C++ Neural Networks and Fuzzy Logic
by Valluru B. Rao
M&T Books, IDG Books Worldwide, Inc.
ISBN: 1558515526 **Pub Date:** 06/01/95

C++ Neural Networks and Fuzzy Logic
by Valluru B. Rao
M&T Books, IDG Books Worldwide, Inc.
ISBN: 1558515526 **Pub Date:** 06/01/95

Table of Contents

Preface

The number of models available in neural network literature is quite large. Very often the treatment is mathematical and complex. This book provides illustrative examples in C++ that the reader can use as a basis for further experimentation. A key to learning about neural networks to appreciate their inner workings is to experiment. Neural networks, in the end, are fun to learn about and discover. Although the language for description used is C++, you will not find extensive class libraries in this book. With the exception of the backpropagation simulator, you will find fairly simple example programs for many different neural network architectures and paradigms. Since backpropagation is widely used and also easy to tame, a simulator is provided with the capacity to handle large input data sets. You use the simulator in one of the chapters in this book to solve a financial forecasting problem. You will find ample room to expand and experiment with the code presented in this book.

There are many different angles to neural networks and fuzzy logic. The fields are expanding rapidly with ever-new results and applications. This book presents many of the different neural network topologies, including the BAM, the Perceptron, Hopfield memory, ART1, Kohonen's Self-Organizing map, Kosko's Fuzzy Associative memory, and, of course, the Feedforward Backpropagation network (aka Multilayer Perceptron). You should get a fairly broad picture of neural networks and fuzzy logic with this book. At the same time, you will have real code that shows you example usage of the models, to solidify your understanding. This is especially useful for the more complicated neural network architectures like the Adaptive Resonance Theory of Stephen Grossberg (ART).

The subjects are covered as follows:

- Chapter 1 gives you an overview of neural network terminology and nomenclature. You discover that neural nets are capable of solving complex

problems with parallel computational architectures. The Hopfield network and feedforward network are introduced in this chapter.

- Chapter 2 introduces C++ and object orientation. You learn the benefits of object-oriented programming and its basic concepts.
- Chapter 3 introduces fuzzy logic, a technology that is fairly synergistic with neural network problem solving. You learn about math with fuzzy sets as well as how you can build a simple fuzzifier in C++.
- Chapter 4 introduces you to two of the simplest, yet very representative, models of: the Hopfield network, the Perceptron network, and their C++ implementations.
- Chapter 5 is a survey of neural network models. This chapter describes the features of several models, describes **threshold** functions, and develops concepts in neural networks.
- Chapter 6 focuses on learning and training paradigms. It introduces the concepts of supervised and unsupervised learning, self-organization and topics including backpropagation of errors, radial basis function networks, and conjugate gradient methods.
- Chapter 7 goes through the construction of a backpropagation simulator. You will find this simulator useful in later chapters also. C++ classes and features are detailed in this chapter.
- Chapter 8 covers the Bidirectional Associative memories for associating pairs of patterns.
- Chapter 9 introduces Fuzzy Associative memories for associating pairs of fuzzy sets.
- Chapter 10 covers the Adaptive Resonance Theory of Grossberg. You will have a chance to experiment with a program that illustrates the working of this theory.
- Chapters 11 and 12 discuss the Self-Organizing map of Teuvo Kohonen and its application to pattern recognition.
- Chapter 13 continues the discussion of the backpropagation simulator, with enhancements made to the simulator to include momentum and noise during training.
- Chapter 14 applies backpropagation to the problem of financial forecasting, discusses setting up a backpropagation network with 15 input variables and 200 test cases to run a simulation. The problem is approached via a systematic 12-step approach for preprocessing data and setting up the problem. You will find a number of examples of financial forecasting highlighted from the literature. A resource guide for neural networks in finance is included for people who would like more information about this area.
- Chapter 15 deals with nonlinear optimization with a thorough discussion of the Traveling Salesperson problem. You learn the formulation by Hopfield and the approach of Kohonen.
- Chapter 16 treats two application areas of fuzzy logic: fuzzy control systems and

fuzzy databases. This chapter also expands on fuzzy relations and fuzzy set theory with several examples.

- Chapter 17 discusses some of the latest applications using neural networks and fuzzy logic.

In this second edition, we have followed readers' suggestions and included more explanations and material, as well as updated the material with the latest information and research. We have also corrected errors and omissions from the first edition.

Neural networks are now a subject of interest to professionals in many fields, and also a tool for many areas of problem solving. The applications are widespread in recent years, and the fruits of these applications are being reaped by many from diverse fields. This methodology has become an alternative to modeling of some physical and nonphysical systems with scientific or mathematical basis, and also to expert systems methodology. One of the reasons for it is that absence of full information is not as big a problem in neural networks as it is in the other methodologies mentioned earlier. The results are sometimes astounding, even phenomenal, with neural networks, and the effort is at times relatively modest to achieve such results. Image processing, vision, financial market analysis, and optimization are among the many areas of application of neural networks. To think that the modeling of neural networks is one of modeling a system that attempts to mimic human learning is somewhat exciting. Neural networks can learn in an unsupervised learning mode. Just as human brains can be trained to master some situations, neural networks can be trained to recognize patterns and to do optimization and other tasks.

In the early days of interest in neural networks, the researchers were mainly biologists and psychologists. Serious research now is done by not only biologists and psychologists, but by professionals from computer science, electrical engineering, computer engineering, mathematics, and physics as well. The latter have either joined forces, or are doing independent research parallel with the former, who opened up a new and promising field for everyone.

In this book, we aim to introduce the subject of neural networks as directly and simply as possible for an easy understanding of the methodology. Most of the important neural network architectures are covered, and we earnestly hope that our efforts have succeeded in presenting this subject matter in a clear and useful fashion.

We welcome your comments and suggestions for this book, from errors and oversights, to suggestions for improvements to future printings at the following E-mail addresses:

V. Rao rao@cse.bridgeport.edu

H. Rao ViaSW@aol.com

Table of Contents

C++ Neural Networks and Fuzzy Logic
by Valluru B. Rao
M&T Books, IDG Books Worldwide, Inc.
ISBN: 1558515526 **Pub Date:** 06/01/95

Table of Contents

Dedication

To the memory of

Vydehamma, Annapurnamma, Anandarao, Madanagopalarao, Govindarao, and Rajyalakshamma.

Acknowledgments

We thank everyone at MIS:Press/Henry Holt and Co. who has been associated with this project for their diligence and support, namely, the Technical Editors of this edition and the first edition for their suggestions and feedback; Laura Lewin, the Editor, and all of the other people at MIS:Press for making the book a reality.

We would also like to thank Dr. Tarek Kaylani for his helpful suggestions, Professor R. Haskell, and our other readers who wrote to us, and Dr. V. Rao's students whose suggestions were helpful. Please E-mail us more feedback!

Finally, thanks to Sarada and Rekha for encouragement and support. Most of all, thanks to Rohini and Pranav for their patience and understanding through many lost evenings and weekends.

Table of Contents

C++ Neural Networks and Fuzzy Logic
by Valluru B. Rao
M&T Books, IDG Books Worldwide, Inc.
ISBN: 1558515526 **Pub Date:** 06/01/95

Previous | Table of Contents | Next

Chapter 1
Introduction to Neural Networks

Neural Processing

How do you recognize a face in a crowd? How does an economist predict the direction of interest rates? Faced with problems like these, the human brain uses a web of interconnected processing elements called *neurons* to process information. Each neuron is autonomous and independent; it does its work *asynchronously*, that is, without any synchronization to other events taking place. The two problems posed, namely recognizing a face and forecasting interest rates, have two important characteristics that distinguish them from other problems: First, the problems are complex, that is, you can't devise a *simple* step-by-step *algorithm* or precise formula to give you an answer; and second, the data provided to resolve the problems is equally complex and may be noisy or incomplete. You could have forgotten your glasses when you're trying to recognize that face. The economist may have at his or her disposal thousands of pieces of data that may or may not be relevant to his or her forecast on the economy and on interest rates.

The vast processing power inherent in biological neural structures has inspired the study of the structure itself for hints on organizing human-made computing structures. *Artificial neural networks*, the subject of this book, covers the way to organize synthetic neurons to solve the same kind of difficult, complex problems in a similar manner as we think the human brain may. This chapter will give you a sampling of the terms and nomenclature used to talk about neural networks. These terms will be covered in more depth in the chapters to follow.

Neural Network

A *neural network* is a computational structure inspired by the study of biological neural processing. There are many different types of neural networks, from relatively simple to very complex, just as there are many theories on how biological neural processing works. We will begin with a discussion of a *layered feed-forward* type of neural network and branch out to other paradigms later in this chapter and in other chapters.

A layered feed-forward neural network has *layers*, or subgroups of processing elements. A layer of processing elements makes independent computations on data that it receives and passes the results to another layer. The next layer may in turn make its independent computations and pass on the results to yet another layer. Finally, a subgroup of one or more processing elements determines the output from the network. Each processing element makes its computation based upon a weighted sum of its inputs. The first layer is the *input layer* and the last the *output layer*. The layers that are placed between the first and the last layers are the *hidden layers*. The processing elements are seen as units that are similar to the neurons in a human brain, and hence, they are referred to as *cells*, *neuromimes*, or *artificial neurons*. A *threshold function* is sometimes used to qualify the output of a neuron in the output layer. Even though our subject matter deals with artificial neurons, we will simply refer to them as neurons. Synapses between neurons are referred to as *connections*, which are represented by edges of a directed graph in which the nodes are the artificial neurons.

Figure 1.1 is a layered feed-forward neural network. The circular nodes represent neurons. Here there are three layers, an input layer, a hidden layer, and an output layer. The directed graph mentioned shows the connections from nodes from a given layer to other nodes in other layers. Throughout this book you will see many variations on the number and types of layers.

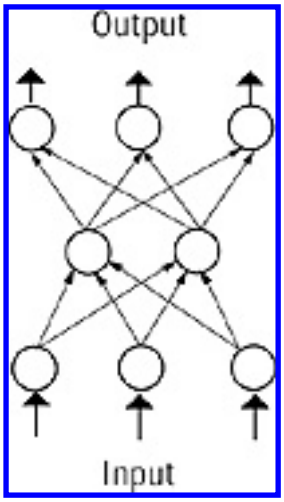

Figure 1.1 A typical neural network.

Output of a Neuron

Basically, the *internal activation* or raw output of a neuron in a neural network is a weighted sum of its inputs, but a threshold function is also used to determine the final value, or the output. When the output is 1, the neuron is said to *fire*, and when it is 0, the neuron is considered not to have fired. When a threshold function is used, different results of activations, all in the same interval of values, can cause the same final output value. This situation helps in the sense that, if precise input causes an activation of 9 and noisy input causes an activation of 10, then the output works out the same as if noise is filtered out.

To put the description of a neural network in a simple and familiar setting, let us describe an example about a daytime game show on television, *The Price is Right*.

Previous | Table of Contents | Next

C++ Neural Networks and Fuzzy Logic
by Valluru B. Rao
M&T Books, IDG Books Worldwide, Inc.
ISBN: 1558515526 **Pub Date:** 06/01/95

Previous | Table of Contents | Next

Cash Register Game

A contestant in *The Price is Right* is sometimes asked to play the *Cash Register Game*. A few products are described, their prices are unknown to the contestant, and the contestant has to declare how many units of each item he or she would like to (pretend to) buy. If the total purchase does not exceed the amount specified, the contestant wins a special prize. After the contestant announces how many items of a particular product he or she wants, the price of that product is revealed, and it is rung up on the cash register. The contestant must be careful, in this case, that the total does not exceed some nominal value, to earn the associated prize. We can now cast the whole operation of this game, in terms of a neural network, called a *Perceptron*, as follows.

Consider each product on the shelf to be a neuron in the input layer, with its input being the unit price of that product. The cash register is the single neuron in the output layer. The only connections in the network are between each of the neurons (products displayed on the shelf) in the input layer and the output neuron (the cash register). This arrangement is usually referred to as a neuron, the cash register in this case, being an *instar* in neural network terminology. The contestant actually determines these connections, because when the contestant says he or she wants, say five, of a specific product, the contestant is thereby assigning a weight of 5 to the connection between that product and the cash register. The total bill for the purchases by the contestant is nothing but the weighted sum of the unit prices of the different products offered. For those items the contestant does not choose to purchase, the implicit weight assigned is 0. The application of the dollar limit to the bill is just the application of a threshold, except that the threshold value should not be exceeded for the outcome from this network to favor the contestant, winning him or her a good prize. In a Perceptron, the way the threshold works is that an output neuron is supposed to fire if its activation value *exceeds* the threshold value.

Weights

The weights used on the connections between different layers have much significance in the working of the neural network and the characterization of a network. The following actions are possible in a neural network:

1. Start with one set of weights and run the network. (NO TRAINING)
2. Start with one set of weights, run the network, and modify some or all the weights, and run the network again with the new set of weights. Repeat this process until some predetermined goal is met. (TRAINING)

Training

Since the output(s) may not be what is expected, the weights may need to be altered. Some rule then needs to be used to determine how to alter the weights. There should also be a criterion to specify when the process of successive modification of weights ceases. This process of changing the weights, or rather, updating the weights, is called *training*. A network in which learning is employed is said to be subjected to *training*. Training is an external process or regimen. Learning is the desired process that takes place internal to the network.

Feedback

If you wish to train a network so it can recognize or identify some predetermined patterns, or evaluate some function values for given arguments, it would be important to have information fed back from the output neurons to neurons in some layer before that, to enable further processing and adjustment of weights on the connections. Such feedback can be to the input layer or a layer between the input layer and the output layer, sometimes labeled the *hidden layer*. What is fed back is usually the error in the output, modified appropriately according to some useful paradigm. The process of feedback continues through the subsequent cycles of operation of the neural network and ceases when the training is completed.

Supervised or Unsupervised Learning

A network can be subject to *supervised* or *unsupervised* learning. The learning would be supervised if external criteria are used and matched by the network output, and if not, the learning is unsupervised. This is one broad way to divide different neural network approaches. Unsupervised approaches are also termed *self-organizing*. There is more interaction between neurons, typically with feedback and intralayer connections between neurons promoting self-organization.

Supervised networks are a little more straightforward to conceptualize than unsupervised networks. You apply the inputs to the supervised network along with an expected response, much like the Pavlovian conditioned stimulus and response regimen. You mold the network with stimulus-response pairs. A stock market forecaster may present economic data (the *stimulus*) along with metrics of stock market performance (the *response*) to the neural network to the present and attempt to predict the future once training is complete.

You provide unsupervised networks with only stimulus. You may, for example, want an unsupervised network to correctly classify parts from a conveyor belt into part numbers, providing an image of each part to do the classification (the stimulus). The unsupervised network in this case would act like a look-up memory that is indexed by its contents, or a *Content-Addressable-Memory (CAM)*.

Previous | Table of Contents | Next