FUZZY EXPERT SYSTEMS AND FUZZY REASONING

FUZZY EXPERT SYSTEMS AND FUZZY REASONING

William Siler
Kemp-Carraway Heart Institute,
Birmingham, AL 35234

James J. Buckley
Mathematics Dept., University of Alabama at Birmingham,
Birmingham, AL 35294

JOHN WILEY & SONS, INC.

This book is printed on acid-free paper. ♾

Published by John Wiley & Sons, Inc., Hoboken, New Jersey.
Published simultaneously in Canada.

Library of Congress Cataloging-in-Publication Data:

Siler, William.
Fuzzy expert systems and fuzzy reasoning / by William Siler, James J. Buckley.
p. cm.
Includes bibliographical references and index.
ISBN 0-471-38859-9
1. Expert systems (Computer science) 2. Fuzzy systems. I. Buckley, James J., 1936-II.
Title

QA76.76.E95S557 2005
006.3′3–dc22

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

CONTENTS

Preface **xiii**

1 Introduction **1**

1.1 Characteristics of Expert Systems 2
1.1.1 Production Systems 3
1.1.2 Data-Driven Systems 4
1.1.3 Special Features of Fuzzy Systems 4
1.1.4 Expert Systems for Fuzzy Control and for Fuzzy Reasoning 5
1.2 Neural Nets 6
1.3 Symbolic Reasoning 7
1.4 Developing a Rule-Based Expert System 8
1.5 Fuzzy Rule-Based Systems 9
1.6 Problems in Learning How to Construct Fuzzy Expert Systems 10
1.7 Tools for Learning How to Construct Fuzzy Expert Systems 11
1.8 Auxiliary Reading 12
1.9 Summary 13
1.10 Questions 13

2 Rule-Based Systems: Overview **15**

2.1 Expert Knowledge: Rules and Data 15
2.2 Rule Antecedent and Consequent 16
2.2.1 Antecedents 16
2.2.2 Admissible Data Types 17
2.2.3 Consequents 17
2.3 Data-Driven Systems 18
2.4 Run and Command Modes 19
2.4.1 Run Mode: Serial and Parallel Rule Firing 21
2.4.2 Checking which Rules are Fireable: The RETE Algorithm 22
2.4.3 Serial Rule Firing 22
2.4.4 Parallel Rule-Firing 23
2.5 Forward and Backward Chaining 24
2.6 Program Modularization and Blackboard Systems 24
2.7 Handling Uncertainties in an Expert System 25

2.8 Summary 26
2.9 Questions 28

3 Fuzzy Logic, Fuzzy Sets, and Fuzzy Numbers: I 29

3.1 Classical Logic 29
3.1.1 Evaluating A AND B and A OR B 30
3.1.2 The Implication Operator 34
3.2 Elementary Fuzzy Logic and Fuzzy Propositions 36
3.3 Fuzzy Sets 38
3.3.1 Discrete Fuzzy Sets 39
3.3.2 Fuzzy Numbers 40
3.3.3 Linguistic Variables and Membership Functions 42
3.4 Fuzzy Relations 43
3.4.1 Matrices of Truth Values 43
3.4.2 Relations Between Sets 44
3.5 Truth Value of Fuzzy Propositions 47
3.5.1 Comparing Single-Valued Data 47
3.5.2 Comparing Fuzzy Numbers 48
3.6 Fuzzification and Defuzzification 49
3.6.1 Fuzzification 49
3.6.2 Defuzzification 51
3.7 Questions 54

4 Fuzzy Logic, Fuzzy Sets, and Fuzzy Numbers: II 57

4.1 Introduction 57
4.2 Algebra of Fuzzy Sets 57
4.2.1 T-Norms and t-Conorms: Fuzzy AND and OR Operators 57
4.2.2 Correlation Fuzzy Logic 60
4.2.3 Combining Fuzzy Numbers 63
4.3 Approximate Reasoning 65
4.4 Hedges 69
4.5 Fuzzy Arithmetic 72
4.5.1 Extension Principle 72
4.5.2 Alpha-Cut and Interval Arithmetic 73
4.5.3 Comparison of Alpha-Cut and Interval Arithmetic Methods 74
4.6 Comparisons between Fuzzy Numbers 74
4.6.1 Using the Extension Principle 74
4.6.2 Alternate Method 76
4.7 Fuzzy Propositions 78
4.8 Questions 82

5 Combining Uncertainties **85**

5.1 Generalizing AND and OR Operators 85
5.1.1 Correlation Logic: A Family of Fuzzy Logical Operators that Obeys Excluded Middle and Non-Contradiction Laws 86
5.2 Combining Single Truth Values 88
5.3 Combining Fuzzy Numbers and Membership Functions 89
5.4 Bayesian Methods 92
5.5 The Dempster–Shafer Method 94
5.6 Summary 96
5.7 Questions 97

6 Inference in an Expert System I **99**

6.1 Overview 99
6.2 Types of Fuzzy Inference 100
6.3 Nature of Inference in a Fuzzy Expert System 100
6.4 Modification and Assignment of Truth Values 102
6.4.1 Monotonic Inference 102
6.4.2 Non-monotonic Inference 103
6.4.3 Downward Monotonic Inference 103
6.5 Approximate Reasoning 105
6.6 Tests of Procedures to Obtain the Truth Value of a Consequent from the Truth Value of Its Antecedent 105
6.6.1 Desirable Properties 105
6.6.2 Summary of Candidate Methods 106
6.6.3 Tests of Methods Against Desirable Properties 107
6.6.4 Implementation of Choices Among Types of Reasoning 109
6.7 Summary 110
6.7.1 Data Types and Truth Values 111
6.7.2 Types of Fuzzy Reasoning 112
6.8 Questions 113

7 Inference in a Fuzzy Expert System II: Modification of Data and Truth Values **115**

7.1 Modification of Existing Data by Rule Consequent Instructions 116
7.2 Modification of Numeric Discrete Fuzzy Sets: Linguistic Variables and Linguistic Terms 117
7.3 Selection of Reasoning Type and Grade-of-Membership Initialization 118
7.4 Fuzzification and Defuzzification 119
7.4.1 Fuzzification and Evaluation of Antecedent Confidence 119
7.4.2 Modification of Consequent Membership Functions 119

7.4.3 Aggregation of Consequent Membership Functions for Each Consequent Linguistic Variable 120
7.4.4 Determination of Defuzzified Value for Consequent Attribute 121
7.5 Non-numeric Discrete Fuzzy Sets 122
7.6 Discrete Fuzzy Sets: Fuzziness, Ambiguity, and Contradiction 124
7.6.1 Fuzziness and Ambiguity 124
7.6.2 Ambiguities and Contradictions 125
7.6.3 Handling Ambiguities and Contradictions 126
7.7 Invalidation of Data: Non-monotonic Reasoning 127
7.8 Modification of Values of Data 129
7.9 Modeling the Entire Rule Space 129
7.9.1 Conventional Method: The Intersection Rule Configuration (IRC) 131
7.9.2 The Combs Union Rule Configuration 132
7.9.3 Performance of the Combs Method 133
7.9.4 Sample IRC and URC Programs 133
7.9.5 Exercises Iris.par and IrisCombs.par 134
7.9.6 Data Mining and the Combs Method 135
7.9.7 Combs Method Summary 135
7.10 Reducing the Number of Classification Rules Required in the Conventional Intersection Rule Configuration 135
7.11 Summary 136
7.11.1 Data Types and Their Truth Values 137
7.11.2 Types of Fuzzy Reasoning 137
7.12 Questions 138

8 Resolving Contradictions: Possibility and Necessity 141

8.1 Definition of Possibility and Necessity 142
8.2 Possibility and Necessity Suitable for MultiStep Rule-Based Fuzzy Reasoning 142
8.2.1 Data Types and Their Truth Values 142
8.2.2 Initialization of Data and Truth Values 144
8.3 Modification of Truth Values During a Fuzzy Reasoning Process 144
8.4 Formulation of Rules for Possibility and Necessity 146
8.5 Resolving Contradictions Using Possibility in a Necessity-Based System 146
8.5.1 Input Data: Useful Ambiguities 147
8.5.2 Output Data: Contradictions 147
8.5.3 Reaching Preliminary Classifications Using Supporting Evidence and Necessities 147
8.5.4 Resolving Contradictions Using Refuting Evidence and Possibilities 148

8.6 Summary 149
8.7 Questions 149

9 Expert System Shells and the Integrated Development Environment (IDE) 151

9.1 Overview 151
9.2 Help Files 153
9.3 Program Editing 153
9.4 Running the Program 154
9.5 Features of General-Purpose Fuzzy Expert Systems 154
9.6 Program Debugging 155
9.7 Summary 156
9.8 Questions 157

10 Simple Example Programs 159

10.1 Simple FLOPS Programs 159
10.2 Numbers.fps 159
10.2.1 Program Listing 159
10.2.2 Program Structure 161
10.2.3 Running the Program 161
10.2.4 Using Basic Debugging Commands 162
10.2.5 Confidence Terminology: Antecedent Confidence, Rule Confidence, and Posterior Confidence (pconf) 163
10.3 Sum.fps 164
10.3.1 Program Listing 164
10.3.2 Running sum.fps 166
10.3.3 Running sum.fps with Debugging Commands prstack; and Run 1; 167
10.4 Sum.par 168
10.4.1 Program Listing 169
10.4.2 Running sum.par 170
10.4.3 Running sum.par with prstack; and Run 1; Commands 171
10.5 Comparison of Serial and Parallel FLOPS 173
10.6 Membership Functions, Fuzzification and Defuzzification 173
10.6.1 Membership Functions in FLOPS/ 173
10.6.2 Fuzzifying Numbers in FLOPS 175
10.6.3 Defuzzification in FLOPS 177
10.7 Summary 179
10.8 Questions 180

11 Running and Debugging Fuzzy Expert Systems I: Parallel Programs 181

11.1 Overview 181
11.2 Debugging Tools 181

11.3 Debugging Short Simple Programs 182
11.4 Isolating the Bug: System Modularization 189
11.5 The Debug Run 189
11.6 Interrupting the Program for Debug Checks 190
11.7 Locating Program Defects with Debug Commands 191
11.7.1 Program Structure 191
11.7.2 Sample Debugging Session 192
11.8 Summary 196
11.9 Questions 197

12 Running and Debugging Expert Systems II: Sequential Rule-Firing 199

12.1 Data Acquisition: From a User Versus Automatically Acquired 199
12.2 Ways of Solving a Tree-Search Problem 201
12.3 Expert Knowledge in Rules; auto1.fps 202
12.3.1 auto1.fps Program: Partial Listing 202
12.3.2 Running auto1.fps 204
12.4 Expert Knowledge in a Database: auto2.fps 207
12.4.1 Expert Knowledge Database Structure 207
12.4.2 auto2.fps Program Listing 208
12.4.3 Running and Debugging auto2.fps 210
12.4.4 Advantages of Database-Defined Tree Search Method 214
12.5 Other Applications of Sequential Rule Firing 214
12.5.1 Missionaries and Cannibals 214
12.6 Rules that Make Themselves Refireable: Runaway Programs and Recursion 217
12.7 Summary 218
12.8 Questions 218

13 Solving "What?" Problems when the Answer is Expressed in Words 219

13.1 General Methods 219
13.2 Iris.par: What Species Is It? 220
13.2.1 Planning Our Data Elements and Membership Functions 221
13.2.2 Writing Our Rules: Getting Started 223
13.2.3 Writing Our Rules: Classifying the Specimens 224
13.2.4 Writing Our Rules: Reporting Results 225
13.2.5 Setting Our Initial Data 225
13.2.6 Running iris.par 226
13.2.7 Improving iris.par 227
13.3 Echocardiogram Pattern Recognition 227
13.3.1 Echocardiogram Pattern Recognition: Data Declarations 228
13.3.2 Echocardiogram Pattern Recognition: Creating Classification Rules from a Database 230

13.3.3 The Organization of echo.par 231
13.3.4 Metarules to Control Activation of Rule Blocks 231
13.3.5 New Features of Rules in echo.par 233
13.3.6 Running echo.par 235
13.4 Schizo.par 235
13.4.1 Data Declarations in schizo.par 235
13.4.2 Structure of schizo.par 236
13.4.3 Rules for schizo.par; Block 0, Inputting Data 237
13.4.4 Rules for schizo.par; Block 1, Getting Truth Values of Generalized Symptoms 237
13.4.5 Rules for schizo.par; Block 2, Getting Basic Diagnoses 238
13.4.6 Rules for schizo.par; Block 3, Getting Final Diagnoses 238
13.4.7 Rules for schizo.par; Block 4, Output of Diagnoses 239
13.4.8 Rules for schizo.par; Block 5, Block Firing Control 239
13.4.9 Running schizo.par with Debugging Commands 240
13.5 Discussion 240
13.6 Questions 240

14 Programs that Can Learn from Experience 243

14.1 General Methods 243
14.2 Pavlov1.par: Learning by Adding Rules 244
14.2.1 Data for Pavlov1.par 244
14.2.2 Rules for Pavlov1.par 246
14.2.3 Exercise: Pavlov1.par 249
14.3 Pavlov2.par: Learning by Adding Facts to Long-Term Memory 250
14.3.1 Data for Pavlov1.par 250
14.3.2 Rules for Pavlov2.par 251
14.3.3 Running: Pavlov2.par 252
14.4 Defining New Data Elements and New: RULEGEN.FPS 253
14.4.1 Running: RULEGEN.FPS 253
14.5 Most General Way of Creating New Rules and Data Descriptors 254
14.6 Discussion 254
14.7 Questions 255

15 Running On-Line in Real-Time 257

15.1 Overview of On-Line Real-Time Work 257
15.2 Input/Output On-Line in Real-Time 258
15.3 On-Line Real-Time Processing 259
15.3.1 Detection of Obvious Artifacts in the Input Data Stream 260
15.3.2 Data Smoothing and Time Delays and Baselines: Exponential Smoothing 260

15.3.3 Rates of Change: Differentiating Noisy Data 263
15.3.4 Rates of Change: Differentiating Wandering Data 264
15.4 Types of Rules Useful in Real-Time On-Line Work 265
15.4.1 Fuzzy Control Rules 265
15.4.2 Focused Control-Type Rules and Focused Membership Functions 267
15.4.3 Fuzzy Reasoning with Approximate Numerical Comparisons 269
15.5 Memory Management 270
15.6 Development of On-Line Real-Time Programs 270
15.7 Speeding Up a Program 272
15.8 Debugging Real-Time Online Programs 272
15.9 Discussion 273
15.10 Questions 273

Appendix 275

Answers 377

References 399

Index 403

PREFACE

In this book, we will explore the emulation of human thought, capable of dealing with uncertainties, ambiguities, and contradictions.

We agree with Anderson (1993) that much human thought can be expressed in rules (Anderson, 1993). To handle uncertainties, ambiguities, and contradictions, we will use fuzzy systems techniques, implemented by a fuzzy expert system. We supply the fuzzy expert system language FLOPS with this book, so that the readers can actually use our supplied example programs and write their own programs.

An overwhelmingly important fact about human reasoning is that it is not a static process. Data are gathered; some preliminary hypotheses are advanced and tested; some of these may be rejected, and new hypotheses advanced; more data may be required, until finally some conclusion is reached. A computer program to emulate reasoning must proceed similarly. Unfortunately, in much mathematical description of the thought process the dynamic nature is lost. We cannot afford to make this error.

Expert systems are computer programs, designed to make available some of the skills of an expert to nonexperts. Since such programs attempt to emulate in some way an expert's thinking patterns, it is natural that the first work here was done in Artificial Intelligence (AI) circles. Among the first expert systems were the 1965 Dendral programs (Feigenbaum and Buchanan, 1993), which determined molecular structure from mass spectrometer data; R1 (McDermott, 1980) used to configure computer systems; and MYCIN (Shortliffe, 1976) for medical diagnosis. Since the middle 1960s there have been many expert systems created for fields ranging from space shuttle operations through intensive-care-unit patient alarm systems to financial decision making.

There is a variety of ways in which the problem of creating computer programs to act like an expert has been approached; a valuable reference is Jackson (1999). One of the earliest methods employs *rule-based systems*, which use "If . . . Then . . . " rules to represent the expert's reasoning process (*if* the data meet certain specified conditions, *then* take appropriate actions). Other approaches include *semantic or associative nets* (Quillian, 1968), *frames* (Minsky, 1975) and *neural nets* (Haykin, 1994), currently very popular in a wide variety of fields. Of these, clearly dominant are the complementary rule-based systems and neural net approaches.

Neural nets do not require that the thinking patterns of an expert be explicitly specified. Instead, two sets of data are required from the real world. These data include all the inputs to the system, and the correct outputs corresponding to

these input values. The first data set or *training set* is used to train the neural network so that, as nearly as possible, the correct outputs are produced for each set of input values. The second data set or *validation set* is used after the neural net has been trained to make sure that correct answers are produced on different input data. An advantage of neural nets is that it is not necessary to extract the thinking patterns of the expert and to render these explicit. Disadvantages are that a substantial training set is required, and that while the neural net may produce reasonably correct answers, in general, we have little or no idea how it does this. Considerable work has been done on extracting rules from a trained neural net, but this work is not yet advanced to a very satisfactory state.

Rule-based systems require that the expert's knowledge and thinking patterns be explicitly specified. Usually two persons (or groups) develop a system. These are the *domain expert*, who knows how to solve the problem at hand but who is seldom acquainted with computer programming; and the *knowledge engineer*, who is thoroughly familiar with the computer technology involved and expert systems but who has little or no knowledge of the problem at hand. Obtaining this knowledge and writing proper rules is called the *knowledge acquisition* phase (Scott et al., 1991). After the system has been written, it must be tuned for accuracy using a tuning data set similar to the training set of a neural net, but usually much smaller. After tuning, a rule-based system must be validated in the same way as a neural net. Rule-based systems have two advantages. A large training set is usually not required, and since the expert's thinking is explicitly spelled out we now know how he thinks about the problem. They have the disadvantage that the knowledge acquisition phase may be difficult. A great advantage of fuzzy expert systems is that most rules can be written in language that the expert can directly understand, rather than in computer jargon; communication between domain expert and knowledge engineer is greatly eased.

Another advantage of rule-based expert systems is the potential ability of rule-based expert systems to learn by creation of new rules and addition of new data to the expert knowledge data base. Probably the first example of a rule-based expert system to rival human experts was DENDRAL, which deduced the molecular structure of organic compounds from knowledge about fragments into which the compound had been broken (Jackson, 1999, pp. 383 ff). One set of DENDRAL's programs worked directly with the data to produce candidate structures. An additional program, Meta-DENDRAL, worked directly with the DENDRAL rules to improve them and discover new rules, thus discovering new concepts about the data. Meta-DENDRAL was not itself written as a rule-based expert system, but the ability of a rule to generate new rules and new expert factual knowledge opens the possibility for writing expert systems that can create new rules and store new expert factual knowledge. This exciting possibility has not as yet been well explored, perhaps due to the common (and, we think, quite incorrect) assumption among conventional AI practitioners that expert systems are no longer to be considered as artificial intelligence.

Rules, called *production rules* or simply *productions* have a long history in computer science, ranging from the simple "if . . . then . . ." statements employed in such

computer languages as BASIC, FORTRAN, and C to complex systems especially designed for processing rules such as the OPS family of languages by Charles Forgy (Brownston et al., 1985) and the AI language Prolog and its fuzzy version Fril (Baldwin et al., 1995). Their use in AI for psychological modeling was pioneered by Newell and Simon (1972), and in expert systems by a number of AI pioneers (Buchanan and Shortliffe, 1984). Certainly rule-based expert systems are capable of emulating human thought patterns that are well defined; they are also capable of emulating human learning, as we shall show. An appreciable body of thought, especially among cognitive psychologists, agrees (Anderson, 1993). Since the authors are deeply interested in the emulation of human thought, this book is concerned with rule-based expert systems.

The systems we describe require that the knowledge engineer/programmer learn three important new concepts: non-procedural data-driven languages; fuzzy systems theory (fuzzy logic, fuzzy sets, and fuzzy numbers); and a parallel language, rather than the conventional one-statement-at-a-time languages that dominate programming at the present.

Most of the systems we shall describe are *data-driven* and *nonprocedural*. Common computer languages (C, Fortran, Basic) are *procedural*; that is, program statements are executed in the order in which they appear, unless explicit transfers of control are executed. In data-driven rule-based programs, rules may be fired (executed), whenever the data permit and the rules are enabled; the sequence in which the rules appear in the program has little or nothing to do with the order in which they are executed.

The systems are based on fuzzy systems theory, and include data types new to most programmers: *fuzzy sets*, *fuzzy numbers*, and *truth values*. The use of discrete fuzzy sets permits convenient handling of ambiguities and contradictions. All data and rules are associated with *truth values* or *confidences*.

Finally, rules may be fired either *sequentially*, one rule at a time, or may be fired effectively in *parallel*. (Few programmers have any experience with parallel languages.)

The effect of these new concepts is a considerable increase in both power and speed of conventional expert systems, at the expense of some mind stretching.

The FTP site that accompanies this book has a complete demonstration version of a fuzzy expert system Integrated Development Environment (IDE) and run-time package, and a number of example programs. Whenever possible, we use extremely simple examples to illustrate the techniques involved. We hope the reader will not confuse *simple* with *trivial*. For example, in illustrating a blackboard system example programs will exchange one word of information; if we can exchange one word, we can exchange a dozen or a thousand. Our examples of programs that learn are equally simple.

Most books written for academic use concentrate on presenting didactic knowledge. This book, while presenting a fair amount of didactic knowledge, concentrates on teaching a *skill*: actually writing and debugging a fuzzy expert system. This is by no means an easy task. Learning how to construct a fuzzy expert system by reading a book is much like learning how to play tennis by reading a book; you have to play

the game, hit keys, write and debug programs on the computer. The theory of fuzzy mathematics is highly advanced; with the exception of fuzzy control systems, the theory behind fuzzy expert systems for other than numeric outputs is quite ill developed. Much of the fuzzy expert systems theory in this book is original, and sometimes differs substantially from existing theory. For fuzzy reasoning, as distinct from fuzzy control (in which there are several excellent books), there is little literature to which we can refer, except for the work of William Combs, Earl Cox, James Baldwin and his colleagues and Nikola Kasabov; consequently, there are far fewer references listed than is usual in a technical book. We hope that the theory in this book will stimulate others to develop the theory of multistep fuzzy reasoning further.

The debt that those of us in the fuzzy field owe to Professor Lotfi Zadeh is incalculable; see Klir and Yuan (1996) for a selection of his most important papers, and Klir and Yuan (1995) for an exposition of the most important concepts in fuzzy systems theory. Not only did he originate the entire field and define its basic concepts many years ago, but also he continues through the years to challenge and inspire us with new ideas. We can only thank him and wish him many more productive years.

WILLIAM SILER
JAMES J. BUCKLEY