



图灵原版计算机科学系列



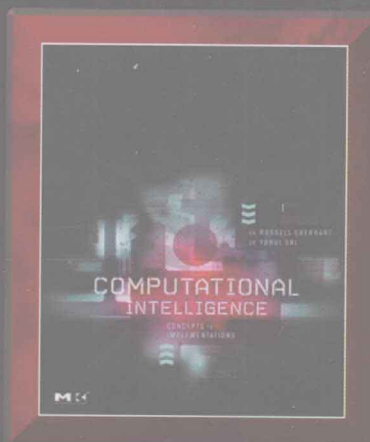
Computational Intelligence
Concepts to Implementations

计算智能

从概念到实现

(英文版)

[美] Russell C. Eberhart 著
Yuhui Shi



人民邮电出版社
POSTS & TELECOM PRESS



图灵原版计算机科学系列

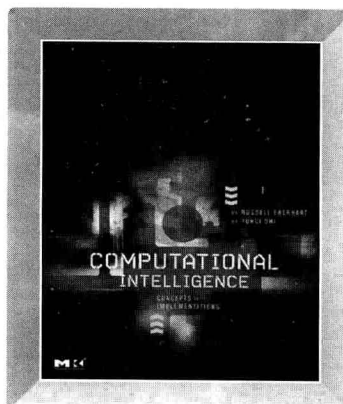
Computational Intelligence
Concepts to Implementations

计算智能

从概念到实现

(英文版)

[美] Russell C. Eberhart 著
Yuhui Shi



人民邮电出版社
北京

图书在版编目 (CIP) 数据

计算智能：从概念到实现 = Computational Intelligence: Concepts to Implementations, First Edition: 英文/ (美) 埃伯哈特 (Eberhart, R. C.), (美) 史玉回著. —北京: 人民邮电出版社, 2009.2

(图灵原版计算机科学系列)

ISBN 978-7-115-19403-9

I. 计… II. ①埃… ②史… III. 人工智能—神经网络—计算—英文 IV. TP183

中国版本图书馆CIP数据核字 (2008) 第198509号

内 容 提 要

本书面向智能系统学科的前沿领域, 系统地讨论了计算智能的理论、技术及其应用, 比较全面地反映了计算智能研究和应用的最新进展。书中涵盖了模糊控制、神经网络控制、进化计算以及其他一些技术及应用的内容。本书提供了大量的实用案例, 重点强调实际的应用和计算工具, 这些对于计算智能领域的进一步发展是非常有意义的。本书取材新颖, 内容深入浅出, 材料丰富, 理论密切结合实际, 具有较高的学术水平和参考价值。

本书可作为高等院校相关专业高年级本科生或研究生的教材及参考用书, 也可供从事智能科学、自动控制、系统科学、计算机科学、应用数学等领域研究的教师和科研人员参考。

图灵原版计算机科学系列

计算智能：从概念到实现 (英文版)

-
- ◆ 著 [美] Russell C. Eberhart Yuhui Shi
责任编辑 杨海玲
 - ◆ 人民邮电出版社出版发行 北京市崇文区夕照寺街14号
邮编 100061 电子函件 315@ptpress.com.cn
网址 <http://www.ptpress.com.cn>
北京铭成印刷有限公司印刷
 - ◆ 开本: 800 × 1000 1/16
印张: 30.5
字数: 586千字 2009年2月第1版
印数: 1-2 000册 2009年2月北京第1次印刷

著作权合同登记号 图字: 01-2008-5827号

ISBN 978-7-115-19403-9/TP

定价: 69.00元

读者服务热线: (010) 88593802 印装质量热线: (010) 67129223

反盗版热线: (010) 67171154

Preface

Several computational analytic tools have matured in the last 10 to 15 years that facilitate solving problems that were previously difficult or impossible to solve. These new analytical tools, known collectively as *computational intelligence tools*, include artificial neural networks, fuzzy systems, and evolutionary computation. They have recently been combined among themselves as well as with more traditional approaches, such as statistical analysis, to solve extremely challenging problems. Diagnostic systems, for example, are being developed that include Bayesian, neural network, and rule-based diagnostic modules, evolutionary algorithm-based explanation facilities, and expert system shells. All of these components work together in a “seamless” way that is transparent to the user, and they deliver results that significantly exceed what is available with any single approach.

At a system prototype level, computational intelligence (CI) tools are capable of yielding results in a relatively short time. For instance, the implementation of a conventional expert system often takes one to three years and requires the active participation of a “knowledge engineer” to build the knowledge and rule bases. In contrast, computational intelligence system solutions can often be prototyped in a few weeks to a few months and are implemented using available engineering and computational resources. Indeed, computational intelligence tools are capable of being applied in many instances by “domain experts” rather than solely by “computer gurus.”

This means that biomedical engineers, for example, can solve problems in biomedical engineering without relying on outside computer science expertise such as that required to build knowledge bases for classical expert systems. Furthermore, innovative ways to combine CI tools are cropping up every day. For example, tools have been developed that incorporate knowledge elements with neural networks, fuzzy logic, and evolutionary computing theory. Such tools are able to solve quickly classification and clustering problems that would be extremely time consuming using other techniques.

The concepts, paradigms, algorithms, and implementation of computational intelligence and its constituent methodologies—evolutionary computation, neural networks, and fuzzy logic—are the focus of this book. In addition, we emphasize practical applications throughout, that is, how to apply the concepts, paradigms, algorithms, and implementations discussed to practical problems in engineering and computer science. This emphasis culminates in the real-world case

studies in a final chapter, which are available on this book's web site at <http://www.computelligence.org/issue/CICI/CICI.html>.

Computational intelligence is closely related to the field called “soft computing.” There is, in fact, a significant overlap. According to Lotfi Zadeh (1998), the inventor of fuzzy logic and one of the leading proponents of soft computing:

Soft computing is not a single methodology. Rather, it is a consortium of computing methodologies which collectively provide a foundation for the conception, design and deployment of intelligent systems. At this juncture, the principal members of soft computing are fuzzy logic (FL), neurocomputing (NC), genetic computing (GC), and probabilistic computing (PC), with the last subsuming evidential reasoning, belief networks, chaotic systems, and parts of machine learning theory. In contrast to traditional hard computing, soft computing is tolerant of imprecision, uncertainty and partial truth. The guiding principle of soft computing is: “exploit the tolerance for imprecision, uncertainty and partial truth to achieve tractability, robustness, low solution cost and better rapport with reality.”

Zadeh also believes that soft computing is serving as the foundation for the emerging field of computational intelligence, and that “In this perspective, the difference between traditional AI [artificial intelligence] and computational intelligence is that AI is based on hard computing whereas CI is based on soft computing” (Zadeh 1994). We believe that soft computing is a large subset of computational intelligence. We heartily agree with him when he says, “Hybrid intelligent systems are definitely the wave of the future” (Zadeh 1994).

Some of the material in this book is adapted from *Computational Intelligence PC Tools* by Eberhart, Dobbins, and Simpson (Academic Press 1996). The extensive rewrite and reorganization of that material reflect the change in our perception of computational intelligence that has occurred over the years. That change is reflected in an increased emphasis on evolutionary computation as providing a foundation for CI. It also features significant recent developments in particle swarm optimization and other evolutionary computation tools.

The primary intended audience for *Computational Intelligence: Concepts to Implementations* comprises researchers and graduate students with engineering or computer science backgrounds and those with a special interest in computational intelligence and/or system adaptation. One of the authors [RE] has taught a CI introductory course for several years; the material in this book was developed to support that course. Other audiences include researchers in fields such as cognitive science and the physical sciences and those who are motivated to learn about computational intelligence via self-study. We assume this book's users understand the basic concepts of classical (two-valued) logic, classical set theory, and elementary probability theory. We also assume that readers have a familiarity with computers and a very basic familiarity with calculus. Knowledge of a computer language such as Java, C, or Visual BASIC is very helpful but not required.

The implementation chapters frequently refer to and list portions of computer code. In Chapters 4 and 6 we use the most common general-purpose, procedural programming language, C, to implement the evolutionary algorithms and the artificial neural networks. Data structures, routines, and finite state machines are used extensively in the C programming. In Chapters 8 and 9, reflecting programming language evolution trends, we use an object-oriented programming language instead of the procedural programming language C to implement the fuzzy systems and evolutionary fuzzy systems. There are a variety of object-oriented languages, such as C++, Java, and C#. We use C++ here primarily because it can be looked at as an extension of the C language.

Organization of the Book

This book is divided into twelve chapters. Chapters 1 and 2 lay the groundwork for the topic, introducing computational intelligence and its foundations. The next portion of the book includes the “backbone” chapters on the three main constituents of CI: evolutionary computation, neural networks, and fuzzy logic, in that order. This order provides an initial focus on evolutionary computation, which is presented as providing a foundation for development of computational intelligence tools involving neural networks and fuzzy logic. For instance, when we discuss neural networks, we see how evolutionary computation can be used to evolve the weights and structure of feedforward neural networks, and with fuzzy logic, we examine evolutionary computation applications to tools built using fuzzy logic. In other words, the evolutionary computation theme pervades this book. Within each backbone chapter, we discuss the histories of computational intelligence, evolutionary computation, neural networks, and fuzzy logic.

We follow each backbone chapter with a chapter discussing implementation and examples. Each one contains a section on implementation considerations that addresses features frequently incorporated into these implementations, which features we chose and why we chose them, and the guidelines to using them, as well as interactions among them. The implementation chapters are intended to provide readers with the insight to clearly understand “canned,” commercially packaged software applications and to enable a more thorough understanding of software and hardware implementation issues for CI paradigms.

Each chapter ends with exercises.

Chapters' Contents

Chapter 1, Foundations, defines terms used throughout the book and briefly reviews biological and behavioral motivations for the constituent methodologies of computational intelligence. This is followed by a brief review of the major application areas

for each methodology, as well as of CI. The chapter concludes with a review of major computational intelligence application areas.

Chapter 2, Computational Intelligence, launches directly into the core subject of this book. We first review the concepts of adaptation and self-organization, key to our view of computational intelligence. Then we summarize the brief history of the CI field, viewing it from the perspectives of other researchers. This leads us into a discussion of the relationships among the three major components and how they cooperate and/or are integrated into a computational intelligence system. We present our definition of computational intelligence, supported by diagrams that place it into context.

Chapter 3, Evolutionary Computation: Concepts and Paradigms, has been adapted from the Evolutionary Computation Theory and Paradigms chapter in *Swarm Intelligence* (Kennedy, Eberhart, and Shi 2001) with updates and augmentations, including recent developments in particle swarm optimization and other evolutionary computation approaches. After reviewing the history of evolutionary computation and giving an overview of the field, we discuss its main paradigms: genetic algorithms, evolutionary programming, evolution strategies, genetic programming, and particle swarm optimization.

Chapter 4, Evolutionary Computation Implementations, discusses factors to consider when implementing evolutionary computation paradigms and presents two implementation examples: a canonical genetic algorithm and a real-valued particle swarm that can be run in single-swarm or multiswarm configurations.

Chapter 5, Neural Network Concepts and Paradigms, first briefly presents an overview of the history of neural networks, then examines what they are and why they are useful. A discussion of neural network components and terminology follows, with a review of neural network topologies. A more detailed look at neural network learning and recall comes next, focusing on three of the most common neural network paradigms: back-propagation, learning vector quantization, and self-organizing feature map networks. These networks represent the two basic learning types: supervised learning (back-propagation) and unsupervised learning (learning vector quantization and self-organizing feature maps). We also briefly discuss hybrid networks and recurrent networks. Finally, considerations of preprocessing and post-processing are evaluated.

Chapter 6, Neural Network Implementations, discusses factors to consider when implementing artificial neural networks and presents four implementation examples: back-propagation, learning vector quantization, self-organizing feature maps, and evolutionary neural networks.

Chapter 7, Fuzzy Systems Concepts and Paradigms, leads off with a brief review of the history of the field, followed by an examination of fuzzy sets and fuzzy logic, the concepts of fuzzy sets, and approximate reasoning. We stress the differences between fuzzy logic and probability, and we present both Mamdani and Takagi–Sugeno–Kang approaches to the design and analysis of fuzzy systems. The chapter

concludes with a look at some design considerations and special topics related to fuzzy systems.

Chapter 8, Fuzzy System Implementations, discusses factors to consider when implementing fuzzy systems and presents two implementation examples: a traditional fuzzy rule system and an evolutionary fuzzy rule system. The evolutionary fuzzy rule system provides a transition into computational intelligence systems.

Chapter 9, Computational Intelligence Implementations, reflects recent developments in the field, including evolutionary fuzzy systems and approaches to system adaptation using computational intelligence. We expand the discussion of the interaction and cooperation among the three basic components of CI and include a section on adaptive evolutionary computation using fuzzy systems.

Chapter 10, Performance Metrics, includes a number of system performance measures not generally used in other disciplines. Included are percent correct, sum-squared error, absolute error, normalized error, receiver operating characteristic curves, recall and precision, confusion matrices, and the chi-squared test.

Chapter 11, Analysis and Explanation, presents several tools that are helpful in assessing and explaining how well a computational intelligence tool is doing its job. Included are sensitivity analyses, Hinton diagrams for neural networks, and the use of evolutionary computing tools for analysis. An example of using particle swarm to develop an explanation facility is included in this chapter.

The book concludes with Chapter 12, Case Study Summaries, which provides examples of practical applications. This “virtual” chapter is located on the book’s web site. Having it there makes it a “living” chapter that can be updated periodically. We will add new case studies from time to time and delete older ones as they become obsolete. We invite you, the reader, to submit case studies you would like to have considered for inclusion. (Please see the web site for more information about this.) Among the initial case studies posted are two based on recent work by us, the authors, including one on human EEG analysis and another on optimization of logistics operations. Other case studies discussed in detail are schedule optimization and control system design. Several other case study examples are briefly reviewed.

A bibliography concludes the book. The glossary is a “virtual” one that is located, with Chapter 12, on this book’s web site <http://www.computelligence.org/issue/CICI/CICI.html>.

Our Approach: What This Book Is, and Is Not, About

This book asserts that computational intelligence rests on a foundation of evolutionary computation. This is certainly not the only way to view computational intelligence, but so far in the authors’ experience, it has proved useful and effective.

It is about *computational tools* that you can use in practical applications. Although the authors have backgrounds in engineering and computer science, CI tools are just as applicable to problems in other fields such as cognitive science and business.

This book is about self-organization, which is closely related to *emergent computation*. *Self-organization* involves simple processes that lead to complex results, and the whole being greater than the sum of its parts. As Stephen Wolfram (1994) said, “It is possible to make things of great complexity out of things that are very simple. There is no conservation of simplicity.”

It is about *complex adaptive systems*, a term that describes nonlinear systems comprising the interaction of numerous adaptive elements, or entities. The concepts of self-organization and complexity are related, as we discuss later.

This book is not an exhaustive treatise on all permutations and variations of computational intelligence and its constituent methodologies. If you want an exhaustive discussion of artificial neural network paradigms, for instance, you’ll need to turn to another book. We present only those paradigms we believe provide the most useful tools for someone solving practical problems.

It is not a compendium of mathematical derivations and proofs. We present only those few we believe are essential to gaining a working-level understanding of how and why the computational tools work.

This book is not about agents. Most of our computational intelligence tools do not qualify as “agents” because they lack the required autonomy and specialization. They can, however, be incorporated into intelligent agents and agent systems.

It is not about life. We nip around the edges of artificial life in a few places, but we don’t address the question “What is alive?” (We do, however, share some preliminary thoughts on that subject.) We also do not address the search for artificial intelligence (whatever that is) or even for a computational intelligence tool from which intelligent behavior will emerge. Our focus is on solving problems.

Throughout the text, additional aspects of our approach and philosophy should become evident, perhaps a little bit at a time. First, when considering computational intelligence tools and systems, traditional distinctions between hardware and software get a bit blurred; distinctions between data and program are often almost nonexistent. Second, our emphasis is on problem solving and applications rather than physiological, biological, or behavioral plausibility. We do not pay too much attention to whether the CI tools reflect what actually goes on in the brain or any other part of a biological organism. Third, we believe that the activities of a computational intelligence application developer and user are often somewhat different from those in other technical areas.

Developing computational intelligence applications requires the developer to play two roles. The first is the hands-on active design, develop, test, and debug role that is fairly common in other technical areas. The second, as important as the first, is a more passive observation and analytical thinking role. Results from a computational intelligence tool are often not what was expected. Most of the time, if the developer takes the time to observe and think, rather than “bash to fit and paint to match,” something very useful can be learned.

Web Site Details

The authors' web site for this book is <http://www.computelligence.org/issue/CICI/CICI.html>. (There is a link to this site from the publisher's web site.) Software implementations are written for the Windows and/or Java environment, and executable versions of software described in the implementation chapters are located and maintained on the web site. Included as part of each implementation are the ancillary files—a run file and a data file—needed to run the implementation. In addition, output (results) files, obtained by the authors using the executable and ancillary files, are provided. You may want to rename these output files, or move them to another directory, so that you can compare your results with those of the authors.

We'd like to emphasize that the software is not just for demonstration; you can use it for many real-world applications. The C and C++ source code has been written using the Borland C++ 4.5 development environment. The Java code will run on any computer that supports the Java Virtual Machine; this includes machines running Windows, Unix, and Macintosh operating systems.

Of special note are the recent variations of particle swarm optimization that have been integrated into the EC theory and paradigms chapter and the EC implementations chapter. Source code is provided on the web site for some of the implementations so that you can modify the software for specific applications.

Some of our software can be run using a web browser. Other software, including source code, is useful only after downloading it from the book's web site. Approximately 600 slides that cover the material in this book are available to instructors (or anyone else) at no cost. These slides, configured as Word files, are downloadable from the web site. The site also contains hyperlinks to other resource information on the Internet related to subjects in this book.

A significant amount of source code is also on the web site. A total of eight software modules are available, both as executables and as source code:

- Genetic algorithm
- Particle swarm optimization (including multiple swarms)
- Back-propagation neural network
- Learning vector quantization neural network
- Self-organizing feature map neural network
- Evolutionary back-propagation neural network
- Fuzzy rule system
- Evolutionary fuzzy rule system

We ask that you send the authors a payment of US \$25 per software module of source code (\$150 for all of the source code) if you find it useful. We are relying on your honesty. (The address is on the web site with the software.)

Finally, as described previously, Chapter 12, Case Studies, is available on the web site.

Acknowledgments

Each of us has numerous people who should be acknowledged; we mention only a few.

Russ Eberhart: First, I want to acknowledge my wife Francie and son Sean who put up with a higher than usual absence rate of their spouse and father, respectively. I also want to acknowledge my son Mark, a three-time cancer survivor, who has taught me what courage is. Special thanks go to my students in ECE 536, Introduction to Computational Intelligence. They were the guinea pigs. Sometimes, just from their eyes glazing over, I knew that a section needed to be rewritten (or deleted). Their patience is appreciated, and their input has been invaluable.

Yuhui Shi: I would like to thank my parents and parents-in-law for taking good care of my daughter Melissa Xueyin Shi and my son Nicholas Yuge Shi so that I had plenty of quality time to work on this book. My thanks also go to professors Zhenya He of Southeast University, M. N. S. Swamy and M. Omair Ahmad of Concordia University, Xin Yao of the University of Birmingham, Jinhung Kim of the Korean Advanced Institute of Science and Technology, and to Russell C. Eberhart, who are my mentors and have paved the way for me in my career development.

Both of us acknowledge the contributions of our technical reviewers. Their insights resulted in improvements in both the organization and content of this book. Finally, we are grateful to the team at Morgan Kaufmann Publishers who worked diligently with us throughout the process of writing, editing, and production. Working with Denise Penrose, Diane Cerra, Emilia Thiuri, Marilyn Rash, and Mary James has been a pleasure and a learning experience.

Contents

chapter one

Foundations

1

- Definitions 2
- Biological Basis for Neural Networks 4
 - Neurons 4
 - Biological versus Artificial Neural Networks 5
 - Biological Basis for Evolutionary Computation 7
 - Chromosomes 7
 - Biological versus Artificial Chromosomes 8
- Behavioral Motivations for Fuzzy Logic 9
- Myths about Computational Intelligence 10
- Computational Intelligence Application Areas 11
 - Neural Networks 12
 - Evolutionary Computation 13
 - Fuzzy Logic 14
- Summary 14
- Exercises 14

chapter two

Computational Intelligence

17

- Adaptation 18
 - Adaptation versus Learning 19
 - Three Types of Adaptation 20
 - Three Spaces of Adaptation 25
- Self-organization and Evolution 26
 - Evolution beyond Darwin 28
- Historical Views of Computational Intelligence 29
- Computational Intelligence as Adaptation and Self-organization 30
- The Ability to Generalize 34

Computational Intelligence and Soft Computing versus Artificial Intelligence and Hard Computing	35
Summary	36
Exercises	38

chapter three

Evolutionary Computation Concepts and Paradigms

39

History of Evolutionary Computation	40
Genetic Algorithms	40
Evolutionary Programming	44
Evolution Strategies	44
Genetic Programming	45
Particle Swarm Optimization	45
Toward Unification	47
Evolutionary Computation Overview	47
EC Paradigm Attributes	48
Implementation	49
Genetic Algorithms	51
Overview of Genetic Algorithms	51
A Sample GA Problem	52
Review of GA Operations in the Simple Example	56
Schemata and the Schema Theorem	64
Comments on Genetic Algorithms	67
Evolutionary Programming	68
Evolutionary Programming Procedure	69
Finite State Machine Evolution for Prediction	69
Function Optimization	74
Comments on Evolutionary Programming	75
Evolution Strategies	75
Selection	78
Key Issues in Evolution Strategies	80
Genetic Programming	81
Particle Swarm Optimization	87
Developments	87
Resources	92
Summary	92
Exercises	93

chapter four

Evolutionary Computation Implementations

95

Implementation Issues	97
Homogeneous versus Heterogeneous Representation	97

Population Adaptation versus Individual Adaptation	98
Static versus Dynamic Adaptation	99
Flowcharts versus Finite State Machines	100
Handling Multiple Similar Cases	100
Allocating and Freeing Memory Space	102
Error Checking	102
Genetic Algorithm Implementation	103
Programming Genetic Algorithms	103
Running the GA Implementation	116
Particle Swarm Optimization Implementation	118
Programming the PSO Implementation	118
Programming the Co-evolutionary PSO	125
Running the PSO Implementation	140
Summary	142
Exercises	142

chapter five

Neural Network Concepts and Paradigms 145

Neural Network History	146
Where Did Neural Networks Get Their Name?	146
The Age of Camelot	147
The Dark Age	153
The Renaissance	159
The Age of Neoconnectionism	164
The Age of Computational Intelligence	165
What Neural Networks Are and Why They Are Useful	165
Neural Network Components and Terminology	168
Terminology	168
Input and Output Patterns	169
Network Weights	170
Processing Elements	171
Processing Element Activation Functions	173
Neural Network Topologies	176
Terminology	176
Two-layer Networks	176
Multilayer Networks	178
Neural Network Adaptation	179
Terminology	180
Hebbian Adaptation	181
Competitive Adaptation	182
Multilayer Error Correction Adaptation	183
Summary of Adaptation Procedures	187

Comparing Neural Networks and Other Information Processing Methods	188
Stochastic Approximation	188
Kalman Filters	188
Linear and Nonlinear Regression	188
Correlation	189
Bayes Classification	189
Vector Quantization	189
Radial Basis Functions	190
Computational Intelligence	190
Preprocessing	190
Selecting Training, Test, and Validation Datasets	191
Preparing Data	192
Postprocessing	195
Denormalization of Output Data	195
Summary	196
Exercises	196

chapter six

Neural Network Implementations

197

Implementation Issues	198
Topology	199
Back-propagation Network Initialization and Normalization	199
Learning Vector Quantizer Network Initialization and Normalization	202
Feedforward Calculations for the Back-propagation Network	203
Feedforward Calculations for the LVQ-I Net	206
Back-propagation Supervised Adaptation by Error Back-propagation	206
LVQ Unsupervised Adaptation Calculations	210
The LVQ Supervised Adaptation Algorithm	211
Issues in Evolving Neural Networks	212
Advantages and Disadvantages of Previous Evolutionary Approaches	214
Evolving Neural Networks with Particle Swarm Optimization	216
Back-propagation Implementation	218
Programming a Back-propagation Neural Network	218
Running the Back-propagation Implementation	233
The Kohonen Network Implementations	235
Programming the Learning Vector Quantizer	235
Running the LVQ Implementation	249
Programming the Self-organizing Feature Map	250
Running the SOFM Implementation	260
Evolutionary Back-propagation Network Implementation	262
Programming the Evolutionary Back-propagation Network	262

Running the Evolutionary Back-propagation Network	264
Summary	265
Exercises	265

chapter seven

Fuzzy Systems Concepts and Paradigms

269

History	270
Fuzzy Sets and Fuzzy Logic	275
Logic, Fuzzy and Otherwise	275
Fuzziness Is Not Probability	276
The Theory of Fuzzy Sets	277
Fuzzy Set Membership Functions	279
Linguistic Variables	281
Linguistic Hedges	282
Approximate Reasoning	283
Paradoxes in Fuzzy Logic	283
Equality of Fuzzy Sets	284
Containment	285
NOT: The Complement of a Fuzzy Set	285
AND: The Intersection of Fuzzy Sets	286
OR: The Union of Fuzzy Sets	287
Compensatory Operators	288
Fuzzy Rules	290
Fuzzification	290
Fuzzy Rules Fire in Parallel	293
Defuzzification	294
Other Defuzzification Methods	296
Measures of Fuzziness	297
Developing a Fuzzy Controller	301
Why Fuzzy Control	301
A Fuzzy Controller	302
Building a Mamdani-type Fuzzy Controller	303
Fuzzy Controller Operation	310
Takagi–Sugeno–Kang Method	310
Summary	313
Exercises	314

chapter eight

Fuzzy Systems Implementations

315

Implementation Issues	316
Fuzzy Rule Representation	316

Evolutionary Design of a Fuzzy Rule System	317
An Object-oriented Language: C++	320
Fuzzy Rule System Implementation	320
Programming Fuzzy Rule Systems	320
Running the Fuzzy Rule System	345
Iris Dataset Application	351
Evolving Fuzzy Rule Systems	353
Programming the Evolutionary Fuzzy Rule System	353
Running the Evolutionary Fuzzy Rule System	366
Summary	371
Exercises	371

chapter nine

Computational Intelligence Implementations	373
Implementation Issues	374
Adaptation of Genetic Algorithms	375
Fuzzy Adaptation	375
Knowledge Elicitation	377
Fuzzy Evolutionary Fuzzy Rule System Implementation	378
Programming the Fuzzy Evolutionary Fuzzy Rule System	378
Running the Fuzzy Evolutionary Fuzzy Rule System	381
Choosing the Best Tools	382
Strengths and Weaknesses	382
Modeling and Optimization	383
Practical Issues	384
Applying Computational Intelligence to Data Mining	385
An Example Data Mining System	386
Summary	387
Exercises	388

chapter ten

Performance Metrics	389
General Issues	390
Selecting Gold Standards	390
Partitioning the Patterns for Training, Testing, and Validation	391
Cross Validation	392
Fitness and Fitness Functions	393
Parametric and Nonparametric Statistics	394
Percent Correct	395
Average Sum-squared Error	396