fuzzy Relations

Let R be a fuzzy relation on $A \times A$ then for any $x, y, z \in A$, R is said to be:

- reflexive if $\mu_R(x, x) = 1$ and $\mu(x, y) < 1, y \neq x$.

  Example:

$$M_R = \begin{bmatrix} 1 & 0.3 & 0.2 \\ 0.4 & 1 & 0.7 \\ 0.5 & 0.9 & 1 \end{bmatrix}$$

- antireflexive if $\mu_R(x, x) = 0$.

  Example:

$$M_R = \begin{bmatrix} 0 & 0.3 & 0.4 \\ 0.1 & 0 & 0.7 \\ 0.1 & 0.5 & 0 \end{bmatrix}$$

- symmetric if $\mu_R(x, y) = \mu_R(y, x)$ (columns and rows are the same).

  Example:

$$M_R = \begin{bmatrix} 0.2 & 0.7 & 0.4 \\ 0.7 & 0.9 & 0.1 \\ 0.4 & 0.1 & 0.3 \end{bmatrix}$$

- antisymmetric if $\mu_R(x, y) \neq \mu_R(y, x)$ (columns and rows are not the same).

  Example:

$$M_R = \begin{bmatrix} 0.3 & 0.5 & 0.7 \\ 1 & 0.5 & 0.2 \\ 0.3 & 0.2 & 0.8 \end{bmatrix}$$

  max-min transitive if $\mu_R(x, z) \geq \max \min(\mu(x, y), \mu(y, z))$.

A fuzzy relation is said to be a pre-order if it is reflexive and has max-min transitivity. If it is reflexive, symmetric, and has max-min transitivity, it is said to be a similarity relation. If it is reflexive and symmetric, it is said to be a resemblance relation.

A fuzzy relation is said to be an identity relation, I, if the diagonal elements of M_R are ones.

It is a zero relation, 0, if all the elements of M_R are zeros, and a universe relation, E, if all elements are ones. For example,

$$I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}; \qquad 0 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}; \qquad E = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

The composition of fuzzy relations plays an important role in fuzzy control as we discuss it in Chapter 5.

3.5 Geometrical Illustration

A geometrical model of fuzzy sets was put forward by Kosko [10]. The model is very helpful in illustrating various concepts pertaining to fuzzy sets. In the following we give a simplified, brief account of that model.

Consider a crisp set of two elements such as X = $\{x_1, x_2\}$. The power set is then given by $\{\Phi, X, \{x_1\}, \{x_2\}\}$. Each of the four subsets may be coded by two bits, i.e., Φ, X, $\{x_1\}$, and $\{x_2\}$ may be coded as [0 0], [1 1], [1 0], and [0 1], respectively. The presence of the ith element, x_i, is represented by 1 and its absence of 0. Each of these points may then be assigned to a vertex of a two-dimensional unit cube (i.e., a square) as shown in Figure 3.6. If X had been a crisp set of n elements, each crisp subset would have been represented by one of the vertices of a unit hypercube in n dimensions.

Fuzzy subsets can be represented by other points within the hypercube (within a square in our case). Figure 3.6 also shows the representation of fuzzy subset A = [0.3 0.8]. Element x_1 belongs to A to the degree 0.3, while element x_2 belongs to A to the degree 0.8.

The subsets resulting from the complement, union, and intersection operations may also be represented by points within the square as shown in Figure 3.7. The point at the middle represents a maximally fuzzy set. The middle point set has the property

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

Kosko [10] suggested further that the properties of the middle point may be used to measure the fuzziness of a subset. A subset A is properly fuzzy iff $\mathrm{A} \cap \overline{\mathrm{A}} \neq \Phi$ and $\mathrm{A} \cup \overline{\mathrm{A}} \neq \mathrm{X}$. The closer the point representing A to the middle, the fuzzier the set. A measure of fuzziness can then be defined as

$$E(\mathrm{A}) = \frac{a}{b}$$

where a is the distance of A to the nearest vertex and b is the distance of A to the farthest vertex.

From the geometry of Figure 3.7 one may conclude that

$$E(\mathrm{A}) = \frac{a}{b} = \frac{M(\mathrm{A} \cap \overline{\mathrm{A}})}{M(\mathrm{A} \cup \overline{\mathrm{A}})}$$

where a and b are defined as "fuzzy Hamming distance," rather than Euclidean distance. The Hamming distance between sets A and B is defined as

$$\sum_i \left| m_{\mathrm{A}}(x_i) - m_{\mathrm{B}}(x_i) \right|,$$

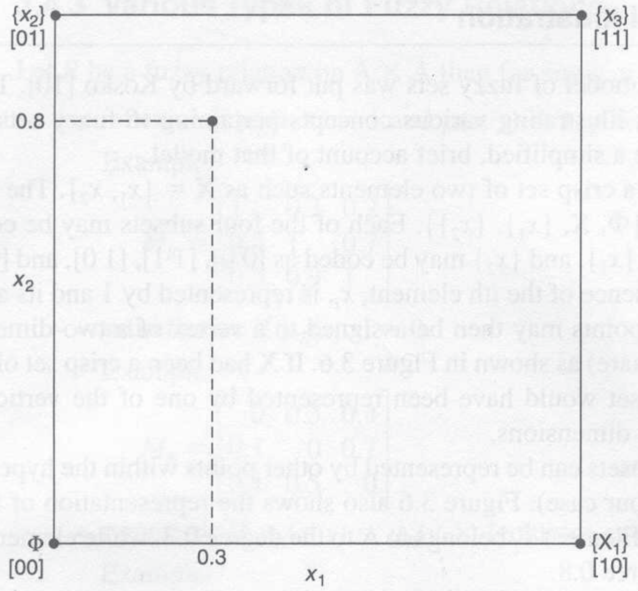


Figure 3.6 Point representation of fuzzy sets. The vertices of the square represent crisp subsets. Other points represent fuzzy subsets; for example, fuzzy subset A is represented by the bit vector [0.3 0.8].

and $M(\mathrm{A})$ is a cardinality measure defined by

$$M(\mathrm{A}) = \Sigma\, \mathrm{m_A}(\mathrm{x}_i).$$

EXAMPLE: Let A = [0.3 0.8].

Then the near vertex $\mathrm{A_{near}} = [0\ 1]$ and

the far vertex $\mathrm{A_{far}} = [1\ 0]$.

Accordingly, $a = 0.3 + 0.2 = 0.5$ and $b = 0.7 + 0.8 = 1.5$.

Thus $E(\mathrm{A}) = \dfrac{a}{b} = \dfrac{1}{3}$

Equivalently, we could have used the relation

$$E(\mathrm{A}) = \frac{M(\mathrm{A} \cap \overline{\mathrm{A}})}{M(\mathrm{A} \cup \overline{\mathrm{A}})}.$$

Thus, if A = [0.3 0.8], then $\overline{\mathrm{A}}$ = [0.7 0.2], and
$\mathrm{A} \cap \overline{\mathrm{A}}$ = [min (0.3, 0.7) min (0.8, 0.2)] = [0.3 0.2], leading to
$M(\mathrm{A} \cap \overline{\mathrm{A}}) = 0.2 + 0.3 = 0.5$.
Similarly, $M(\mathrm{A} \cup \overline{\mathrm{A}}) = 1.5$, and hence

$$E(\mathrm{A}) = \frac{1}{3}.$$

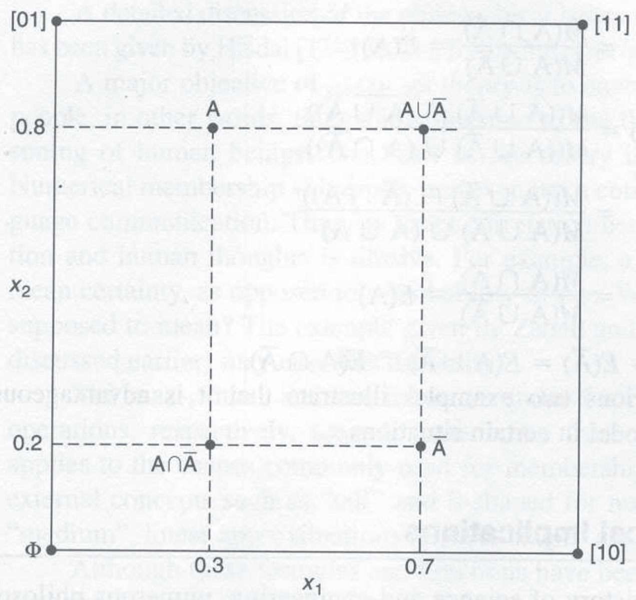


Figure 3.7 Point representation of various fuzzy subsets.

Note that at a vertex $a = 0$, and hence E(A) = 0, indicating a non-fuzzy set. At the middle point $a = b$, and hence E(A) = 1, indicating a maximally fuzzy set.

EXAMPLE: Show that

$$E(\text{A}) = E(\bar{\text{A}}) = E(\text{A} \cap \bar{\text{A}}) = E(\text{A} \cup \bar{\text{A}}).$$

The validity of the above identities may be realized from Figure 3.7. From that figure, the distance between A, $\bar{\text{A}}$, A ∩ $\bar{\text{A}}$, and A ∪ $\bar{\text{A}}$ and the nearest vertices $\{x_2\}$, $\{x_1\}$, Φ, and X, respectively, are the same; the distances to the farthest vertices $\{x_1\}$, $\{x_2\}$, X, and Φ, respectively, are also the same.

Equivalently, we could have used the following relations:

$$E(\text{A}) = \frac{M(\text{A} \cap \bar{\text{A}})}{M(\text{A} \cup \bar{\text{A}})} = \frac{M(\bar{\text{A}} \cap \text{A})}{M(\bar{\text{A}} \cup \text{A})} = E(\bar{\text{A}})$$

$$E(\text{A} \cap \bar{\text{A}}) = \frac{M((\text{A} \cap \bar{\text{A}}) \cap (\overline{\text{A} \cap \bar{\text{A}}}))}{M((\text{A} \cap \bar{\text{A}}) \cup (\overline{\text{A} \cap \bar{\text{A}}}))}$$

$$= \frac{M((\text{A} \cap \bar{\text{A}}) \cap (\bar{\text{A}} \cup \text{A}))}{M((\text{A} \cap \bar{\text{A}}) \cup (\bar{\text{A}} \cup \text{A}))}$$

$$= \frac{M((\text{A} \cap \bar{\text{A}}) \cup (\text{A} \cap \bar{\text{A}}))}{M((\text{A} \cup \bar{\text{A}}) \cap (\text{A} \cup \bar{\text{A}}))}$$

$$= \frac{M(\mathrm{A} \cap \bar{\mathrm{A}})}{M(\mathrm{A} \cup \bar{\mathrm{A}})} = E(\mathrm{A})$$

$$E(\mathrm{A} \cup \bar{\mathrm{A}}) = \frac{M((\mathrm{A} \cup \bar{\mathrm{A}}) \cap \overline{(\mathrm{A} \cup \bar{\mathrm{A}})})}{M((\mathrm{A} \cup \bar{\mathrm{A}}) \cup (\mathrm{A} \cap \bar{\mathrm{A}}))}$$

$$= \frac{M((\mathrm{A} \cup \bar{\mathrm{A}}) \cap (\bar{\mathrm{A}} \cap \mathrm{A}))}{M(\mathrm{A} \cup \bar{\mathrm{A}}) \cup (\bar{\mathrm{A}} \cup \mathrm{A})}$$

$$= \frac{M(\mathrm{A} \cap \bar{\mathrm{A}})}{M(\mathrm{A} \cup \bar{\mathrm{A}})} = E(\mathrm{A})$$

Hence, $E(\mathrm{A}) = E(\bar{\mathrm{A}}) = E(\mathrm{A} \cup \bar{\mathrm{A}}) = E(\mathrm{A} \cap \bar{\mathrm{A}})$.

The previous two examples illustrate that it is advantageous to use the geometrical model in certain situations.

3.6 Philosophical Implications

Through the history of science and engineering, numerous philosophical questions have been raised and varied logical conflicts have been encountered. Some of them remain as paradoxes. A famous paradox in classical set theory is that of the army barber who was ordered to shave everyone in the camp who does not shave himself. Then who shaves the barber? If he shaves himself, then he should not. If he does not, then he should! Another well-known paradox is that of a glass that can hold up to, say, 300 mL of water. If the glass has in it 150 mL, is it full or empty?

Electronics is defined as the study and design of control, communication, and computing devices that rely on the movement of electrons in circuits containing semiconductors, thermoionic valves, resistors, capacitors, and inductors [11]. Thus, the essence of electronics is the electron—so what is an electron? It is of paradoxical nature; some experimental results can be explained if the electron is assumed to be a particle, while other experimental results can only be explained if the electron is assumed to be a wave. So, what is an electron? An electron is an electron [12]. Engineers may accept that as long as the behavior of the electron can be predicted. The behavior may be predicted through the so-called Schrödinger's equation.

The tendency in engineering and technology is that if an argument or a proposal can lead to a device or a system that works, then the argument is vindicated. If the device or the system can be sold (and no one sues you), then we are talking about a celebrated idea—all paradoxes can wait to be solved later!

Fuzzy set theory is no exception; it has its own paradoxes. Some of the paradoxes can somehow be eased, while some may need more thinking. Scientists and engineers do not have to wait until everything is crystal-clear before looking into the practical applications of the theory.

A detailed discussion of the philosophical issues raised by the fuzzy set theory has been given by Hisdal [13–15]. The following is an outline of some of these issues.

A major objective of fuzzy set theory is to enable a machine to "think" like people, in other words, to enable a machine to use the natural language and reasoning of human beings. The heart of the theory is the membership function. Numerical membership values are, however, not a common practice in natural language communication. Thus, an exact correlation between the membership function and human thoughts is illusive. For example, a possibility of one does not mean certainty, as opposed to a probability of one. What physically is possibility supposed to mean? The example given by Zadeh and the elaboration of Giles (as discussed earlier) may ease this difficulty.

The max-, min-, and one-minus-formulas for the OR, AND, and negation operations, respectively, seem to have come from nowhere! The same thing applies to the shapes commonly used for membership functions (bell-shaped for external concepts such as "tall" and S-shaped for non-external concepts such as "medium", linear approximations of these shapes are also used.)

Although these formulas and functions have been used successfully in practical applications, they lead to some paradoxical situations; for example,

1. The union of a fuzzy set and its complement is, in general, not equal to the universe of discourse. (Ironically, this property was used as a measure of fuzziness as outlined in Section 3.5.) If we refrain from thinking in terms of classical sets, this may not really be a paradox. It seems, however, there is a duality in the way people think. Although a fuzzy set concept may be close to the way people think, the idea of producing everything (the universe of discourse) by combining what we select and what we leave out (objects and their complements) seems to be deeply engraved in the way people think!
2. The occurrence of a depression in the OR-curve, where none should have existed. Figure 3.8b shows an example of such a situation.
3. The lack of distinction between the outcome of an AND and an OR operation at the crossover point. Figure 3.8c shows another example.

If you are intrigued by these ideas, then you will enjoy reading the original work on the topic by Hisdal [13–15].

Chapter 3 Questions

3-1. Given the fuzzy set A = $\{(\alpha/0.4), (\alpha_2/0.3), (\alpha_3/0.7)\}$:

a) Write the set in an alternative format.

b) Explain the meaning of the set.

3-2. Would the membership values in a fuzzy set add up to unity always? Explain the significance of your answer.

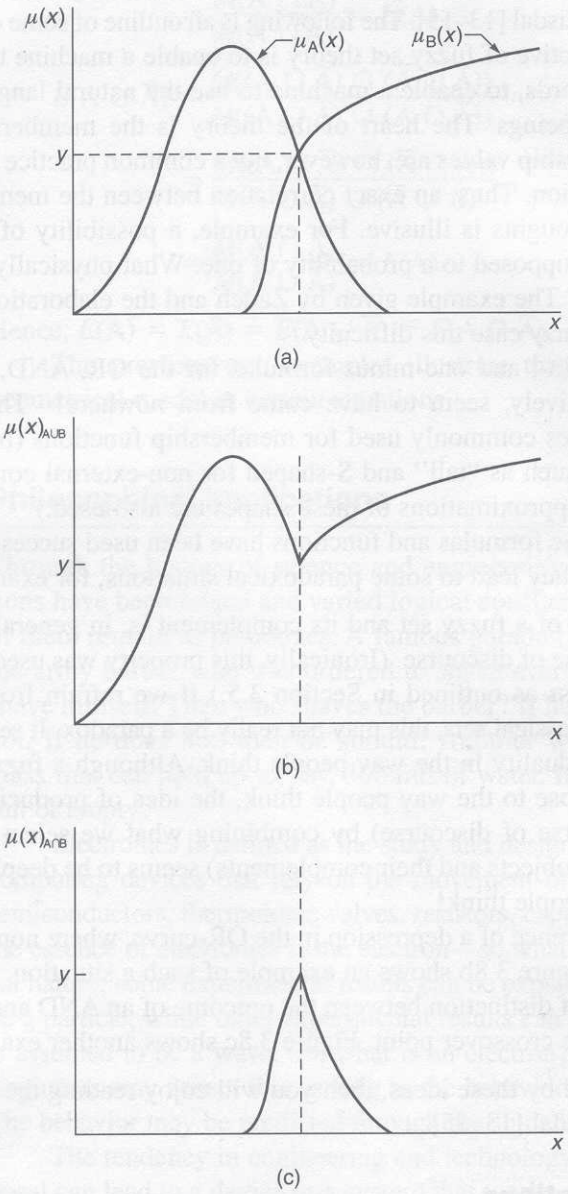


Figure 3.8 a) $\mu_A(x)$ and $\mu_B(x)$ are membership functions of concepts such as "medium" and "tall," respectively; b) The depression of value y should not be there (according to the way people think); c) $\mu_{A\cup B}(x) = \mu_{A\cap B}(x)$ at the crossover point.

3-3. If an item has a partial membership in several sets, would all membership values add up to unity always? Explain the significance of your results.

3-4. Suggest a fuzzy set to describe the statement "comfortable temperature."
a) from your point of view
b) from a polar bear's point of view
c) from a camel's point of view
Assume the universe of discourse, in absolute temperatures, to be {260, 270, 280, 300, 310, 320}.

3-5. Determine all possible α-level sets of
A = {(1, 0.3), (2, 0.4), (3, 0.5), (4, 0.6), (5, 0.3), (6, 0.2)}.

3-6. Determine all possible strong α-level sets of
A = {(1, 0.3), (2, 0.4), (3, 0.8), (4, 0.5), (6, 0.3)}.

3-7. Let A = {(0, 0.2), (1, 0.3), (2, 0.5), (3, 0.6), (4, 0.8)} and B = {(0, 0.4), (1, 0.5), (2, 0.7), (3, 0.8), (4, 0.9)}; determine
a) $\overline{A}$ **b)** $\overline{B}$ **c)** (B – A)

3-8. Given that A = {(α, 0.2), (β, 0.4), (γ, 0.7), (δ, 0.4)} and B = {(α, 0.4), (β, 0.6), (γ, 0.2), (δ, 0.8)}, determine
a) $A \cup B$ **b)** $A \cap B$ **c)** $\overline{A} \cap \overline{B}$ **d)** $\overline{A} \cup \overline{B}$

3-9. Given that A = {(α, 0.2), (β, 0.7), (δ, 0.4)} and B = {(α, 0.4), (β, 0.3), (γ, 0.3)}, determine
a) $A \cup B$ **b)** $A \cap B$

3-10. Given that A = 0.2/3 + 0.5/4 + 0.8/5 and B = 0.8/5 + 0.2/8, determine the Cartesian product of the two sets; A × B.

3-11. Given that A = 0.2/3 + 0.5/4 + 1/5 + 0.4/6 and B = 0.3/3 + 0.2/4 + 0.7/5 + 0.6/6, determine the algebraic product of the two sets.

3-12. Given that A = 0.2/α + 0.4/1β + 0.3/δ, and B = 0.4/α + 0.3/β + 0.2/δ, determine the algebraic product of the two sets.

3-13. Given that A = 0.2/3 + 0.5/4 + 0.3/5 + 1/6, determine
a) CON(A) **b)** DIL(A)

3-14. Given that A = 0.3/4 + 0.5/5 + 0.6/6 + 1/7 and B = 0.2/4 + 0.7/5 + 0.2/6 + 0.5/7, determine
a) CON(A) **b)** CON(B) **c)** CON($A \cap B$)

3-15. Given that A = 0.2/4 + 0.8/5 + 0.3/6 and B = 0.3/4 + 0.2/6, determine
a) the algebraic sum of the two sets
b) the bound sum of the two sets

3-16. Given that A = 0.8/4 + 0.7/5 + 0.3/6 and B = 0.5/4 + 0.1/5 + 0.8/6, determine the bounded sum of the two sets.

3-17. Given that A = 0.6/4 + 0.5/6 + 0.7/8 and B = 0.3/4 + 0.5/8, determine the bounded difference $A \ominus B$.

3-18. Let A = $\{\alpha_1, \alpha_2\}$, B = $\{\beta_1, \beta_2\}$, and C = $\{\gamma_1, \gamma_2, \gamma_3\}$. The relation R from A to B is given by $R = 0.1/(\alpha_1, \beta_1) + 0.2/(\alpha_1, \beta_2) + 0.4/(\alpha_2, \beta_2)$, and the relation S from B to C is given by $S = 0.2/(\beta_1, \gamma_1) + 0.4/(\alpha_1, \beta_2) + 0.2/(\beta_2, \gamma_1) + 0.8/(\beta_1, \gamma_2) + 0.1/(\beta_2, \gamma_3)$. Construct a fuzzy arrow diagram to show the relations S and R.

3-19. Let the universe of discourse be given by
U = {5, 15, 20, 30, 40, 60, 80, 90}.
a) Suggest a fuzzy set to describe the term "young."
b) Suggest a fuzzy set to describe the term "old."
c) Derive a fuzzy set to describe "not old."
d) Derive a fuzzy set to describe "very young."

3-20. Østergaard (in *Fuzzy Automata and Decision Process*, ed. Gupta et al. (North Holland, 1977) defined the linguistic variables "large positive," and "medium positive," "small positive," and "large negative" analytically by

$$1 - \exp\left[-\left(\frac{0.5}{|1-x|}\right)^{2.5}\right],\ 1 - \exp\left[-\left(\frac{0.25}{|0.7-x|}\right)^{2.5}\right],$$

$$1 - \exp\left[-\left(\frac{0.25}{|0.4-x|}\right)^{2.5}\right], \text{ and}$$

$$1 - \exp\left[-\left(\frac{0.5}{|-1-x|}\right)^{2.5}\right],$$ respectively. Sketch a graphical representation of these variables.

3-21. Develop graphically membership functions to describe the linguistic variables "cold," "warm," and "hot." The temperature range is 0°C to 100°C.

3-22. Use the graphs developed in the previous question (3–21) to define graphically the following:
a) NOT warm
b) warm OR cold
c) warm AND hot

3-23. Give possible reasons for triangular membership functions being frequently used, particularly when the height of intersection of each two successive fuzzy sets is equal to one-half.

***3-24.** Blin and Whinston in their paper "Fuzzy Sets and Social Choices," *Journal of Cybernetics*, (1974): 3, 4, 28–35, reported on an application of fuzzy sets and fuzzy relations.
a) What was the aim of the paper?
b) Explain briefly how the problem discussed illustrates the distinction between fuzziness and probability.

***3-25.** Compare and contrast the approach of Echauz and Vachtsevanos (*IEEE Trans. Education* 38 (1995): 158–165) and that of Biswas (*Fuzzy Sets and Systems* 74 (1995): 187–194) in evaluating and grading students' performance using the concepts of fuzzy sets.

3-26. Mark the following statements as true or false; correct the false statements meaningfully.

1) This question uses binary logic only.
2) A classical set will typically have a normal distribution graph describing its membership function.
3) The support of a fuzzy set is a crisp set.
4) The law of excluded-middle does not apply to fuzzy sets.
5) DeMorgan's theorems do not apply to fuzzy sets.
6) A membership function for a classical set cannot be defined.
7) The values of membership in a given set can never add up to one (or 100%).
8) The fuzzier a set is, the more similar it is to its complement.
9) Precision and accuracy are the same thing.
10) The CON operation always reduces the value of membership.
11) The CON operation makes a fuzzy set less fuzzy.
12) In the limit, the DIL operation leads to a singleton set.
13) There is more than one algorithm for fuzzification.
14) A fuzzy set ANDed with its complement will not necessarily result in the universe of discourse.
15) The order in which the fuzzy AND operation is done does not make any difference on the outcome.
16) The order in which the Cartesian product of two fuzzy sets is done does not make any difference on the outcome.
17) A fuzzy set is a collection of items that we know nothing about.
18) Membership functions are commonly normalized by setting the maximum value to one and scaling all other membership values proportionally.
19) Fuzzy sets describing a universe of discourse have to be symmetric.
20) Fuzzy sets describing a universe of discourse should not overlap.
21) An α-level represents a threshold restriction based on a membership value.
22) Continuous membership functions have to be of a triangular shape.

References

1. L. A. Zadeh, "Fuzzy Sets," *Information Control* 8 (1965): 338–353.
2. L. A. Zadeh, "Outline of a New Approach to the Analysis of Complex Systems and Decision Processes," *IEEE Trans. Sys. Man. Cybern.* SMC-3 (1973): 28–44.
3. A. Kandel and S. C. Lee, *Fuzzy Switching and Automata — Theory and Applications* (London: Edward Arnold, 1979).
4. R. K. Ragade and M. M. Gupta, "Fuzzy Set Theory: Introduction," in *Fuzzy Automata and Decision Process*, ed. M. M. Gupta, G. N. Saridis, and B. R. Gaines (North Holland, 1977).
5. T. Terano, K. Asai, M. Sugeno, *Fuzzy Systems Theory and Its Applications* (Academic Press, 1992).
6. L. A. Zadeh, "A Fuzzy-Set-Theoretic Interpretation of Linguistic Hedges," *J. Cybern.*, 2, 3, (1972): 4–34.
7. Y. Shirai, T. Yamakawa, and F. Ueno, "A CAD-oriented Synthesis of Fuzzy Logic Circuits," *Systems. Computers. Controls* 15 (1984): 76–83.
8. L. A. Zadeh, "Fuzzy Sets as Basis for a Theory of Possibility," *Fuzzy Sets and Systems* 1 (1978): 3–28.

9. R. Giles, "Semantics for Fuzzy Reasoning," *Int. J. Man-Machine Studies* 17 (1982): 401–415.
10. B. Kosko, "Fuzziness vs. Probability," chap. 7 in *Neural Networks and Fuzzy Systems* (New Jersey: Prentice Hall, 1992).
11. *A Concise Dictionary of Physics* (Oxford University Press, 1990).
12. L. Solymar and D. Walsh, *Lectures on the Electrical Properties of Materials* (Oxford University Press, 1982).
13. E. Hisdal, "Infinite-valued Logic Based on Two-valued Logic and Probability," part 1.1: Difficulties with Present-day-fuzzy-set Theory and Their Resolution in the TEE Model," *Int. J. Man-Machine Studies* 25 (1986): 89–111.
14. E. Hisdal, "Infinite-valued Logic Based on Two-valued Logic and Probability," part 1.2 "Different Sources of Fuzziness," *Int. J. Man-Machine Studies* 25 (1986): 113–138.
15. E. Hisdal, "The Philosophical Issues Raised by Fuzzy Set Theory," *Fuzzy Sets and Systems* 25 (1988): 349–356.

4 Fuzzy Logic and Algebra

4.1 Fuzzy Logic

Fuzzy logic is based on the concepts of fuzzy sets. The truth value of a formula can assume any value in the interval [0, 1], as opposed to Boolean logic, where two values (0 and 1) are allowed.

Let T(S) denote the truth value of a formula S. The evaluation procedure for a formula in fuzzy logic may be described as follows:

(1) $T(S) = T(A)$ if $S = A$
(2) $T(S) = 1 - T(R)$ if $S = \overline{R}$
(3) $T(S) = \min[T(S_1), T(S_2)]$ if $S = S_1 \cdot S_2$
(4) $T(S) = \max[T(S_1), T(S_2)]$ if $S = S_1 + S_2$

Note that two-valued logic is a special case of fuzzy logic; all the rules stated above are applicable in two-valued logic.

EXAMPLE: Let $S = (P + Q) \cdot \overline{R}$.

Assume $T(P) = 0.1, T(Q) = 0.7$, and $T(R) = 0.6$; then

$$T(S) = \min\{\max[T(P), T(Q)], [1 - T(R)]\}$$
$$= \min\{0.7, 0.4\} = 0.4$$

If we assume T(P) = 1, T(Q) = 0, and T(R) = 0, then

$$\begin{aligned} T(S) &= \min\{\max[T(P), T(Q)], [1 - T(R)]\} \\ &= \min\{1,1\} = 1 \end{aligned}$$

(In Boolean algebra, $S = (1 + 0) \cdot 1 = 1 \cdot 1 = 1$.)

In systems using two-valued logic, one stores a statement A if the truth value of A is 1 and a statement $\overline{A}$ if the truth value of A is 0. In fuzzy logic systems, one stores a statement A if the truth value of A is greater than or equal to that of $\overline{A}$, i.e., if $T(A) \geq 1 - T(A)$, which leads to $T(A) \geq 0.5$.

A formula S is said to be satisfied if $T(S) \geq 0.5$, and falsified if $T(S) \leq 0.5$. With this definition, one should realize that not satisfying is different from falsifying, and not falsifying is different from satisfying.

4.2 Fuzzy Algebra

The concepts of crisp sets (where the membership function is either 1 or 0) were used in conjunction with Boolean algebra. Similarly, the concepts of fuzzy sets are used in conjunction fuzzy algebra; the term "fuzzy variable" replaced the term "membership grade" of a fuzzy variable in a set.

There is a close similarity between Boolean algebra and fuzzy algebra; however, it should be noted that in Boolean algebra the conditions: $x\overline{x} = 0$ and $x + \overline{x} = 1$ always exist, which is not the case in fuzzy algebra. Accordingly, every Boolean algebra is a fuzzy algebra but not vice versa.

4.3 Truth Tables

Fuzzy logic is multivalued, i.e., a variable may assume one of many possible values (more than two in general). Let us consider the simplest fuzzy logic scheme to see how it may work. Assume a three-valued logic: true (T), false (F), and unknown (T + F). One then may define the following truth tables for AND, OR, and complement.

$\wedge$	T	F	T + F
T	T	F	T + F
F	F	F	F
T + F	T + F	F	T + F

$\vee$	T	F	T + F
T	T	T	T
F	T	F	T + F
T + F	T	T + F	T

	$\neg$
T	F
F	T
T + F	T + F

4.4 Fuzzy Functions

A fuzzy logic function $f(x, y)$ may assume infinitely many values in the range assigned (closed range [0, 1] for normalized functions). The range may be subdivided into m classes, leading to an m-valued logic problem. The objective of the analysis of a given fuzzy logic function is to determine the conditions that must be satisfied by fuzzy variables in order for the function, f, to belong to a certain class m where

Class 1: $\alpha_1 \leq f \leq 1$

Class 2: $\alpha_2 < f < \alpha_1$

.

.

.

Class m: $0 \leq f \leq \alpha_{m-1}$

with $1 > \alpha_1 > \alpha_2 > ... > \alpha_{m-1} > 0$.

EXAMPLE: If $f(x, y) = x + \bar{y}$, determine the conditions that must be satisfied by the fuzzy variables x and y so that $f(x, y)$ belongs to Class 2, where classes are defined as follows:

Class 1: $0.6 \leq f(x, y) \leq 1$
Class 2: $0.3 < f(x, y) < 0.6$
Class 3: $0 \leq f(x, y) \leq 0.3$

For $f(x, y)$ to belong to Class 2, the following condition applies

$0.3 < f(x, y) < 0.6$

This can be divided into two groups:

Group 1: $x + \overline{y} > 0.3$, and
Group 2: $x + \overline{y} < 0.6$.
Note that $x + \overline{y} > 0.3$ gives

$$x > 0.3 \text{ or } \overline{y} > 0.3$$
$$y < 0.7$$

S $x > \overline{y}$
$\mu(x + \overline{y}) = \text{m}~~\text{ax}~~ \; x > 0.3$
$\overline{y} > x$
$\mu(x + \overline{y}$ ~~=~~ $\mu(\overline{y}) > 0.3$
$\mu(y) < 0.7$

and

$x + \overline{y} < 0.6$ gives
$\overline{x}y > 0.4$ (using DeMorgan's Law).

This leads to the condition $y > 0.4$ and $\overline{x} > 0.4$, or in other words, $y > 0.4$ and $x < 0.6$. Thus, for $f(x, y)$ belong to the class $0.3 < f(x, y) < 0.6$, x and y must satisfy the following conditions:

Group 1: $[x > 0.3 \text{ or } y < 0.7]$, and
Group 2: $[x < 0.6 \text{ and } y > 0.4]$.
In general, if $f(x, y) = x + \overline{y}$ belongs to class m, i.e.

$$a_m \le f(x, y) < a_{m-1}$$

then the conditions would have been

Group 1: $[x \ge a_m \text{ or } y \le 1 - a_m]$, and
Group 2: $[x < a_{m-1} \text{ and } y > 1 - a_{m-1}]$.

EXAMPLE: If $f(x, y, z) = xz + y\overline{z}$, determine the conditions that must be satisfied by the fuzzy variables x, y, and z so that $f(x, y, z)$ belongs to class m, in other words,

$$a_m \le f(x, y, z) < a_{m-1}.$$

$xz + y\,\overline{z} \ge a_m$ gives
Group 1: $x \ge a_m$ and $z \ge a_m$ or $y \ge a_m$ and $z \le 1 - a_m$,
and $xz + y\,\overline{z} < a_{m-1}$ gives
Groups 2: $x < a_{m-1}$ and $z < a_{m-1}$ or $y < a_{m-1}$ and $z > 1 - a_{m-1}$.

EXAMPLE: Determine the conditions that must be satisfied by the fuzzy variables x, y, and z so that

$$f(x, y, z) = (x + z)(\overline{y} + z) \text{ satisfies } a_m \le f(x, y, z) < a_{m-1}.$$

The condition $a_m \le (x + y)(y + z)$ leads to

$$\text{Group 1} = \left\{ \begin{bmatrix} x \ge a_m \\ \text{or } y \ge a_m \end{bmatrix} \text{ and } \begin{bmatrix} z > a_m \\ \text{or } y < 1 - a_m \end{bmatrix} \right\}.$$

The condition $(x + y)(\overline{y} + z) < a_{m-1}$ leads to

$$\text{Group } 2 = \left\{ \begin{bmatrix} x < a_{m-1} \\ \text{or } y < a_{m-1} \end{bmatrix} \text{ and } \begin{bmatrix} z < a_{m-1} \\ \text{or } y > 1 - a_{m-1} \end{bmatrix} \right\}.$$

4.5 Concepts of Fuzzy Logic Circuits

Any Boolean function may be implemented using three basic circuits: AND, OR, and INVERTER. Likewise, any fuzzy function may be implemented using fuzzy AND, OR, and INVERTER. One more circuit may be needed: a discriminator to separate the output into classes. In the following we discuss the concept of basic circuits of fuzzy AND, OR, and INVERTER.

4.5.1 Fuzzy Inverter

Fuzzy complement is defined by $\mu_{\overline{A}}(x) = 1 - \mu_A(x)$. Hence, a circuit that has an output voltage V_{out} given by $V_{out} = V_{Ref.} - V_{in}$ may perform as an inverter, with V_{ref} being a reference voltage and V_{in} the input voltage. Such a circuit will have a normalized input-output relation given by

$V_{out} = 1 - V_{in}$; V_{out} represents $\mu_{\overline{A}}(x)$ and V_{in} represents $\mu_A(x)$.

A circuit that can do this is shown in Figure 4.1.

The circuit is composed of an op-amp with negative feedback resistors. Assuming an ideal op-amp, the output is

$$V_{out} = \left(\frac{R_1 + R_2}{R_1}\right)\left(\frac{R_2}{R_1 + R_2}\right) V_{ref.} - V_{in}\left(\frac{R_2}{R_1}\right)$$

$$= \left(\frac{R_2}{R_1}\right)(V_{ref.} - V_{in}).$$

If V_{ref} is selected to be logic 1, then

$$V_{out} = \kappa(1 - V_{in})$$

where κ is a constant.

4.5.2 Fuzzy OR

Fuzzy ORing is defined by $\max[\mu_A(x), \mu_B(x)]$. Hence, a circuit that has an output voltage V_{out} that is equal to the value of the largest of the input voltages will perform as an OR circuit:

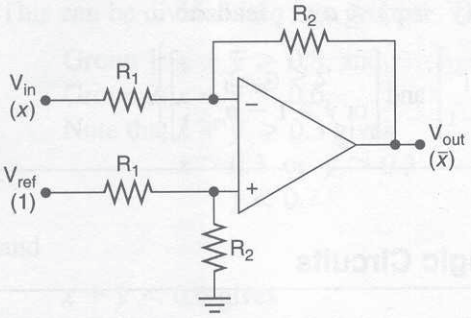


Figure 4.1 Fuzzy Logic Inverter.

$$V_{\text{out}} = V_A \quad \text{if } V_A > V_B$$
$$\text{or} \quad V_{\text{out}} = V_B \quad \text{if } V_B > V_A.$$

In binary circuits V_A, V_B, and V_{out} may assume one of two values, say $0V$ and $+5V$, while in fuzzy circuits V_A, V_B, and V_{out} may assume any value in the range $0V$ to $+5V$. Diodes were used in binary logic without considering the voltage needed to turn a diode on ($V_D \cong 0.7V$). A "diode" with zero voltage (or almost zero) is needed for fuzzy logic circuits.

Such a "diode" can be produced using the circuit known as precision rectifier. The circuit uses an operational amplifier to make it sensitive to voltages as low as V_D/A_{OL}, where V_{OL} = open loop gain of the operational amplifier, which is in the range of 10^5. Such a circuit is shown in Figure 4.2.

Such a circuit may be used to produce an OR circuit as shown in Figure 4.3.

4.5.3 Fuzzy AND

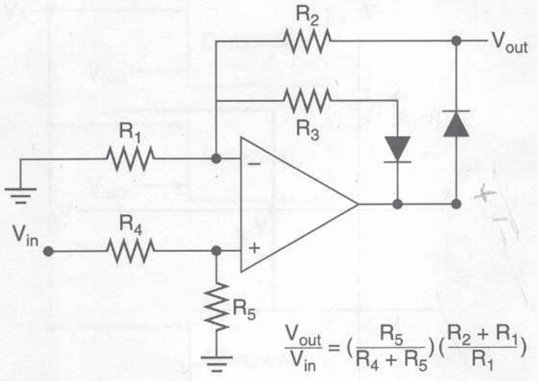
From DeMorgan's laws we have

$$A \cdot B = \overline{\overline{A} + \overline{B}},$$

and hence an AND can be realized using an inverting precision rectifier (to produce $\overline{A} + \overline{B}$), followed by an inverter as shown in Figure 4.4.

4.5.4 Discriminator

An *n*-class discriminator is composed of *n* comparators. A comparator can be realized using operational amplifiers to compare two voltages (see Figure 4.5). The



$$\frac{V_{out}}{V_{in}} = \left(\frac{R_5}{R_4 + R_5}\right)\left(\frac{R_2 + R_1}{R_1}\right)$$

Figure 4.2 Precision rectifier circuit (non-inverting).

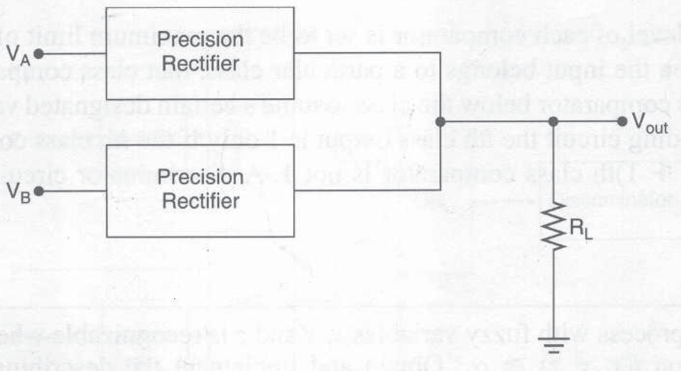


Figure 4.3 Fuzzy OR circuit

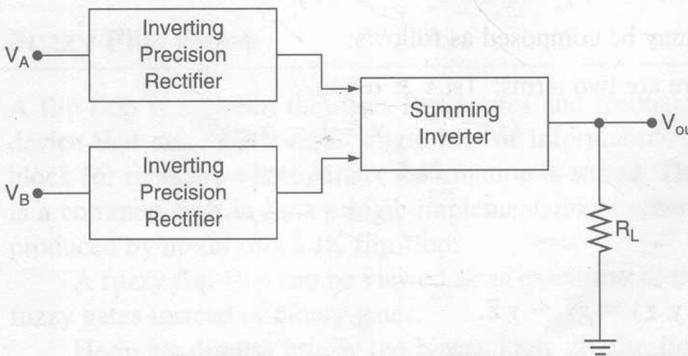


Figure 4.4 Concept of fuzzy AND circuit.

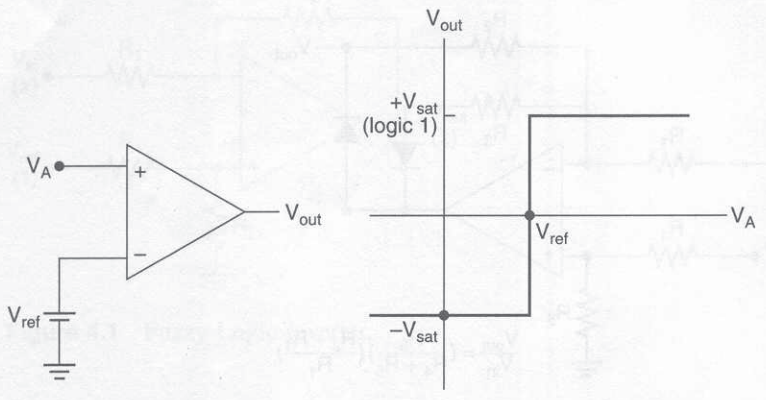


Figure 4.5 Concept of comparator circuit.

reference level of each comparator is set to be the maximum limit of the previous class. When the input belongs to a particular class, that class comparator and the other class comparator below the class assume a certain designated value (logic 1). With decoding circuit the *i*th class output is 1 only if the *i*th class comparator is 1 and the $(i + 1)$th class comparator is not 1. A discriminator circuit is shown in Figure 4.6.

EXAMPLE: A certain process with fuzzy variables *x*, *y* and *z* is recognizable when its describing function $f(x, y, z) \geq \alpha_1$. Obtain and implement the describing function, if $f(x, y, z) \geq \alpha_1$ when

$$\left\{\begin{matrix} x \geq \alpha_1 \\ \text{and } y \leq \alpha_1 \end{matrix}\right\} \text{ or } \left\{\begin{matrix} y \geq \alpha_1 \\ \text{and } z \leq 1 - \alpha_1 \end{matrix}\right\}.$$

$f(x, y, z)$ may be composed as follows:

$$\begin{aligned} \text{There are two terms: 1st } & x \geq \alpha_1 \rightarrow x \\ & y \leq 1 - \alpha_1 \rightarrow \overline{y} \\ \text{2nd } & y \geq \alpha_1 \rightarrow y \\ & z \leq 1 - \alpha_1 \rightarrow \overline{z} \end{aligned}$$

Then

$$f(x, y, z) = x\overline{y} + y\,\overline{z}.$$

This function can be implemented as shown in Figure 4.7.

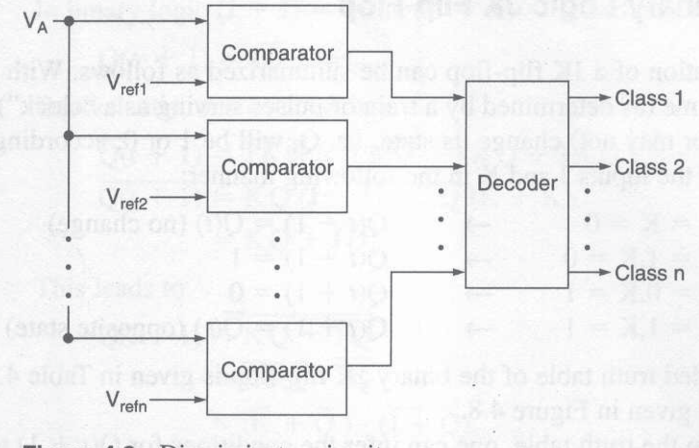


Figure 4.6 Discriminator circuit.

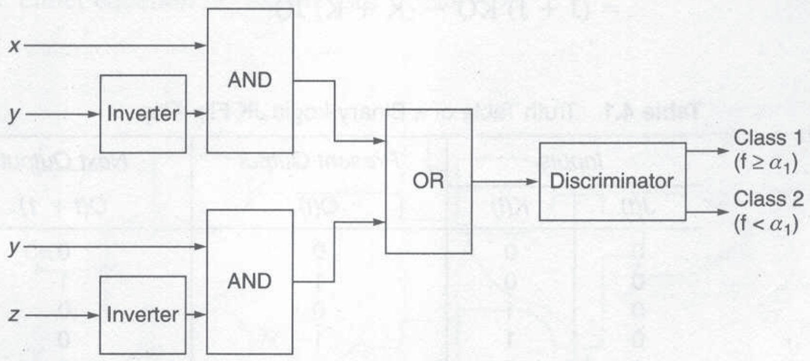


Figure 4.7 Implementation of $f(x, y, z) \geq \alpha_1$.

4.6 Fuzzy Flip-Flops

A flip-flop is a circuit that uses logic gates and feedback to produce a bistable device that can "memorize" single bits of information. It is the basic building block for registers where binary information is stored. The so-called JK flip-flop is a common type in binary logic implementations; other flip-flop types may be produced by modifying a JK flip-flop.

A fuzzy flip-flop can be viewed as an extension of the binary flip-flop using fuzzy gates instead of binary gates.

Here, we discuss briefly the binary logic JK flip-flop, then extend the concept to fuzzy flip-flop.

4.6.1 Binary Logic JK Flip-Flop

The operation of a JK flip-flop can be summarized as follows. With every increment of time (as determined by a train of pulses serving as a "clock"), the output, Q, may (or may not) change its state, i.e. Q, will be 1 or 0, according to the conditions of the inputs J and K in the following manner:

1. $J = K = 0 \quad \rightarrow \quad Q(t+1) = Q(t)$ (no change)
2. $J = 1, K = 0 \quad \rightarrow \quad Q(t+1) = 1$
3. $J = 0, K = 1 \quad \rightarrow \quad Q(t+1) = 0$
4. $J = 1, K = 1 \quad \rightarrow \quad Q(t+1) = \overline{Q}(t)$ (opposite state)

An extended truth table of the binary JK flip-flop is given in Table 4.1. The logic symbol is given in Figure 4.8.

From the truth table, one can infer the conditions for $Q(t+1)$ to be 1 as

$$\begin{aligned} Q(t+1) &= \overline{J}\,\overline{K}Q + J\overline{K}\,\overline{Q} + J\overline{K}Q + JK\overline{Q} \\ &= (J + \overline{J})\,\overline{K}Q + (K + \overline{K})\,J\overline{Q}. \end{aligned} \qquad (1)'$$

Table 4.1 Truth Table of a Binary-Logic JK Flip-Flop

Inputs		*Present Output*	*Next Output*
J(t)	*K(t)*	*Q(t)*	*Q(t + 1)*
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

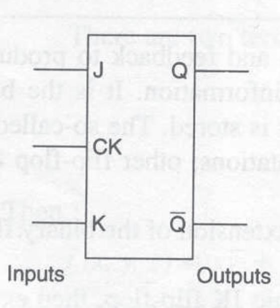


Figure 4.8 Logic symbol of JK flip-flop.

In binary logic $(J + \bar{J}) = 1$ and $(K + \bar{K}) = 1$, and hence

$$Q(t + 1) = J\bar{Q} + \bar{K}Q. \qquad (1)$$

Equivalently, one could write

$$\overline{Q(t+1)} = \bar{J}\,\bar{K}\,\bar{Q} + \bar{J}\,K\,\bar{Q} + \bar{J}\,K\,Q + J\,K\,Q \qquad (2)'$$

$$\begin{aligned}\overline{Q(t+1)} &= K\,Q\,(J + \bar{J}) + \bar{J}\,\bar{Q}\,(K + \bar{K})\\ &= K\,Q + \bar{J}\,\bar{Q}\end{aligned}$$

This leads to

$$\begin{aligned}Q(t+1) &= \overline{K\,Q + \bar{J}\,\bar{Q}}\\ &= \overline{(K\,Q)} \cdot \overline{\bar{J}\,\bar{Q}}\\ &= (\bar{K} + \bar{Q}) \cdot (J + Q) \qquad (2)\end{aligned}$$

In binary logic equations (1) and (2) are equal, and hence implementation using either equation is possible (see Figure 4.9).

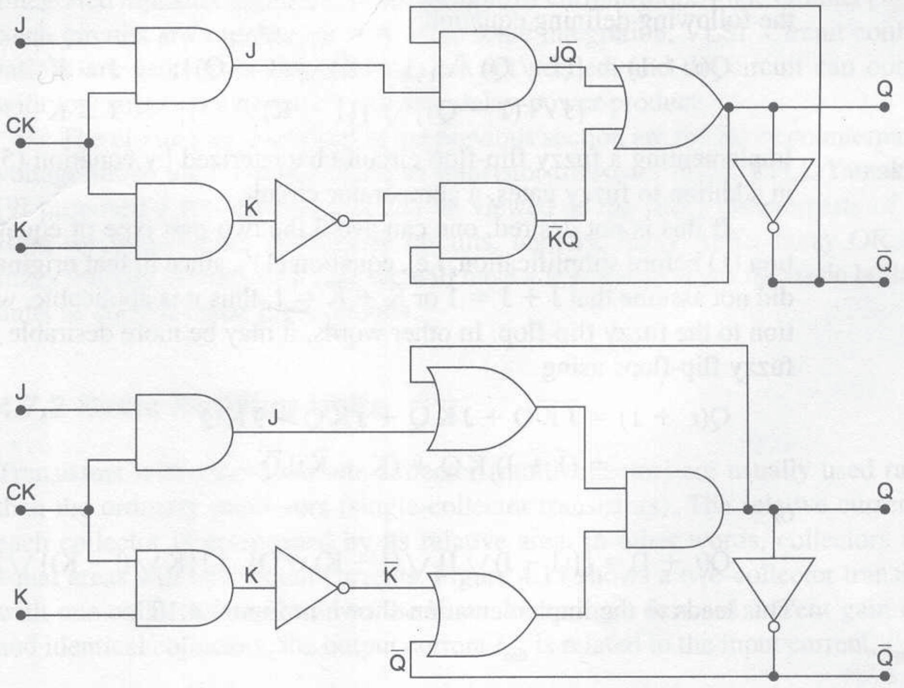


Figure 4.9 Two equipment implementations of binary JK flip-flop.

4.6.2 Fuzzy Logic JK Flip-Flop

Equations similar to (1) and (2) may be obtained for fuzzy variables using fuzzy OR, AND, and complement;

$$Q_R\,(t+1) = \{J \wedge (1-Q)\} \vee \{(1-K) \wedge Q\} \tag{3}$$
$$Q_S\,(t+1) = \{(J \vee Q)\} \wedge \{(1-K) \vee (1-Q)\} \tag{4}$$

These two equations, contrary to binary logic, are not equivalent in general.

Let us examine equations (3) and (4) more closely under various conditions of J, K, and Q.

1. If $J = K < Q$;
   $Q_R\,(t+1) = \{J \wedge (1-Q)\} \vee \{(1-J) \wedge Q\} = (1-J) \wedge Q$
   $Q_S\,(t+1) = (J \vee Q) \wedge \{(1-K) \vee (1-Q)\} = Q \wedge (1-J)$
2. If $J = K = Q$;
   $Q_R(t+1) = Q_S\,(t+1) = J \wedge (1-J)$
3. If $J = K > Q$;
   $Q_R(t+1) = Q_S\,(t+1) = J \wedge (1-Q)$

Therefore, to extend the binary JK flip-flop smoothly to fuzzy flip-flop, we write the following defining equation:

$$Q(t+1) = \begin{cases} \{(J \vee Q) \wedge \{(1-K) \vee (1-Q)\}; & J \leq K \\ \{J \wedge (1-Q)\} \vee \{(1-K) \wedge Q\}; & J \leq K \end{cases} \tag{5}$$

Implementing a fuzzy flip-flop circuit characterized by equation (5) will require, in addition to fuzzy gates, a comparator circuit.

If this is not desired, one can avoid the two-part type of equations by equation (1) before simplification, i.e., equation (1)′, since in that original equation we did not assume that $J + \overline{J} = 1$ or $K + \overline{K} = 1$, thus it is applicable, without restriction to the fuzzy flip-flop. In other words, it may be more desirable implementing fuzzy flip-flops using

$$Q(t+1) = \overline{J}\,\overline{K}Q + J\overline{K}\,\overline{Q} + J\overline{K}Q + JK\overline{Q}$$
$$= (\overline{J} + J)\,\overline{K}Q + (K + \overline{K})J\,\overline{Q}$$

or

$$Q(t+1) = \{\{(1-J) \vee J\} \vee (1-K) \vee Q\} \vee \{\{K \vee (1-K)\} \vee J \vee (1-Q)\}.$$

This leads to the implementation shown in Figure 4.10.

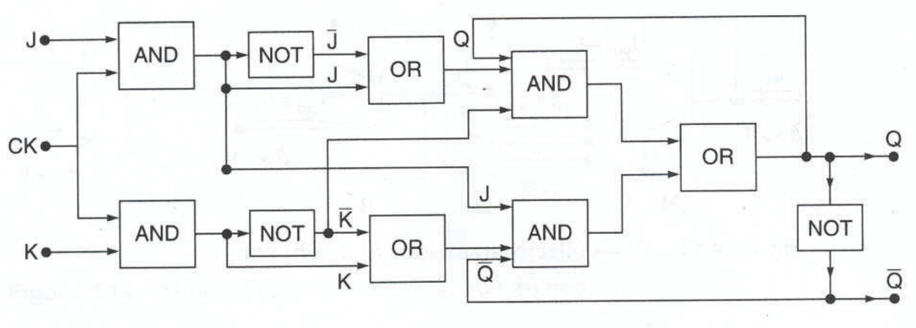


Figure 4.10 Implementation of a fuzzy flip-flop.

4.7 Fuzzy Logic Circuits in Current-Mode

4.7.1 Introduction

In current-mode logic circuits the amplitude and direction of signal current, rather than amplitude and polarity of signal voltage, are used to represent information. Integrated injection logic, I^2L, is an example of current-mode logic circuits [7–10]. Such circuits are suitable for very large scale integration, VLSI. Circuit configurations are usually simple, resistors are not needed, and the circuit can operate with low supply voltage and has a low delay-power product.

The circuits we discussed in the previous section are the fuzzy counterpart of voltage-mode logic circuits (such as transistor-transistor logic, TTL). Yamakawa [9] proposed a few circuits that can be viewed as the fuzzy counterparts of I^2L. Here we will discuss three basic circuits, namely, fuzzy AND, fuzzy OR, and fuzzy inverter circuits. But first we will have a few words about the basic building units of current-mode logic circuits.

4.7.2 Basic Building Units

Transistors with more than one collector (multicollector) are usually used rather than the ordinary transistors (single-collector transistors). The relative current in each collector is determined by its relative area. In other words, collectors with equal areas will have equal currents. Figure 4.11 shows a two-collector transistor with one collector connected to the base. Assuming a forward current gain of β and identical collectors, the output current I_{out} is related to the input current, I_{in}, by

$$I_{\text{out}} = \left(\frac{\beta}{1+\beta}\right) I_{\text{in}}.$$

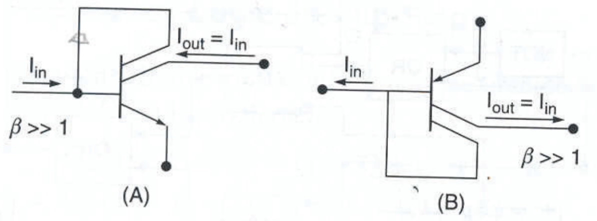


Figure 4.11 Two-collector transistors a) npn b) pnp.

If $\beta >> 1$, we get a current mirror with

$$I_{\text{out}} = I_{\text{in}}.$$

Such a circuit is used when changing direction of the signal current.

Figure 4.12 shows a multicollector circuit with identical collectors. The circuit generates a large number of outputs with the same signal current. Such a circuit is used for single-to-multifan-out conversion, i.e., to enable a logic gate to drive more than one load, as shown in Figure 4.13.

A multicollector transistor may be designed to have unequal collector areas. If the area of the n^{th} collector is k_n times the area of the collector connected to the base, then

$$I_{\text{out}_n} = k_n\, I_{\text{in.}}$$

Such a circuit (with $k_n > 1$) may be used as a current amplifier at the output stage of a current-mode circuit.

Figure 4.14 shows a two-input circuit; one input is applied to the current mirror I_A, the other, I_B, is subtracted from I_A by wiring (wired-subtraction). The difference is applied to a diode. The output current I_Z is then given by

$$I_Z = \begin{cases} I_A - I_B & \text{when } I_A \geq I_B \\ 0 & \text{when } I_A < I_B \end{cases}$$

In other words, the circuit of Figure 4.14 is an implementation of bounded difference operation (see Section 3.3.5). This circuit is the basic building unit of current-mode fuzzy logic circuits.

4.7.3 Basic Fuzzy Logic Circuits

In fuzzy logic the information may assume any value in the interval [0, 1]. This is represented by current in the interval [0 A, 10μA] in current-mode logic circuits.

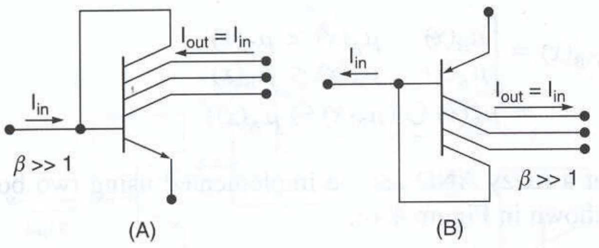


Figure 4.12 Multicollector transistors a) npn, b) pnp.

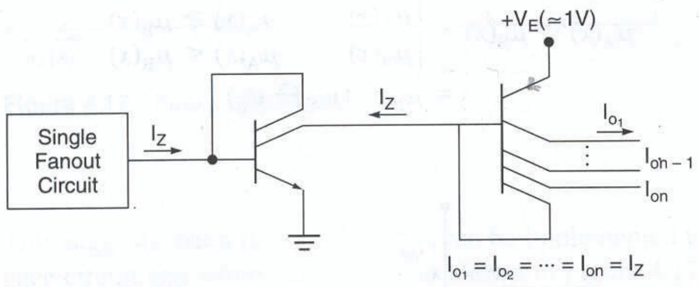


Figure 4.13 Single-to-multifan-out conversion.

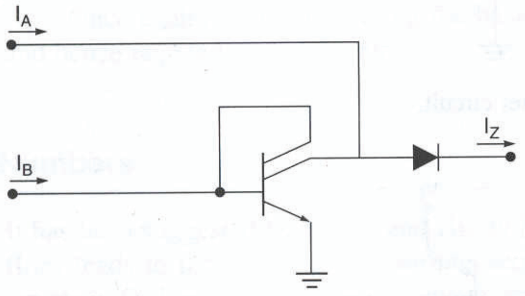


Figure 4.14 Bounded difference circuit.

Following we present three basic circuits, the inverter, AND, and OR circuits.

1. Fuzzy Inverter
   Fuzzy complement is defined by $\mu\overline{A}\ (x) = 1 - \mu_{\overline{A}}(x)$. This operation is implemented by the circuit shown in Figure 4.15. I_{ref} is set to logic 1, which is typically 10 μA.
2. Fuzzy AND
   Fuzzy ANDing is defined by min $[\mu_A(x), \mu_B(x)]$. This can be written as

$$\mu_{A\wedge B}(x) = \begin{cases} \mu_B(x) & \mu_A(x) \geq \mu_B(x) \\ \mu_A(x) & \mu_A(x) < \mu_B(x) \end{cases}$$
$$= \mu_B(x) \ominus (\mu_B(x) \ominus \mu_A(x))$$

This suggests that a fuzzy AND can be implemented using two bounded-difference circuits, as shown in Figure 4.16.

3. Fuzzy OR
   Fuzzy ORing is defined by $\max[\mu_A(x), \mu_B(x)]$. This can be written as

$$\mu_A(x) \vee \mu_B(x) = \begin{cases} \mu_A(x) & \mu_A(x) \geq \mu_B(x) \\ \mu_B(x) & \mu_A(x) < \mu_B(x) \end{cases}$$
$$= \mu_B + (\mu_A \ominus \mu_B)$$

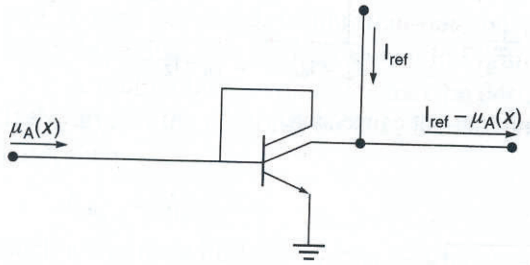


Figure 4.15 Fuzzy inverter circuit.

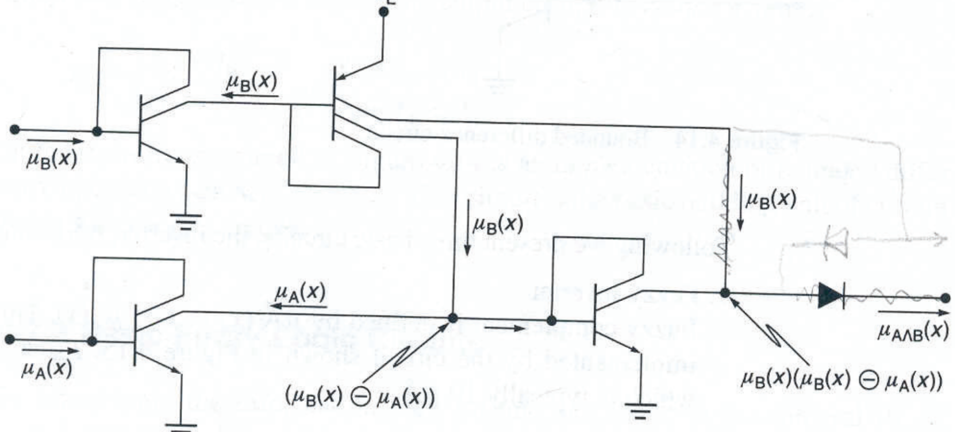


Figure 4.16 Fuzzy AND circuit.

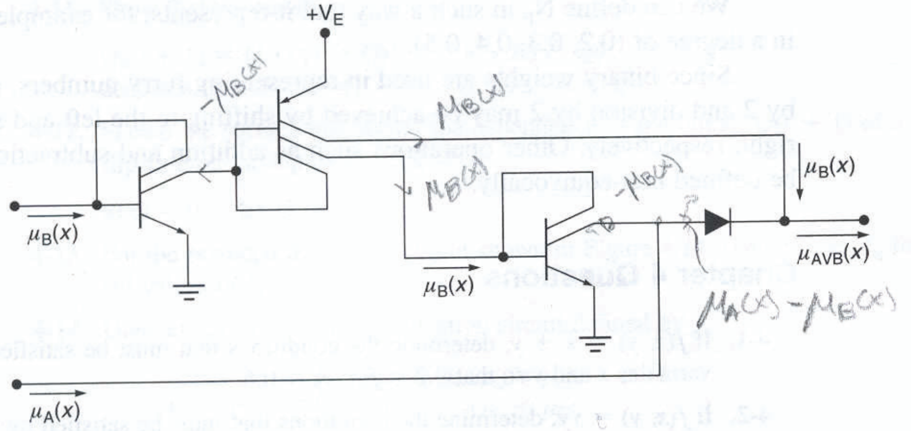


Figure 4.17 Fuzzy OR circuit.

This suggests that a fuzzy OR circuit can be implemented using a bounded-difference circuit and wired summation, as shown in Figure 4.17.

Each of the circuits discussed can drive only one gate. A fan-out converter (see Figure 4.13) may be used if the circuit is to drive more than one gate. Of course, other implementations of the logic functions are possible.

Once again, by implementing the basic gates one can implement flip-flops and hence registers, counters, etc.

4.8 Furry Numbers

It has been suggested by Kóczy and Hirota [5] that the introduction of fuzzy flip-flops leads to the generation of various sequential circuits such as registers and counters. Ordinary registers and counters process binary digits (bits); fuzzy registers and counters process fuzzy binary digits (fits). Binary digits lead to a binary number that is an array of bits. The binary weights are (from right to left) 2^0, 2^1, 2^2, 2^3, etc. Since a bit may assume either 1 or 0 only, a binary number may appear as 1001_2 (the subscript 2 denotes the number as binary). Similarly, fuzzy digits lead to fuzzy binary numbers (furry numbers). In other words, a furry number is an array of fits with binary weights. Since a fit can assume values in the interval [0, 1], a furry number may appear as $(0.4\ 1.0\ 0.0\ 0.3)_F$. Each fit is a fuzzy set of bits 0 and 1, i.e., the universe of discourse is $X = \{0, 1\}$, and the membership function expresses the degree to which a given fit is equal to 1. For example, the first fit, 0.3, indicates that the first fit has a degree of 0.3. Remember, in binary numbers, the degrees are either 1 or 0, and nothing in between.

We can define N_F in such a way that it represents, for example, $9_{10} = 1001_2$ in a degree of (0.2, 0.3, 0.4, 0.5).

Since binary weights are used in representing furry numbers, multiplication by 2 and division by 2 may be achieved by shifting to the left and shifting to the right, respectively. Other operations such as addition and subtraction have yet to be defined non-equivocally.

Chapter 4 Questions

4-1. If $f(x, y) = \overline{x} + y$, determine the conditions that must be satisfied by the fuzzy variables x and y so that $0.2 < f(x, y) < 0.6$.

4-2. If $f(x, y) = x\overline{y}$, determine the conditions that must be satisfied by the fuzzy variables x and y so that $0.3 < f(x, y) < 0.6$.

4-3. If $f(x, y, z) = xz + y\overline{z}$, determine the conditions that must be satisfied by the fuzzy variables x, y, and z so that $0.2 < f(x, y, z) < 0.7$.

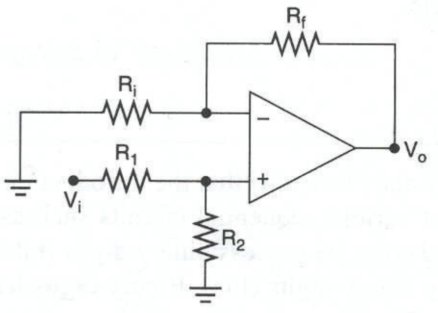
4-5. What are the characteristics of an ideal op-amp?

4-6. Give the circuit diagram and derive an expression for the output voltage for the following circuits built using op-amps.

a) inverting amplifier

b) non-inverting amplifier

4-7. Derive an expression for the output voltage for the circuit shown.



4-8. What is the operational difference between a rectifier using diodes and a precision rectifier using diodes and op-amps.

4-9. Explain the operation of the circuit shown in Figure 4.2.

4-10. Show that a binary circuit, e.g., an AND gate, can be realized using diodes and resistors. Explain why precision rectifiers, as opposed to diodes, are needed for fuzzy circuits.

4-11. Show that the equation

$Q_R(t + 1) = \{J \wedge (1 - Q)\} \vee \{(1 - K) \wedge Q\}$
becomes $Q_R(t + 1) = (1 - J) \wedge Q$ if $J = K < Q$.

4-12. Sketch the surface that shows the dependence of the output $Q(t + 1)$ of a fuzzy flip-flop on the inputs J and K when:

a) $Q = 0$ **b)** $Q = 1.0$

4-13. For the bounded difference circuit shown in Figure 4.14, sketch I_Z vs. I_B for varied values of I_A.

4-14. Give a realization of the implication circuit defined by

$$\mu_z = \mu_{x\to y} = \begin{cases} 1 - \mu_x + \mu_y \ (\mu_x \geq \mu_y) \\ 1 \quad (\mu_x < \mu_y) \end{cases}$$

4-15. Give a realization of the bounded-sum circuit defined by

$$\mu_z = \mu_{x\oplus y} = \begin{cases} 1 \quad (\mu_x + \mu_y \geq 1) \\ \mu_x + \mu_y \ (\mu_x + \mu_y < 1) \end{cases}$$

***4-16.** Obtain, e.g., using the Internet, a list of available fuzzy IC's and their applications.

References

1. P.N. Marinos, "Fuzzy Logic and Its Applications to Switching Systems," *IEEE Trans. Computers* C-18 (1969): 343-348.
2. A. Kaufmann, *Introduction to the Theory of Fuzzy Subsets* (New York: Academic Press, 1985).
3. A. Kandel and S.C. Lee; *Fuzzy Switching and Automata—Theory and Applications* (London: Edward Arnold, 1979).
4. K. Hirota, "The Concept of Fuzzy Flip-flop," *IEEE Trans. Sys., Man. and Cyber.* 19 (1989): 980–997.
5. L.T. Kóczy and K. Hirota, "Digital Circuits Based on Algebraic Fuzzy Operations," Progress in Fuzzy Sets and Systems, ed. W.H. Janko et al. (Kluwer Academic Publishers, the Netherlands, 1990): 100–114.
6. M. Togai and H. Watanabe, "Expert Systems on a Chip: An Engine for Real-time Approximate Reasoning," *IEEE Expert* (Fall 1986): 55–62.
7. K. Hart and A. Slob, "Integrated Injection Logic: A New Approach to LSI," *IEEE J. Solid State Circuits* SC-7 (5) (1972): 346.
8. H.H. Floke, "I^2L Design in Standard Bipolar Process," *IEEE J. Solid State Circuits* SC-13(6) (1978): 914.
9. Y. Yamakawa, "Fuzzy Logic Circuits in Current Mode," Chap. 17 in vol. I of *Analysis of Fuzzy Information*, ed. J. C. Bezdek (CRC Press, 1987): 242-262.
10. T. Yamakawa, Y. Shirai, T. Inoue, and F. Ueno, "Realization of Fuzzy Logic Operations by Current-Mode Circuits", Trans. (c), I.E.C.E., Japan, J63-6, 10, 772–773, 1980.

5 Fuzzy Control

5.1 Introduction

An operator can control a process adequately without the need of any mathematical model for the system. For example, you control the process of car driving without the knowledge of a mathematical model of driving. It may even appear ridiculous to raise the question whether such a model exists or not. You want to reach your destination safely and you know how to do that. "How to do that" is based on the ability to interpret linguistic statements about the system and to reason qualitatively. For example, you see that you are too close to the car ahead of you, then you slow down a bit. "Too close" and "slow a bit" can be defined in terms of fuzzy sets. Of course your final action, i.e., the pressure you apply on the brakes, is very specific and not fuzzy. The value you select depends on your criteria of driving. The situation starts to become entangled if there are more conditions to consider before you act. For example, if you are getting close, you slow a bit, if it is raining, slow down a lot. The final outcome, again the pressure on the brakes, comes as a result of selecting a single value from the fuzzy set that results from considering the two conditions. Obviously, two steps were carried out, one to infer the possible actions and the other to select one particular action. Sometimes, during the course of driving you respond with clear definite action to a clear definite situation. For example, when you see a red traffic light, you stop.

Red and stop can still be described in terms of fuzzy sets—remember the good old singleton? So you have been driving fuzzy (not fuzzily) all these years.

Fuzzy control was the first application of fuzzy set theory that drew attention to the practical potential of the theory. Fuzzy controllers have been reported to perform better than conventional PID or DDC controllers (see section 5.2 for definitions), especially when noise is taken into account [1,2].

5.2 Conventional Control Systems [3–6]

A typical automatic control system is shown in Figure 5.1. The output of the process under control is measured and converted to an electrical signal that represents the output, $c(t)$. This signal is compared to a reference or an input signal, $r(t)$. As a result, an error signal, $e(t)$, is generated. The error signal actuates the controller to generate a control action signal, $m(t)$.

To design such a system one needs a mathematical model for the process and the specifications or the model of the overall system. A mathematical model for the controller can then be produced and implemented.

In a manually controlled system, similar actions to those depicted in Figure 5.1 take place qualitatively. A driver (same silly example again) "measures" the distance to the car ahead (the eyes act as a transducer in this case), and compares the distance to a preset distance (stored in the brain). The difference actuates the control action, speeding up or slowing down (although the control is usually carried out by the foot pressing the gas or brake pedals, it is still called manual control!). The wider the experience of the driver, the smoother the driving in general. Intelligent drivers can infer actions, based on their driving experience, to handle situations they never faced before.

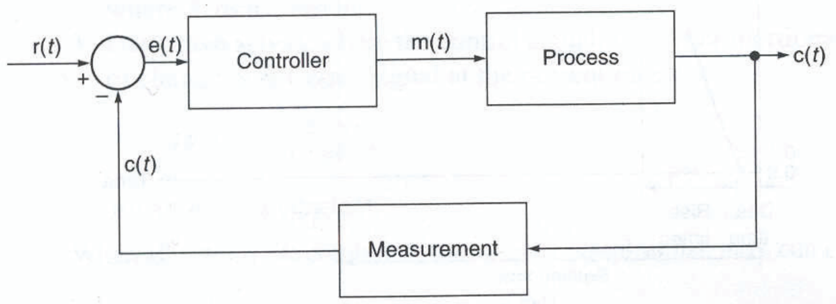


Figure 5.1 A typical closed-loop feedback system. $r(t)$: reference (or setpoint), $e(t)$: error signal, $m(t)$: control action, $c(t)$: control variable.

In digital systems, the proportional action becomes

$$P(t_k) = K_p e(t_k), \tag{5.5}$$

where t_k is the sampling moment and $k = 0, 1, 2, ...$

The integral action becomes

$$I(t_k) = K_I h[e(0) + e(h) + e(2h) + \ ....], \tag{5.6}$$

where the integration is replaced by the summation of successive rectangles of width h and length $e^*(t)$.

The derivative action is given by

$$D(t_k) = K_D \left[\frac{e\,(t_k) - \ e(t_{k\,-\,1})}{h}\right]. \tag{5.7}$$

When all the above actions are combined, the overall expression becomes

$$m(t_k) = K_P e(t_k) + K_I h[e(0) + \ e(h) + \ e(2h) + \ ....]$$
$$+ \frac{K_D}{h}[e\,(t_k) - \ e(t_{k\,-\,1})]. \tag{5.8}$$

PI control is adequate for processes where the dynamics are essentially first order, i.e., the process can be described by a first-order differential equation; e.g., level control in single tanks. PID control is adequate for processes where the dynamics are essentially second-order, e.g., motion with friction. For more complicated systems, a simple PID control may not be sufficient.

5.3 Fuzzy Logic Control Systems [7–14]

Fuzzy logic control (FLC) describes the algorithm for process control as a fuzzy relation between information on the condition of the process to be controlled and the control action. It is thus distinguished from the conventional control algorithms, since information (linguistic or fuzzy model) about the system rather than a mathematical model is what the designer needs. It is also different from expert systems in the sense that the rules of FLC systems are not extracted from the human expert through the system but formulated explicitly by the FLC designer (there are, of course, fuzzy expert systems).

A possible fuzzy control system is shown in Figure 5.4. There is an aspect of similarity between the system shown and any other control system—a control action is produced based on measurement of the output. In conventional systems, control action is reached through an algorithm based on multiplication by a constant, taking a derivative, integration, or a combination of all three (simple addition in PID control and some more entangled combination in more complicated cases).

The essence of fuzzy control algorithms is a conditional statement between a fuzzy input variable A and a fuzzy output variable B. This is expressed by a linguistic implication statement such as

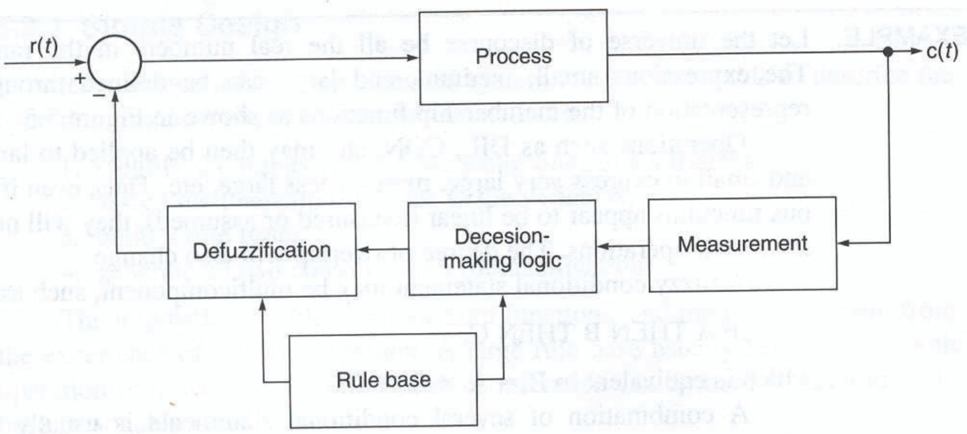


Figure 5.4 A possible fuzzy logic control system.

$A \rightarrow B$ (condition A implies condition B),

which may be written as

IF A THEN B.

There is an equivalency between this expression and the relation obtained by Cartesian multiplication, i.e.

$R = A \times B \equiv$ IF A THEN B.

A fuzzy variable is expressed through a fuzzy set, which in turn is defined by a membership function μ. The fuzzy variable may be continuous or discrete. A continuous variable can be quantized and expressed as if it were discrete.

EXAMPLE: Let the universe of discourse be all the integer numbers in the range 0 to 10. Then we can define the expressions small, medium, and large by the fuzzy sets A, B, and C as

$$A \equiv \text{small} = 1/0 + 0.8/1 + 0.5/2 + 0.2/3$$
$$B \equiv \text{medium} = 0.5/4 + 1/5 + 0.6/6$$
$$C \equiv \text{large} = 0.2/7 + 0.5/8 + 0.7/9 + 1/10.$$

These discrete fuzzy variables may be tabulated as shown in the following table (membership matrix table).

Table 5.1 Membership Matrix

	0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
small	1	0.8	0.5	0.2	0	0	0	0	0	0	0
medium	0	0	0	0	0.5	1	0.6	0	0	0	0
large	0	0	0	0	0	0	0	0.2	0.5	0.7	1

EXAMPLE: Let the universe of discourse be all the real numbers in the range 0 to 10. The expressions small, medium, and large can be defined through graphical representation of the membership functions as shown in Figure 5.5.

Operations such as DIL, CON, etc. may then be applied to large, medium, and small to express very large, more-or-less large, etc. Thus, even if the continuous functions appear to be linear (measured or assumed), they will not stay linear after such operations. The degree of overlap will also change.

A fuzzy conditional statement may be multicomponent, such as

IF A THEN B THEN C,

which is equivalent to $R = A \times B \times C$.

A combination of several conditional statements is usually required for adequate control in a given situation. When an actual input is given, the output is calculated by a defuzzification process.

Since it is impractical to have an explicit rule for every conceivable situation, a composition rule of inference may be used to produce a control rule. For example, assume the explicit control rule is "if A then B." Then it may be inferred that if the condition A' occurs, it implies an outcome B', i.e., "if A' then B'," where B' is inferred from a composition rule of inference such as

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

The designer may, for simplicity, decide that for a given input, the control algorithm checks if there exists a corresponding rule. If no rule exists, then the rule in the immediate neighborhood, within a predetermined distance, will be used.

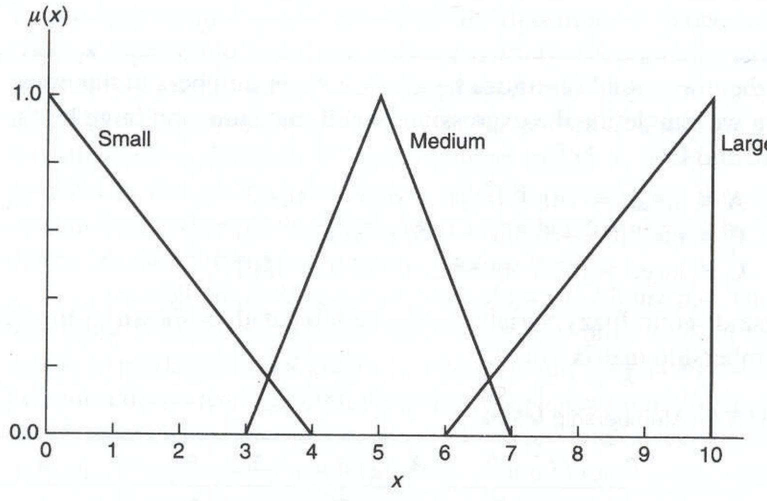


Figure 5.5 Graphical representation of the expressions small, medium, and large in a universe of discourse defined as all the real numbers in the range 0 to 10. (Triangular graphs are used for illustration only, graphs of other shapes may be used.)

5.3.1 Simple Design

In order to design a fuzzy logic control system one has to be able to describe the operation linguistically. In other words, one has to

1. identify the inputs and outputs using linguistic variables;
2. assign membership functions to the variables;
3. build a rule base;
4. generate a crisp control action (defuzzification).

The linguistic variables, membership functions, and the rule base stem from the experience of a skilled operator. A large rule base usually leads to a smooth operation; however, composition rules of inference can be used to keep the rule base reasonably sized.

The rule base in a fuzzy system takes the form IF ... AND ... OR ..., THEN ..., with the operations AND, OR, etc. being, of course, fuzzy logic operations. In a non-fuzzy system that has a rule base control, one rule fires at a time. In fuzzy systems, more than one rule may fire at the same time, but with varied strengths. This mixture of rules firing with varied strengths leads to a crisp control action through the process of defuzzification.

The process of defuzzification in fuzzy logic control systems is not standardized. There are several methods in use such as

- min-max operation (AND–OR operation);
- center of gravity method;
- using output singleton sets.

These ideas are best illustrated by a numerical example.

EXAMPLE: Consider the case of a subway train approaching or leaving a station.

The inputs are the distance from the station and the speed of the train. (More inputs, such as an angle of elevation, radius of curvature, number of cars, etc., can be considered for a better design). The output is the cantidad frenos amount of brakes power used.

The membership functions for distance, speed, and brakes are shown in Figure 5.6. They may not be very realistic, but they will do to illustrate the concepts.

The rule base for this example is composed of the following rules

1. If speed is very_slow and distance very_close, then brakes power is very_light.
2. If speed is slow and distance very_close, then brakes power is heavy.
3. If speed is fast and distance very_close, then brakes power is heavy.
4. If speed is very_fast and distance very_close, then brakes power is very_heavy.
5. If speed is very_slow and distance is close, then the brakes power is light.
6. If speed is slow and distance is close, then brakes power is light.

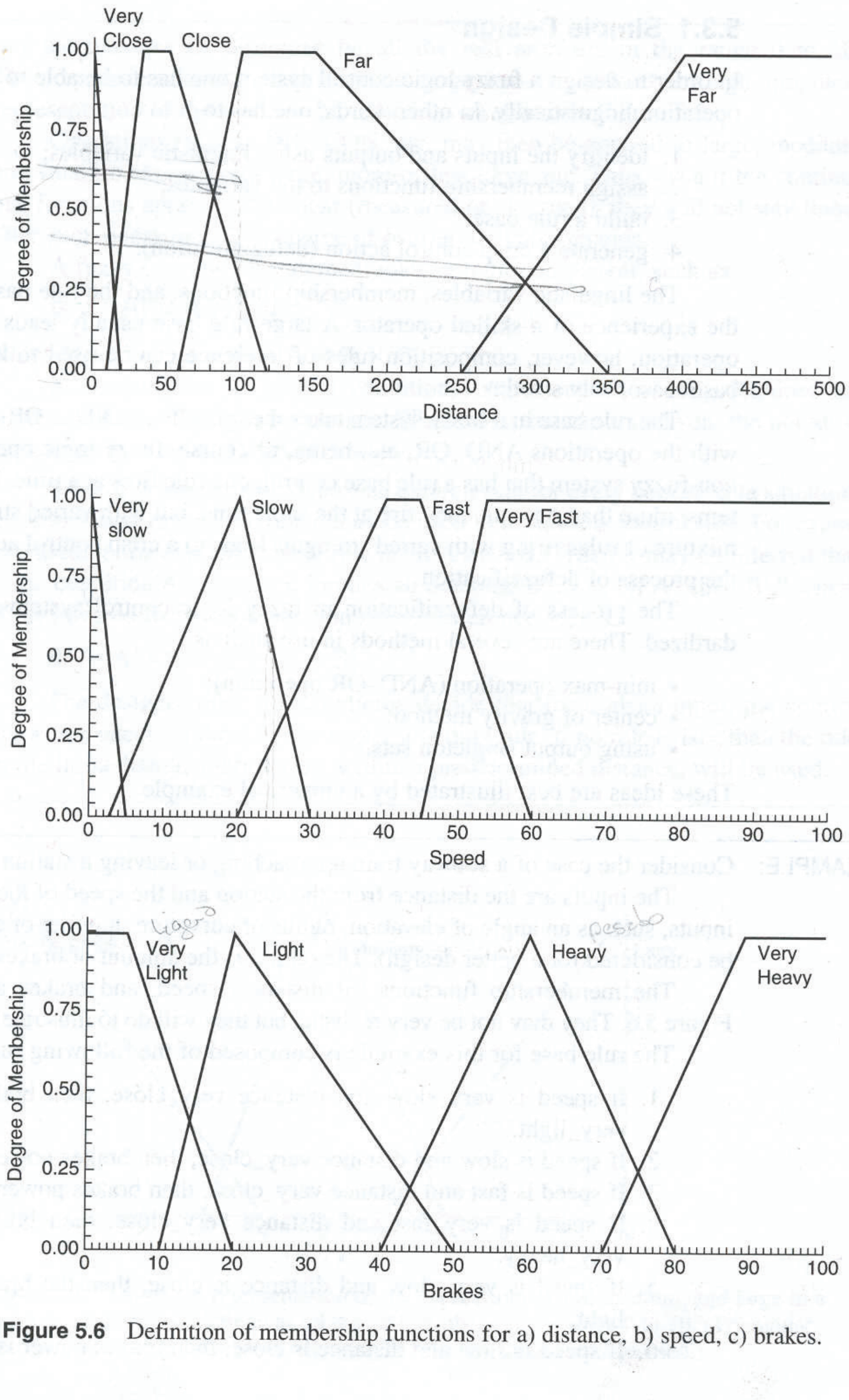


Figure 5.6 Definition of membership functions for a) distance, b) speed, c) brakes.

7. If speed is fast and distance is close, then brakes power is heavy.
8. If speed is very_fast and distance is close, then brakes power is heavy.
9. If speed is very_slow and distance is far, then brakes power is light.
10. If speed is slow and distance is far, then brakes power is very_light.
11. If speed is fast and distance is far, then brakes power is light.
12. If speed is very_fast and distance is far, then brakes power is heavy.
13. If speed is very_slow and distance is very_far, then brakes power is very_light.
14. If speed is slow and distance is very_far, then brakes pressure is very_light.
15. If speed is fast and distance is very_far, then brakes power is light.
16. If speed is very_fast and distance is very_far, then brakes power is light.

These rules are summarized in Table 5.2.

Now, let us consider the control action if the distance is 100 m and the speed is 24.6 km/h.

A speed of 24.6 km/h has a membership of 0.58 in Slow and 0.21 in Fast, i.e.,

$$\mu_{\text{slow}} = 0.58$$
$$\mu_{\text{fast}} = 0.21$$

A distance of 100 m has a membership of 0.29 in Close and 0.88 in Far, i.e.,

$$\mu_{\text{close}} = 0.29$$
$$\mu_{\text{far}} = 0.88$$

Accordingly, the rules that will fire are

- Rule #6 (slow/close) with a strength of 0.58 AND 0.29 = 0.29
- Rule #7 (fast/close) with a strength of 0.21 AND 0.29 = 0.21

Table 5.2 Input matrix with action. The shaded area gives the action (power of the brakes).

Distance / Speed	*Very_Slow*	*Slow*	*Fast*	*Very_Fast*
Very_Close	Light	Heavy	Very_Heavy	Very_Heavy
Close	Light	6 Light	7 Heavy	Very_Heavy
Far	Light	10 Very_Light	11 Light	Heavy
Very_Far	Very_Light	Very_Light	Light	Light

- Rule #10 (slow/far) with a strength of 0.58 AND 0.88 = 0.58
- Rule #11 (fast/far) with a strength of 0.21 AND 0.88 = 0.21

If using min-max defuzzification, an OR operation follows, leading to 0.29 OR 0.21 OR 0.58 OR 0.21= 0.58

The crisp control action will be to apply 58% of the maximum braking power.

If using center-of-gravity defuzzification, the crisp output is determined by the weighted sum of the centroids of the membership functions. The centroid is the center of the output membership function adjusted to the degree of rule firing. The triangular membership functions become trapezoidal, as shown in Figure 5.7.

One can see from Figure 5.7 that the crisp control action will be to apply much less than 58% of the braking power.

Instead of defining the output membership functions as shown in Figure 5.6c, singleton sets could have been used. Figure 5.8 shows an example of using singleton sets to define the output control actions.

The crisp control action is determined by the weighted average defined by:

$$\text{Crisp action} = \frac{\sum^{n} (\text{rule strength})_n \cdot (\text{action})}{\sum^{n} (\text{rule strength})_n}$$

This leads to

$$\text{Brakes power} = \frac{(0.29)(5) + (0.21)(70) + (0.58)(5) + (0.21)(30)}{0.29 + 0.21 + 0.58 + 0.21}$$
$$= 19.65\% \text{ of maximum braking power.}$$

In Figure 5.9 are the results of a simulation carried out using Fuzzy Logic Designer from Byte Dynamics Inc. The results of simulation using six and three rules, respectively, instead of sixteen are shown in Figure 5.10. The control surface becomes progressively less smooth as the number of rules is decreased.

5.4 Fuzzy Logic Control vs. PID Control

PID control is well-established in classical control systems. It is often used as a benchmark against which other control systems are evaluated. One can design a PID Controller using fuzzy logic approach, but why would one be interested in doing that?

A possible reason for the interest in producing a PID-like system using fuzzy logic could have been to gain confidence in a fuzzy logic approach. Fuzzy logic, however, has passed this stage; we know it works well. Further, we know that, in general, a system that can be described mathematically can be described linguistically, though the opposite is not always true.

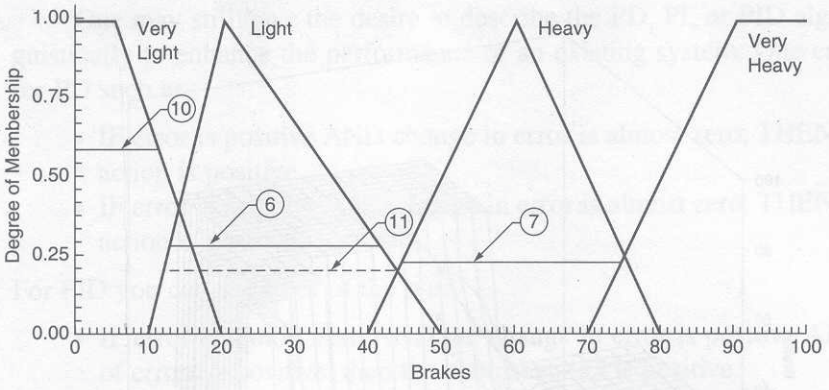


Figure 5.7 Membership functions adjusted to the degree of rule firing.

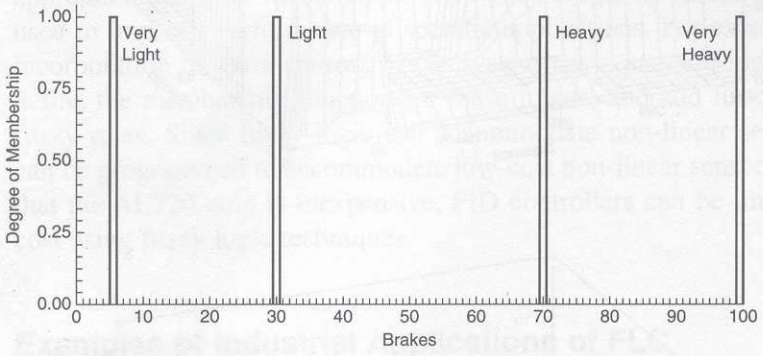


Figure 5.8 Output membership functions defined by singleton sets.

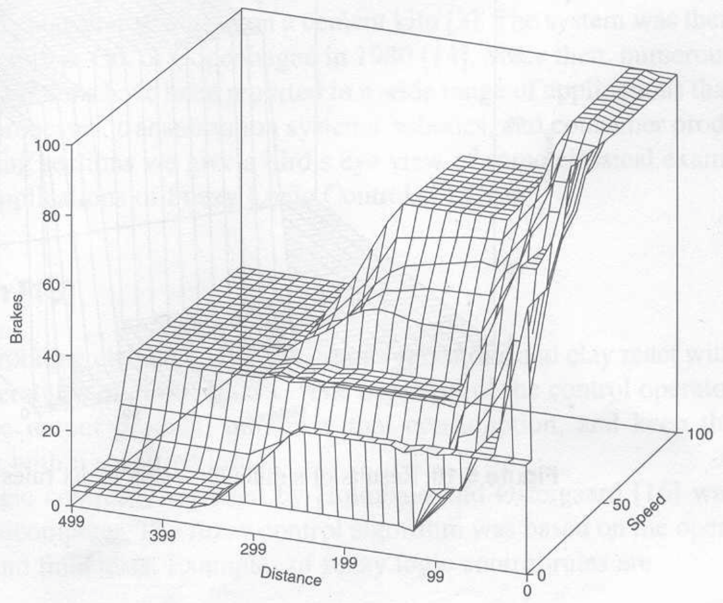
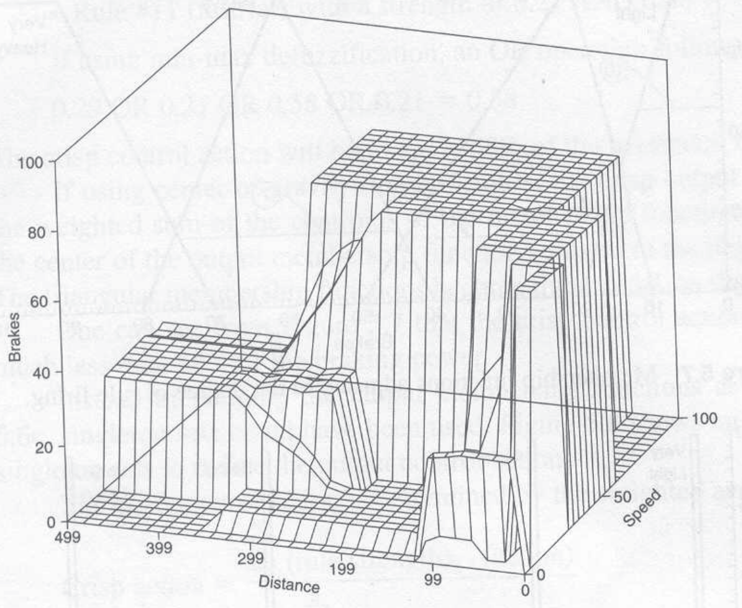


Figure 5.9 Results of simulation using sixteen rules.



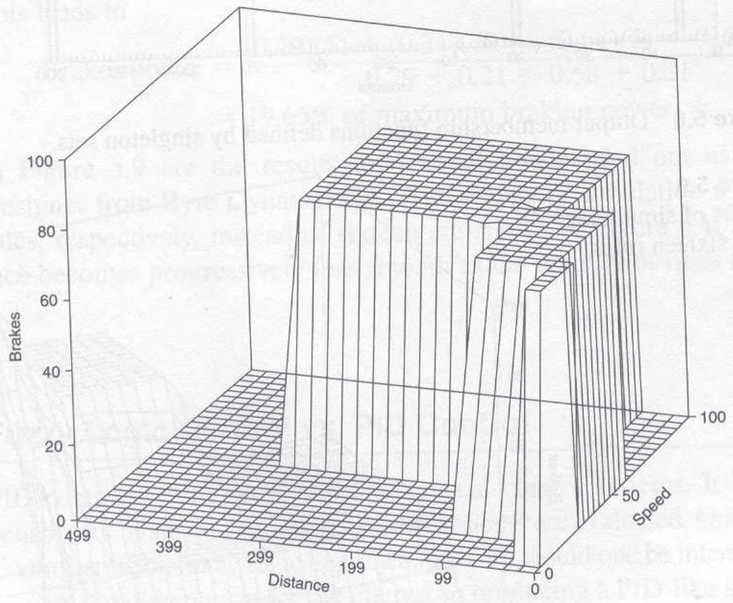


Figure 5.10 Results of simulation using a) six rules b) three rules

One may still have the desire to describe the PD, PI, or PID algorithms linguistically to enhance the performance of an existing system. One can use rules for PD such as

- IF error is positive AND change in error is almost zero, THEN the control action is positive.
- IF error is negative AND change in error is almost zero, THEN the control action is positive.

For PID you can use rules of the type

- IF error is almost zero AND the change in error is positive AND the sum of errors is positive, then the control action is positive.

The AL220 Fuzzy Controller (see Appendix B) was shown to be able to create an approximation of a PID controller [15]. The ease of modification it allows can be used to improve performance in specific applications. For example, it makes the incorporation of extra sensors in the system easy; just add inputs from sensors, define the membership function for these inputs, and add fuzzy variables to the fuzzy rules. Since fuzzy logic can accommodate non-linear sensors, the AL220 can be programmed to accommodate low-cost non-linear sensors. Adding the fact that the AL220 chip is inexpensive, PID controllers can be implemented at low cost using fuzzy logic techniques.

5.5 Examples of Industrial Applications of FLC

One of the early applications of fuzzy logic control occurred in 1978 in Denmark, where a fuzzy logic controller was used on a cement kiln [3]. The system was then marketed by F.L. Smith & Co. of Copenhagen in 1980 [14]. Since then, numerous fuzzy logic control systems have been reported in a wide range of applications that include industrial processes, transportation systems, robotics, and consumer products. In the following sections we give a bird's eye view of some classical examples of industrial applications of Fuzzy Logic Control (FLC).

A. Cement kiln [15]

A cement kiln is a rotating chamber in which ground limestone and clay react with each other at temperatures of 1000–1400°C. The objective of the control operator is to maximize the output product, minimize fuel consumption, and keep the machinery running within specified limits.

The fuzzy logic controller reported by Holmbald and Østergaard [16] was realized with a minicomputer. The fuzzy control algorithm was based on the operators' experience and field tests. Examples of fuzzy logic control rules are

- If TC is ZE and FL is LO, then FA is MN
- If TC is ZE and FL is OK, then FA is ZE
- If TC is SN and FL is LO, then FA is SP

where TC = kiln drive torque change, FL = free lime, FA = fuel flow adjustment, ZE = zero, LO = low, OK = okay, SN = small negative, and SP = small positive.

The fuzzy variables LO, HI, etc. were defined in the algorithm. The extent of combination of each control rule is judged based on the actual conditions of the kiln.

The FLC was reported to reduce fuel consumption slightly compared to the operator controlled system.

B. Heat exchanger process [17]

The core of a heat exchanger is two hydraulic systems: hot and cold water circuits. Water in a reservoir is heated electrically. It is then pumped into a pipeline to the neighborhood of a cold water pipe. The cold water gets heated and the hot water loses heat. The hot water then goes back to the reservoir for reheating.

The control problem reported was to adjust the hot water flow rate, FH, and the reservoir heating power, W, so that the cold water outlet temperature, TCO, and hot water inlet temperature, HI, take on setpoint values TCOS and THIS, respectively. Two 7 × 9 decision tables were needed. Using an IBM/1800 computer, 7 hours of computing time were used.

As an example of control rules, one of the rules for controlling the cold water outlet TCO states that

- If TCOE is LP or MP and THIE is NOT LP or MP, then FH is MP where LP = large positive, MP = medium positive.

The experimental results showed characteristics comparable to those of a PI controller.

C. Boiler and steam engine [18, 19]

The steam engine speed was controlled by adjusting the throttle opening at the input of the engine cylinder, while the boiler pressure was controlled by the heat input to the boiler.

The operating experience and technical knowledge of the process was used to specify the control rules. The algorithm was implemented on a PDP8 computer.

As an example of control rules,

- If PE = NB then if CPE = NOT (NB or NM) then HC = PB
- If PE = (NB or NM) then if CPE = NS then HC = PM

where PE = pressure error, CPE = change in pressure error, HC = heat input change, NB = negative big, NM = negative medium, PB = positive big, NS = negative small, PM = positive medium.

When FLC was compared with DDC, the FLC system was found to be much less sensitive to process parameter changes.

D. Water purification plant [20]

In a water purification plant, raw water is purified by injecting chemicals at rates related to water quality. The feed rate of chemicals is judged by a skilled operator. Aluminum sulfate or PAC (polymerized aluminum chloride) is used as a coagulant. Aluminum sulfate is less expensive than PAC but is not as effective in low temperature water. An alkaline agent, such as lime or caustic soda, is used for pH adjustment. A chlorine agent is used for sterilization. All chemical agents, except the coagulant agent are usually fed at a fixed rate that is proportional to the flow of raw water. The reaction of the coagulant with water impurities is a complicated process to model; no universally accepted method of a feed rate exists.

Yagishita et al. [20] structured a system based on fuzzy logic so that the feed rate of the coagulant can be judged automatically even by an unskilled operator. Ten control rules were used. As an example,

- If TUUP = MM, then DDOS = PS
- If TUUP = ML, then DDOS = PM

where TUUP = grade of water, DDOS = amount of compensation dose, MM = medium, PS = positive small, ML = medium large, and PM = positive medium.

The fuzzy variables MM, PS, etc., were assumed to have a normal distribution. A particular set was defined by specifying the average value and the standard deviation.

The system was used successfully in a water purification plant in Akita City, Japan in 1983.

E. Refuse incineration plant [21]

In a refuse incineration plant, a refuse supply feeder takes the refuse to a stoker in which refuse if burnt. The major purposes of the combustion control are:

1. providing adequate supply of refuse (oversupply extinguishes the flame, insufficient supply leads to poor combustion)
2. keeping the evaporation rate and furnace temperature at their specified values.

The fuzzy control for the plant in Shizuoka City reported by Ono et al. [21] consists of fuzzy sensors, fuzzy controllers, and conventional combustion control

(ACC). A total of 45 control rules based on the knowledge and experience of the operator were incorporated. For example,

- If the evaporation rate is in the target zone, then leave it to automatic operation by ACC.
- If the evaporation rate is low and the refuse layer thickness is high, then increase amount of combustion air and decrease feeder speed.

F. Arc welding robot [22]

A hazardous environment surrounds the arc-welding process. Very bright welding sparks, high temperature, and toxic fumes are associated with the process. It also takes a long time to train a human welder. Thus, the process is a suitable field of application for a robot.

The system has two inputs and two outputs representing the composition of a vertical and a horizontal control system. The FLC was designed for the two systems independently. Eight control rules were used. The membership functions of the fuzzy variables P_H and N_H were defined by empirical formulae.

The experimental data indicated that using FLC, the system produced less vibration of tracking locus compared to the case where PI controller was used.

5.6 Stability

Traditionally, stability analysis is a fundamental aspect in determining the reliability of a control system. Conventional stability analysis is based on the availability of a mathematical model of the system to be controlled. Difficulty arises in fuzzy control systems where the major advantage is in devising a control system without the need for a mathematical model.

Mamdani [23] voiced the opinion that a fuzzy controller can be analyzed qualitatively to assure that a runaway instability does not occur. Changes to the control system structure to improve performance can be introduced by monitoring the system running open-loop.

For example, King and Mamdani [19] reported on temperature control of stirred tanks. Their initial results showed a tendency to oscillate around the setpoint. Simulation of the process was carried out to determine the cause of the instability. Time delay was observed to be a primary cause of the instability. An effective way to solve the problem was to modify the universe of discourse by changing the quantization levels, and by adjusting the overall loop gain. The results were confirmed to be satisfactory on the real process.

Mamdani [23] suggested that the discussion on stability is irrelevant if it implies that no attempt be made to control difficult processes unless a rigorous theory can be found to design fuzzy logic controllers. This was accepted implicitly by numerous reports in the literature of fuzzy logic systems design. The concern about a systematic method of studying the stability, however, existed since the early stages of fuzzy logic control applications. Nevertheless, the literature has much work reporting on applications but much less reporting on stability studies (see for example references [24–38]).

If we have a mathematical model, we expect a mathematical analysis of stability. But if we have a linguistic model, why should we expect mathematical analysis of stability? Would it be possible to have a linguistic stability criterion? Stability can be defined linguistically of course. Probably the closest analysis to such an idea is the one by Kiszka et al. [28].

The basic equation of a simple fuzzy dynamic system has the form

$$X_{k+1} = X_k \circ U_k \circ R \qquad k = 0, 1, 2, 3, \ldots$$

where X_k and X_{k+1} are the fuzzy sets of the states at kth and $(k-1)$th time instants, respectively, U_k is the input at the kth instant, and R is the fuzzy relation between the input and output.

A free or unforced dynamic system is realized by setting the input to zero. In this case, we have

$$X_{k+1} = X_k \circ P$$

where $P = U_k \circ R$ *for* $U_k =$ zero for all k, $k = 0, 1, 2, \ldots$

A state of X_s of a free dynamic system is an equilibrium state if $X_{k+1} = X_k$ for all k, i.e.

$$X_s = X_s \circ P.$$

Various authors [38–40] put forward algorithms to calculate the greatest equilibrium state. Kiszka et al. [28] suggested an algorithm based on the energy of the system. A dynamic system is stable if its total energy is minimal and constant. The system is unstable if its energy increases with time, and oscillatory if its energy fluctuates periodically. They proposed an intuitive measure of energy of a fuzzy relation. It relies on the most salient physical characteristics of the fuzzy relation, such as the position in the support set, maxima of membership function, etc. They defined the energy of a fuzzy relation P as

$$E(P) = \frac{1}{n \cdot m} \sum_{i=1}^{n} \sum_{j=1}^{m} w(x_i, y_j) \cdot f(\mu_p(x_i, y_j))$$

where $w(x_i, y_j)$ is the function responsible for the position of the fuzzy relation in the universe of discourse, $f(\mu_p(x_i, y_j))$ is the function responsible for the shape, spread, etc., of the fuzzy relation, and m and n are the cardinalities.

Further, they defined the incremental rate of energy change or the relational characteristic energy function as

$$\Delta E(P, i) = E(P^i) - E(P^{i-1})$$

where $P^i = P \circ P \circ P \ldots P$ is the i-times max-min composition of the fuzzy relation P.

The energistic stability algorithm ESA, states the following:

If $\Delta E(P, i) \leq 0$ for $i \rightarrow \infty$, then the system is stable.

If $\Delta E(P, i) > 0$ for $i \rightarrow \infty$, then the system is unstable.

If $|\Delta E(P, i)| = |\Delta \mathrm{E}(\mathrm{P}, i + \tau)|$, for $i \rightarrow \infty$, then the system oscillates with frequency $1/\tau$.

In summary, to investigate the stability of a fuzzy dynamic system, the linguistic algorithm is translated into a look-up table or a relation P. The system is stable if the energy of the relation P is minimal and constant.

In the following we give some simple examples to illustrate the type of calculations done by ESA as given by Kiszka et al. [28].

EXAMPLE: Let the fuzzy relation P be given by

$$\mathrm{P} = \begin{bmatrix} 0.1 & 0.5 \\ 0.3 & 0.2 \end{bmatrix},$$

and the universe of discourse $X = Y = [1, 2]$. Determine whether the system is stable, oscillating, or unstable.

To determine the stability of the system we start by calculating several consecutive values of $E(P^i)$, where $P^i = P \circ P \circ P \ldots \circ P$. The max min composition rule discussed in section 3.4.2 gives

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \circ \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix}$$

where,

$$c_{11} = \max\{\min(a_{11}, b_{11}), \min(a_{12}, b_{21})\}$$
$$c_{12} = \max\{\min(a_{11}, b_{12}), \min(a_{12}, b_{22})\}$$
$$c_{21} = \max\{\min(a_{21}, b_{11}), \min(a_{22}, b_{21})\}$$
$$c_{22} = \max\{\min(a_{21}, b_{12}), \min(a_{22}, b_{22})\}$$

Since, $P^2 = P \circ P$, then

$$P^2 = \begin{bmatrix} 0.1 & 0.5 \\ 0.3 & 0.2 \end{bmatrix} \circ \begin{bmatrix} 0.1 & 0.5 \\ 0.3 & 0.2 \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix}$$

where,

$$c_{11} = \max\{\min(0.1, 0.1), \min(0.5, 0.3)\} = \max\{0.1, 0.3\} = 0.3$$

$$c_{12} = \max\{\min(0.1, 0.5), \min(0.5, 0.2)\} = \max\{0.1, 0.2\} = 0.2$$

$$c_{21} = \max\{\min(0.3, 0.1), \min(0.2, 0.3)\} = \max\{0.1, 0.2\} = 0.2$$

$$\begin{aligned} c_{22} &= \max\{\min(0.3, 0.5), \min(0.2, 0.2)\} \\ &= \max\{0.3, 0.2\} = 0.3 \end{aligned}$$

Thus,

$$P^2 = \begin{bmatrix} 0.3 & 0.2 \\ 0.2 & 0.3 \end{bmatrix}.$$

Similar procedures lead to

$$\begin{aligned} P^3 &= P^2 \circ P \\ &= \begin{bmatrix} 0.3 & 0.2 \\ 0.2 & 0.3 \end{bmatrix} \circ \begin{bmatrix} 0.1 & 0.5 \\ 0.3 & 0.2 \end{bmatrix} = \begin{bmatrix} 0.2 & 0.3 \\ 0.3 & 0.2 \end{bmatrix} \\ P^4 &= P^3 \circ P \\ &= \begin{bmatrix} 0.2 & 0.3 \\ 0.3 & 0.2 \end{bmatrix} \circ \begin{bmatrix} 0.1 & 0.5 \\ 0.3 & 0.2 \end{bmatrix} = \begin{bmatrix} 0.3 & 0.2 \\ 0.2 & 0.3 \end{bmatrix} \end{aligned}$$

From the definition of the energy of the fuzzy relation P

$$E(P) = \frac{1}{n \cdot m} \sum_{i=1}^{n} \sum_{j=1}^{m} w(x_i, y_j) \cdot f(\mu_p(x_i, y_j))$$

we get

$$\begin{aligned} E(P) &= \frac{1}{2 \times 2}\Big[(1 \times 1) \times 0.1 + (1 \times 2) \times 0.5 + (2 \times 1) \times 0.3 \\ &\qquad + (2 \times 2) \times 0.2\Big]. \\ &= \frac{1}{4}(2.5) \end{aligned}$$

Similar calculations lead to

$$E(P^2) = \frac{1}{4}(2.3),\ E(P^3) = \frac{1}{4}(2.2),\ E(P^4) = \frac{1}{4}(2.3), \text{ and } E(P^5) = \frac{1}{4}(2.2).$$

Thus, the fuzzy system is oscillating.

EXAMPLE: Let

$$P = \begin{bmatrix} 0.1 & 0.3 \\ 0.7 & 1.0 \end{bmatrix} \text{ and the universe of discourse } X = Y = [1, 2].$$

Then,

$$P^2 = \begin{bmatrix} 0.1 & 0.3 \\ 0.7 & 1.0 \end{bmatrix} \circ \begin{bmatrix} 0.1 & 0.3 \\ 0.7 & 1.0 \end{bmatrix} = \begin{bmatrix} 0.3 & 0.3 \\ 0.7 & 1.0 \end{bmatrix}$$

$$P^3 = \begin{bmatrix} 0.3 & 0.3 \\ 0.7 & 1.0 \end{bmatrix} \circ \begin{bmatrix} 0.1 & 0.3 \\ 0.7 & 1.0 \end{bmatrix} = \begin{bmatrix} 0.3 & 0.3 \\ 0.7 & 1.0 \end{bmatrix}$$

6 Electronic Neural Networks

6.1 Introduction

Through the attempts to understand the human computational power two fields emerged: fuzzy logic and electronic (or artificial) neural networks. The two fields were developed independently. Merging the two fields can get us closer to understanding the human computational power, or at least provide us with very powerful computational capabilities.

As discussed earlier, fuzzy logic started with the goal of using approximate human reasoning in knowledge-based systems. Fuzzy logic provides inference mechanisms under cognitive uncertainty [1]. Neural networks, on the other hand, may owe their start to the quest for mimicking the biological sensory systems in pattern recognition. Elementary neural networks for detecting simple forms were proposed by early researchers. It was then immediately realized that the performance of the biological systems is very difficult to mimic. It was also realized that the biological sensory systems depend on the cognitive processes of the brain.

Although pattern recognition is still at the heart of neural networks applications, they have been proven useful in many other applications, such as optimization computations and decision making. Neural networks have the advantage of learning, adaptation, and fault tolerance.

Several authors (see for example references 1–25) have reported on techniques and approaches of merging the two fields and the emerging applications.

Two possible things that one can imagine achieving are enabling neural networks to deal with cognitive uncertainty and generating fuzzy rules using neural networks.

The objective of this chapter is to introduce concepts of neural networks to pave the way for understanding fuzzy neural networks. We will briefly discuss the biological neural system—the system electronic neural networks are trying to mimic. Then we will introduce several ideas of electronic neural networks and discuss in simple terms representative examples of neural network models.

Before going any further, we should emphasize that the commonly used expression "mimic the biological system" is an exaggeration; electronic neural networks, at best, are nothing more than crude approximations of the biological system. Probably you have heard the quip "if our brains were simple to understand, we would be too simple to understand them."

6.2 The Biological System

We present here a simplified picture of the neuron, the basic building unit of the brain and related structures. Then we discuss the learning process.

6.2.1 The Neuron

A neuron is a living cell. Contrary to other cells, it does not duplicate itself. It can, however, develop more connections with other neurons as a person learns more. The brain is estimated to have billions of neurons; each neuron has thousands of connections to other neurons [26].

The basic anatomy of a biological neuron is shown in Figure 6.1. It is basically composed of the cell body (soma), dendrites, axon, and terminal buttons (synaptic terminals). The dendrites are branch-like structures that pick up input signals to the soma. The soma performs the central functions of the cell. The axon carries the output signal; it is terminated with the so-called terminal buttons.

Almost half of the brain weight is made of another cell type known as neuroglia (nerve glue) [27]. There are two main types of glial cells in the central nervous system: the astrocytes and the oligodendrocytes. The astrocytes have a star-like appearance, with several long arms radiating out of a central cell body. The oligodendrocytes also have a central body, but with shorter and more numerous arms than the astrocytes. The oligodendrocytes form the myelin sheath around axons in the central nervous system. In the peripheral nervous system another type of glial cell, the Schwann cell, is responsible for forming the myelin sheath [28]. The myelin sheath surrounds many axons. It reduces the capacitance between the cell membrane (cytoplasm) and the outside fluid. This leads to an increase in the velocity of signal propagation along the axon (it is about 120 m/s) [29]. The myelin sheath is segmented; each segment is a few millimeters

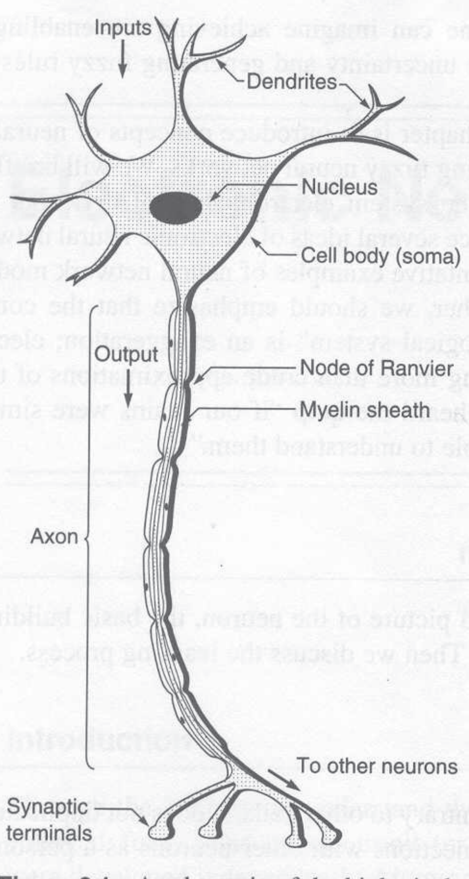


Figure 6.1 A schematic of the biological neuron.

in length. The gaps between the segments are known as the nodes of Ranvier. These nodes act as repeater sites to restore the propagating signal [29]. Other roles for glia have been speculated; for example, it is thought [28] that the glia might provide structural support for neurons, segregate groups of neurons, supply metabolic components to the neurons, and play a role in information handling and memory storage.

The point at which two or more neurons interconnect is known as the synapse. The two neurons are connected through the so-called synaptic grieta cleft. In other words, the synaptic cleft is the area between the terminal buttons of one neuron and the dendritic ends of another. The signal is transmitted across by chemicals called neurotransmitters. The signal propagates by these chemicals at a relatively low velocity (about 2 mm/min.). The chemicals can be modified to increase or decrease the velocity of signal propagation. Such modification occurs by a signal from the brain, for example, in the face of danger, or due to drugs. If this break in the transmission circuit did not exist, one would be alerted with a muscle twitch at every step and every touch.

The operation of the neurons can be summarized in simple terms as follows. The dendrites receive synaptic inputs from other neurons. The input signal current builds up an electric charge across the cell-membrane capacitance. When a certain threshold is reached, a nerve pulse, known as the action potential, is generated. The action potential propagates along the axon to other neurons (the term potential, rather than voltage, is commonly used by neurophysiologists).

At rest, a live neuron has a potential across it, the so-called rest potential. The existence of such voltage can be explained as follows. The fluid outside is basically a sodium chloride (NaCl) solution (40% of blood volume is NaCl). The inside fluid is rich in potassium (K). A metabolically driven pump in the cell membrane keeps the inside fluid rich in potassium and depleted of sodium, and the reverse for the outer fluid. The gradient of charged particles leads to a voltage difference developed across the membrane. That voltage, V_r, is given by

$$V_r = -\frac{K_B T}{q} \ln\left(\frac{N_{\text{in}}}{N_{\text{ex}}}\right), \tag{6.1}$$

where N_{in} and N_{ex} are the ionic concentrations in the internal and external fluids, respectively. K_B is the Boltzmann constant, T is the absolute temperature, q is the electronic charge, and V_r is the reverse voltage developed.

Each ion gradient will produce a corresponding voltage. The overall picture can be visualized by the circuit shown in Figure 6.2.

The rest potential, V_0, across the membrane is the voltage that appears when the net current flow is zero. This occurs when

$$I = 0 = \frac{(V_K - V_0)}{R_K} + \frac{(V_{\text{Na}} - V_0)}{R_{\text{Na}}} + \frac{(V_{\text{Cl}} - V_0)}{R_{\text{Cl}}}. \tag{6.2}$$

The chloride current was found to be very small [30], and hence we can write

$$V_0 = \frac{V_K R_{\text{Na}} + V_{\text{Na}} R_K}{R_K + R_{\text{Na}}} \tag{6.3}$$

The value of the rest potential varies from one experiment to another; it has a value of around 90 m V.

The capacitance of the membrane is about 1 μF/cm^2, and since $C = \epsilon_0 \epsilon_r A/\ell$, ϵ_r is about 8.5. The value of RC (the time constant) is about 4 ms, indicating a resistance of about 0.4235 Ω/cm^2, or a resistivity of about $5.647 \times 10^9 \Omega \cdot$ cm[31].

The previous analysis shows that a neuron at rest is polarized to a negative potential and that the membrane under these conditions is a good insulator. Now, applying an excitory signal depolarizes the membrane (decreases the negative voltage). Applying an inhibitory signal hyperpolarizes the membrane (increases its negative voltage).

A biological network is usually modeled by a network that has nodes (or processing units) and input and output links. The processing units simulate the neurons (or actually the soma) and the links simulate the synapses. Weights are

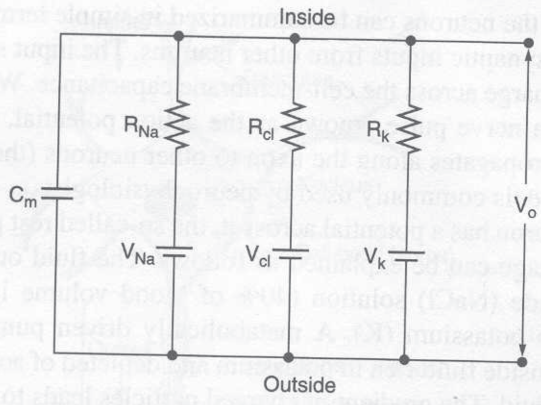


Figure 6.2 Voltage across cell membrane at rest. V_{Na}, V_{Cl}, V_K represent the reverse potential, V_r, of sodium, chlorine, and potassium, respectively. R_{Na}, R_{Cl}, and R_K represent the corresponding resistance of the membrane. C_m is the membrane capacitance. V_0 is the rest potential.

assigned to the input links to simulate the action of the neurotransmitters. Such modeling can be done with software or hardware (or a combination). An algorithm is used to adjust the weights of the input links so that the neurons produce the desired output. Such an algorithm is called a learning algorithm.

6.3 Learning

Learning may be defined as *a relatively permanent change in behavior that is caused by practice or experience* [32]. Learning can usually be estimated by measuring performance; however, learning may also occur without any immediate measurable change in behavior. It may behave as a catalyst for measurable learning. The relatively permanent change in behavior is also called *self-organization*, or *adaptation* in some cases. Learning is described as relatively permanent, rather than absolutely permanent, to allow for forgetting. It is also defined as due to training to exclude effects due to damage to the neurons (a dead neuron has a permanent change in behavior!).

Learning in biological systems is thought to occur when the effective coupling between the neurons is modified. Similarly, learning in artificial neural networks is thought to occur when the weight of the input links are modified. There are, of course, good learning and bad learning, i.e., you may learn how to do things right or you may learn to do things wrong. Usually we are after good learning. We will see later that the central idea of learning in neural networks is the associative memory: store a set of patterns in such a way that when presented with a new pattern, the network can associate the new pattern with one of the stored patterns.

There are two modes of learning: supervised and unsupervised. Unsupervised learning is very common in biological systems. We are continuously receiving information without any obvious relationships or any form of instructions. The question "what use can be made of this flood of information?" has been addressed by several authors (for a review see reference 34). This mode of learning is very important for electronic neural networks, since in many practical applications the training data are not available.

In supervised learning mode, the desired output is known and the system is trained to produce it; it is learning with a teacher. In electronic neural networks, a training set is used. The training set is composed of examples from which the network can learn. Each example consists of a pair, an input and the corresponding output data.

There are several learning laws. Some of these follow:

Hebbian learning

Hebbian learning is the first and probably the best known learning law. It states "when an axon of cell A is near enough to fore (excite) cell B and repeatedly takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased" [35]. This law is powerful yet simple; however, the law as stated needs modifications to take into account some practical aspects, such as the following:

- Biological systems forget as well as learn.
- There are excitory and inhibitory inputs.
- A limit has to exist on the increase of a reinforced input.

Grossberg (Neo-hebbian) learning

In this model biological forgetting is incorporated into the Hebbian learning law (reference 36 is a collection of research papers focusing on this area).

Kohonen's learning

In Kohonen's model the neurons compete with each other in learning. The neuron with largest output wins; it and its neighbor are allowed to adjust the weights of their inputs (allowed to learn). The competing neurons are suppressed. The size of the neighborhood becomes smaller as the training time increases (reference 38 is devoted to this topic).

Back-propagation learning

The back-propagation model is a commonly used learning algorithm for electronic neural networks. Nevertheless, there is no evidence to suggest that this method occurs in biological systems. This is another example where we are interested in a model as long as it works, regardless of its relation to reality.

The weights are set randomly at the beginning. Learning takes place in two phases: forward and backward phases. In the forward phase, an input is applied and the output is observed. The error between the actual output and the output desired is calculated. The second phase starts by propagating the error backward to adjust the weights. When the two phases are complete, a new input from the training set is applied, and so on. A long training time is usually required, since hundreds or even thousands of iterations may be needed. A collection of monographs on this topic is given in reference 39. The concept is discussed in more detail in section 6.4.3.

Fuzzy learning

In various conventional neural networks, a training set is composed of numerical data to train the network. Fuzzy learning utilizes expert knowledge represented by if-then rules to train the network. This takes neural networks a step closer to the biological system. The topic is discussed further in chapter 7.

Several other techniques for training neural networks are used. Drive-reinorcement learning, stochastic learning, simulated annealing, counter back propagation, and adaptive resonance are examples. In section 6.4 we discuss various learning models in some detail.

Explanation

A network with perfect training inspires complete confidence in its decision. To approach perfect training, a neural network needs a large training set, a very powerful learning technique, and *enough* time to complete the training. Practical networks, however, have less than perfect training. Doubts may arise about the validity of their decisions. Explanation becomes important for the user to accept the results presented by the network.

Intelligent systems (including most people) can explain the decision they make. They can, for example, provide the intermediate steps of their reasoning process. Although explanation supplied by a machine may be elementary and limited, it can, nevertheless, be useful for the user.

Various authors have reported on the explanation that neural networks can provide. A review of research in that area is given in reference 40.

6.4 Neural Networks Models

Electronic neural networks are inspired by the biological neural system. Not all the models, however, have biological justifications. The major function of a neural network is classification of some sort. This leads to applications in pattern recognition (visual and auditory), communications, and control.

To learn about the models, it may be tempting to start by classifying the models themselves. Several possible classification schemes exist. It depends on who is doing the classification and what point of view is to be conveyed. For example, we can say there are two major classes: biologically-justified and not biologically-justified, with subclasses in each case. We can also say that the two major classes are supervised and unsupervised networks, with the subclasses feedforward and feedback in each case. The same classification can be presented differently by considering the major classes to be the feedforward and feedback, and the subclasses to be the supervised and unsupervised. Of course, the classification can be based on the inputs—binary or continuous-valued—or even on the potential applications.

Here we present you with several models; it will be up to you to re-classify them to carry on generalization and adaptation to your needs.

6.4.1 The Perceptron

A simple model of the biological neuron is shown in Figure 6.3. It can be built using software or VLSI technology.

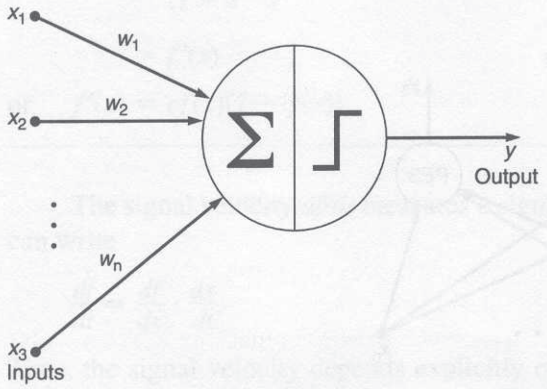


Figure 6.3 A simple model of the neuron (McCulloch-Pitts model). The inputs are x_1, x_2, ... x_n. The output is y. The unit sums the inputs and has a threshold function.

The synaptic inputs are represented by $x_1, x_2, ..., x_n$, with weights $w_1, w_2, ... w_n$ assigned to each input. The axon output is represented by y. The processing unit represents the neuron (or actually the soma). It sums the weighted inputs. If the sum (minus any offset) exceeds a certain threshold, y is set to one (the neuron fires); otherwise, it is zero. This action can be described by

$$y = f\left[\sum_{i=1}^{n} (w_i x_i - \theta)\right] \tag{6.4}$$

where θ is the neuron's off-set or bias, and f is an activation function, which is usually monotonically increasing with upper and lower bounds, as in the case of a step function.

The model neurons, or processing units, connected in a simple fashion, as shown for example in Figure 6.4, form the perceptron.

EXAMPLE: Suggest an algorithm to train a perceptron to classify hand-written letters as A or B.

Hand-written letters have variations in their appearance; however, they are quite distinct. Let the perceptron output be one when the letter detected is an A and zero when the letter detected is B. Then, the algorithm may be

- Assign random values to the weights and the threshold.
- Present an input (digitized picture).
- Calculate the actual output:

$$y = f\left[\sum_{i=1}^{n} [(w_i(t) x_i(t) - \theta)\right].$$

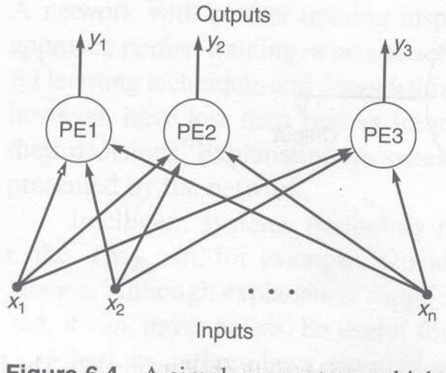


Figure 6.4 A simple perceptron, which (by definition) has only one layer of processing units (PE = processing unit).

- Adjust the weights to reduce the output error:
  if y is correct; $w_i(t+1) = w_1(t)$
  if $y = 0$ and should be 1; $w_i(t+1) = w_i(t) + \alpha x_i(t)$
  if $y = 1$ and should be 0; $w_i(t+1) = w_i(t) - \alpha x_i(t)$,

where $0 \le \alpha \le 1$ is a positive constant that controls the adaptation rate.

A commonly used activation function in neural networks is a signoid (s-shaped) function such as

$$f(x) = \frac{1}{1 + e^{-cx}} \tag{6.5}$$

where c is a positive scaling constant. This function is particularly interesting because its derivative is very easy to compute.

EXAMPLE: Given that

$$f(x) = \frac{1}{1 + e^{-cx}}, \text{ show that } f'(x) = cf(x)[1 - f(x)].$$

$$\text{RHS} = f'(x) = c(1 + e^{-cx})^{-2}(e^{-cx})$$

$$\text{LHS} = cf(x)[1 - f(x)]$$

$$= \frac{c}{1 + e^{-cx}}\left[1 - \frac{1}{1 + e^{-cx}}\right]$$

$$= \frac{c}{1 + e^{-cx}}\left\{\frac{1 + e^{-cx} - 1}{1 + e^{-cx}}\right\}$$

$$= \frac{ce^{-cx}}{(1 + e^{-cx})^2}$$

$$= f'(x)$$

or $\quad f'(x) = cf(x)[1 - f(x)]$.

The signal velocity *df/dt* measures a signal's instantaneous time change. One can write

$$\frac{df}{dt} = \frac{df}{dx} \cdot \frac{dx}{dt}. \tag{6.6}$$

Thus, the signal velocity depends explicitly on activation velocities. This dependence leads to an increase in the number of unsupervised learning laws that adapt with locally available information [15].

6.4.2 Pictorial Illustration of Simple Perceptron Behavior

The summation

$\sum_{i=1}^{n}(w_i x_i - \theta)$ in equation (6.4) can be rewritten as

$\sum_{i=0}^{n}(w_i x_i)$, where the term $w_0 x_0$ takes care of the bias θ. Now we can write the perceptron input as a vector,

$$\mathbf{X} = (x_0, x_1, \ldots x_n). \tag{6.7}$$

We can also write the corresponding weights as a vector,

$$\mathbf{W} = (w_0, w_1, \ldots w_n). \tag{6.8}$$

The weighted sum can be replaced by the dot product, $\mathbf{W} \cdot \mathbf{X}$. The result is a line in an n-dimensional space. It partitions the space such that the group A pattern is in one half and group B is in the other half.

To get a better picture, let us examine a two-dimensional example. Consider the case of a two-input logic OR function. The output is one if the input patterns are 01, 10, or 11. The output is zero for the pattern 00 as illustrated in Figure 6.5.

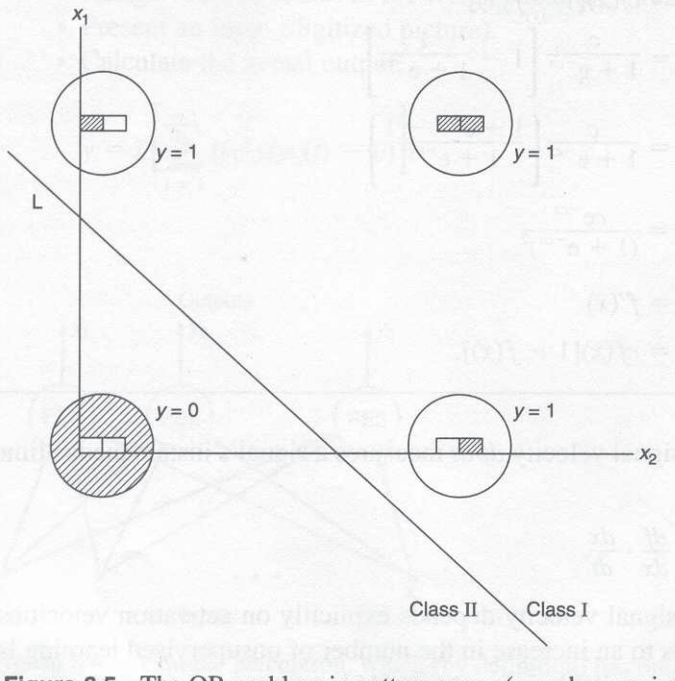


Figure 6.5 The OR problem in pattern space (x_1 and x_2 are inputs, y is an output).

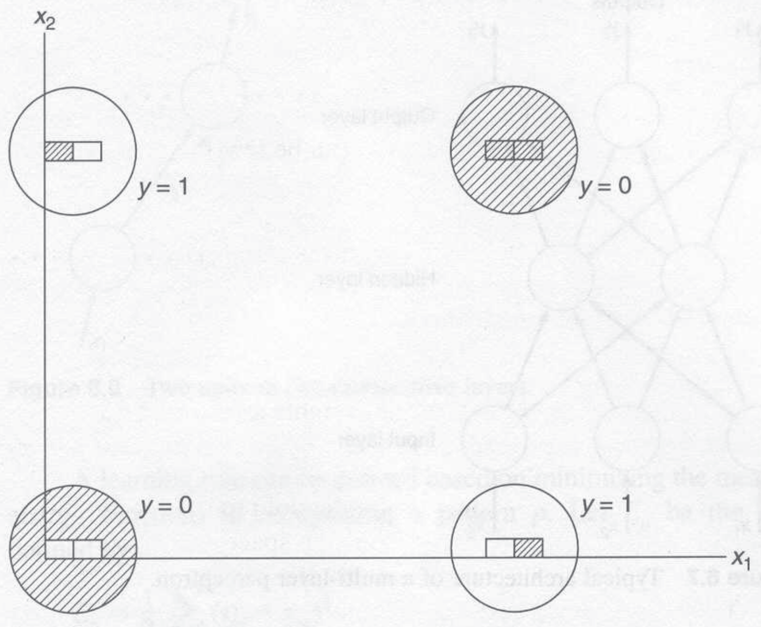


Figure 6.6 The XOR problem in pattern space.

A line L can be found such that the space is divided into two halves, one for class I (giving output $y = 1$) and one for class II (giving output $y = 0$).

The objective of a training algorithm is to adjust the weights so that the perceptron produces that line. Obviously, not all patterns are linearly separable. A simple classical example is the case of the exclusive-OR (XOR) logic function. An output of one represents the pattern 01 or 10, while an output of zero represents the patterns 00 or 11. The pattern space is illustrated in Figure 6.6. There is no linear division of space to produce the classification required. The perceptron cannot solve the problem.

It is sometimes useful to view the operation of the neural processing element as a non-linear mapping operation. The mapping converts the input vector **X** to a scalar y. The mapping can be thought to occur in two steps: a linear weighted mapping of **X** to u, and a non-linear mapping of u to y through the function f.

$$\begin{aligned} y &= f(u) \\ &= f(\mathbf{W} \cdot \mathbf{X}) \end{aligned}$$

An operation other than the dot product may be used depending on how the similarity between **W** and **X** is measured. So in general, $y = f(\mathbf{W} \circledcirc \mathbf{X})$, where © is confluence operation (scalar or distance measure).

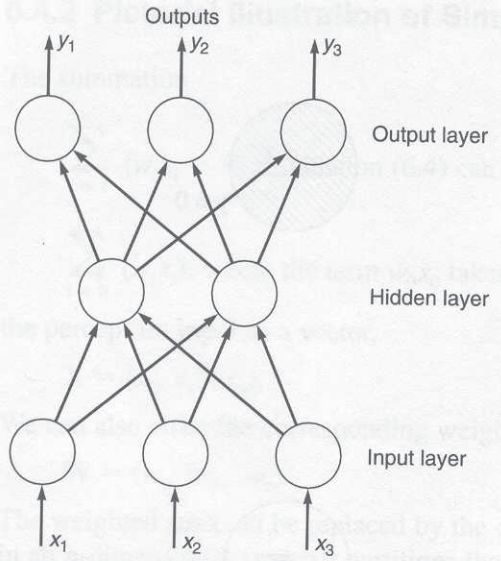


Figure 6.7 Typical architecture of a multi-layer perceptron.

6.4.3 The Multilayer Perceptron and Back Propagation

To overcome the difficulty of classifying linearly inseparable patterns, a multi-layer perceptron is used. A typical architecture is shown in Figure 6.7.

The new model has three layers: an input layer, an output layer, and a layer in between referred to as a hidden layer. In the literature, some authors do not count the input layer as one of the layers in the architecture because the neurons of the input layer do not learn (and hence they refer to Figure 6.7 as a two-layer perceptron). The architecture shown in Figure 6.7 is referred to, in general, as layered feed-forward network. The neural structure of the brain is generally multi-layer, and has a mostly feed-forward structure.

Suppose the output of node *i*, y_i, is to be sent to node *j*. Then the input to node *j* will be

$$x_{\mathrm{j}} = \sum_i (w_{ij} y_i), \tag{6.9}$$

where w_{ij} is the weight between nodes *i* and *j*. The output of node *j* is y_j, given by

$$y_{\mathrm{j}} = f_{\mathrm{j}} \sum_i (w_{ij} y_{pi}), \tag{6.10}$$

where f_j is the activation function of layer *j*. Figure 6.8 illustrates the relation between the units in two consecutive layers.

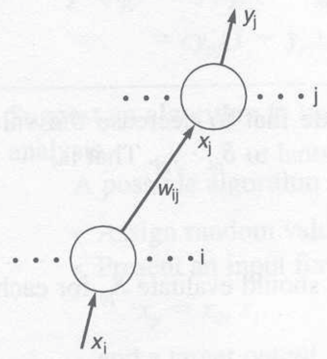


Figure 6.8 Two units in two consecutive layers.

A learning rule can be defined based on minimizing the mean-square error (or energy function) in recognizing a pattern p. Let E_p be the measure of error defined by

$$E_p = \frac{1}{2}\sum_j (t_{pj} - y_{pj})^2 \tag{6.11}$$

where t_{pj} is the target output from node j for pattern p and y_{pj} is the actual output. Now, we can write the change in the error as a function of the change in weight between two units in consecutive layers using the chain rule as

$$\frac{\partial E_p}{\partial w_{ij}} = \frac{\partial E_p}{\partial x_{pj}} \cdot \frac{\partial x_{pj}}{\partial w_{ij}} \tag{6.12}$$

We can define the change in error as a function of the change in inputs to a unit as

$$-\frac{\partial E_p}{\partial x_{pj}} = \delta_{pj}. \tag{6.13}$$

The second term of equation (6.12) can be written as

$$\frac{\partial x_{pj}}{\partial w_{ij}} = \frac{\partial}{\partial w_{ij}}\Big(\sum_k (w_{kj} \cdot y_{pk})\Big)$$
$$= \sum_k \Big(\frac{\partial w_{kj}}{\partial w_{ij}} \cdot y_{pk}\Big). \tag{6.14}$$

Since $\frac{\partial w_{kj}}{\partial w_{ij}} = 0$ except when $k = i$, then

$$\frac{\partial x_{pj}}{\partial w_{ij}} = y_{pi}. \tag{6.15}$$

From equations (6.13) and (6.15) we get

$$-\frac{\partial E_p}{\partial w_{ij}} = \delta_{pj} y_{pi}\,. \tag{6.16}$$

From equation (6.16), one may conclude that to decrease the value of E_p, the weight changes, $\Delta_p w_{ij}$, are to be proportional to $\delta_{pj} \cdot y_{pi}$. That is,

$$\Delta_p w_{ij} = \eta \delta_{pj} y_{pi} \tag{6.17}$$

where η is a constant.

In order to use equation (6.17), we should evaluate δ_{pj} for each of the units. We can use the chain rule to write

$$\delta_{pj} = -\frac{\partial E_p}{\partial x_{pj}} = -\frac{\partial E_p}{\partial y_{pj}} \cdot \frac{\partial y_{pj}}{\partial x_{pj}}. \tag{6.18}$$

Since $y_{pj} = f_j(x_{pj})$, then

$$\frac{\partial y_{pj}}{\partial x_{pj}} = f'_j(x_{pj}). \tag{6.19}$$

Also, since $E_p = \frac{1}{2}\sum_j (t_{pj} - y_{pj})^2$ then

$$\frac{\partial E_p}{\partial y_{pj}} = -(t_{pj} - y_{pj}). \tag{6.20}$$

Thus, we get

$$\delta_{pj} = f'_j(x_{pj})\,(t_{pj} - y_{pj}). \tag{6.21}$$

Equation (6.21) is useful for the output units, since the target outputs, t_{pj}, and the actual output, y_{pj}, are available. The same does not apply to units in the hidden layer, since the target output is not known. If j is not an output unit, we can modify equation (6.21) as follows:

$$\begin{aligned}\delta_{pj} &= \frac{\partial E_p}{\partial y_{pj}} = \sum_k \left(\frac{\partial E_p}{\partial x_{pk}} \cdot \frac{\partial x_{pk}}{\partial y_{pj}}\right)\\ &= \sum_k \frac{\partial E_p}{\partial x_{pk}} \cdot \frac{\partial}{\partial y_{pj}}\left(\sum w_{ik} y_{pi}\right)\\ &= -\sum_k \delta_{pk} w_{jk}\end{aligned} \tag{6.22}$$

This leads to

$$\delta_{pj} = f'_j(x_{pj}) \sum_k (\delta_{pk} w_{jk}). \tag{6.23}$$

If $f_j(x_{pj})$ is selected as the sigmoid function given by equation (6.5), its derivative is easy to evaluate, as illustrated earlier, leading to

$$f'(x_{pj}) = cf(x_{pj})[1 - f(x_{pj})]$$
$$= cy_{pj}(1 - y_{pj}). \tag{6.24}$$

EXAMPLE: Suggest an algorithm to train a multilayer perceptron network using the previous analysis.

A possible algorithm can be as follows.

- Assign random values to weights and thresholds.
- Present an input for pattern p,

  $$x_p = x_0, x_1, \ldots$$

  and a target output

  $$T_p = t_0, t_1, \ldots$$

- Calculate the actual output for each layer and present it as an input to the next layer.

  $$y_{pj} = f \sum_i w_i x_i$$

- Adjust the weights starting from the output layer,

  $$w_{ij}(t+1) = w_{ij}(t) + \eta \delta_{pj} x_{pj},$$

  using the appropriate expression for δ_{pj} (either equation (6.24) or equation (6.23).

The algorithm of the previous example is referred to as error back-propagation or just back-propagation. The concept of back-propagation to train multilayer networks was proposed independently by several authors [41–44]. There are several variations of the above algorithm that use the concept of back-propagation. Back-propagation is a widely used method to train neural networks.

The training time depends on the convergence rate, or how fast the network error approaches zero. Learning difficulty may occur. To see a possible difficulty, suppose we consider a network in which we can change just one weight. The error E vs. the weight w may be as shown in Figure 6.9.

If we have two variable weights, instead of one, we need to add a third axis to Figure 6.9, leading to a three-dimensional landscape. In general, n variable weights lead to an n-dimensional landscape. A complex energy surface leads to slower learning. The objective of back-propagation is to adjust the weights so that E is minimized. Each minimum (called a basin of attraction in an n-dimensional surface) represents a solution. A difficulty occurs if the network settles into a local minimum rather than a global minimum (B, for example, rather than A in Figure 6.9). The network stops learning.

A local minimum occurs when two or more disjoint classes are categorized as being the same. This difficulty may be avoided by using clearly distinguishable training examples. Adding more hidden layers reduces the occurrence of local minima.

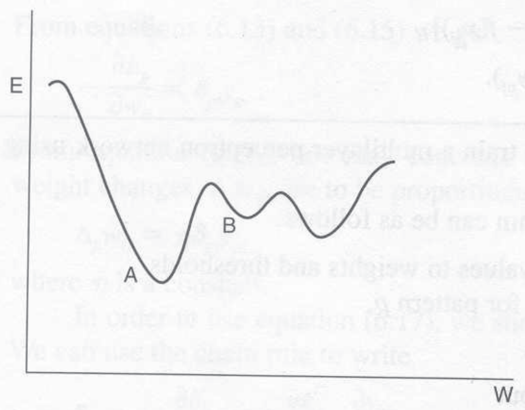


Figure 6.9 One-dimensional error vs. weight.

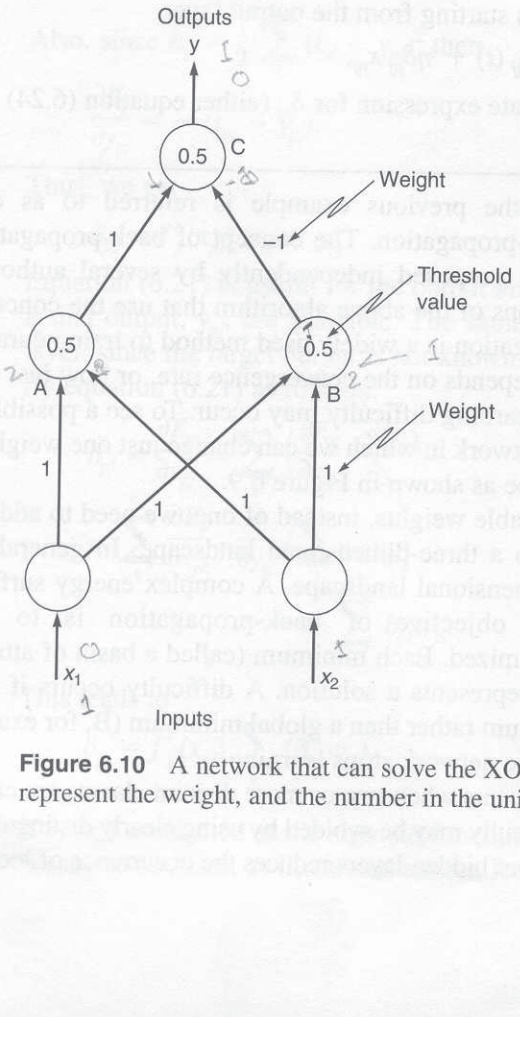


Figure 6.10 A network that can solve the XOR problem (the numbers on the arrows represent the weight, and the number in the units represents the threshold value).

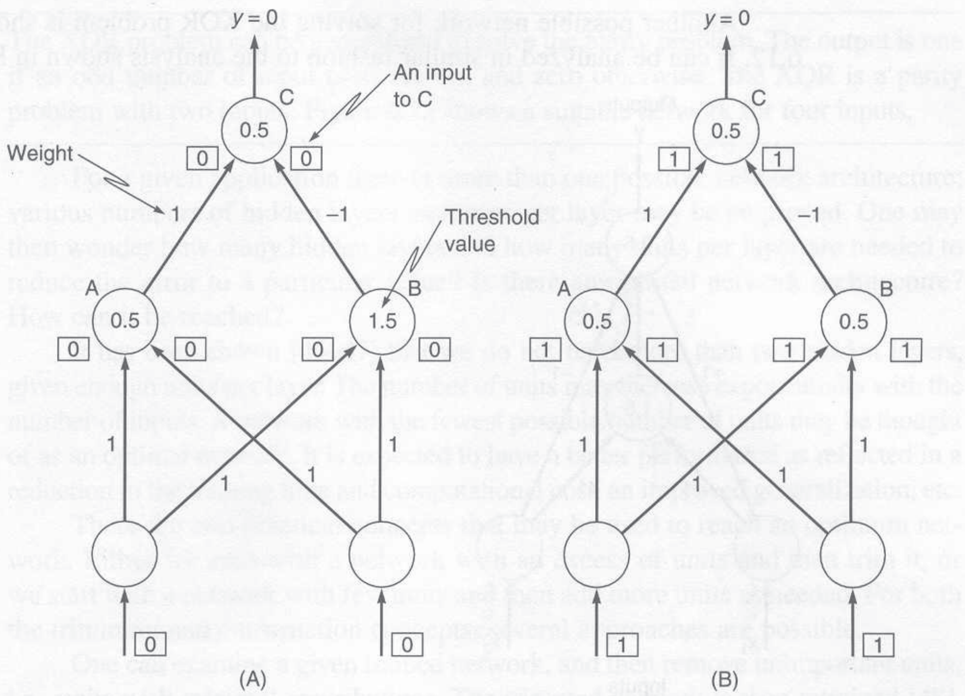


Figure 6.11 The response of the network of Figure 6.10 to various input patterns (numbers in the squares are inputs to the units).

The network may be made to escape settling in a local minimum by adding random values to the weights. This can have the effect of moving the network to some random position away from the local minimum.

A multilayer network that can solve the XOR problem is shown in Figure 6.10.

EXAMPLE: Show that the network of Figure 6.10 solves the XOR problem.

The output is to be zero for patterns 00 and 11. The output is to be one for patterns 01 and 10.

In Figure 6.11-a, the inputs x_1 and x_2 are 00.

It follows that the input to unit A is $0 \times 1 + 0 \times 1 = 0$; the same applies to unit B. Since the inputs in both cases are below the thresholds, 0.5 and 1.5, respectively, the output of units A and B will be zero and the inputs to unit C will be zero, which is below the threshold of 0.5. This leads to an output of $y = 0$. Similar reasoning applies to Figure 6.11-b and Figure 6.11-c.

Another possible network for solving the XOR problem is shown in Figure 6.12. It can be analyzed in similar fashion to the analysis shown in Figure 6.11.

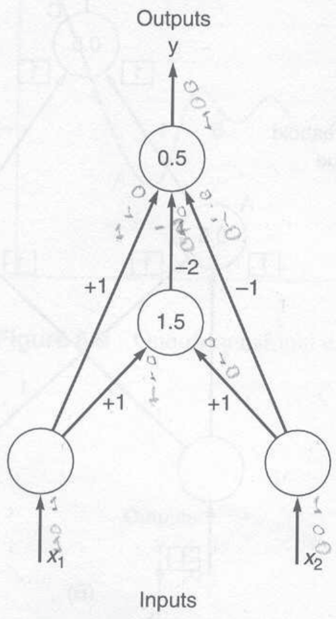


Figure 6.12 A network for solving the XOR problem that has one unit in the hidden layer.

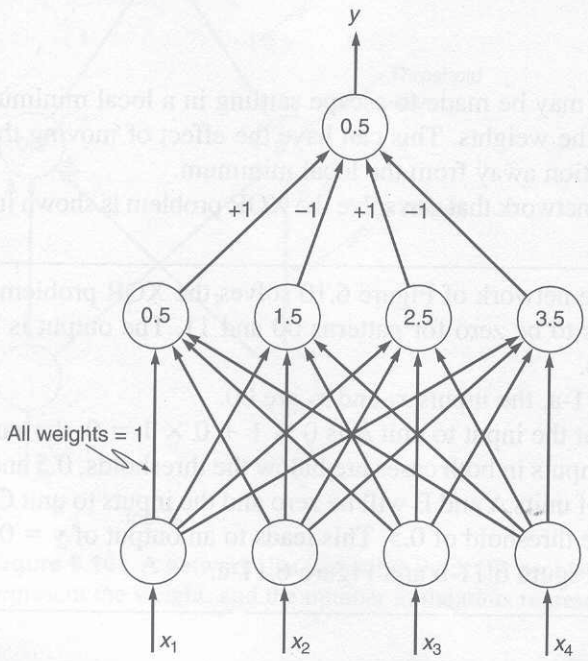


Figure 6.13 A network to solve the parity problem for four inputs.

EXAMPLE The XOR problem can be generalized to solve the parity problem. The output is one if an odd number of input units are on, and zero otherwise. The XOR is a parity problem with two inputs. Figure 6.13 shows a suitable network for four inputs.

For a given application there is more than one possible network architecture; various numbers of hidden layers and units per layer may be employed. One may then wonder how many hidden layers and how many units per layer are needed to reduce the error to a particular value? Is there an optimal network architecture? How can it be reached?

It has been shown [45–47] that we do not need more than two hidden layers, given enough units per layer. The number of units may increase exponentially with the number of inputs. A network with the fewest possible number of units may be thought of as an optimal network. It is expected to have a better performance as reflected in a reduction in the training time and computational cost, an improved generalization, etc.

There are two practical concepts that may be used to reach an optimum network. Either we start with a network with an excess of units and then trim it, or we start with a network with few units and then add more units as needed. For both the trimming and construction concepts, several approaches are possible.

One can examine a given trained network, and then remove unimportant units, i.e., units with minimal contributions. The trimmed network is then retrained [48]. Another approach is to modify the training algorithm so that each weight is allowed to decay to zero. Thus the network, during training, optimizes itself by removing unnecessary connections. Unneeded units can also be removed [49, 50].

On the other hand, several approaches have been suggested to construct an optimum network starting with fewer units. For example, the tiling algorithm [51] builds a network by starting with several units at the bottom of the architecture, then adding layers. Each layer has fewer units than the previous one. Eventually, the process comes to an end with a single output unit. The upstart algorithm [52] builds the network unit by unit, starting with a unit (it becomes the top unit of the architecture) with all inputs connected to it. All cases of wrong output are noted and a layer with two units is added (if needed), with one unit to correct the wrong ON cases and the other to correct the wrong OFF cases. The connections of the new units to the inputs are given large weights to override the previous output. Layers may be added in a similar fashion if necessary. The resulting architecture can then be converted to another architecture with a more conventional appearance. The single hidden-layer algorithm [53] builds a network using a single hidden layer. Hidden units are added one by one, each separating one or more of the target patterns.

6.4.4 Recurrent Networks

In the previous section we presented examples of supervised learning in feedforward networks, i.e., networks that only allow unilateral connections between units

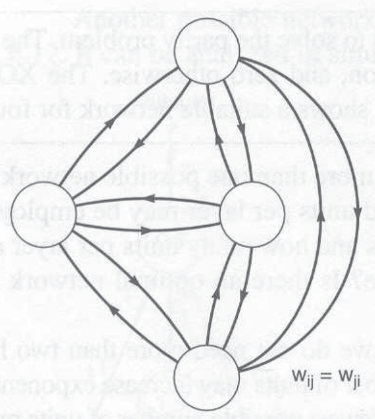


Figure 6.14 An example of Hopfield architecture with four nodes (processing units).

from inputs toward outputs. In this section we still present examples of supervised learning, but in networks that allow bilateral connections between units, i.e., recurrent networks. We will describe Hopfield networks and Boltzmann machine networks.

A. Hopfield Networks

The architecture of a Hopfield network [54] is shown in Figure 6.14. All nodes are connected bilaterally to one another. The network can be used as an associative memory, or to solve an optimization problem, or as content-addressable memory. The network is most suitable when exact binary (1 and −1 or 1 and 0) representations are possible, for example, with black and white images where input elements are pixel values, or with ASCII text where input values are sets of bits [55].

The network can be trained to recognize a set of patterns. Then when presented with a corrupt pattern it can recognize it in spite of the change in its appearance. Figure 6.15 shows a set of patterns that can be used in training the network and Figure 6.16 shows a pattern (L) corrupted to various degrees.

The network is considered to have a state determined by each individual unit. Now when a trained network is presented with a corrupted pattern of one of the images it was trained to recognize and then left alone, the state of the network keeps changing until it reaches a stable steady-state. The node outputs after convergence give the pattern as the network remembers it.

The number of patterns that the network can memorize and recall accurately is very limited. Hopfield [54] showed that the number of classes to be stored in the network should be less than 15% of the number of nodes in the network. For example, to store 10 classes, more than 70 nodes will be needed. If we try to store more than that, the network may have faulty recollection and spurious output patterns may occur, i.e., the network may respond with a pattern that is not from the training

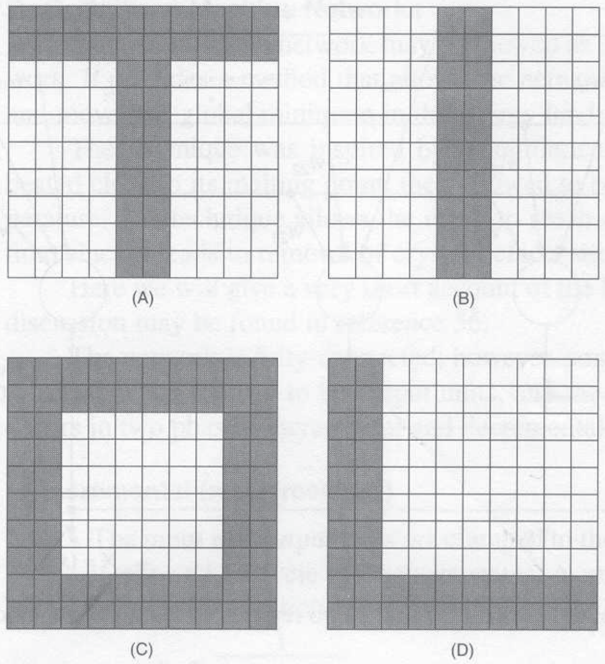


Figure 6.15 A set of training patterns.

patterns. Problems may also occur if the training patterns share many bits. In this case, they will not be suitable for training and cannot be stored and recalled accurately.

An alternative view of a Hopfield network is shown in Figure 6.17. The network is rearranged to represent n nodes systematically, making its implementation easier.

The operation of a Hopfield network can be divided into two stages, the learning stage and the recognition stage.

1) Learning stage. During this stage exemplar patterns are presented to the network. The connection weights are assigned so that the network associates each

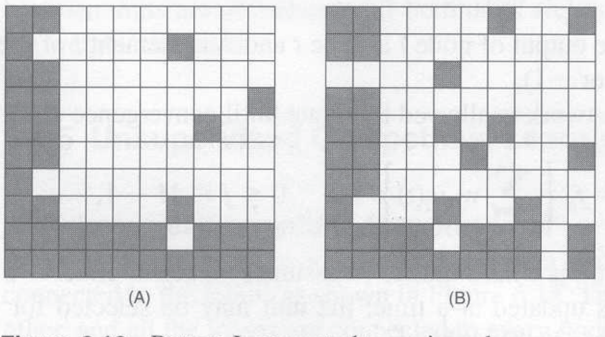


Figure 6.16 Pattern L corrupted to various degrees.

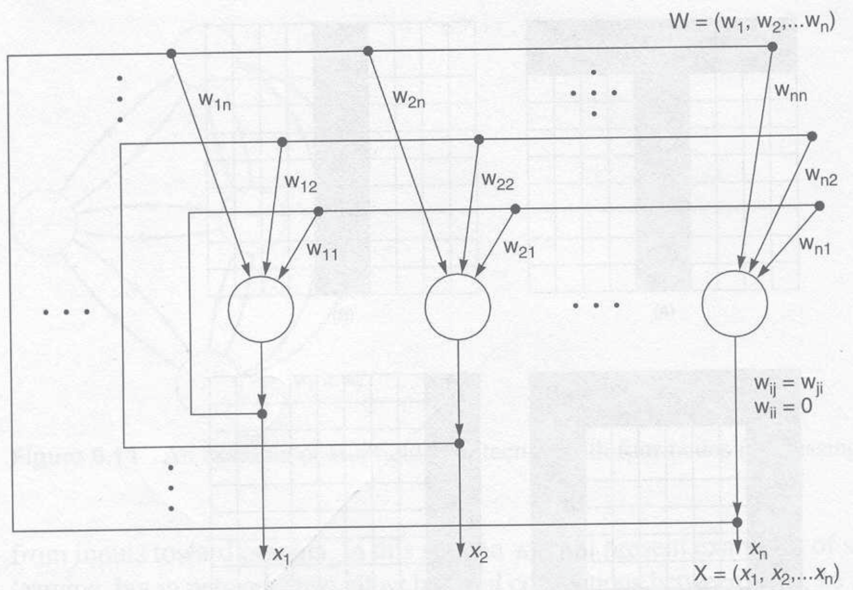


Figure 6.17 Hopfield network rearranged to represent n nodes systematically.

pattern from the exemplar with itself. This is done according to

$$w_{ij} = \begin{cases} \sum_{s=0}^{M-1} x^s{}_i\, x^s{}_j & \text{for } i \neq j \\ 0 & \text{for } i = j \end{cases}$$

where w_{ij} is the connection weight from unit i to unit j, $x^s{}_i$ is element i of the exemplar pattern for class s($x^s{}_i$ is either $+1$ or -1), and M is the number of patterns.

2) Recognition Stage. An unknown pattern is presented at time zero,

$$\mu_i(0) = x_i \qquad i = 0, 1, 2, ...,$$

where $\mu_i(t)$ is the output of node i at time t and x_i is element i of the input pattern (x_i is either $+1$ or -1).

Then the network is allowed to iterate until convergence according to

$$\mu_j(i+1) = f_h\left\{\sum_i w_{ij}\mu_i(t)\right\} \qquad 0 \leq j \leq M-1,$$

where the function is a hard limiter (it assumes values of either $+1$ or -1).

One unit is updated at a time; the unit may be selected for updating randomly so that all units have the same average updating rate.

B. Boltzmann Machine Networks

A Boltzmann machine network may be viewed as an extension of a Hopfield network. It provides a method that allows the network to escape from local minima and move to a global minimum in the energy landscape.

The technique was inspired by metallurgical annealing, where a metal is heated close to its melting point, then allowed to cool slowly down to room temperature. The technique allows the metal to reach a stable low-energy configuration, since it leads to removal of crystal defects such as dislocations.

Here we will give a very short account of the Boltzmann machine; a detailed discussion may be found in reference 56.

The network is fully-connected; however, some units are chosen, arbitrarily, to be input units, some to be output units, and the rest as hidden units. Learning occurs in two phases, incremental and decremental.

1) Incremental (reinforcement)

- The input and output units are clamped to their correct values. The network is allowed to cycle through its states. A unit is selected randomly and a probability function is calculated. For the *i*th unit,

$$p_i = \frac{1}{1 + e^{-\Delta E_i/T}},$$

where $\Delta E_i = \sum_j w_{ij} y_j$ can be viewed as an activation energy.

- A random number generator (which uses a uniform probability density function) is then used to generate a random value, ξ, between 0.0 and 1.0. If $\xi \leq p$, the unit sets its output to $+1$; otherwise, it sets its output to -1.
- The temperature parameter, T, is decremented until the output is stable.
- The weight between two units is incremented if they are both on.

2) Decremental (forget bad associations) In this phase, only the inputs are clamped. The network is allowed to reach thermal equilibrium again. The weights between units are decremented if both units are on. The process is repeated until the weights are stable.

6.4.5 Unsupervised Competitive Learning

A. Kohonen Self-organizing Networks

A typical Kohonen network is composed of a two-dimensional layer (flat grid) connected to the inputs as shown in Figure 6.18. The nodes are connected to each other, and all the inputs are connected to every node. Each node in the grid represents an output node.

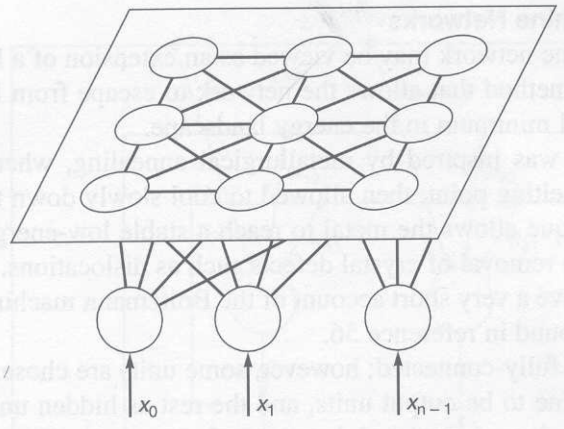


Figure 6.18 A typical Kohonen's network.

The objective is to convert this random set of nodes into subsets of organized nodes. Each subset recognizes an input pattern. The Kohonen algorithm may be described as follows:

- Start with small random values for the weights $w_{ij}(t)$ (connecting n inputs to m output nodes). Assign a large value for the neighborhood size NE(t).
- Present an input $x_0(t), x_1(t), \ldots x_n(t)$.
- Calculate the distance d_j in n-dimensional space between the output vector and the weight vector. The node with the smallest distance has the closest association to the training input.

$$d_j = \sum_{i=0}^{n-1} (x_i(t) - w_{ij}(t))^2$$

- Select the output node with the minimum distance, and give it the designation j^* (the winning neuron).
- Update weights to node j^* and its neighbors; the training law used is

$$w_{ij}(t+1) = w_{ij}(t) + \eta(t)(x_i(t) - w_{ij}(t)),$$

where $j \in \text{NE}_{j^*}(t)$, $0 \leq i \leq n-1$, and the term $\eta(t)$ is a gain term that represents the rate of learning $(0 < \eta(t) < 1)$ that decreases with time.

The neighborhood $\text{NE}_{j^*}(t)$ decreases with time, thus localizing the area of maximum activity.

- The process is repeated, and the weights eventually converge and are fixed after the gain term is reduced to zero.

B. Adaptive Resonance Theory (ART) Networks

When people learn, they can add more to what they have learned without the need for complete retraining and without forgetting all that they have already learned. In electronic networks, however (in Kohonen's network, for example), a catastrophic forgetting may occur if we try to add more patterns to a stable trained network. There is a trade-off between the stability and the ability to learn new data. This is known as the stability-plasticity dilemma. The so-called ART networks were developed [57–60] to overcome this dilemma, i.e., to produce a stable network that can learn new data. The key idea is to spare some of the output units for new patterns. If the input and a stored pattern are sufficiently similar, they are said to be resonating. If there is not sufficient similarity, a new class of patterns is formed utilizing the unused output units. The network gives no response if all output units are used.

The ART network has been implemented in several versions: ART1, ART2, ART3, and ARTMAP [61], which is a supervised version of ART. ART1 can stably learn to categorize binary input patterns presented in an arbitrary order; ART2 does the same thing with either analog or binary input patterns; ART3 can carry out parallel search, or hypothesis testing, or distributed recognition codes in a multilevel network hierarchy. ARTMAP is a supervised version that can rapidly self-organize stable categorical mappings between n-dimensional inputs and m-dimensional output vectors. The following outline relates to ART1.

The network structure is shown in Figure 6.19. The network has an input (comparison) layer and an output (recognition) layer. The units in the output layer are connected so that lateral inhibition can occur, i.e., units with larger output can suppress other units. A feedforward and feedback connection exists between every unit in the input and output layers. Each layer has a logic control, designated control-1 and control-2, connected to each unit in the layer. Control-1 is one whenever a valid input is present, and forced to zero if any unit in the output layer is active. Control-2 will be one for any valid input pattern, and goes to zero after a failed vigilance test. A vigilance test evaluates the similarity of an input to a stored exemplar by finalizing the ration count of the number of ones in each. It controls the generation of new class patterns. The stored exemplar weight vectors are sometimes called long term memory, as opposed to the transient states, which are called short-term memory.

The operation of the network may be summarized as follows.

1) The initialization phase

- The feedback weights are set to 1, $t_{ij}(0) = 1$.
- The feedforward weights are set such that

$$w_{ij}(0) = \frac{1}{1 + N},$$

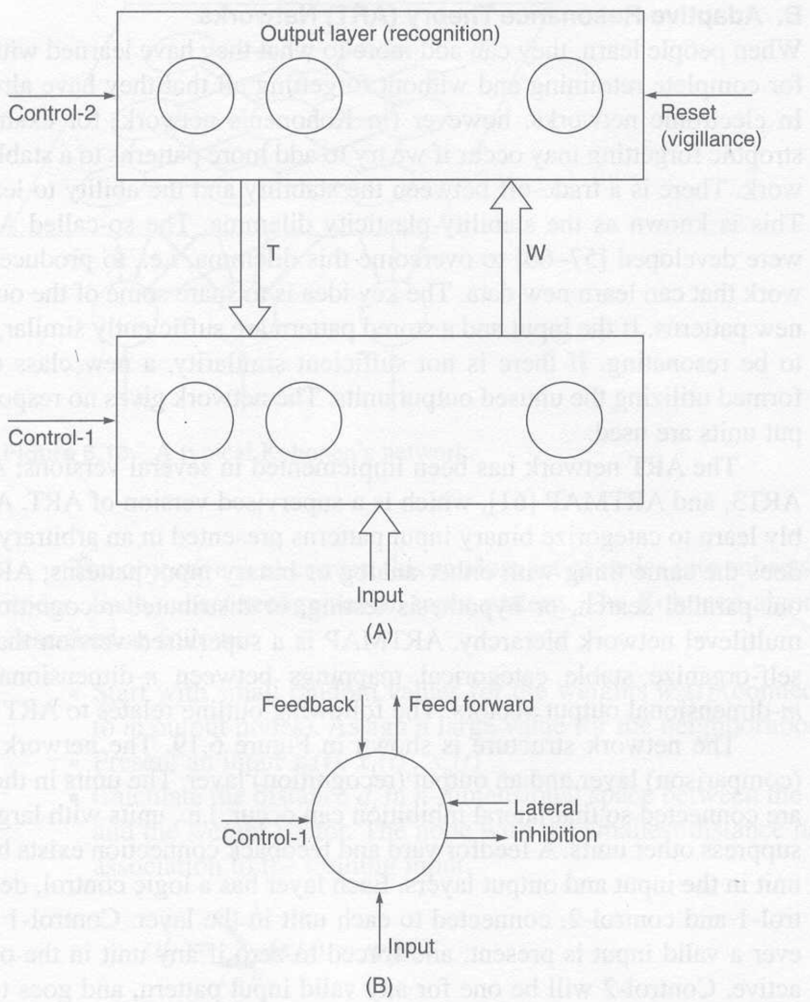


Figure 6.19 a) A schematic of ART network. b) A unit in the input layer.

$$0 \le i \le N - 1, 0 \le j \le M - 1$$

where N is the number of input nodes, and M is the number of output nodes.

- The vigilance threshold, ρ, is given a value such that $0 \le \rho \le 1$. ρ controls the resolution of the classification.

2) The Recognition Phase. A new input is applied and the matching is computed.

$$\mu_j = \sum_{i=0}^{N-1} w_{ij}(t)x_i, \; 0 \le j \le M - 1$$

The largest output is designated μ_{j*} and found from

$$\mu_{j*} = \max_j [\mu_j].$$

3) The Comparison Phase. The winning unit feeds back its pattern by adjusting the weights T_{ij} to the input. The ratio S is calculated by

$$S = \frac{\Sigma t_{ij} x_i}{\Sigma x_i}.$$

If $S > \rho$, then the classification is complete.
If $S < \rho$, then a search phase starts.

4) The Search Phase. First the present active output unit is zeroed and control-1 is thus forced to zero (all outputs are zero now). Then, the input is reapplied; the recognition and comparison phases are repeated. The process goes on until $S > \rho$. If no classification is possible, the input is declared unknown and is allocated to a previously unassigned unit in the output layer.

6.5 VLSI Implementation of Neural Networks

VLSI implementation has been extensively discussed in the literature (see for example reference 29 and references 62–70). The advantage of using hardware to realize neural functions is the higher speed and density on one hand and the lower power dissipation and cost on the other compared to simulations running on conventional computers. In general, neural hardware can be tailored for the needs of an application that requires a single neural network of fixed size and topology as well as for an application that requires a set of different neural networks. It can also be designed for general purpose neurocomputing.

There are two general approaches to VLSI implementation. The first has the objective of solving the neural system equations using either standard or custom VLSI chips. The second approach is not very much concerned with attempting to replicate the biological system but rather with the design of a fault-tolerant adaptive VLSI chip guided by the concepts of biological systems. The algorithms used for software implementation may not be directly usable for VLSI implementation. The neuron transfer function is an exact mathematical function in software implementation, while it is an approximate function in VLSI implementation.

Although analog computation may have its advantages, digital techniques are more advantageous in communication. Analog computation can also be achieved using digital techniques.

The need for information storage for less than 1 s can be satisfied with a simple capacitor. Larger storage times require periodic restorations of charge, as in

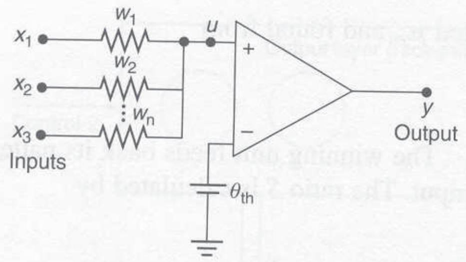


Figure 6.20 A possible VLSI model of a neuron

dynamic RAMS. Floating polysilicon gate technology, which is used in EPROM and EEROM, may be suitable for very long storage times.

There are several limitations on VLSI implementation including the chip area, the number of package pins, and the number of required connections.

In the following we give some illustrative examples of using amplifiers, resistors, and capacitors to model a neuron and neural network.

EXAMPLE: **VLSI model of a neuron**

Figure 6.20 shows a circuit that can be used to simulate the action of a neuron. The resistors associated with each input (with conductances w_1, w_2, ... w_n) simulate the synaptic weights the neuron offset. As mentioned earlier, the transfer function of the neuron in this realization is less flexible than in software implementation. Also, the synaptic weights are less flexible; they have positive values only. Two amplifiers are needed to simulate the effects of excitory and inhibitory inputs.

The net input of the amplifier can be determined by

$$(x_1 - u) + (x_2 - u)w_2 + \dots + (x_n - u)w_2 = u \cdot \frac{1}{Req.},$$

where x_1, x_2, ... x_n are the input voltages, and u is the input to the amplifier.

w_1, w_2, ...w_n are the conductance values representing the weight of each input, and *R*eq. is the input resistance of the amplifier. Assuming *R*eq. to be very large, we get

$$u = \frac{\sum_{i=1}^{n}(w_i x_i)}{\sum_{i=1}^{n} w_i}.$$

The output voltage, y, is then given by

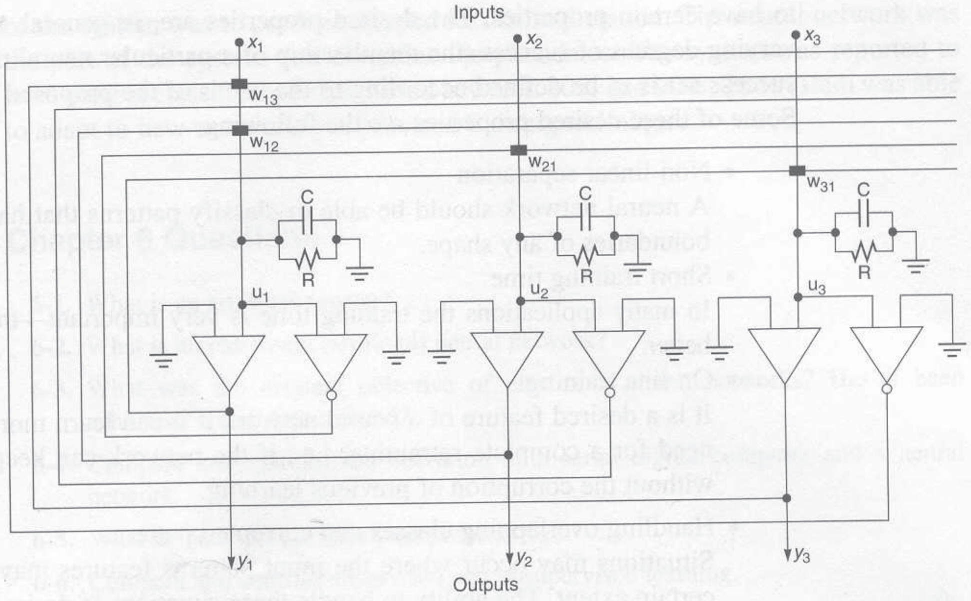


Figure 6.21 An implementation of a Hopfield network.

$$y = A\left\{\frac{\sum_{i=1}^{n}(w_i xi)}{\sum_{i=1}^{n} w_i} - \theta\right\}$$

where A is the amplifier gain and θ is the offset voltage.

EXAMPLE: **VLSI Hopfield neural network**

Figure 6.21 shows an implementation of a Hopfield network. Each neuron is represented by inverting and noninverting amplifiers so that it can handle both positive and negative values of weights. A densely connected resistive network is used for feedback; the network is symmetric and has no self-feedback connections.

A comprehensive survey and compilation of VLSI implementation of neural networks reported in the literature up to the year 1994 is given in reference [79].

6.6 Concluding Remarks

In the previous sections of this Chapter we discussed a few neural network systems with varied properties. For practical applications, neural networks are desired to have certain properties. The desired properties are, in general, achieved with varying degrees of success (the membership of a particular neural network in the success set is to be defined according to the needs of the proposed application!). Some of these desired properties are the following:

- Non-linear separation
  A neural network should be able to classify patterns that have separation boundaries of any shape.
- Short training time
  In many applications the training time is very important—the shorter, the better.
- On-line training
  It is a desired feature of a neural network if it can learn more without the need for a complete retraining, i.e., if the network can keep on learning without the corruption of previous learning.
- Handling overlapping classes and corrupted patterns
  Situations may occur where the input patterns features may overlap to a certain extent. The ability to handle these situations is desired.
- Accepting linguistic inputs
  In some cases, it is more convenient to have a linguistic description of the input rather than exact numerical values. (The incorporation of fuzzy logic becomes a must in these cases).

Our objective in the previous sections was to pave the way to understanding fuzzy-neural systems. The discussion was mainly concerned with neural networks concepts rather than their applications. A curious reader may wonder about the practical applications of neural networks on their own (without fuzziness). Such applications have been discussed extensively in the literature (see for example references [71–78]). One of the early applications was NET Talk. It is a hierarchical back-propagation neural system [76]. It is composed of three fully-connected layers. The objective was to read aloud a printed English text. It was the first application of its kind using neural networks. The input to the network was a printed text that stepped in front of the input sensors one letter at a time. The output was a phoneme code, which was directed to a speech generator to give the pronunciation of the letter at the center of the network's field of view. The field of view consisted of seven letters. The extra letters provided contextual information to help the network in choosing the best pronunciation. The network was trained using 1024 words from a side-by-side English text/phoneme source. After 50 training epochs an accuracy of 95% was achieved. The network was shown to degrade gracefully with rapid recovery upon retraining.

Another classical application example is the phonetic typewriter [77, 78]. It is a system that uses a Kohonn network as one of its building units. The objective of the system was to produce a typed text from dictation. The neural network was the part of the system that classifies the phonemes. The system was reported to have correct classification that varied between 80% and 90%. The system was able to adapt to new users using supervised training techniques.

Chapter 6 Questions

6-1. What is an artificial neuron?

6-2. What is an electronic (artificial) neural network?

6-3. What was the original objective of electronic neural networks? Has it been achieved? Can it be achieved?

6-4. Compare and contrast the operation of a serial digital computer and a neural network.

6-5. What is the XOR classification problem?

6-6. Compare and contrast supervised and unsupervised learning.

6-7. Show that the following functions have the derivatives indicated ($y' = \frac{dy}{dx}$, $c =$ constant). What can they be used for in neural networks?

a) $y = \tanh(cx)$; $\quad y' = c(1 - y^2)$

b) $y = e^{-cx2}$; $\quad y' = -2cxe^{-x^2}$

c) $y = \dfrac{x^n}{c + x^n}$; $\quad y' = \dfrac{cnx^{n-1}}{(c + x^n)^2}$

6-8. Sketch the function $y = 1/(1 + ce^{-cx})$ for $c = 0.5$, 1, 5, and 10. Which value of c makes the function closer to a hard limiter?

6-9. A neuron has two inputs, $x_1 = 1$, and $x_2 = 1$. The corresponding weights are $w_1 = -3$ and $w_2 = 2$. The threshold, $\theta = 1$. The transfer function is given by $y = 1/(1 + e^{-cx})$.

a) Calculate $x = \sum_i^n (x_i w_i) - \theta$.

b) Calculate the output, y, for $c = 0.5$ and $c = 10$.

6-10. Give a simple analog circuit that simulates a neuron.

6-11. State some of the advantages of using simple analog circuits in simulating neural networks.

6-12. State some of the features that one may consider when designing a neural network.

***6-13.** Lewenstein and Olko (*Network* 2 (1991): 207–230) put forward a new class of neural network models; they referred to it as Quantum Neural Networks. Describe qualitatively their approach.

***6-14.** Select one of the following applications of neural networks and discuss it briefly.

a) Manufacturing ICs (G.S. May, *IEEE Spectrum* (September 1994): 47–51.)

b) Audio Synthesizer (M. Thorson, *Dr. Dobb's Journal* (February 1993).
c) Character Recognition (K. Karnofsky, *Dr. Dobb's Journal* (June 1993).
d) Control of Automative Fuel-Injection System (M. Majors et al., *IEEE Control Systems* (June 1994): 31–36.)

***6-15.** Obtain, for example using the Internet, a list of recent neural products (hardware and software).

6-16. Mark the following statements as true or false; correct the wrong statements.

1) The brain may be viewed as a parallel, distributed processing system.
2) Dendrites carry the output from the neuron.
3) All learning laws are biologically justified.
4) Hebian learning implies increasing effectiveness of active junctions.
5) Learning corresponds to adjusting the value of the weight of the network.
6) Perceptrons can solve the XOR problem.
7) XOR is an example of a linearly separable problem.
8) AND is an example of a linearly separable problem.
9) The axon can be modeled as if it were a transmission line.
10) The training set is usually the same as the test set of a neural network.
11) Perceptrons fail in solving linearly separable problems.
12) Learning process will always converge.
13) ART3 handles binary inputs only.
14) ART is an unsupervised competitive learning algorithm.
15) ART did not solve the stability-plasticity problem of neural networks.
16) ART2 is implemented for real and binary inputs.
17) Von Neumann architecture is used in neural networks.
18) Hopfield networks are self-organizing.
19) Hopfield networks are symmetric and fully connected.
20) A Hopfield network is a single-layer attractor network.
21) Kohonen network is a neural network algorithm that is based on statistical mechanics.
22) Basins of attraction are the lowest parts in neural network architecture.
23) The traveling salesman problem addresses the problem of selling neural networks at a good price.
24) The energy surface is n-dimensional, where n is the number of varying weights.
25) The stability-elasticity problem does not exist in multilayer perceptrons.
26) A hypercube is a cube in an n-dimensional space.
27) Artificial stupidity may have useful applications.

References

1. M.M. Gupta, "Fuzzy Logic and Neural Networks," *Tenth Conf. on Multicriterion Decision Making* (Taipei, July 19–24, 1992) 281–294.
2. J.C. Bezdek and S.K. Pal, *Fuzzy Models for Pattern Recognition* (New York: IEEE Press, 1992).

3. M.M. Gupta and D.H. Rao, "On the Principles of Fuzzy Neural Networks," *Fuzzy Sets and Systems* 61 (1994): 1–18.
4. M.E. Cohen and D.L. Hudson, "An Expert System on Neural Network Techniques," in *The Proc. of NAFIP*, ed. I. B. Turksen (Toronto, June 1990) 117–120.
5. M.M. Gupta and G.K. Knopf, "Fuzzy Neural Network Approach to Control Systems," in *Proc. of First Int. Symp. on Uncertainty Modeling and Analysis* (Maryland, Dec. 3–5, 1990) 483–488.
6. J.C. Bezdek, *Pattern Recognition with Fuzzy Objective Algorithm* (New York: Plenum Press, 1991).
7. Y.-H. Kuo, C-I Kao, and J.J. Chen, "A Fuzzy Neural Network Model and its Hardware Implementation," *IEEE Trans. Fuzzy Systems* 1 (1993): 171–183.
8. S.K. Pal and S. Mitra, "Multilayer Perceptron, Fuzzy Sets, and Classification," *IEEE Trans. Neural Networks* 3 (1992): 683–697.
9. W. Pedrycz, "Fuzzy Neural Networks with Reference Neurons as Pattern Classifiers," *IEEE Trans. Neural Networks* 3 (1992): 770–775.
10. S. Horikawa et al., "On Fuzzy Modeling Using Fuzzy Neural Networks with the Back-propagation Algorithm," *IEEE Trans. Neural Networks* 3(1992): 801–806.
11. C.-T. Lin and C.S.G. Lee, "Neural-network-based Fuzzy Logic Control and Decision System," *IEEE Trans. Comput.* 40 (1991): 1320–1336.
12. N.P. Archer and S. Wang, "Fuzzy Set Representation of Neural Network Classification Boundaries," *IEEE Trans. Sys. Man., and Cyber.* 21 (1991): 735–742.
13. J.M. Keller, R.R. Yoger, and H. Tahani, "Neural Network Implementation of Fuzzy Logic," *Fuzzy Sets and Systems* 45 (1992): 1–12.
14. J.V. de Oliveria, "Neuron Inspired Learning Rules for Fuzzy Relational Structures," *Fuzzy Sets and Systems* 57 (1993): 41–53.
15. B. Kosko, *Neural Networks and Fuzzy Systems* (Englewood Cliffs, N.J.: Prentice Hall, 1992).
16. C. H. Chen, ed., *Fuzzy Logic and Neural Network Handbook*, New York: McGraw-Hill, Inc. 1996.
17. J.J. Buckley, "Hybrid Neural Nets Can Be Fuzzy Controllers and Fuzzy Expert Systems," *Fuzzy Sets and Systems* 60 (1993): 135–142.
18. H.-M. Lee and W.-T. Wang, "A Neural Network Architecture for Classification of Fuzzy Inputs," *Fuzzy Sets and Systems* 63 (1994): 159–173.
19. L.-X. Wang and J.M. Mandel, "Generating Fuzzy Rules by Learning from Examples," *IEEE Trans. Sys., Man., and Cyber.* 22 (1992): 1414–1427.
20. S.-I. Horikawa, T. Furuhashi, and Y. Uchikawa, "On Fuzzy Modeling Using Fuzzy Neural Networks with Back-propagation Algorithm," *IEEE Trans. Neural Networks* 3 (1992): 801–806.
21. J.J. Buckley and Y. Hayashi, "Can Fuzzy Neural Nets Approximate Fuzzy Functions?", *Fuzzy Sets and Systems* 61 (1994): 43–51.
22. Q. Sang and G. Bortolan, "Some Properties of Defuzzification Neural Networks," *Fuzzy Sets and Systems* 61 (1994): 83–89.
23. M.E. Cohen and D.L. Hudson, "Approaches to the Handling of Fuzzy Input Data in Neural Networks," *IEEE Int. Conf. on Fuzzy Systems* (March 8–12, 1992) 93–100 San Diego, CA.

24. Y. Hayashi, E. Czogala, and J.J. Buckley, "Fuzzy Neural Controller," *IEEE Int. Conf. on Fuzzy Systems* (March 8–12, 1992) 197–202 San Diego, CA.
25. H. Nomura, I. Hayashi, and Wakami, "A Learning Method of Fuzzy Inference Rules by Descent Method", *IEEE Int. Conf. on Fuzzy Systems* (March 8–12, 1992) 203–210 San Diego, CA.
26. B. Soper and G. Rosenthal, "The Number of Neurons in the Brain: How We Report on What We Do Not Know," *Teaching Psychology* 15 (1988) 153–156.
27. H.K. Kimelberg, *Glial Cell Receptors* (New York: Raven Press, 1988).
28. I.B. Levitan and L.K. Kaczmarek, *The Neuron: Cell and Molecular Biology* (Oxford: Oxford University Press, 1991).
29. C. Mead, *Analog VLSI and Neural Systems* (Adison-Wesley, 1989) Reading MA.
30. A.L. Hodgkin and A.F. Huxley, "Current Carried by Sodium and Potassium Ions through the Membrane of the Giant Axion of Loligo," *Journal of Physiology* 116 (1952): 449.
31. S. Deutsch and A. Deutch, *Understanding the Nervous System* (New York, IEEE Press, 1993).
32. L.T. Benjamin, Jr., J.R. Hopkins, and J.R. Nation, *Psychology* (Toronto: Maxwell MacMillan, 19??).
33. R. Beale and T. Jackson, *Neural Computing: An Introduction* (Bristol: Adam Hilger, 1990).
34. H.B. Barlow, "Unsupervised Learning," *Neural Computing*, 1 (1989): 295–311.
35. D.O. Hebb, *The Organization of Behaviour* (New York: Wiley, 1949).
36. S. Grossberg, *Studies of Mind and Brain* (Boston: Ridel Publishing Company, 1982).
37. M. Caudill and C. Butler, *Naturally Intelligent Systems* (Cambridge: MIT Press, 1990).
38. T. Kohonen, *Self Organization and Associative Memory* (Berlin: Springer-Verlag, 1990).
39. D.E. Rumelhart, J.H. McClelland, and the PDP Research Group, *Parallel Distributed Processing*, Vol. 1 and 2 (Cambridge: MIT Press, 1986).
40. J. Diederich, "Explanation and Artificial Neural Networks," *Int. J. Man-Machine Studies* 37 (1992): 335–355.
41. A.E. Bryson and Y.-C. Ho, *Applied Optimal Control* (New York: Blaisdell, 1969).
42. P. Werbos, "Beyond Regression: New Tools for Prediction and Analysis in Behavioral Sciences" (Ph.D. Harvard University, 1974).
43. D.B. Parker, "Learning Logic," *Technical Report TR-47* (Cambridge: Center for Computational Research in Economics and Management Science, MIT, 1985).
44. D.E. Rumbelhart, G.H. Hinton, and R.J. Williams, "Learning Representations by Back-Propagation Errors," *Nature* 323 (1986): 533–536.
45. G. Cybenko, "Continuous Valued Neural Networks with Two Hidden-layers are Sufficient," (Medford, MA: Technical report, Department of Computer Science, Tufts University, 1988).
46. G. Cybenko, "Approximation by Superpositions of a Signoidal Function," *Mathematics of Control, Signals, and Systems* 2 (1989) 203–314.
47. A. Lapedes and R. Farber, "How Neural Nets Work," in *Neural Information Processing Systems*, ed. D.Z. Anderson (New York: American Institute of Physics, 1988) 442–456.
48. J. Sietsma and R.J.F. Dow, "Neural Net Pruning—Why and How," in *IEEE International Conference on Neural Networks* Vol 1 (New York: IEEE, 1988) 325–333.

49. R. Scalettar and A. Zee, "Emergence of Grandmother Memory in Feed forward Networks: Learning with Noise and Forgetfulness," in *Connectionist Models and Their Implications: Readings from Cognitive Science*, ed. D. Waltz and J.A. Feldman (Norwood: Ablex, 1988) 309–332.
50. A.H. Kramer and A. Sangiovanni-Vincentelli, "Efficient Parallel Learning Algorithms for Neural Networks," in *Advances in Neural Information Processing Systems I*, ed. D.S. Touretazky (San Mateo: Morgan Kaufmann, 1989).
51. M. Mézard and J.-P. Nadal, "Learning in Feedforward Layered Networks: The Tiling Algorithm," *Journal of Physics A* 22 (1989): 2191–2204.
52. M. Frean, "The Upstart Algorithm: A Method for Constructing and Training Feedforward Neural Networks," *Neural Computation* 2 (1987): 198–209.
53. M. Marchand, M. Golea, and P. Ruján, "A Convergence Theorem for Sequential Learning in Two-layer Perceptrons," *Europhysics Letters* 11 (1990): 487–492.
54. J.J. Hopfield, "Neural Networks and Physical Systems with Emergent Collective Computational Abilities," *Proc. Natl. Acad. Sci., USA* 79 (1982): 2254–2558.
55. R.P. Lippmann, "An Introduction to Computing with Neural Nets," *IEEE ASP Magazine* 4 (1987): 4–22.
56. E. Aarts and J. Korst, *Simulated Annealing and Boltzmann Machines* (New York: Wiley, 1989).
57. G.A. Carpenter and S. Grossberg, "A Massively Parallel Architecture for a Self-organizing Neural Pattern Recognition Machine," *Computer Vision, Graphics and Image Processing* 37(1987): 54–115.
58. G.A. Carpenter and S. Grossberg, "ART2: Self-organization of Stable Category Recognition Codes for Analog Input Patterns," *Applied Optics* 26 (1987): 4919–4930.
59. G.A. Carpenter and S. Grossberg, "The ART of Adaptive Pattern Recognition by Self-organizing Neural Network," *Computer* 3 (1988): 77–88.
60. S. Grossberg, *Neural Networks and Neural Intelligence* (Cambridge: MIT Press, 1988).
61. G.A. Carpenter, S. Grossberg, and J.H. Reynolds, "ARTMAP: Supervised Real Time Learning and Classification of Non-stationary Data by a Self-organizing Neural Network," *Neural Networks* 4 (1991): 565–588.
62. F. Faggin and C. Mead, "VLSI Implementation of Neural Network," in *An Introduction to Neural and Electronic Networks*, ed. S.F. Zornetzer, J.L. Davis, and C. Lau (New York: Academic Press, 1990) 275–292.
63. L.A. Akes, D.K. Ferry, and R.O. Grondin, "Synthetic Neural Systems in VLSI," ed. S.F. Zornetzer, J.L. Davis, and C. Lau (New York: Academic Press, 1990) 317–337.
64. U. Ramacher and U. Rückert, eds., *VLSI Design of Neural Networks* (Boston: Kluwer Academic Publishers, 1991).
65. B.W. Lee and B.J. Sheu, *Hardware Annealing in Analog VLSI Neurocomputing*, (Boston: Kluwer Academic Publishers, 1991).
66. S.W. Tsay and R.W. Newcomb, "VLSI Implementation of ART1 Memories," *IEEE Trans. Neural Networks* 2 (1991): 214–221.
67. A.G. Andreou, K.A. Boahen, P.O. Pouliquen, A. Pavasović, R.E. Jenkins, and K. Strohbehn, "Current-mode Subthreshold MOS Circuits for Analog VLSI Neural Systems," *IEEE Trans. Neural Networks* 2 (1991): 205–213.
68. W.A. Fisher, R.J. Fujimoto, and R.C. Smithson, "A Programmable Analog Neural Network Processor," *IEEE Trans. Neural Networks* 2 (1991): 222–229.

69. S.Y. Foo, L.R. Anderson, and Y. Takefuji, "Analog Components for the VLSI of Neural Networks," *IEEE Circuits and Devices Magazine* 7(1990): 18–26.

70. J.M. Zurada, "Analog Implementation of Neural Networks," *IEEE Circuits and Devices Magazine* 9 (1992): 36–41.

71. W.T. Miller, III, R.S. Sutton, and P.J. Werbos, eds., *Neural Networks for Control* (Cambridge: MIT Press, 1991).

72. B. Yuhas and N. Ansari, eds., *Neural Networks in Telecommunications* (Boston: Kluwer Academic Publishers, 1994).

73. A. Cichocki and R. Unbehauen, *Neural Networks for Optimization and Signal Processing* (Toronto: John Wiley & Sons, 1993).

74. J.A. Freeman and D.M. Skapura, *Neural Networks: Algorithms, Applications, and Programming Techniques* (New York: Addison-Wesley, 1991).

75. M.A. Lehr and B. Widrow, "Commercial and Industrial Applications of Neural Networks," *Proc. of the Fifth Workshop on Neural Networks* (San Francisco, California, 1993) 165–170.

76. T. Sejnowski and C. Rosenberg, "Parallel Networks that Learn to Pronounce English text." *Complex Systems* 1 (1987): 145–168.

77. T. Kohonen, "The Neural Phonetic Typewriter," *Computer* 21(1988): 11–22.

78. T. Kohonen, "Speech Recognition Based on Topology-preserving Neural Maps," in *Neural Computing Architecture*, ed. I. Aleksander (Cambridge: MIT Press, 1989) 26–40.

79. T.A. Doung, S.P. Eberchardt, T. Daud, and A. Thakoor, "Learning in Neural Networks: VLSIImplementation Strategies," in *Fuzzy Logic and Neural Networks Handbook*, ed., C.H. Chen, (New York: McGraw Hill, Inc. 1996. 27.1–27.48.)

7 Fuzzy Neural Networks

7.1 Introduction

Fuzzy neural networks and neural fuzzy systems are powerful techniques for various computational and control applications. The area is still under a great influx from both theoretical and applied research. There is no systematic or unified approach for incorporating the concepts of fuzziness and neural processing. The objective of this chapter is not to encompass all the existing ideas up to the present, but rather to provide a view of some representative ideas in this area so that the reader can pursue the progressing literature with ease.

It may be educational to attempt a classification along the same lines of the previous chapter, i.e., discuss fuzzy neurons [1–5], fuzzy perceptrons [6–10], fuzzy recurrent networks, fuzzy competitive learning incorporating fuzzy Kohonen's networks [11–17], fuzzy ART [18, 19], and fuzzy ARTMAP [20, 21]. Then, we can discuss fuzzy associative memory (FAM)[22], which uses fuzzy matrices instead of fuzzy neurons to represent fuzzy associations.

Other approaches to viewing the combination of fuzziness and neural networks are possible. For example, we may consider three classes (with fuzzy boundaries!); the first encompasses neural networks that are used to generate fuzzy rules and membership functions using traditional neural network architectures. The second encompasses fuzzy systems used to interpret linguistic statements for neural networks. The third class encompasses systems that do not belong to either of the previous classes. Still other approaches are possible (see for example references 23–26).

In the following sections we do not start with fuzzy neurons, since most fuzzy neural networks do not employ fuzzy neurons as such. We first present fuzzy multilayer perceptrons, fuzzy Kohonen networks, and fuzzy ART and related models. Then, we discuss a model of a fuzzy neuron and a network built around it. Finally, we give an insight into neural fuzzy control.

7.2 Fuzzy Multilayer Perceptron

Several models have been proposed to incorporate fuzziness into the backpropagation multilayer perceptron networks. Pal and Mitra [7] put forward a particularly interesting model that is capable of fuzzy classifications of patterns. Both numeric and linguistic inputs are expressed in terms of membership values to the linguistic properties *low, medium,* and *high.* When the input is numerical, it is fuzzified first using a bell-shaped membership function. An n-dimensional input feature is thus converted into a $3n$-dimensional feature to be processed by the neural network.

In the proposed model, the network passes through two phases: training and testing. A supervised learning algorithm is used for training; it assigns output membership values that correspond to the training inputs. Errors in membership assignment are backpropagated and used to adjust the connecting weights. The error backpropagating from nodes with higher membership will have more weight. After convergence, the network is tested using a part of the same training data.

Although the model is time-consuming in training, the test of the network performance for fuzzy classification of speech data have given favorable results.

7.3 Fuzzy Competitive Learning

The competitive learning models discussed in section 6.4.5 usually utilize a single-layer feedforward network. The neurons compete; the winning neuron learns and the losing neurons do not learn—a concept referred to as *winner-take-all.*

Such models may underutilize the available neurons. In addition, some neurons may never win if their initialization weights are far away from what is required by the training set [27, 28]. The concept of winner-take-all leads to loss of information regarding how close other neurons were to winning [29]. It has also been reported that the winner-take-all networks require $\log_2(m)$ steps to perform one competition [30].

Fuzzy competitive learning can be useful in overcoming the above drawbacks of the traditional competitive learning. The concept of membership to the set of winners is introduced. Instead of having one dominating winning neuron, we assign a degree of winning to each neuron depending on its distance to the current training pattern. Neurons learn according to their membership to the winning set during competition. Instead of a winner-take-all rule, a learn-according-to-how-well-it-wins rule results [31]. A measure of confidence in the resulting output can also be reached.

The Kohonen algorithm presented in section 6.4.5 can be modified as follows [31].

- Start with small random values for the weights, $w_{ij}(t)$, that connect n inputs to m output nodes.
- Present an input $x_0(t), x_1(t), \ldots, x_n(t)$.
- Calculate the distance in n-dimensional space between the output vector and the weights vector

$$d_j = \sum_{i=0}^{n-1} (x_i(t) - w_{ij}(t))^2$$

- Based on the computed distance, d_j, determine the membership, μ_j, of each neuron, j, to the set of winners.
- Update each competing neuron's weights:

  $w_{ij}(t+1) = w_{ij}(t) + \eta(t)Z_{ij}(t)[x_i - w_{ij}(t)]$,

  where $\eta(t)$ is the learning rate and $Z_{ij}(t)$ is a fuzzy scaling factor that is a function of μ_j.
- The process is repeated until convergence.

In the previous example of a fuzzy Kohonen network, the input is assumed to be crisp; fuzziness was utilized to change the winner-take-all concept and retain the degree of winning information.

Kohonen networks can be modified using fuzzy concepts in a different way by considering fuzzy inputs. The input patterns may have imprecise or incomplete features due to noise corruption, for example. Sometimes, it is costly to extract features of a given pattern. In such situations, linguistic hedges become useful, that is, inputs such as *low, medium,* and *high* may be used instead of exact numerical values.

Such a model was put forward by Mitra and Pal [32]. The input consists of membership values for linguistic properties and some contextual class membership information. They also proposed a new definition for the gain factor. The effectiveness of their algorithm was demonstrated on the speech recognition problem.

7.4 Fuzzy ART

Fuzzy ART is an extension of ART1. It incorporates computations from fuzzy set theory into ART1 neural networks [18, 19].

In section 6.4.5b we presented the basic idea of ART1; a key expression in the ART1 algorithm is the comparison

$$\frac{\sum x_i t_{ij}}{\sum x_i} > \rho, \tag{7.1}$$

where x_i is the input, t_{ij} is the feedback weight, and ρ is the vigilance threshold.

That expression can be written in set theory notation as

$$\frac{|\mathbf{X} \cap \mathbf{T}|}{|\mathbf{X}|} > \rho \tag{7.2}$$

where $\cap$ is the logical AND intersection.

Now, the fuzzy set theory calculations can be incorporated in the algorithm by changing the logical AND calculations into fuzzy AND calculations. This leads to the expression

$$\frac{|\mathbf{X} \wedge \mathbf{T}|}{|\mathbf{X}|} > \rho \tag{7.3}$$

where $\wedge$ is the fuzzy AND defined by $(\mathbf{x} \wedge \mathbf{y})_i \equiv \min(x_i, y_i)$, and the norm $|\cdot|$ is defined by

$$|\mathbf{X}| \equiv \sum_{i=1}^{n} |x_i|. \tag{7.4}$$

Proliferation of categories is avoided in fuzzy ART if inputs are normalized; that is, for some $\gamma > 0$,

$$|\mathbf{X}| \equiv \gamma \text{ for all inputs } \mathbf{X}. \tag{7.5}$$

Normalization can be achieved for each incoming vector $\mathbf{a}$ by setting

$$\mathbf{X} = \frac{\mathbf{a}}{|\mathbf{a}|}. \tag{7.6}$$

Complement coding is an alternative normalization rule; it has the advantage of preserving amplitude information. The technique uses on-cell and off-cell pairs to normalize input vectors.

Let $\mathbf{a}$ represent the ON-response and $\mathbf{a}^c$ (the complement of $\mathbf{a}$) represent the OFF-response. The complement coded input $\mathbf{X}$ is defined by

$$\mathbf{X} = (\mathbf{a}, \mathbf{a}^c) \equiv (a_1, a_2, \dots a_n, a_1^c, a_2^c, \dots a_n^c). \tag{7.7}$$

Now, we have

$$\begin{aligned} |\mathbf{X}| &= |(\mathbf{a}, \mathbf{a}^c)| \\ &= \sum_{i=1}^{n} a_i + \left(n - \sum_{i=1}^{n} a_i\right) \\ &= n. \end{aligned} \tag{7.8}$$

In other words, inputs preprocessed into complement coding form are automatically normalized.

Fuzzy ARTMAP

ARTMAP, also referred to as predictive ART, is a supervised neural network system. It uses two ART1 modules (ART_a and ART_b) that are capable of self-organizing stable recognition categories. The two modules are linked by an inter-ART associative memory network and an internal controller. During training, the ART_a receives streams of input patterns $[\mathbf{a}^{(p)}]$ and $[\mathbf{b}^{(p)}]$, respectively, where $\mathbf{b}^{(p)}$ is the correct prediction given $\mathbf{a}^{(p)}$.

The ARTMAP system was reported [20, 21] to learn much more quickly and accurately than other algorithms such as back propagation. As with ART, ARTMAP can continue learning until its memory is fully utilized.

The concept was generalized by replacing ART_a and ART_b modules by their fuzzy counterparts. The system thus structured is known as fuzzy ARTMAP. The system can learn each input as it is received on-line rather than performing off-line optimization. The system's on-line learning was demonstrated to have an accuracy that ranges from 88.6% to 98.0% for training set sizes that range from 100 to 100,000 randomly chosen points.

Comparison of on-line training of fuzzy ARTMAP and back propagation indicated a better performance of fuzzy ARTMAP. The test task was to distinguish two spirals. About 20,000 epochs were needed for the back propagation system, while five epochs were sufficient for fuzzy ARTMAP (an epoch is defined as one cycle of training on an entire set of input exemplars).

7.5 Fuzzy Min-Max Neural Networks

There are similarities between fuzzy ART and fuzzy min-max neural networks [33, 34]. Both networks stemmed from the fuzzification of ART networks; however, the details of the two algorithms are different.

The fuzzy min-max system divides the n-dimensional pattern space into hyperboxes. Each hyperbox is defined by a pair of min-max points, the min points being

$$V_j = (v_{j1}, v_{j2}, \dots v_{jn}), \tag{7.9}$$

and the max points being

$$W_j = (w_{j1}, w_{j2}, \dots w_{jn}). \tag{7.10}$$

These two points in hyper-space are enough to define a hyperbox.

There is a membership function associated with pixels of every input pattern. The j^{th} hyperbox fuzzy set, B_j, can then be defined by an ordered-pair set given by

$$B_j = \{A_h, V_j, W_j, b_j(A_h, V_j, W_j)\} \quad \text{for all } h = 1, 2, \dots m, \tag{7.11}$$

where $A_h = (a_{h1}, a_{h2}, \dots a_{hn})$ is the h^{th} pattern in the input data, and $0 \le b_j(A_h, V_j, W_j) \le 1$ is the membership function of the j^{th} hyperbox.

The membership function, b_j, is a measure of the degree to which the h^{th} input pattern, A_h, is contained within the hyperbox defined by the max and min points. The membership will be one if the pixel is contained completely within the hyperbox.

Simpson [33, 34] proposed a membership function that meets the above requirements as the sum of two complements. This leads to

$$\begin{aligned} b_j &= \frac{1}{2n}\sum_{i=1}^{n}[\max(0, 1 - \max(0, \gamma \min(1, a_{hi} - w_{j2}))) \\ &\quad + \max(0, 1 - \max(0, \gamma \min(1, v_{ji} - a_{hi})))] \\ &= \frac{1}{n}\sum_{i=1}^{n}[1 - f(a_{hi} - w_{ji}, \gamma) - f(v_{ji} - a_{hi}, \gamma)], \end{aligned} \tag{7.12}$$

$$\text{where } f(x, \gamma) = \begin{cases} 1 & ; \quad x\gamma > 1 \\ x\gamma & ; \quad 0 \le x\gamma \le 1 \\ 0 & ; \quad x\gamma < 0 \end{cases}$$

The sensitivity parameter, γ, controls how the membership values decrease as the distance between A_h and B_j increases. When γ is large, the fuzzy set becomes less fuzzy—it moves towards becoming crisp. Thus, by selecting a large value for the parameter γ, it is possible to define crisp cluster boundaries as a special case of fuzzy boundaries.

Fuzzy Min-Max Learning

The fuzzy min-max learning algorithm proposed [33] consists of four steps: initialization, expansion, overlap test, and contraction. These steps can be described as follows.

Initialization

The algorithm uses two sets of clusters: the committed set, C, and the uncommitted set, U. The uncommitted clusters are the ones available for adjusting their min-max points. Once the min-max points are defined, they become committed.

The initialization process occurs by setting the min points, V_j, and the max points, W_j as follows:

$$V_j = \underline{1} \text{ and } W_j = \underline{0} \qquad \text{for all } B_j \in U,$$

where $\underline{1}$ and $\underline{0}$ are n-dimensional vectors of all ones and all zeros, respectively.

This initialization ensures that the first pattern committed to a hyperbox results in a single point that is identical to the input pattern.

Hyberbox expansion

The second step is to identify the closest hyperbox, which can be expanded, if needed, to the input pattern.

For a hyperbox B_j to expand so that it includes A_h, the following constraint must be satisfied

$$\sum_{i=1}^{n} [\max(w_{ji}, a_{\text{hi}}) - \min(v_{\text{ji}}, a_{\text{hi}})] \leq \theta,$$

where $0 \leq \theta \leq 1$ is a parameter, defined by the user, that sets a maximum bound on the size of the hyperbox.

If that constraint is satisfied, then the min and max points can be expanded as follows:

$$v_{ji}^{\text{new}} = \min(v_{ji}^{\text{old}}, a_{\text{hi}}) \quad \text{, and}$$
$$w_{ji}^{\text{new}} = \max(w_{ji}^{\text{old}}, a_{hi}) \quad \text{for all } i = 1, 2, \ldots, n.$$

A hyperbox $B_j \in U$ is selected, and the above expansion equations are used if all $B_j \in C$ are exhausted without any possible expansion.

Overlap test

After expanding $B_j \in C$, an overlap test between B_j and the remaining $B_k \in C$ is performed. An overlap occurs if any of the following conditions is satisfied for each of the n-dimensions:

$$v_{ji} < v_{ki} < w_{ji} < w_{ki}, \text{ or}$$
$$v_{ki} < v_{ji} < w_{ki} < w_{ji}, \text{ or}$$
$$v_{ji} < v_{ki} \leq w_{ki} < w_{ji}, \text{ or}$$
$$w_{ki} < v_{ji} \leq w_{ji} < w_{ki}.$$

Hyperbox contraction

If an overlap between two hyperboxes is identified, an immediate contraction is done to eliminate that overlap. The elimination of the overlap is done on dimension-by-dimension basis. The contraction process is carried out as follows.

If $v_{ji} < v_{ki} < w_{ji} < w_{ki}$, then

$$v_{ki}^{\text{new}} = w_{ji}^{\text{new}} = \frac{w_{ki}^{\text{old}} + w_{ji}^{\text{old}}}{2}.$$

If $v_{ki} < v_{ji} < w_{ki} < w_{ji}$, then

$$v_{ji}^{\text{new}} = w_{ki}^{\text{new}} = \frac{v_{ji}^{\text{old}} + w_{ki}^{\text{old}}}{2}.$$

If $v_{ji} < v_{ki} < w_{ki} < w_{ji}$; then the contraction is performed on the smaller overlap, i.e.,

$$v_{ji}^{\text{new}} = w_{ki}^{\text{old}} \text{ if } w_{ki} - v_{ji} < w_{ji} - v_{ji} < w_{ji} - v_{ki};$$

otherwise, $w_{ji}^{\text{new}} = v_{ki}^{\text{old}}$.

If $v_{ki} < v_{ji} \leq w_{ji} < w_{ki}$, then

$$v_{ki}^{\text{new}} = w_{ji}^{\text{old}} \text{ if } w_{ji} - v_{ki} < w_{ki} - v_{ji};$$

otherwise, $v_{ki}^{\text{new}} = v_{ji}^{\text{old}}$.

A different contraction technique can be used to suit the requirements of a particular application. The above steps are repeated until cluster stability is reached.

Figure 7.1a shows an implementation of the fuzzy min-max cluster [34]. The input layer consists of n neurons, and the output layer consists of m neurons. There are two connections from each input neuron to each output neuron, as clarified in Figure 7.1b. Each output neuron gives the degree to which the input pattern A_h belongs to each of the available clusters. The full membership occurs for one hyperbox fuzzy set. Thus, this network architecture implements each hyperbox fuzzy set as an output neuron.

7.6 Fuzzy Neurons

The fuzzy neural networks discussed in the previous sections used non-fuzzy neurons. It is possible, however, to put forward a model of a fuzzy neuron, and then construct a network using it as a processing unit.

In this section we give an overview of a fuzzy neuron model proposed by Kwan and Cai [5]. We also discuss a network built around that model for pattern recognition.

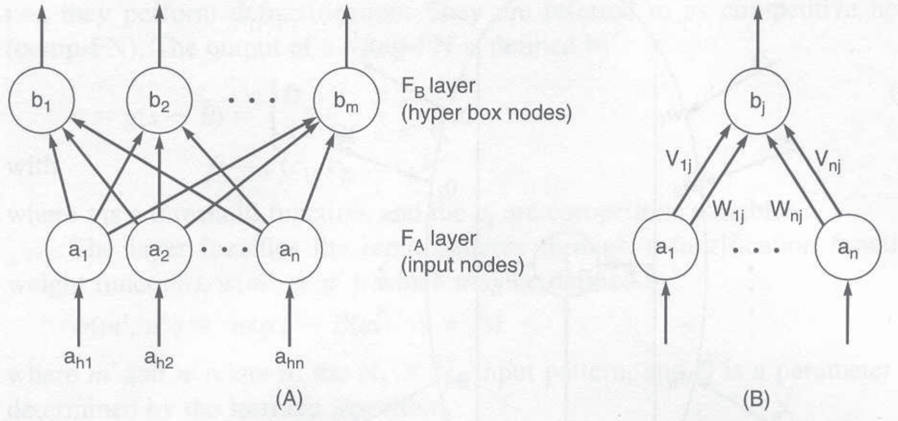


Figure 7.1 (a) A fuzzy min-max neural network (each connecting arrow represents a dual connection, as illustrated in (b)).

The fuzzy neuron model is similar to the classical neuron model in that it has n weighted inputs, x_i, with weights w_i, where $i = 1, 2, 3, \ldots n$. Both x_i and w_i have real values. Also, it has m outputs, y_j, where $j = 1, 2, \ldots, m$. In general, the output may have real values in the interval [0, 1].

Fuzziness is introduced by associating each output with a membership value that indicates to what degree an input pattern $(x_1, x_2, \ldots x_n)$ belongs to a fuzzy set. The output is expressed as

$$y_j = g_j\,(s) \qquad j = 1, 2, 3, \ldots, m, \tag{7.13}$$

where the g_j are the m output functions of the fuzzy neuron, which represent the membership functions of the input pattern in all the m fuzzy sets, and s is the state of the fuzzy neuron given by

$$s = f(z - \theta), \tag{7.14}$$

where f is the activation function, θ is the activation threshold, and z is the net input to the fuzzy neuron. It is given by

$$z = h(w_i x_i) \qquad i = 1, 2, 3, \ldots n, \tag{7.15}$$

where h is an aggregation function. The above description is summarized in Figure 7.2

A network built around the fuzzy neuron discussed above will have these fuzzy neurons in the input, output, and hidden layers. The requirement of each layer is different. Such differences can be modeled by the choice of the aggregation and output functions.

Kwan and Cai proposed a feedforward neural network and competitive algorithm, but supervised. The concept is very close to that of the previous networks,

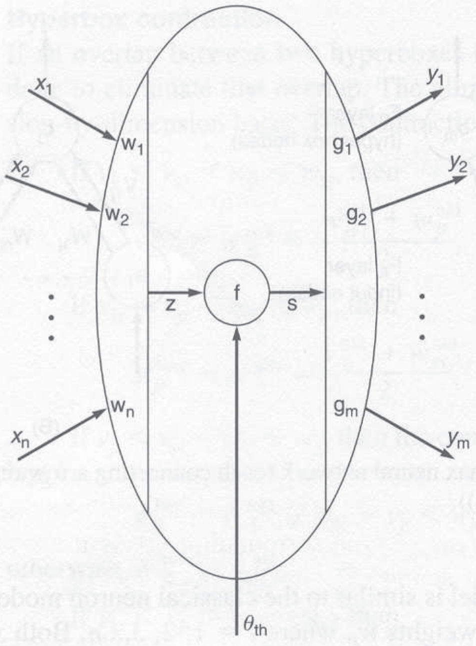


Figure 7.2 A fuzzy neuron.

but the network has a different architecture. The network can be trained to recognize an input pattern composed of $N_1 \times N_2$ pixels. The network proposed consists of four layers: input, output, and two hidden layers.

The input layer accepts the data of an input pattern. Its objective is to transform the pixel values of the input pattern into normalized values between zero and one. The neurons of this layer are called input fuzzy neurons (in-FN). Each neuron has only one input; this leads to $z = x$.

The hidden layers are composed of a layer of max-FNs followed by another one composed of min-FNs. A maximum fuzzy neuron (max-FN) has a max function as its aggregation function, i.e.,

$$z = \max_{i=1}^{n} (w_i x_i). \tag{7.16}$$

A minimum fuzzy neuron has a min function for aggregation, i.e.,

$$z = \min_{i=1}^{n} (w_i x_i). \tag{7.17}$$

The max-FN layer fuzzifies the input pattern, while the min-FN layer gives the similarities of the input pattern to all of the learned patterns.

The neurons of the output layer accept fuzzy input and produce crisp outs, i.e., they perform defuzzification. They are referred to as competitive neurons (comp-FN). The output of a comp-FN is defined by

$$y = g(s - \theta) = \begin{cases} 0 & ; \quad s < \theta \\ 1 & ; \quad s \geq \theta, \end{cases} \tag{7.18}$$

with $\theta = t\,(c_1, c_2, \dots c_k)$

where t is a threshold function, and the c_k are competitive variables.

The layer fuzzifies the input patterns through a fuzzification function (a weight function), $w(m' + n')$, which may be defined by

$$w(m', n') = \exp\,(-\beta(m'^2 + n'^2)) \tag{7.19}$$

where m' and n' relate to the $N_1 \times N_2$ input pattern, and β is a parameter to be determined by the learning algorithm.

Using max-FNs and fuzzification weights leads to affecting the state of several FNs in the second layer by one dark pixel in the input pattern. The network thus focuses on one pixel but sees the surrounding pixels to some degree. The parameter β determines that degree. The smaller the value of β, the more pixels seen. If β is too small, the network may become unable to separate some distinct training patterns. If β is too large, the network may lose its ability to recognize distorted patterns.

The third layer presents the similarities of the input pattern to all of the learned patterns. The output layer uses one comp-FN for each of the m learned patterns. If an input pattern is close enough to the mth learned pattern, the output of the mth comp-FN will be one and all the other outputs will be zero.

The network proposed was simulated using C language [5]. After training using 36 alphanumeric exemplar patterns, the network was able to achieve 100% recognition rate. The same recognition rate was achieved with one pixel shifted in eight directions (up, down, left, right, up-left, up-right, down-right, and down-left). An average recognition rate of about 92% was achieved with two pixels shifted in eight directions. The recognition rate of the two-pixels shift cases became about 98.6% after training with 36 exemplar patterns and 72 distorted patterns. The one-pixel shift case stayed at 100% recognition rate.

7.7 Fuzzy Neural Control Systems

The basic advantage of fuzzy control systems, as discussed in Chapter 5, is that they do not require a mathematical model of the process under control. Fuzzy systems depend on linguistic description of the process. The inputs are given in terms of linguistic variables and membership functions. The variables are processed with a predetermined set of fuzzy logic IF-THEN rules. The response to each rule is reached through a fuzzy implication rule. A defuzzification process then leads to a crisp output. The membership functions and the fuzzy rules are usually determined through the observation of a system operated by a human expert. The membership functions and the rules represent the knowledge or experience of the system. They are time-consuming to determine, test, and adjust. The learning ability of neural networks can be used to produce a fuzzy control system that can learn by example and generalize.

A few questions may arise. Can the fuzzy networks discussed in the previous sections (or similar ones) be used? Would it be more advantageous to separate the fuzzy controller from the neural network? or is it more advantageous to implement a fuzzy controller using neural network concepts? What determines stability, response time, etc.? Then finally, one may wonder about a systematic design technique with the simplest possible architecture. Various authors [35, 37–42] have reported on neuro-fuzzy systems with varied design approaches. Following we give examples of neuro-fuzzy control applications.

EXAMPLES: **A. Neuro-Fuzzy Control of a Car [35]**

Lin and Lee [35] implemented the idea of a fuzzy car [36] using neural network concepts. The core of a fuzzy control system, i.e., rule base, fuzzification, defuzzification, inference, and feedback (see Figure 5.5), was realized by a recurrent five-layer neural network. The car becomes able to learn from examples to move along a track that has rectangular turns.

Architecture. The first layer of the network is composed of linguistic nodes; each node represents an input linguistic variable. Three variables are used for car control: x_0, x_1, and x_2. The variables x_0 and x_1 represent the distance of the center of the car from the horizontal and vertical boundaries at the corner, respectively. The variable x_2 represents the current steering angle. The linguistic variable, y, represents the next steering angle.

The second layer of the network is composed of term nodes. They act as input membership functions for the input variables. The layer can be thought of as performing the fuzzification process.

The third layer is composed of rule nodes. Each node represents one fuzzy rule. The layer performs a fuzzy rule base.

The fourth layer is composed of term nodes. They act as membership functions for the output, with the links from the third layer to perform as the decision-making logic (inference engine).

The fifth layer is composed of two sets of linguistic nodes. One set is for the training data, i.e., the desired output. The output is fed back to the fourth layer nodes. The second set of nodes gives the actual control output. Figure 7.3 gives a summary of the tasks performed by the neural network.

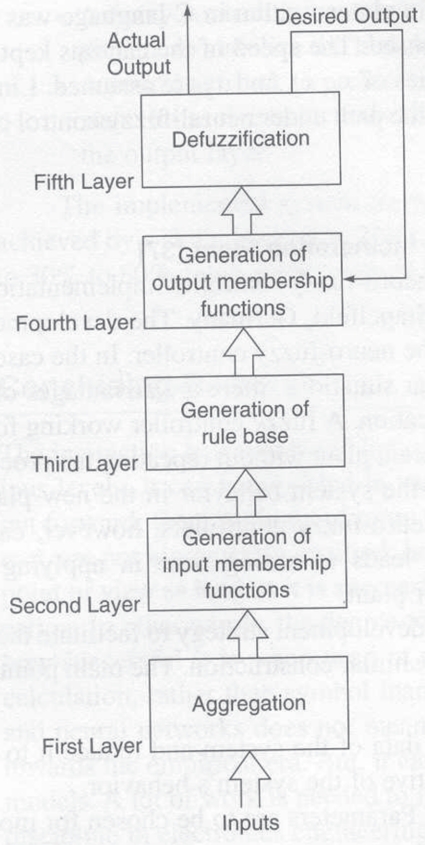


Figure 7.3 Summary of the tasks performed by the neural network implementation of the fuzzy car.

Learning Algorithm. Lin and Lee [35] used a hybrid learning algorithm to train the network. It has two phases. The first phase uses the concepts of self-organization clustering to locate the membership functions by finding their centers and widths, and to find fuzzy logic rules by competitive learning. The training data are input-output pairs, fuzzy partitions, and the desired shapes of the membership functions.

The second phase uses supervised back-propagation to adjust the widths and centers of membership functions optimally. The training data in this phase are input-output pairs, fuzzy partition, and fuzzy logic rules.

In controlling the fuzzy car, 61 fuzzy rules were learned. For example, they found that when $x_0 = 2$, $x_1 = 3$, and $x_2 = 2$, the output y is 12 is interpreted as:

IF x_0 is term 2, and x_1 is term 3, and x_2 is term 2, THEN y is term 6.

Results. A general purpose simulator written in C language was used to simulate the car control system proposed. The speed of the car was kept constant, and sensors to provide the state values of x_0, x_1, and x_2 are assumed. Lin and Lee [35] found that the training path and the path under neural-fuzzy control coincide under various initial steering angles.

B. Neuro-Fuzzy Control of an Incineration Plant [37]

Krause et al. [37] reported on a neuro-fuzzy controller implementation for a refuse incineration plant in Hamburg-Stapelfeld, Germany. The development tool fuzzy TECH was used to implement the neuro-fuzzy controller. In the case of the incineration plant, and in other similar situations, there are advantages of neuro-fuzzy control over fuzzy control application. A fuzzy controller working for a particular plant cannot be applied to a different plant without repeating the process of acquiring dedicated knowledge about the system behavior in the new plant, a process that can be time consuming. Neuro-fuzzy controllers, however, can learn from examples and generalize. This leads to saving time in applying a controller designed for one plant to another plant.

Krause et al. put forward a development strategy to facilitate the use of fuzzy controllers in several plants with similar construction. The main points of the strategy are as follows:

- Collect input and output data of the system and reduce it to a reasonable amount that is representative of the system's behavior.
- Select a training method. Parameters are to be chosen for modification by the neuro-fuzzy algorithm.
- The learning algorithm is executed until convergence, i.e., until it reaches the selected error thresholds. If it does not converge, use a different system setting or a different learning algorithm and repeat learning.
- On-line optimization for security assurance.

C. Neuro-Fuzzy Control of Tank Level [42]

In the process of producing lubricant oil, vacuum distillation is used. It results in distillate oil and reduced oil that includes wax. The wax is removed in a solvent dewaxing plant, in which a tank receives the solvent and lubricant oil for further processing.

The control objective is to change the rate of flow from the tank smoothly and to keep the fluid level in the tank stable.

The neuro-fuzzy controller used has a different architecture compared to the controllers discussed in the previous examples. It is composed of three components:

- A statistical component to calculate the flow rate tendency in the flow rate based on operation data.

- A fuzzy controller to correct flow rate, determined by the statistical component, to stabilize the tank level.
- A neural network to predict the inflow rate of the tank during the transient state. It has five units in the input layer, ten in the hidden layer, and one for the output layer.

The implemented system showed that the flow rate was as smooth as that achieved by a skilled operator. The tank level varied from 35% to 75% as opposed to 30% to 80% achieved by manual operation.

7.8 Concluding Remarks

The interaction between fuzzy logic theory and neural networks may occur at various levels. It can be modeled in varied ways. Several models have already been put forward; there is plenty of room for many more to appear. The models, in general, are not biologically justified; however, a model is useful from an engineering point of view as long as it is successful in improving or initiating a practical application. In other words, the degree to which a model is acceptable is measured by how successful it is when used in practical application. The fact that numerical calculation, rather than symbol manipulation, is fundamental to both fuzzy logic and neural networks does not mean that electronics engineering is heading back towards the empirical era. But, it can lead to a change in our view of theories and models. A lot of work is needed to make the neuro-fuzzy systems as established a discipline in electronics engineering as is digital electronics.

Chapter 7 Questions

7-1. Compare and contrast the foundation of neural networks and fuzzy logic.

7-2. Discuss how fuzzy systems and neural networks can be combined.

7-3. What are the merits of using neural networks in fuzzy control systems?

***7-4.** Learning fuzzy systems have several applications; select one of the following applications and discuss it briefly:

a) Pattern recognition (H. Kwan, *IEEE Trans. Fuzzy Systems* 2 (1994): 185–193).

b) Rice taste analysis (H. Ishibuchi et al., *Fuzzy Sets and Systems* 64 (1994): 129–144).

c) Control of refuse incineration plant (B. Krause et al., *Fuzzy Sets and Systems* 63 (1994): 329–338).

7-5. Mark the following statements as true or false; correct the wrong statements.

1. Fuzzy associative memory uses fuzzy matrices instead of fuzzy neurons.
2. Neural networks can be used to generate fuzzy rules.

3. Neural networks can do anything fuzzy systems can do.
4. Fuzzy systems can do anything neural networks can do.
5. Fuzzy neural systems are the ultimate in electronics; no further advances are possible.
6. A neural network may be used to interpret linguistic statements for a fuzzy system.
7. In Pal and Mitra's fuzzy perceptron, a supervised learning algorithm generates output membership assignments.
8. "Winner-take-all" refers to the action of supervised fuzzy learning.
9. Fuzzy competitive learning assigns a degree of winning to competing neurons.
10. "Learn according to how well the neuron learns" is the learning rule of competitive learning.
11. Fuzziness may be incorporated in a Kohonen network through one standard technique.
12. An ARTMAP system learns much faster than a back-propagation algorithm.
13. A fuzzy min-max learning algorithm is similar to fuzzy back-propagation.
14. A fuzzy neural network is always built using fuzzy neurons.
15. Fuzzy control systems can learn by example.
16. The neuro-fuzzy control model of Lin and Lee uses both self-organization and back-propagation.
17. In general, there is no biological justification for neuro-fuzzy systems.

7-6. Suggest a possible implementation of a fuzzy flip-flop using neural networks.

References

1. S.C. Lee and E.T. Lee, "Fuzzy Neural Networks," *Mathematical Biosciences* 23 (1975): 151–177.
2. T. Yamakawa and S. Tomada, "A Fuzzy Neuron and Its Applications to Pattern Recognition," *Proc. of the Third IFSA Congress* (Seattle, Washington, August 1989) 943–948.
3. T. Yamakawa, "Pattern Recognition Hardware System Employing a Fuzzy Neuron," *Proc. the Int. Conf. on Fuzzy Logic and Neural Networks* (Japan, July 1990) 30–38.
4. T. Yamakawa and M. Furukawa, "A Design Algorithm of Membership Functions of a Fuzzy Neuron Using Example-based learning," *IEEE Int. Conf. on Fuzzy Systems* (San Diego, March 1992) 75–82.
5. H.K. Kwan and Y. Cai, "A Fuzzy Neural Network and Its Application to Pattern Recognition," *IEEE Trans. Fuzzy Systems* 2 (1994): 185–193.
6. J.M. Keller and D.J. Hunt, "Incorporating Fuzzy Membership Functions into Perception Algorithm," *IEEE Trans. Pattern Analysis Machine Intell.* PAMI-7 (1985): 693–699.
7. S.K. Pal and S. Mitra, "Multilayer Perceptron, Fuzzy Sets, and Classification," *IEEE Trans. Neural Networks* 3 (1992): 683–697.
8. S. Horikawa, T. Furuhashi, and Y. Uchikawa, "On Fuzzy Modeling Using Fuzzy Neural Networks with the Back-propagation Algorithm," *IEEE Trans. Neural Networks* 3 (1992): 801–806.

9. Q. Hu and D.B. Hertz, "Improving the Convergence of Back-propagation Using Fuzzy Logic Control," *WCNN '93*, vol. 2, (1993): 47–51.
10. J.J. Shann and H.C. Fu, "Back-propagation Learning for Acquiring Fine Knowledge of Fuzzy Neural Networks," *WCNN '93*, vol. 2 (1993): 66–69.
11. C. Isik and F. Zia, "Fuzzy Logic Control Using a Self-organizing map," *WCNN '93*, II–56, 1993.
12. P.S. Khedkar and H. R. Berenji, "Generating Fuzzy Rules with Linear Consequences from Data," *WCNN '93*, II–18, 1993.
13. T.L. Huntsberger and P. Ajimarangsee, "Parallel Self-organizing Feature Maps for Unsupervised Pattern Recognition," *International J. General Systems* 16 (1990): 357–372.
14. D. Zhang, M. Kamel, and M.I. Elmasry, "Fuzzy Clustering Neural Network (FCNN) Using Fuzzy Competitive Learning," World Congress on Neural Networks, 1993, II–22, July 11–15, Portland, Oregon.
15. J. Nie, "Fuzzy Modeling Using Self-organizing CPN Networks," *WCNN '93*, II–43, 1993.
16. F.L. Chung and T. Lee, "Fuzzy Competitive Learning," *Neural Networks* 3 (1994): 539–551.
17. S. Mitra and S.K. Pal, "Self-organizing Neural Network as a Fuzzy Classifier," *IEEE Trans. Systems. Man. and Cypernetics,* V. 24 (1994): 385–399.
18. G.A. Carpenter, S. Grossberg, and D.B. Rosen, "Fuzzy ART: Fast Stable Learning and Categorization of Analog Patterns by Adaptive Resonance System," *Neural Networks* 4 (1991): 759–771.
19. M.J. Healy and T.P. Candell, "Discrete Stack Interval Representation and Fuzzy ART1," *WCNN '93*, II–82, 1993.
20. G.A. Carpenter, S. Grossberg, N. Markuzon, J.H. Reynolds, and D.B. Rosen, *IEEE Trans. Neural Networks* 3 (1992): 698–713.
21. G.A. Carpenter, S. Grossberg, and J.H. Reynolds, "Fuzzy ARTMAP, Slow Learning and Probability Estimation," *WCNN '93*, vol. 2 (1993): 26.
22. B. Kosko, *Neural Networks and Fuzzy Systems* (Prentice Hall, 1992). Englewood Cliffs, NJ
23. A.F. Rocha and R. Yager, "Neural Nets and Fuzzy Logic," in *Hybrid Architectures for Intelligent Systems*, ed. A. Kandel and G. Langholz (CRC Press, 1992) 4–28 Boca Raton, FL.
24. H. Takagi, "Fusion Techniques of Fuzzy Systems and Neural Networks, and Fuzzy Systems and Genetic Algorithms," *SPIE Proc. of Technical Conference on Applications of Fuzzy Logic Technology v.* 2061 (September 1993): 402–413.
25. J.J. Buckley and Y. Hayashi, "Fuzzy Neural Networks: A Survey," *Fuzzy Sets and Systems* 66 (1994): 1–13.
26. M.M. Gupta and D.H. Rao, "On the Principles of Fuzzy Neural Networks," *Fuzzy Sets and Systems* 61 (1994): 1–18.
27. S.C. Ahalt, A.K. Krishnamurthy, and D.E. Melton, "Competitive Learning Algorithms for Vector Quantization," *Neural Networks* 3 (1990): 277–290.
28. S. Grossberg, "Adaptive Pattern Classification and Universal Recording: I. Parallel development and coding of Neural Feature Detectors," *Biological Cybernetics* 23 (1976): 121–134.

29. C. Dacaestecker, "Competitive Clustering," *Proceedings Int. Neural Networks Conference* 2 (1990): 833.
30. J.H. Winters and C. Rose, "Minimum Distance Automata in Parallel Networks for Optimum Classification," *Neural Networks* 2 (1989): 127–132.
31. F.L. Chung and T. Lee, "Fuzzy Competitive Learning," *Neural Networks* 7 (1994): 539–551.
32. S. Mitra and S.K. Pal, "Self-organizing Neural Network as a Fuzzy Classifier," *IEEE Trans. Systems, Man. and Cyber*, 24 (1994): 385–399.
33. P.K. Simpson, "Fuzzy Min-max Neural Networks-Part 1: Classification," *IEEE Trans. Neural Networks* 3 (1992): 776–786.
34. P.K. Simpson, "Fuzzy Min-max Neural Networks-Part 2: Clustering," *IEEE Trans. Fuzzy Systems* 1 (1993): 32–45.
35. C.-T. Lin and C.S. Lee, "Neural-network-based Fuzzy Logic Control and Decision System," *IEEE Trans. Computers* 40 (1991): 1320–1336.
36. M. Sugeno and M. Nishida, "Fuzzy Control of Model Car," *Fuzzy Sets and Systems* 16 (1985): 103–113.
37. B. Krause, C. von Altrock, K. Limper, and W. Schäfers, "A Neuro-fuzzy Adaptive Control Strategy for Refuse Incineration Plants," *Fuzzy Sets and Systems* 63 (1994): 329–338.
38. R.R. Yager, "Implementing Fuzzy Logic Controllers Using a Neural Network Framework," *Fuzzy Sets and Systems* 48 (1992): 53–64.
39. J.J. Buckley and Y. Hayashi, "Hybrid Neural Nets Can Be Fuzzy Controllers and Fuzzy Expert Systems," *Fuzzy Sets and Systems* 60 (1993): 135–142.
40. F. Bouslama and K. Ichikawa, "Application of Neural Networks to Fuzzy Control," *Neural Networks* 6 (1993): 791–799.
41. C.-L. Chen and W.-C. Chen, "Fuzzy Controller Design by Using Neural Network Techniques," *IEEE Trans. Fuzzy Systems* 2 (1994): 235–244.
42. T. Tani, S. Murakoshi, T. Sato, M. Umano, and K. Tanaka, "Applications of Neuro-fuzzy Hybrid Control System to Tank Level Control," in *Fuzzy Logic Technology and Applications* ed. R.J. Marks II, (1994): 247–252.

10

GLOSSARY

Adaptive: A system that can be modified during operation to meet a specified criteria.

Algorithm (algorism): A step-by-step procedure that can be carried out mechanically; it can be realized by software or hardware. Alkhawarizmi (780–850) wrote: "with my two algorithms, one can solve all problems—without error, if God will."

ART: Adaptive Resonance Theory: a self-organizing network. The first version, ART1, can process only binary input patterns. The second version, ART2, can process real input patterns; ART3 is an improved ART2 in which the processing is more stable.

Attractor Network: A neural network that has an energy surface that attracts the current state of the network to a particular energy state or states.

Antecedent: The clause that implies the other clause in a conditional statement.

Associative Memory: A system that stores data in parallel and recalls them based on some feature of the data.

Adaptive Fuzzy System: A fuzzy system that does not require rules from a human expert; it generates and tunes its own rules. A neuro-fuzzy system or fuzzy-neural systems are adaptive fuzzy systems.

Back-propagation: A supervised learning rule for multilayer perceptrons that operates by calculating the value of the error function for a known input, then back-propagating the error from one layer to the previous one. Each neuron has its weights adjusted so that it reduces the value of the error function until a stable state is reached.

BAM: Bidirectional Associative Memory. An attractor network in which activation moves back and forth between two layers of neurons until a stable state is reached.

Boltzmann machine: A neural network algorithm that is based on statistical mechanics. It uses simulated annealing to reach stable states.

Basins of attraction: The valleys of the energy surface of a neural network. The energy surface is a graph of the energy function vs. weights, with the energy function being a measure of the amount by which the input differs from the desired output. Thus, the basins of attraction give all possible solutions, i.e., values of weights that produce correct output for a given input.

Compensatory operators: Non-Zadeh operators, i.e., operators not defined by simple min-max rules. They fall into two general categories; the first encompasses operators based on simple arithmetic transformations, such as the bound sum and difference. The second encompasses operators based on more complex functional transformations, such as Yager operators.

Comparator: A circuit, commonly based on an operational amplifier, that has two inputs and one output. The output is zero, positive, or negative depending on whether the input is equal to, greater than, or smaller than the reference input.

Consequent (succedent): The resultant clause in a conditional statement.

Crisp set: The classical set, as opposed to a fuzzy set. It is a collection of items. An item can be either a member of that set or not. All sets are subsets of the universal set (mother of all sets).

Competitive learning: A learning algorithm that requires neurons to compete with each other in adjusting their weights.

Degree of membership: An expression of the confidence or certainty that an element belongs to a fuzzy set. It is a number that ranges from zero to one.

Defuzzification: Finding the best crisp representation of a given fuzzy set.

Excluded middle law: The principle that every proposition is either true or false. The principle leads to classical set theory. Fuzzy logic and fuzzy sets do not obey this law, since fuzzy sets allow partial membership.

FAM: Fuzzy Associative Memory; an associative memory based on fuzzy logic principles.

Fuzzy flip-flop: A generalization of Boolean JK flip-flop. It uses fuzzy states and can be used for fuzzy information processing.

Fuzzy control: A control system based on an algorithm composed of IF . . . THEN . . . rules. A fuzzy logic operation may be used in the construction of the rule, e.g., IF . . . AND . . . THEN . . . . More than one rule may fire at the same time, each with its own strength. A defuzzification process follows to generate a crisp control action.

Fuzzy operators: Operators that combine fuzzy antecedents to produce truth value. Zadeh defined fuzzy operators in terms of min-max rules. Several other methods of definition exist; the alternatively defined operators are referred to as non-Zadeh operators or compensatory operators.

Fuzzy Set: A set that allows its elements to have degrees of membership. These degrees range from zero, when the element is not in the set, to one, when the element is in the set. The objective is to define precisely what is intrinsically vague. A crisp set allows only two values of membership, either one or zero. An ultrafuzzy set has its membership function itself as a fuzzy set.

Fuzzy logic: A scheme of systematic analysis that uses linguistic variables, such as hot, cold, very, little, large, small, etc., as opposed to Boolean or binary logic, which is restricted to true or false states. The objective of the scheme is to enable the computer to make human-like decisions.

Fuzzy NOT operator: Zadeh defined the fuzzy negation operation by $\mu_{\bar{A}} = 1 - \mu_A$. Alternative definitions exist, although they may not be commonly used. For example, Yager, Sugeno, and threshold NOT operators are defined. The threshold NOT is defined by

$$\mu_{\bar{A}}(x;\, k) = \begin{cases} 1 & \mu_A < k \\ 0 & \mu_A \geq k \end{cases}$$

Fuzzy modifier: An added description of a fuzzy set that leads to an operation that changes the shape (mainly the width and position) of a membership function.

Fuzzification: Changing a crisp number or set to a fuzzy number or set.

Fuzzy entropy: A measure, which ranges from 0% to 100%, of the fuzziness of a set. The more a set resembles its negation, the greater its fuzzy entropy, and the fuzzier it is.

Floating membership function: A feature introduced in the AL220 fuzzy controller (from Adaptive Logic Inc.) whereby the center and width of a membership function can be varied dynamically based on values from input or output registers.

Genetic algorithm: An efficient method of searching for the optimal solution without carrying out an exhaustive search. The search carried out by a series of random choices is governed by the rules of genetics.

Gradient descent system: A system that attempts to reach its stable state by moving consistently down the steepest portion of its energy surface.

Hedges: Linguistic terms that intensify, dilute, or complement a fuzzy set.

Hopfield network: A single-layer attractor network. It consists of a number of neurons (nodes); each is connected to every other neuron. The weights on the link from one neuron to another are the same in both directions. The network takes only two-state inputs.

Hyper-cube: A cube in an *n*-dimensional space. A crisp set defines a corner of a unit hyper-cube, while a fuzzy set defines a point inside the hyper-cube.

Inverted pendulum problem: The problem of controlling the speed of a small vehicle to keep a rigid pole attached to it by a pivot balanced. It was solved classically by mathematical modeling. The problem was also modeled linguistically to illustrate the potential of a fuzzy control approach.

Kalman filter: An algorithm that gives an optimal estimate of the next state of a system, given the present state of a linear system and all of its past states.

Kohonen network: A self-organizing neural network. All the neurons are in one two-dimensional layer with inputs connected to every neuron. Neurons are connected laterally to their immediate neighbors; neighborhood interactions decrease with time.

Learning law: The rule used during training of a neural network for systematic updating of the weights.

Linguistic variable: Common language expression used to describe a condition or a situation, such as "hot," "cold," etc. It can be expressed using a fuzzy set defined by the designer.

Logic circuit: A circuit that compares inputs and produces outputs according to a set of logic rules. The simplest logic circuit performs one simple logic operation, such as an AND, OR, or NOT operation. The operation of a logic circuit may be described using a logic expression or a truth table.

Mapping: A transformation, particularly between abstract spaces.

Membership function: The mapping that associates each element in a set with its degree of membership. It can be expressed as discrete values or as a continuous function. Triangular, trapezoidal, and Gaussian membership functions are commonly used.

Neuron: The biological neuron is a cell that builds up the nervous system. The dendrites are its input parts, the axon is its output part, and the synapse is the junction between the axon of one neuron and the dendrite of another. When a neuron is stimulated on its dendrites, it sums the incoming potentials. If the sum is high enough, it sends an action potential down the axon. The operation is modeled in an electronic neural network by a processing element, PE, that performs a weighted summation and has threshold and a sigmoid-type transfer function. The term neurode is sometimes used to distinguish the artificial neuron from the biological one.

Paradox: A contradiction reached by derivation from unexceptional premises, for example, Russell's paradox: the set of all sets that are not members of themselves is a member of itself only if it is not, and it is not only if it is. Simply put, "an army barber was ordered to shave everyone who does not shave oneself; who shaves the barber?"

Pythagorean theorem: The square of the length of the longest side in a right-angle triangle is equal to the sum of the squares of the lengths of the other two sides. The theorem can lead to a measure of the degree to which one fuzzy set contains another through the subset-hood theorem.

PID Controller: A cascade control device inserted in the forward path of a feedback control system. Its input is the error signal and its output is the control action. It is comprised of three control terms: proportional, integral, and derivative. Increasing the gain of the proportional term increases the speed of the system response and reduces the steady state error, but tends to destabilize the system. The integral term can remove the steady-state error. It tends to destabilize the system because of the extra phase lag it introduces. The derivative term speeds up the transient response; it introduces a phase-lead and thus has a stabilizing effect.

Perceptron: A single-layer neural network that performed the first training algorithm. It can solve only linearly separable problems.

Singleton: A set that has one member only.

Simulated annealing: The process of introducing and then reducing the amount of random noise introduced into the weights and inputs of a neural network. The process is analogous to methods used in solidification, where the system starts with a high temperature to avoid local energy minima, and then gradually the temperature is lowered according to a particular algorithm.

Stability-plasticity problem: The situation when neural networks are not able to learn new information without the destruction of previous learning; a time-consuming retraining will be needed every time new information is to be learned. ART networks solved this problem.

Sugeno Operator: The Sugeno complement is a compensatory operator that uses a class parameter to determine the strength of the negation. It is defined by

$$\mu_{\bar{A}} = \frac{1 - \mu_A}{1 + k\mu_A}; \ -1 < k < \infty.$$

The Sugeno operator becomes the standard Zadeh operator when $k = 0$.

Truth table: A table that describes the operation of a logic circuit, or logic function, by listing all possible combinations of the inputs and the corresponding output values.

TSP: Traveling Salesman Problem. The problem of selecting an optimal path for a traveling salesman to visit several diverse locations once. It is a classical optimization problem.

Uncertainty principle (Heisenberg uncertainty principle): It is not possible to know with certainty both the position and momentum of a particle. It is usually expressed as $\Delta x \, \Delta p_x \geq h/4\pi$, where Δx and Δp_x are the uncertainty in the position and momentum, respectively, and h is Plank's constant.

Yager operators: A type of compensatory operators. The complex function defining the operator uses a parameter whose numerical value depends on the strength of the fuzzy operation. The operators converge to Zadeh's operators as the parameter becomes very large. The objective of such operators is to indicate the importance of the set membership truth functions. Yager AND, OR, and NOT are defined by

$$\mu_{A \cap B} = 1 - (1, ((1 - \mu_A)^k + (1 - \mu_B)^k)^{1/k}),$$
$$\mu_{A \cup B} = \min(1, ((1 - \mu_A)^k + (\mu_B)^k)^{1/k}), \text{ and}$$
$$\mu_{\bar{A}} = (1 - (\mu_A)^k)^{1/k}.$$

Appendix A

Examples of Fuzzy Logic Software

1. FuzGen (shareware)
   Alston Software Labs
   1320 Standiford Ave. #242
   Modesto, CA 95350 USA
   Tel/Fax: 209-522-8666
   76040.2247 @ compuserve.com

2. GA FuzzyWare (shareware)
   RP Huang
   3842 West Sheffield Ave.
   Chandler, AZ 85226 USA
   huang@drwho.iac.honeywell.com

3. Togai Demo
   (a collection of demonstration programs
   for fuzzy control of an inverted pendulum)
   Togai InfraLogic, Inc.
   30 Corporate Park, Suite 107
   Irvine, CA 92714 USA
   Tel: 714-975-8522
   Fax: 714-975-8524

4. FuzzyFan (shareware)
(an example of fuzzy control of a fan)
Henry Hurdon
15 Summit Avenue
Thunder Bay, Ontario
Canada P7B 3N7
hdhurdon@flash.lakeheadu.ca

5. Fuzzy Logic Designer
Byte Dynamics Inc.
14608 E. Olympic Ave.
Spokane, WA 99216 USA
Tel: 800-233-2983
Fax: 509-926-6130

6. FuziCalc
(fuzzy spreadsheet)
FuzziWare, Inc.
Knoxville, TN 37937-1287
Tel: 800-473-1287

7. O'INCA, Design Framework for Windows
Intelligent Machines, Inc.
1153 Bordeaux Drive
Sunnyvale, CA 94089 USA
Tel: 408-745-0881
Fax: 408-745-6408

8. NeuFuz4-C
National Semiconductor Corp.
2900 Semiconductor, POB 58090
Santa Clara, CA 95052 USA
Tel: 800-272-9959
Fax: 800-428-0065

9. CubiCalc and Fuzzy Logic
Hyper Logic Corporation
1855 East Valley Parkway, Suite 210
Escondido, CA 92027 USA
Tel: 619-746-2765
Fax: 619-746-4089

10. TIL Shell
Togai Information, Inc.
5 Vanderbilt
Irvine, CA 92718 USA
Tel: 714-588-3800
Fax: 714-588-3808

11. Fuzzy Decision Maker,
Fuzzy Thought Amplifier, and
Fuzzy Knowledge Builder
Fuzzy Systems Engineering
POB 27390
San Diego, CA 92198 USA
Tel/Fax: 619-748-7384

12. Fuzzy Logic Development Kit (FULDEK)
TSI Enterprises, Inc.
PO Box 14155
Albuquerque, NM 87191-4155 USA
Tel: 505-298-5817

13. MatLab, Fuzzy Logic Toolbox
The Math Works, Inc.
24 Prime Park Way
Natick, MA 01760-1500 USA
info@mathworks.com

14. Mathematica, Fuzzy Logic Pack
Wolfram Research Inc.
info@wri.com

Appendix B

Examples of Fuzzy Logic Hardware

1. From Adaptive Logic Inc.:
   (800 Charcot Ave., Suite 112, San Jose, CA 95131, USA,
   75471.2055@compuserve.com)

AL220	8 bit fuzzy microcontroller
NLX221	4–8 bit digital I/O single chip fuzzy microcontroller with EEPROM
NLX222	4–8 bit analog and digital I/O single chip fuzzy microcontroller
NLX230	8 bit microcontroller (30 million fuzzy rules per second)
NLX1100	Fuzzy pattern comparator
NLX112/113	Fuzzy data controllers

2. From Omron Corporation:
   (One East Commerce Drive, Schaumberg, IL 60173, USA,
   Fax: 708-843-7787/8568)

C500-FZ001	Fuzzy logic processor module for Omron C-series PLC's
E5AF	Fuzzy process temperature controller
FB-30 AT	FP-3000 based PC AT fuzzy inference board
FP-1000/3000	Digital fuzzy controller
FP-5000	Analog fuzzy controller

3. From Togai Infralogic Inc.:
   (5 Vanderbilt, Irvine, CA 92718, USA, info@til.com)

FC110	Digital fuzzy processor
FCAFL	SI cores based on Fuzzy Computational Acceleration
FCD10xxx	FC110 based modules

4. From Toshiba:
   (9775 Toledo Way, Irvine, CA 92718, USA, Fax: 714-8859-3963)

T/FC150	10-bit fuzzy inference processor
LFZY1	FC150-based NEC PC fuzzy logic board

Appendix C

Source code generated automatically by FLD design package for the example given in section 5.3.1.

```
/*
*******************************************************************
**********
Module: train.c
*******************************************************************
**********
*/

/*
*******************************************************************
**********
This source code has been automatically generated by the
  Fuzzy Logic Designer(tm) Version 1.10 design package
      Copyright (c) 1993  Byte Dynamics Inc.
      14608 E. Olympic Ave. Spokane, Wa. 99216
*******************************************************************
**********
*/

/*
*******************************************************************
**********
Include files:
*******************************************************************
**********
*/

#include "train.h"

/*
*******************************************************************
**********
Module variables:
*******************************************************************
**********
*/

static int  result_SPEED_VERY_SLOW=0;
static int  result_SPEED_SLOW=0;
static int  result_SPEED_FAST=0;
static int  result_SPEED_VERY_FAST=0;
static int  result_DISTANCE_VERY_CLOSE=0;
static int  result_DISTANCE_CLOSE=0;
static int  result_DISTANCE_FAR=0;
static int  result_DISTANCE_VERY_FAR=0;

static int result_BRAKES=0;

static int sum_BRAKES=0;

static int rule_result[NUM_RULES];
```

```
/*
*******************************************************************
**********
Function:  and( int fl1, int fl2)
*******************************************************************
**********
*/
static int  and( fl1, fl2)
int fl1;
int fl2;
{
if( fl1 <= fl2 )
  return( fl1);
else
  return( fl2);
}

/*
*******************************************************************
**********
Function:  or( int fl1, int fl2)
*******************************************************************
**********
*/
static int  or( fl1, fl2)
int fl1;
int fl2;
{

if( fl1 > fl2 )
  return( fl1 );
else
  return( fl2);
}

/*
*******************************************************************
**********
Function:  not( int fl1)
*******************************************************************
**********
*/
static int  not( fl1)
int fl1;
{

fl1 = 1 - fl1;
return( fl1);
}

/*
*******************************************************************
**********
```

```
Function:  very( int fl1)
*****************************************************************
**********
*/
static int  very( fl1)
int fl1;
{

fl1 = fl1 * fl1;
return( fl1);
}

/*
*****************************************************************
**********
Function:  not_very( int fl1)
*****************************************************************
**********
*/
static int  not_very( fl1)
int fl1;
{

fl1 = (10000 - (fl1 * fl1)) / 100;
return( fl1);
}

/*
*****************************************************************
**********
Function:  eval_SPEED_VERY_SLOW( )
*****************************************************************
**********
*/
static void eval_SPEED_VERY_SLOW( speed)
int speed;
{

if( speed >= 0 && speed <= 0 )
  result_SPEED_VERY_SLOW=100;
else if( speed >= 0 && speed <= 5 )
  result_SPEED_VERY_SLOW=100-(((speed-(0))*200)/10);
else
  result_SPEED_VERY_SLOW=0;

return;
}

/*
*****************************************************************
**********
Function:  eval_SPEED_SLOW( )
*****************************************************************
**********
```

```
*/
static void eval_SPEED_SLOW( speed)
int speed;
{

if( speed >= 20 && speed <= 20 )
  result_SPEED_SLOW=100;
else if( speed >= 3 && speed <= 20 )
  result_SPEED_SLOW=((speed-(3))*58)/10;
else if( speed >= 20 && speed <= 30 )
  result_SPEED_SLOW=100-(((speed-(20))*100)/10);
else
  result_SPEED_SLOW=0;

return;
}
/*
*****************************************************************
**********
Function:  eval_SPEED_FAST( )
*****************************************************************
**********
*/
static void eval_SPEED_FAST( speed)
int speed;
{

if( speed >= 40 && speed <= 45 )
  result_SPEED_FAST=100;
else if( speed >= 20 && speed <= 40 )
  result_SPEED_FAST=((speed-(20))*50)/10;
else if( speed >= 45 && speed <= 60 )
  result_SPEED_FAST=100-(((speed-(45))*66)/10);
else
  result_SPEED_FAST=0;

return;
}
/*
*****************************************************************
**********
Function:  eval_SPEED_VERY_FAST( )
*****************************************************************
**********
*/
static void eval_SPEED_VERY_FAST( speed)
int speed;
{

if( speed >= 53 && speed <= 100 )
  result_SPEED_VERY_FAST=100;
else if( speed >= 45 && speed <= 53 )
```

```
  result_SPEED_VERY_FAST=((speed-(45))*125)/10;
else
  result_SPEED_VERY_FAST=0;

return;
}

/*
*****************************************************************
**********
Function:  eval_DISTANCE_VERY_CLOSE( )
*****************************************************************
**********
*/
static void eval_DISTANCE_VERY_CLOSE( distance)
int distance;
{

if( distance >= 0 && distance <= 1 )
  result_DISTANCE_VERY_CLOSE=100;
else if( distance >= 1 && distance <= 20 )
  result_DISTANCE_VERY_CLOSE=100-(((distance-(1))*52)/10);
else
  result_DISTANCE_VERY_CLOSE=0;

return;
}

/*
*****************************************************************
**********
Function:  eval_DISTANCE_CLOSE( )
*****************************************************************
**********
*/
static void eval_DISTANCE_CLOSE( distance)
int distance;
{

if( distance >= 30 && distance <= 40 )
  result_DISTANCE_CLOSE=100;
else if( distance >= 10 && distance <= 30 )
  result_DISTANCE_CLOSE=((distance-(10))*50)/10;
else if( distance >= 40 && distance <= 120 )
  result_DISTANCE_CLOSE=100-(((distance-(40))*12)/10);
else
  result_DISTANCE_CLOSE=0;

return;
}

/*
*****************************************************************
**********
```

```
Function: eval_DISTANCE_FAR( )
**********************************************************************
**********
*/
static void eval_DISTANCE_FAR( distance)
int distance;
{

if( distance >= 100 && distance <= 150 )
  result_DISTANCE_FAR=100;
else if( distance >= 60 && distance <= 100 )
  result_DISTANCE_FAR=((distance-(60))*25)/10;
else if( distance >= 150 && distance <= 350 )
  result_DISTANCE_FAR=100-(((distance-(150))*5)/10);
else
  result_DISTANCE_FAR=0;

return;
}

/*
**********************************************************************
**********
Function:  eval_DISTANCE_VERY_FAR( )
**********************************************************************
**********
*/
static void eval_DISTANCE_VERY_FAR( distance)
int distance;
{

if( distance >= 400 && distance <= 500 )
  result_DISTANCE_VERY_FAR=100;
else if( distance >= 250 && distance <= 400 )
  result_DISTANCE_VERY_FAR=((distance-(250))*6)/10;
else
  result_DISTANCE_VERY_FAR=0;

return;
}

/*
**********************************************************************
**********
Function:  evaluate_rules( )
**********************************************************************
**********
*/
static void  evaluate_rules( )
{
int  result;

/*--------------------------------------------------------------------
---------
```

```
Rule 1
if SPEED is VERY_SLOW and
DISTANCE is CLOSE then
BRAKES is LIGHT
--------------------------------------------------------------------
---------*/
result=and( result_SPEED_VERY_SLOW, 100);
result=and( result_DISTANCE_CLOSE, result);
result_BRAKES+=(result*BRAKES_LIGHT);
sum_BRAKES+=result;
rule_result[0]=result;
/*------------------------------------------------------------------
---------
Rule 2
if SPEED is VERY_SLOW and
DISTANCE is FAR then
BRAKES is LIGHT
--------------------------------------------------------------------
---------*/
result=and( result_SPEED_VERY_SLOW, 100);
result=and( result_DISTANCE_FAR, result);
result_BRAKES+=(result*BRAKES_LIGHT);
sum_BRAKES+=result;
rule_result[1]=result;
/*------------------------------------------------------------------
---------
Rule 3
if SPEED is VERY_SLOW and
DISTANCE is VERY_CLOSE then
BRAKES is LIGHT
--------------------------------------------------------------------
---------*/
result=and( result_SPEED_VERY_SLOW, 100);
result=and( result_DISTANCE_VERY_CLOSE, result);
result_BRAKES+=(result*BRAKES_LIGHT);
sum_BRAKES+=result;
rule_result[2]=result;
/*------------------------------------------------------------------
---------
Rule 4
if SPEED is VERY_SLOW and
DISTANCE is VERY_FAR then
BRAKES is VERY_LIGHT
--------------------------------------------------------------------
---------*/
result=and( result_SPEED_VERY_SLOW, 100);
result=and( result_DISTANCE_VERY_FAR, result);
result_BRAKES+=(result*BRAKES_VERY_LIGHT);
sum_BRAKES+=result;
rule_result[3]=result;
```

```
/*-----------------------------------------------------------------------
---------
Rule 5
if SPEED is SLOW and
DISTANCE is VERY_CLOSE then
BRAKES is HEAVY
-------------------------------------------------------------------------
---------*/
result=and( result_SPEED_SLOW, 100);
result=and( result_DISTANCE_VERY_CLOSE, result);
result_BRAKES+=(result*BRAKES_HEAVY);
sum_BRAKES+=result;
rule_result[4]=result;
/*-----------------------------------------------------------------------
---------
Rule 6
if SPEED is SLOW and
DISTANCE is CLOSE then
BRAKES is LIGHT
-------------------------------------------------------------------------
---------*/
result=and( result_SPEED_SLOW, 100);
result=and( result_DISTANCE_CLOSE, result);
result_BRAKES+=(result*BRAKES_LIGHT);
sum_BRAKES+=result;
rule_result[5]=result;
/*-----------------------------------------------------------------------
---------
Rule 7
if SPEED is SLOW and
DISTANCE is FAR then
BRAKES is VERY_LIGHT
-------------------------------------------------------------------------
---------*/
result=and( result_SPEED_SLOW, 100);
result=and( result_DISTANCE_FAR, result);
result_BRAKES+=(result*BRAKES_VERY_LIGHT);
sum_BRAKES+=result;
rule_result[6]=result;
/*-----------------------------------------------------------------------
---------
Rule 8
if SPEED is SLOW and
DISTANCE is VERY_FAR then
BRAKES is VERY_LIGHT
-------------------------------------------------------------------------
---------*/
result=and( result_SPEED_SLOW, 100);
result=and( result_DISTANCE_VERY_FAR, result);
result_BRAKES+=(result*BRAKES_VERY_LIGHT);
sum_BRAKES+=result;
```

```
rule_result[7]=result;

/*------------------------------------------------------------------
---------
Rule 9
if SPEED is FAST and
DISTANCE is VERY_CLOSE then
BRAKES is VERY_HEAVY
--------------------------------------------------------------------
--------*/
result=and( result_SPEED_FAST, 100);
result=and( result_DISTANCE_VERY_CLOSE, result);
result_BRAKES+=(result*BRAKES_VERY_HEAVY);
sum_BRAKES+=result;
rule_result[8]=result;

/*------------------------------------------------------------------
---------
Rule 10
if SPEED is FAST and
DISTANCE is CLOSE then
BRAKES is HEAVY
--------------------------------------------------------------------
--------*/
result=and( result_SPEED_FAST, 100);
result=and( result_DISTANCE_CLOSE, result);
result_BRAKES+=(result*BRAKES_HEAVY);
sum_BRAKES+=result;
rule_result[9]=result;

/*------------------------------------------------------------------
---------
Rule 11
if SPEED is FAST and
DISTANCE is FAR then
BRAKES is LIGHT
--------------------------------------------------------------------
--------*/
result=and( result_SPEED_FAST, 100);
result=and( result_DISTANCE_FAR, result);
result_BRAKES+=(result*BRAKES_LIGHT);
sum_BRAKES+=result;
rule_result[10]=result;

/*------------------------------------------------------------------
---------
Rule 12
if SPEED is FAST and
DISTANCE is VERY_FAR then
BRAKES is LIGHT
--------------------------------------------------------------------
--------*/
result=and( result_SPEED_FAST, 100);
result=and( result_DISTANCE_VERY_FAR, result);
```

```
result_BRAKES+=(result*BRAKES_LIGHT);
sum_BRAKES+=result;
rule_result[11]=result;

/*-------------------------------------------------------------------
---------
Rule 13
if SPEED is VERY_FAST and
DISTANCE is VERY_CLOSE then
BRAKES is VERY_HEAVY
--------------------------------------------------------------------
---------*/
result=and( result_SPEED_VERY_FAST, 100);
result=and( result_DISTANCE_VERY_CLOSE, result);
result_BRAKES+=(result*BRAKES_VERY_HEAVY);
sum_BRAKES+=result;
rule_result[12]=result;

/*-------------------------------------------------------------------
---------
Rule 14
if SPEED is VERY_FAST and
DISTANCE is CLOSE then
BRAKES is VERY_HEAVY
--------------------------------------------------------------------
---------*/
result=and( result_SPEED_VERY_FAST, 100);
result=and( result_DISTANCE_CLOSE, result);
result_BRAKES+=(result*BRAKES_VERY_HEAVY);
sum_BRAKES+=result;
rule_result[13]=result;

/*-------------------------------------------------------------------
---------
Rule 15
if SPEED is VERY_FAST and
DISTANCE is FAR then
BRAKES is HEAVY
--------------------------------------------------------------------
---------*/
result=and( result_SPEED_VERY_FAST, 100);
result=and( result_DISTANCE_FAR, result);
result_BRAKES+=(result*BRAKES_HEAVY);
sum_BRAKES+=result;
rule_result[14]=result;

/*-------------------------------------------------------------------
---------
Rule 16
if SPEED is VERY_FAST and
DISTANCE is VERY_FAR then
BRAKES is LIGHT
--------------------------------------------------------------------
---------*/
```

```
result=and( result_SPEED_VERY_FAST, 100);
result=and( result_DISTANCE_VERY_FAR, result);
result_BRAKES+=(result*BRAKES_LIGHT);
sum_BRAKES+=result;
rule_result[15]=result;

return;
}

/*
*****************************************************************
**********
Function:  initialize_outputs( )
*****************************************************************
**********
*/
static void  initialize_outputs( )
{

result_BRAKES=0;
sum_BRAKES=0;

return;
}

/*
*****************************************************************
**********
Function:  determine_outputs( )
*****************************************************************
**********
*/
static void  determine_outputs( )
{

if( sum_BRAKES > 0 )
  result_BRAKES = result_BRAKES / sum_BRAKES;
else
  result_BRAKES = 0;

return;
}

/*
*****************************************************************
**********
Function:  fld_fuzzy_inputs( )
*****************************************************************
**********
*/
int  fld_fuzzy_inputs(  speed, distance)
int  speed;
int  distance;
{
```

```
int err=0;

if( speed < MIN_SPEED )
  speed = MIN_SPEED;
else if( speed > MAX_SPEED )
  speed = MAX_SPEED;

if( distance < MIN_DISTANCE )
  distance = MIN_DISTANCE;
else if( distance > MAX_DISTANCE )
  distance = MAX_DISTANCE;

eval_SPEED_VERY_SLOW( speed);
eval_SPEED_SLOW( speed);
eval_SPEED_FAST( speed);
eval_SPEED_VERY_FAST( speed);
eval_DISTANCE_VERY_CLOSE( distance);
eval_DISTANCE_CLOSE( distance);
eval_DISTANCE_FAR( distance);
eval_DISTANCE_VERY_FAR( distance);

initialize_outputs( );
evaluate_rules( );
determine_outputs( );

return( err);
}
/*
**********************************************************************
**********
Function:  fld_fuzzy_outputs( )
**********************************************************************
**********
*/
int  fld_fuzzy_outputs(  brakes)
int *brakes;
{
int err=0;

*brakes=(int)result_BRAKES;

return( err);
}
```

```
/*
*******************************************************************
**********
Module: train.h
*******************************************************************
**********
*/
/*
*******************************************************************
**********
This source code has been automatically generated by the
   Fuzzy Logic Designer(tm) Version 1.10 design package
         Copyright (c) 1993  Byte Dynamics Inc.
         14608 E. Olympic Ave. Spokane, Wa. 99216
*******************************************************************
**********
*/

#define NUM_INPUTS    2
#define NUM_OUTPUTS   1
#define NUM_RULES     16

#define MIN_SPEED   0
#define MAX_SPEED  100
#define MIN_DISTANCE  0
#define MAX_DISTANCE  500

#define MIN_BRAKES  0
#define MAX_BRAKES  100
#define BRAKES_VERY_LIGHT    5
#define BRAKES_LIGHT        30
#define BRAKES_HEAVY        70
#define BRAKES_VERY_HEAVY     99
```

Appendix D

Examples of Consumer Products That Use Fuzzy Logic

Air conditioner	Hitachi
	Matsushita
	Mitsubishi
	Sharp
Anti-lock brakes	Nissan
Clothes dryer	Sanyo
	Matsushita
Copy machine	Canon
Dish washer	Matsushita
Electric fan	Sanyo
Kerosene fan heater	Matsushita
	Mitsubishi
	Toshiba
	Fujitsu
	Corona
	Toyotomi
	Sanyo
	Sharp
Microwave oven	Hitachi
	Matsushita
	Sanyo
	Toshiba
	Sony
Refrigerator	Sharp
Rice cooker	Matsushita
	Sanyo
	Hitachi
	Sharp

Still camera	Canon
	Minolta
Television	Goldstar
	Hitachi
	Samsung
	Sony
Toaster	Sony
Vacuum cleaner	Hitachi
	Matsushita
	Toshiba
Video camcorder	Panasonic
Washing machine	Goldstar
	Hitachi
	Matsushita
	Samsung
	Sanyo
	Sharp

Appendix E

Points to Ponder

1. Next to the podium, the logic professor places two glasses: one is full of water, in case he gets thirsty; the other is empty, in case he does not get thirsty.
   Unknown
2. All traditional logic habitually assumes that precise symbols are being employed. It is therefore not applicable to this terrestrial life. . .
   Bertrand Russell
3. So far as the laws of mathematics refer to reality, they are not certain. And so far as they are certain, they do not refer to reality.
   Albert Einstein
4. Every system is its own best analogue.
   Andrew D. Booth
5. The Yellow Emperor said . . . "if we want to return again to the roots, I'm afraid we'll have a hard time of it!"
   Chuang Tzu
6. We have to think that if we know one, we know two because one and one are two. We are finding that we must learn a great deal about *and*.
   Freeman Dyson
7. There is nothing more practical than a good theory.
   Leonid I. Brezhnev
8. The Japanese government had decided that the electronics industry was too important to be left only to businessmen.
   Clyde Prestowitz
9. I cannot think of any problem that could not be solved better by ordinary logic.
   William Kahan
10. Anything that can be done with fuzzy logic . . . can better be done with probability.
    Dennis Lindley

11. *Fuzzification* is a kind of scientific permissiveness . . . I cannot conceive of fuzzification as a viable alternative to scientific method.

    Rudolf E. Kalman

12. Fuzzy logic is the cocaine of science.

    William Kahan

13. Fuzziness is probability in disguise

    Myron Trbus

14. Some day the enormous amount of effort the Japanese have put into fuzzy logic will earn Lotfi a medal that will be awarded by the president of the United States. God knows we have to do something to slow these guys down.

    William Kahan

15. In questions of science the authority of a thousand is not worth the humble reasoning of a single individual.

    Galileo Galilei

16. Do not confuse the moon with the finger that points at it.

    Zen Proverb

17. What we observe is not nature itself, but nature exposed to our method of questioning.

    Werner Heisenberg

18. If fuzziness exists, the physical consequences are universal, and the sociological consequence is startling: scientists, especially physicists, have overlooked an entire mode of reality.

    Bart Kosko

19. In my opinion, a mathematician, in so far as he is a mathematician, need not preoccupy himself with philosophy.

    Henri L. Lebesgue

20. I learned many years ago never to waste time trying to convince my colleagues.

    Albert Einstein

21. We shall have to learn to refrain from doing things merely because we know how to do them.

    Theodore Fox

22. I knew that just by choosing the label fuzzy I was going to find myself in the midst of a controversy.

    Lotfi Zadeh

23. We had a funny discussion over the term fuzzy. The company is highly respected and asked us to find another name.

    Jens-Jorgen Østergaard

24. We fuzzy control theorists have brought control theory from seventh heaven to the ground.

    Michio Sugeno

25. In a matter of two to five years, you'll find most expert systems will use fuzzy logic.

    Lotfi Zadeh

26. Fuzzy logic is saving us thousands if not hundreds of thousands of dollars on a yearly basis.

    Peter Holmblad

27. What is now proved was once only imagined.

    William Blake

INDEX