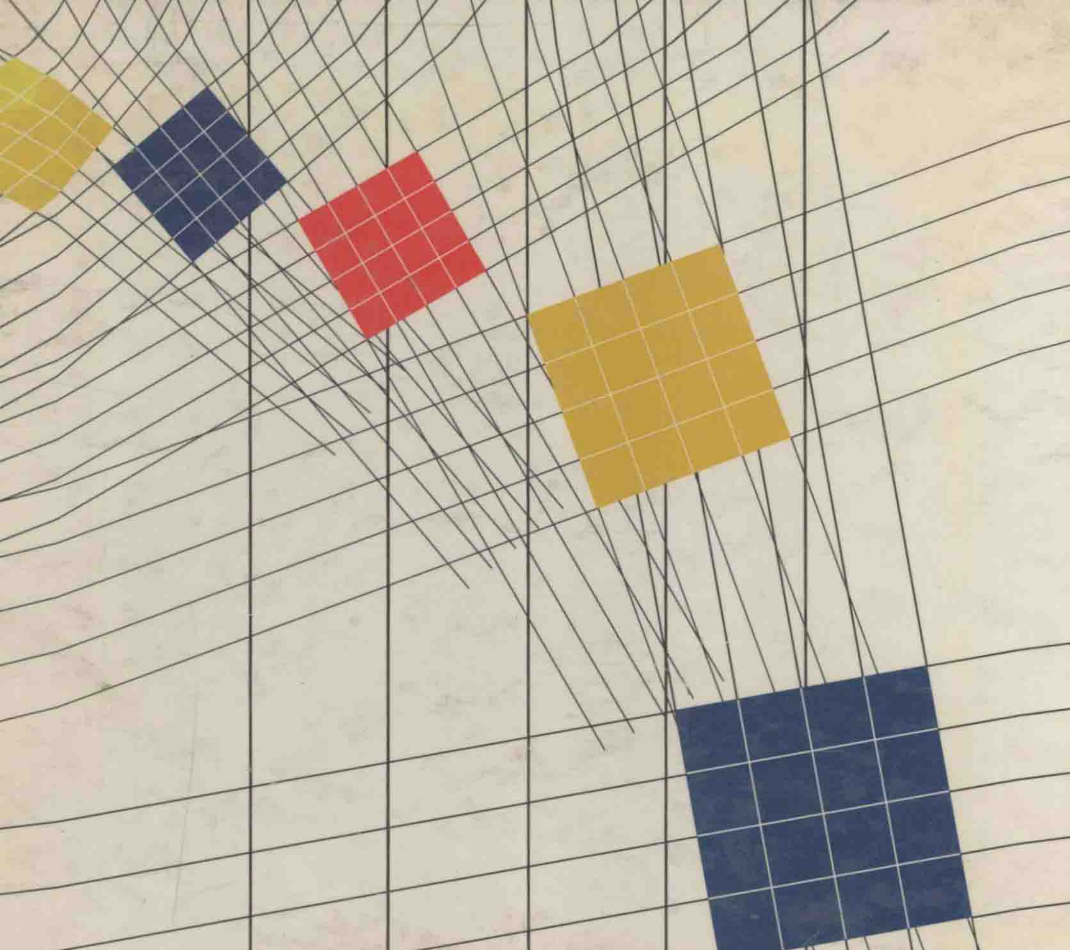

Neural Network PC Tools

A Practical Guide

Edited by
Russell C. Eberhart
Roy W. Dobbins

Neural Network PC Tools

A Practical Guide

with a Foreword by Bernard Widrow

Edited by
Russell C. Eberhart and
Roy W. Dobbins
The Johns Hopkins University Applied Physics Laboratory,
Laurel, Maryland

Academic Press, Inc.
Harcourt Brace Jovanovich, Publishers
San Diego New York Boston London Sydney Tokyo Toronto

This book is printed on acid-free paper. ♾

Academic Press, Inc.
San Diego, California 92101

United Kingdom Edition published by
Academic Press Limited
24–28 Oval Road, London NW1 7DX

Library of Congress Cataloging-in-Publication Data

Neural network PC tools : a practical guide / [edited by] Russell C. Eberhart and Roy W. Dobbins.
p. cm.
ISBN 0-12-228640-5 (alk. paper)
1. Neural computers. 2. Neural circuitry. 3. Microcomputers.
I. Eberhart, Russell C. II. Dobbins, Roy W.
QA76.5.N42827 1990
006.3--dc20 90-727
CIP

Printed in the United States of America
90 91 92 93 9 8 7 6 5 4 3 2 1

Neural Network PC Tools

To Francie, Mark, and Sean;
to Leonie, Lorien, and Audrey;
and in Renée's memory.

CONTRIBUTORS

Numbers in parentheses indicate the pages on which the authors' contributions begin.

Maureen Caudhill (189), 11450 Grassy Trail Drive, San Diego, California 92127

Roy W. Dobbins (9, 35, 59, 81, 111, 161, 215, 295, 393), The Johns Hopkins University Applied Physics Laboratory, Laurel, Maryland 20723

Russell C. Eberhart (9, 35, 59, 81, 111, 161, 177, 215, 295, 393), The Johns Hopkins University Applied Physics Laboratory, Laurel, Maryland 20723

Gary Entsminger (285), Rocky Mountain Biological Laboratory, Crested Butte, Colorado 81224

Larrie V. Hutton (161, 235), The Johns Hopkins University Applied Physics Laboratory, Laurel, Maryland 20723

D. Gilbert Lee, Jr. (137), The Johns Hopkins University Applied Physics Laboratory, Laurel, Maryland 20723

Vincent G. Sigillito (177, 235), The Johns Hopkins University Applied Physics Laboratory, Laurel, Maryland 20723

Thomas Zaremba (251), The Johns Hopkins University Applied Physics Laboratory, Laurel, Maryland 20723

FOREWORD

I am pleased to have been asked by Russ Eberhart and Roy Dobbins to write the foreword to this book. It has been three decades since my frequently referenced article with Hoff, on adaptive switching circuits, that discussed the Least Mean Squares algorithm [13]. My first hardware version of Adaline, shown in the accompanying photograph, is also approaching its 30th birthday. How time flies!

After my original work in the neural network field, I did some developmental work in the adaptive filter area. I still believe that if an electrical engineer had developed the back-propagation algorithm, we'd be working with "massively parallel adaptive filters" instead of neural networks. Oh, well.

A few years ago, about the time of Rumelhart and McClelland's three-volume tome on parallel distributed processing [2,4,5], I said to myself, "What the heck, adaptive filters are in pretty good shape. I think I'll work on neural networks again."

In the past few years, there has been an absolute explosion in the amount of work being done in the neural network field. It seems somewhat analogous to the far-reaching social and political changes sweeping the world as this book goes to press in 1990.

Just as I have to watch the morning news to keep abreast of changes in governments in Eastern Europe, I have to read my morning mail (which now includes e-mail) to keep abreast of developments in neural networks. With all the fine neural network applications both working and under development, I feel that neural networks are here to stay! And I'm sure that the most exciting applications are yet to come.

As president of the International Neural Network Society (INNS) and fellow of the Institute of Electrical and Electronics Engineers, with a special interest in its Neural Networks Council, I'm in a position to see most major publications in the field. In fact, I am asked to review a significant percentage of the new books.

It is thus from a position of some experience that I say that an exposition on the practical applications of neural networks has been greatly needed. I believe that this book fulfills that need in an extremely fine fashion.

Many books have been written that emphasize the theoretical aspects of neural networks. Some have gone as far as presenting equations for various network topologies. One or two have even included demonstration software illustrating different network topologies.

Equations and demonstrations, however, are only a starting point for engineers and computer scientists. What we need, for our real-world practical applications, is a carefully thought-out methodology that takes the systems approach. By that I mean an approach is required that starts with a systems analysis and goes all the way to the computer code necessary to implement the design developed from the analysis.

This book does that. It is a practical and thorough approach to applying neural network tools to everyday problems. And, as the case studies illustrate, these applications aren't limited to the scientific and engineering fields. In this book, you can even learn how to use neural network tools to compose music and analyze the commodities futures market.

Another issue dealt with, at least implicitly, in this book is that of terminology. The glossary near the end of the book contains proposed definitions for a number of terms we use in our everyday neural network efforts. While I personally may not agree with each and every definition, I wholeheartedly endorse moving toward a commonly accepted terminology. It's pretty hard for a person new to the field to sort through literature that refers to processing elements, processing units, units, neurons, nodes, neurodes, etc., all of which refer to exactly the same thing.

Through their participation in the Ad Hoc Standards Committee of the IEEE Neural Networks Council, chaired by Evangelia Tzanakou of Rutgers University, Russ Eberhart and Roy Dobbins, with their colleagues from academia, industry, and government, will be grappling with the issue of definitions. I'm sure that their committee is in for some interesting discussions over the next few years.

Also helpful to folks new to neural nets is the appendix on additional resources. Of course, as president of the INNS, I feel bound to ask that you pay special attention to the information on our society!

As Russ and Roy say in the introductory chapter, you really don't need a supercomputer, a million dollars, and an interdisciplinary team of experts to put neural networks to work. All you need is a personal computer and this book. I'm sure you'll enjoy it!

Bernard Widrow
Electrical Engineering Department
Stanford University

CONTENTS

Contributors xiii
Foreword xv
Introduction 1

1. Background and History 9
Russell C. Eberhart and Roy W. Dobbins

Introduction 9
Biological Basis for Neural Network Tools 10
Introduction 10
Neurons 10
Differences between Biological Structures and NNTs 11
Where Did Neural Networks Get Their Name? 13
Neural Network Development History 14
Introduction 14
The Age of Camelot 14
The Dark Age 21
The Renaissance 28
The Age of Neoconnectionism 33

2. Implementations 35
Russell C. Eberhart and Roy W. Dobbins

Introduction 35
The Back-Propagation Model 36
Introduction 36
Topology and Notation 37
Network Input 39
Feedforward Calculations 40
Training by Error Back-Propagation 43
Running the Back-Propagation NNT 48
The Self-Organization Model 49
Introduction 49
Topology and Notation 50
Network Initialization and Input 53
Training Calculations 54
Testing and Running 58

3. Systems Considerations 59
Russell C. Eberhart and Roy W. Dobbins

Introduction 59
Evaluating Problem Categories 60

The Big Picture 62
Developing a System Specification 63
Specifications and Models 63
Informal Specifications 64
Structured Analysis 64
Formal Specifications 67
Applying Specifications to Neural Networks 68
Choosing Effective Roles for Neural Networks 69
Introduction 69
Network Incarnations and Reincarnations 70
Avoiding Preprocessing Pitfalls 70
Neural Networks versus Expert Systems 74
Successful Application Examples 77

4. Software Tools 81
Roy W. Dobbins and Russell C. Eberhart

Introduction 81
What Is Neural Network Software? 81
The Last of the Programmers? 82
Implementing Neural Networks on the PC 82
Using C and Assembly Language 83
Back-Propagation Networks 83
The Three Rs . . . 84
Iterations . . . Kernel, Brain, or Engine? 84
Forward . . . and Backward 85
Computing Activations 85
Vector and Matrix Operations 87
Storage Allocation 87
Propagating Error Signals 88
Adapting Weights 89
Kohonen Self-Organizing Networks 90
Finding the Winning Unit 91
What Is This Good For? 91
Nondeterministic Response 92
Running Neural Networks 92
Getting Data into and out of the Network 93
Reading Input Patterns 93
Dealing with the Real World 94
Normalizing Data 95
Weights 95
Setting Attributes 97
Average Sum-Squared Error 98
What's It Doing? 98
Implementation Issues 101
Interpretation versus Compilation 101
Optimizing the Code . . . How to Make the Network Scream! 102
Memory Limitations 103
Do You Really Need Floating Point? 107
Making the Most of Coprocessors 107
Debugging Networks 108

5. Development Environments 111
Roy W. Dobbins and Russell C. Eberhart

Introduction 111
What Is a Neural Network Development Environment? 112
Desirable Characteristics of Development Environments 113
Why a Development Environment? 115
Introduction to Network Modeling Languages 117
Specification Languages 118
Parallel Processing and Object-Oriented Languages 118
Conventional Programming Languages 119
A Brief Survey of Neural Network Modeling Languages 119
Specifying Neural Network Models 120
Specifying Network Architecture 122
Activation Functions 123
Learning Rules 123
Specifying the Environment 123
Update Rules 123
Neural Network Paradigms 124
A Brief Survey of Neural Network Development Environments 124
CaseNet: A Neural Network Development Environment 126
Anatomy of CaseNet 126
CaseNet Components 127
Graphical Network Editor 128
Network Parser 129
Network Analyzer 131
Network Code Generator 133
Network Compiler 135

6. Hardware Implementations 137
D. Gilbert Lee, Jr.

The Transputer 138
Interprocess Communications 140
Multitasking 141
Programming Languages 142
Optimizing the Matrix–Vector Multiply 142
Using Transputers in Parallel 147
Processor Farms 147
Pipelining 148
Programming the Transputers 149
Discussion 153
Mini Case Study: Ship Image Recognition 155
Description of Ship Image Preprocessing 155
Neural Network Training Description 156
Results 157
Summary 158
Vendors 158

7. Performance Metrics 161
Russell C. Eberhart, Roy W. Dobbins, and Larrie V. Hutton

Introduction 161
Percent Correct 162
Average Sum-Squared Error 165
Normalized Error 167
Receiver Operating Characteristic Curves 169
Recall and Precision 172
Other ROC-Related Measures 173
Chi-Square Test 174

8. Network Analysis 177
Vincent G. Sigillito and Russell C. Eberhart

Introduction 177
Network Analysis 178
- Introduction 178
- The Divide-by-Three Problem 178
- Other Considerations 182
- The Square-within-a-Square Problem 185
- Distributions of Hidden Neurode Activity Levels 186
- Analyzing Weights in Trained Networks 187

Relation Factors 187

9. Expert Networks 189
Maureen Caudill

Rule-Based Expert Systems 190
Expert Networks 197
- Fuzzy Mathematics 197
- Fuzzy Cognitive Maps 199
- An Expert Bond-Rating Network 204
- Knowledge in an Expert Network 207

Expert Network Characteristics 209
Hybrid Expert Networks 211
- Explanation by Confabulation 212
- Rule Extraction 212
- True Hybrid Expert 213

10. Case Study I: The Detection of Electroencephalogram Spikes 215
Russell C. Eberhart and Roy W. Dobbins

Introduction 215
Goals and Objectives 216
Design Process 217
System Specifications 218

Background 220
Data Preprocessing and Categorization 220
Test Results 229

11. Case Study II: Radar Signal Processing 235
Vincent G. Sigillito and Larrie V. Hutton

Introduction 235
Description of the Radar Facility 236
Operation of the System and Data Collection 236
Goals and Objectives 237
The Design Process 239
Representation 240
Choosing the Number of Hidden Nodes 241
Choosing Training and Test Sets 241
Results and Discussion 242
A Preliminary Analysis 242
The Neural Network Analysis and Results 243
Conclusions 249

12. Case Study III: Technology in Search of a Buck 251
Thomas Zaremba

Introduction 251
Markets to Watch and Markets to Trade 252
Futures Market Forecasting 255
Historical Futures Market Data 256
Sources of Market Model Data 260
Futures Market Model Description 261
Why Neural Networks? 270
Why Excel? 271
Current Status, Future Plans, and Money Made 277

13. Case Study IV: Optical Character Recognition 285
Gary Entsminger

From .PCX to .TXT via a Neural Network 285
Why OCR Is Such a Bear 286
Objects 290
Notes and Conclusions 292
For More Information, Consult the Following 293

14. Case Study V: Making Music 295
Russell C. Eberhart and Roy W. Dobbins

Introduction 295
Representing Music for Neural Network Tools 296

Network Configurations 298
Stochasticity, Variability, and Surprise 308
Playing Your Music with MIDI 310
Now What? 312

Glossary 313
References 321

Appendix A. Batchnet Back-Propagation Source Code with Pattern, Weight, and Run Files 329

Appendix B. Self-Organizing Neural Network Tool Code with Pattern, Run, and Demo Files 345

Appendix C. Turbo Pascal Code for Optical Character Recognition Shell 367

Appendix D. Source Code for Music Composition Files 375

Appendix E. Additional Resources 393
Russell C. Eberhart and Roy W. Dobbins

Introduction 393
Organizations and Societies 394
Conferences and Symposia 396
Journals, Magazines, and Newsletters 397
Computer Bulletin Boards 401
Computer Databases 402
Summary 403

Appendix F. Transputer Matrix Multiply Code 405

Index 411

Introduction

Russell C. Eberhart
Roy W. Dobbins

In the past few years, neural networks have received a great deal of attention and are being touted as one of the greatest computational tools ever developed. Much of the excitement is due to the apparent ability of neural networks to imitate the brain's ability to make decisions and draw conclusions when presented with complex, noisy, irrelevant, and/or partial information. Furthermore, at some primitive level, neural networks appear able to imitate the brain's "creative" processes to generate new data or patterns.

It is hard, especially for a person unfamiliar with the subject, to separate the substance from the hype. Many of the applications being discussed for neural networks are complex and relatively hard to understand, and many of the available hardware and software tools are either too simplistic to be useful or too complicated and expensive to be affordable and understandable for the average engineer or computer scientist.

The hardware and software tools we describe in this book, with few exceptions, are available to most technical people, and we have written the book to help the typical engineer, computer scientist, or other technically oriented person who is interested in solving practical problems with neural networks. You'll need some background in algebra to understand some of the equations for network training and operation, but the algebra required isn't any more involved than most folks have had by the time they graduate from high school. The most complicated mathematics we'll use involves summing a series of subscripted variables.

It is true that a deep understanding of biologically derived neural networks requires knowledge in a variety of fields, including biology, mathematics, and artificial intelligence. But none of this knowledge is needed to understand the neural network tools presented in this book. Probably the best background for getting the maximum benefit from this book is liking to "muck about" with computers. If you're comfortable running a variety of software and occasionally (possibly with some trepidation) fiddling with programming simple stuff in a language such as BASIC or C, you'll feel right at home here.

It's a myth that the only way to achieve results with neural networks is with a million dollars, a supercomputer, and an interdisciplinary team of Nobel laureates, though some commercial vendors out there would like you to believe it.

You don't need a supercomputer or a parallel processing machine to do something useful with neural networks. It's not even necessary to have a MicroVAX or a Sun workstation. A personal computer such as an IBM PC/AT or workalike is a perfectly adequate hardware base. A plain vanilla PC, XT, or workalike is even sufficient; it's just that the slower clock speed is going to make things take longer. With simple hardware and software tools, it is possible to solve problems that are otherwise impossible or impractical. Neural networks really do offer solutions to some problems that can't be solved in any other way known to the authors. That's no hype!

What *is* hype is that neural networks can solve all of your difficult engineering or computer problems faster and cheaper than anything you have ever tried. It is a myth that neural networks can leap tall buildings in a single bound and that they can solve problems single-handedly. They are particularly inappropriate for problems requiring precise calculations: You'll probably never successfully balance your checkbook with a neural network. (But then, how many people have actually used a personal computer for this task?)

Another statement that qualifies as *mostly* myth is that you don't need to do any programming at all to use neural network tools. This is at best misleading. It's true that a neural network trains (learns) and runs on input data and according to a set of rules that update the weights that connect the processing elements, or nodes, and that the learning of the network is not, strictly speaking, programmed. It's also true that computer-aided software engineering (CASE) tools will become more available in the next few years and that little or no programming expertise will be required to use these tools to generate executable neural network code. But it's also true that in the real world of neural network applications, some programming is required to get from where you start to a solution.

Furthermore, although it is accurate to say that neural networks can play a key role in the solution of several classes of problems that are