IEEE Press Series on Computational Intelligence
David B. Fogel, Series Editor

FUNDAMENTALS OF COMPUTATIONAL INTELLIGENCE

NEURAL NETWORKS, FUZZY SYSTEMS, AND EVOLUTIONARY COMPUTATION

JAMES M. KELLER, DERONG LIU, AND DAVID B. FOGEL

IEEE PRESS

WILEY

FUNDAMENTALS OF COMPUTATIONAL INTELLIGENCE

IEEE Press
445 Hoes Lane
Piscataway, NJ 08854

Technical Reviewer
Daniel Sanchez, *Department of Computer Science and Artificial Intelligence, University of Granada, Spain*

FUNDAMENTALS OF COMPUTATIONAL INTELLIGENCE

NEURAL NETWORKS, FUZZY SYSTEMS, AND EVOLUTIONARY COMPUTATION

James M. Keller
Derong Liu
David B. Fogel

IEEE Press Series on Computational Intelligence

IEEE PRESS

WILEY

Library of Congress Cataloging-in-Publication Data is available.

ISBN: 978-1-110-21434-2

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

Jim: To my grandsons, Mack, Shea, Jones, and Jameson who continuously remind me what is really important in life.

Derong: To my dear wife Mingshu for her love and support.

David: To Jonathan and Skyler, for helping me discover what it's all about.

CONTENTS

ACKNOWLEDGMENTS

We are grateful to the many people who have helped us during the past years, who have contributed to work presented here, and who have offered critical reviews of prior publications. We also thank Wiley-IEEE Press for their assistance in publishing the manuscript, the IEEE Computational Intelligence Society for having it be part of the IEEE Press Series on Computational Intelligence, and Zhenbang Ju for his help in conducting simulations and program development in support of the chapters on evolutionary computation, as well as his assistance in formatting all the materials. We are also grateful to Andrew Buck and Alina Zare for using draft chapters from this book in teaching an Introduction to Computational Intelligence course at the University of Missouri, and for providing valuable feedback. Fernando Gomide also offered constructive criticisms that were helpful in improving the content. Jim Keller thanks Mihail Popescu and Derek Anderson for their assistance in implementing fuzzy clustering and fuzzy integral algorithms. Derong Liu thanks Ding Wang for his help with preparing the manuscript. David Fogel thanks the IEEE, Springer, MIT Press, and Morgan Kaufmann (Elsevier) for permissions to reprint materials (cited in the text). He also thanks SPIE for returning rights to materials published previously under its copyright. Finally, we acknowledge the assistance and friendship of our colleagues within the field of computational intelligence who hold the same passion for this material as we do. We hope the reader will enjoy as well.

CHAPTER 1

Introduction to Computational Intelligence

1.1 WELCOME TO COMPUTATIONAL INTELLIGENCE

Welcome to the world of computational intelligence (CI), which takes inspiration from nature to develop intelligent computer-based systems. Broadly, the field of CI encompasses three main branches of research and application: (1) neural networks, which model aspects of how brains function, (2) fuzzy systems, which model aspects of how people describe the world around them, and (3) evolutionary computation, which models aspects of variation and natural selection in the biosphere. These three approaches are often synergistic, working together to supplement each other and provide superior solutions to vexing problems.

1.2 WHAT MAKES THIS BOOK SPECIAL

A unique feature of this textbook is that each of us has been an editor-in-chief for an IEEE Transactions sponsored by the IEEE Computational Intelligence Society (CIS), the main technical society supporting research in CI around the world. This book offers the only systematic treatment of the entire field of CI from the perspectives of three experts who have guided peer-reviewed seminal research published in the top-tier journals in the area of CI.

The publications we've edited include the *IEEE Transactions on Neural Networks* (Derong Liu), the *IEEE Transactions on Fuzzy Systems* (James Keller), and the *IEEE Transactions on Evolutionary Computation* (David Fogel). These publications consistently present the most recent theoretical developments and practical implementations in the field of CI.

As you read through the book, you'll notice that each central area of CI is offered in its own style. That's because each of us has taken the primary lead on the material in our own area of expertise. We've made efforts to be consistent, but you'll certainly

Fundamentals of Computational Intelligence: Neural Networks, Fuzzy Systems, and Evolutionary Computation, First Edition. James M. Keller, Derong Liu, and David B. Fogel.

notice three distinct ways of conveying what we know. We believe that this is one of the advantages of our partnership—you get the whole story, but not from the standpoint of a single author. We made a deal to allow each of us to tell our story in our own way.

You may relate more to one of our styles over the others, but the content is solid and your efforts at studying this material will be rewarding. The theories and techniques described will allow you to create solutions to problems in pattern recognition, control, automated decision making, optimization, statistical modeling, and many other areas.

1.3 WHAT THIS BOOK COVERS

This introduction to CI covers basic and advanced material in neural networks, fuzzy systems, and evolutionary computation. Does it cover all of the possible topics within the field of computational intelligence? Certainly not!

Our goal is to provide fundamental material in the diverse and fast growing area of CI and give you a strong fundamental understanding of its basic concepts. We also provide some chapters with more advanced material. Each chapter offers exercises to test your knowledge and explore interesting research problems. When you master these chapters, you will be ready to dig deeper into the literature and create your own contributions to it.

1.4 HOW TO USE THIS BOOK

The best way for you to use this book is to study all of the chapters. (You knew we would say that, right?) We think that the development from neural networks to fuzzy systems to evolutionary computation provides a logical flow within the framework of a semester-long course. You'll find that each of the three main topics is described with basic chapters upfront, which cover theory, framework, and algorithms. These are followed by more advanced chapters covering more specific issues, fine points, and extensions of the basic constructions.

For instructors, presuming a typical 16-week U.S. university semester, you can easily construct three 4-week modules from the basic material with plenty of time remaining for in-class exercises, homework discussions, and computer projects. There's even some time available to pursue more advanced research in your own favorite area. (This is how the "Introduction to CI" class at the University of Missouri (MU) is organized.)

Alternatively, if you want to focus more on one area of CI, you can certainly use this book to do so. For example, if you wanted a course mainly on fuzzy systems, you could use all four of the chapters on fuzzy systems, and then sample from neural networks (to demonstrate the basis for neuro–fuzzy systems) and evolutionary computation (to develop optimization approaches in the design of fuzzy inference

systems). By analogy, you could focus on neural networks or evolutionary computation, and then supplement those materials with the other chapters in the book.

1.5 FINAL THOUGHTS BEFORE YOU GET STARTED

An introductory course on computational intelligence has been taught at the University of Missouri (Jim's place) since 2005. Various texts have been used, including most recently draft chapters from this book. The class is colisted in the Electrical and Computer Engineering Department and the Computer Science Department and is available to both seniors and beginning graduate students.

In at least one semester, students were given a first-day assignment to provide a list of things that computers can't do as well as humans. The following are some of the items from the combined list:

Qualitative classification
Going from specific to general, or vice versa
Consciousness and emotion
Driving a car
Writing a poem
Chatting
Shopping
Handling inaccuracies in problems
Ethics
Natural language in conversation, with idioms
Face recognition
Aesthetics
Adaptivity
Learning (like humans do)

This was from a group of students with little or no background in intelligent systems. Depending on what you read and/or do, you might say that progress (significant in some cases) has been made on creating systems with attributes from that list, and you'd be right. Amazing things are happening. This book will provide you with the background and tools to join in the fun.

As editors-in-chief of the three main IEEE publications in the area of CI, we've had the good fortune to see novel advancements in our fields of interest even before they've been peer-reviewed and published. We've also had the joy of participating in making some of those advancements ourselves.

In fact, we've devoted our lives to advancing the theory and practice of the methods that you'll read about in this textbook. We've done that because we've often found these techniques to offer practical advantages as well as mental challenges. But in the end, we've pursued these lines of research primarily because they're a lot of fun.

We hope that you'll find not only a mathematically and practically challenging set of material in this book, but also that the material ultimately brings you as much enjoyment as it has brought for us, or even more!

Enjoy!

JAMES KELLER, Ph.D.
DERONG LIU, Ph.D.
DAVID FOGEL, Ph.D.

PART I

NEURAL NETWORKS

CHAPTER 2

Introduction and Single-Layer Neural Networks

All of us have a highly interconnected set of some 10^{11} neurons to facilitate our reading, breathing, motion, and thinking. Each of the biological neurons has the complexity of a microprocessor. Remarkably, the human brain is a highly complex, nonlinear, and parallel computer. It has the capability to organize its structural constituents, that is, neurons, so as to perform certain computations many times faster than the fastest digital computer in existence today.

Specifically, the human brain consists of a large number of highly connected elements (approximately 10^4 connections per element) called neurons [Hagan *et al*., 1996]. For our purposes, these neurons have three principal components: the dendrites, the cell body, and the axon. The dendrites are tree-like receptive networks of nerve fibers that carry electrical signals into the cell body. The cell body effectively sums and thresholds these incoming signals. The axon is a single long fiber that carries the signal from the cell body out to other neurons. The point of contact between an axon of one cell and a dendrite of another cell is called a synapse. It is the arrangement of neurons and the strengths of the individual synapses, determined by a complex chemical process, that establishes the function of the neural network. Figure 2.1 shows a simplified schematic diagram of two biological neurons.

Actually, scientists have begun to study how biological neural networks operate. It is generally understood that all biological neural functions, including memory, are stored in the neurons and in the connections between them. Learning is viewed as the establishment of new connections between neurons or the modification of existing connections. Then, one may have the question: Although we have only a rudimentary understanding of biological neural networks, is it possible to construct a small set of simple artificial neurons and then train them to serve a useful function? The answer is "yes." This is accomplished using artificial neural networks, commonly referred to as neural networks, which have been motivated right from its inception by the recognition that the human brain computes in an entirely different way from the conventional digital computer. Figure 2.2 shows a simplified schematic diagram of

Fundamentals of Computational Intelligence: Neural Networks, Fuzzy Systems, and Evolutionary Computation, First Edition. James M. Keller, Derong Liu, and David B. Fogel.

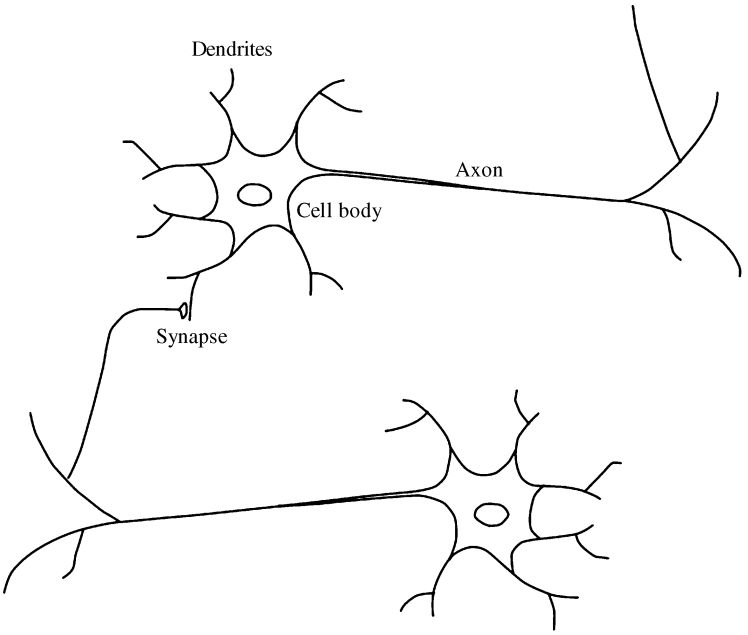

FIGURE 2.1 The schematic diagram of two biological neurons.

two artificial neurons. Here, the two artificial neurons are connected to be a simple artificial neural network and each artificial neuron contains some input and output signals.

The neurons that we consider here are not biological. They are extremely simple abstractions of biological neurons, realized as elements in a program or perhaps as circuits made of silicon. Networks of these artificial neurons do not have a fraction of the power of the human brain. However, they can be trained to perform useful functions. Note that even though biological neurons are very slow compared to electrical circuits, the brain is able to perform many tasks much faster than any conventional computer. One important reason is that the biological neural networks hold massively parallel structure and all of the neurons operate at the same time. Fortunately, the artificial neural networks share this parallel structure, which makes them useful in practice.

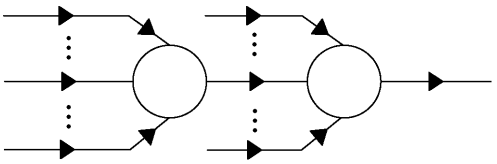

FIGURE 2.2 The schematic diagram of two artificial neurons.

Artificial neural networks do not approach the complexity of the brain. However, there are two main similarities between the biological neural networks and artificial neural networks. One is that the building blocks of both networks are simple computational devices (although artificial neurons are much simpler than biological neurons) that are highly interconnected. The other is that the connections between neurons determine the function of the network.

2.1 SHORT HISTORY OF NEURAL NETWORKS

Generally speaking, the history of neural networks has progressed through both conceptual innovations and implementation developments. However, these advancements seem to have occurred in fits and starts, rather than by steady evolution [Hagan *et al.*, 1996].

Some of the background work for the field of neural networks occurred in the late nineteenth and early twentieth centuries. The primarily interdisciplinary work was conducted by many famous scientists from the fields of physics, psychology, and neurophysiology. In this stage, the research of neural networks emphasized general theories of learning, vision, conditioning, and so on, and did not include specific mathematical models of the neuron operation.

Then, the modern view of neural networks began in the 1940s with the work of Warren McCulloch and Walter Pitts, who showed that networks of artificial neurons could, in principle, compute any arithmetic or logical function. Notice that this important work is often regarded as the origin of the neural network community. Then, scientists proposed a mechanism for learning in biological neurons.

The first practical application of neural networks came in the late 1950s, with the invention of the perceptron network and the associated learning rule [Rosenblatt, 1958]. In this stage, the great success brought large interest to the research of neural networks. However, it was later shown that the basic perceptron network could only solve a limited class of problems. At the same time, scientists introduced a new learning algorithm and used it to train an adaptive linear neural network, which is still in use today. In fact, it was similar in structure and capability to Rosenblatt's perceptron.

Unfortunately, Rosenblatt's networks suffered from the same inherent limitation of what class of problems could be learned. Though Rosenblatt was aware of the limitation and proposed new networks to overcome it, he was not able to modify the learning algorithm to accommodate training more complex networks. Therefore, many people believed that further research on neural networks was a dead end. What's more, considering the fact that there were no powerful digital computers to conduct experiment, the research on neural networks was largely suspended.

Interest in neural networks faltered during the late 1960s because of the lack of new ideas and powerful computers with which to experiment. However, during the 1980s, these impediments were gradually overcome. Hence, research on neural networks increased dramatically. In this stage, new personal computers and workstations, which rapidly grew in capability, became widely available. More importantly, some

new concepts were introduced. Among those, two novel concepts were most responsible for the rebirth of the neural network field. One was the use of statistical mechanics to explain the operation of a certain class of recurrent networks, which could be used as an associative memory. The other was the development of the backpropagation algorithm, which was introduced for helping to train multilayer perceptron networks [Rumelhart *et al*., 1986a, 1986b; Werbos, 1974, 1994].

These new developments reinvigorated the neural network community. In the last several decades, thousands of excellent papers have been written. The neural network technique has found many applications. Now, the field is buzzing with new theoretical and practical work. It is important to notice that many of the advances in neural networks have been related to new concepts, such as innovative architectures and training rules. In addition, the availability of powerful new computers, which test the new concepts, is also of great significance [Hagan *et al*., 1996].

It is apparent that a neural network derives its computing power through its massively parallel distributed structure and also its ability to learn and therefore generalize. The characteristic of generalization refers to the neural network producing reasonable outputs for inputs that were not encountered during training. These two information processing capabilities make it possible for neural networks to solve complex and large-scale problems that are currently intractable. However, in practice, neural networks cannot provide the solution by working individually. Instead, they need to be integrated into a consistent system engineering approach. Specifically, a complex problem of interest is decomposed into a number of relatively simple tasks, and some neural networks are assigned a subset of the tasks that match their inherent capabilities. However, it is important to recognize that we still have a long way to go before we can build a computer architecture that mimics a human brain. Consequently, how to achieve the true brain intelligence via artificial neural network is one of the main research objectives of scientists.

2.2 ROSENBLATT'S NEURON

Artificial neural networks, commonly referred to as "neural networks," represent a technology rooted in many disciplines: neurosciences, mathematics, statistics, physics, computer science, and engineering. Neural networks are potentially massively parallel distributed structures and have the ability to learn and generalize. Generalization denotes the neural network's production of reasonable outputs for inputs not encountered during learning process. Therefore, neural networks can be applied to diverse fields, such as modeling, time series analysis, pattern recognition, signal processing, and system control.

The neuron is the information processing unit of a neural network and the basis for designing numerous neural networks. A fundamental neural model consists of the following basic elements [Haykin, 2009]:

- A set of synapses, or connecting links, each of which is characterized by a weight or strength of its own.

- An adder for summing the input signals, weighted by the respective synaptic strengths of the neuron.
- An activation function for limiting the amplitude of the output of a neuron.
- An externally applied bias, which has the effect of increasing or lowering the net input of the activation function.

The most fundamental network architecture is a single-layer neural network, where the "single-layer" refers to the output layer of computation neurons. Note that we do not count the input layer of source nodes because no computation is performed there.

In neural network community, a signal flow graph is often used to provide a complete description of signal flow in a network. A signal flow graph is a network of directed links that are interconnected at certain points called nodes. The flow of signals in the various parts of the graph complies with the following three rules [Haykin, 1999]:

1. A signal flows along a link only in the direction indicated by the arrow on the link.
2. A node signal equals the algebraic sum of all signals entering the pertinent node via the incoming links.
3. The signal at a node is transmitted to each outgoing link originating from that node, with the transmission being entirely independent of the transfer functions of the outgoing links.

It should be pointed out that there are two different types of links, namely, synaptic links and activation links.

- The behavior of synaptic links is governed by a linear input–output relation. See Figure 2.3, the node signal x_j is multiplied by the synaptic weight w_{kj} to produce the node signal y_k, that is, $y_k = w_{kj}x_j$.
- The behavior of activation links is governed by a nonlinear input–output relation. This is illustrated in Figure 2.4, where $\phi(\cdot)$ is called the nonlinear activation function, that is, $y_k = \phi(x_j)$.

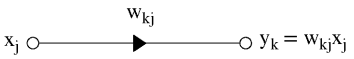

FIGURE 2.3 Illustration of the synaptic link.

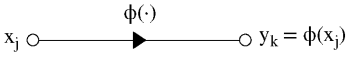

FIGURE 2.4 Illustration of the activation link.

Another expression method that can also be utilized to depict a network is called the architectural graph. Unlike the signal flow graph, the architectural graph possesses the following characteristics [Haykin, 1999]:

1. Source nodes supply input signals to the graph.
2. Each neuron is represented by a signal node called a computation node.
3. The communication links interconnecting the source and computation nodes of the graph carry no weight. They merely provide directions of signal flow in the graph.

Now, we introduce Rosenblatt's neuron [Haykin, 1999, 2009; Rosenblatt, 1958]. Rosenblatt's perceptron occupies a special place in the historical development of neural networks. It was the first algorithmically described neural network, which was built around a nonlinear neuron, namely, the McCulloch–Pitts model. Incidentally, the McCulloch–Pitts model is a neuron stated in recognition of the pioneering work done by McCulloch–Pitts. Rosenblatt's perceptron is an algorithm for learning a binary classifier: a function that maps its input x (a real-valued vector) to an output value f(x) (a single binary value):

$$f(x) = \begin{cases} 1, & \text{if } w \cdot x + b > 0 \\ 0, & \text{otherwise} \end{cases} \tag{2.1}$$

where w is a vector of real-valued weights, $w \cdot x$ is the dot product (which here computes a weighted sum), and b (a real scalar) is the "bias," a constant term that does not depend on any input value. Figure 2.5 shows the signal flow graph of the Rosenblatt's perceptron.

Here, the harder limiter input (i.e., the induced local field) of the neuron is $w \cdot x + b$.

The value of f(x) (0 or 1) is used to classify x as either a positive or a negative instance, in the case of a binary classification problem. The decision rule for the classification is to assign the point represented by the inputs $x_1, x_2, \ldots, x_n$ to class $\aleph_1$ if the perceptron output y is +1 and to class $\aleph_2$ if it is 0. Note that for the case of two input variables, the decision boundary takes the form of a straight line in a two-dimensional plane (see Figure 2.6). If b is negative, then the weighted combination of inputs must produce a positive value greater than |b| in order to push the classifier

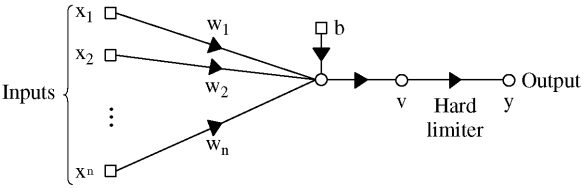

FIGURE 2.5 Signal flow graph of the perceptron.

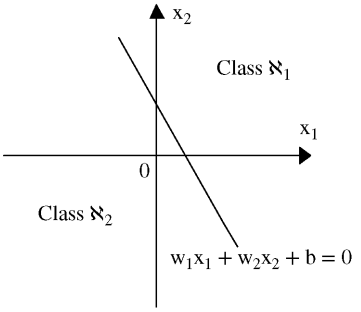

FIGURE 2.6 Illustration of the two-dimensional, two-class pattern classification problem.

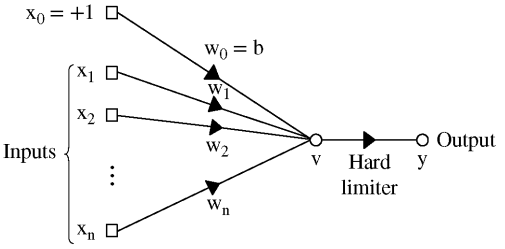

FIGURE 2.7 Equivalent signal flow graph of the perceptron.

neuron over the 0 threshold. Spatially, the bias alters the position (although not the orientation) of the decision boundary.

We now give an equivalent model of the perceptron described in Figure 2.5. Here, the bias b is viewed as a synaptic weight driven by a fixed input equal to +1. Then, the signal flow graph is shown in Figure 2.7.

2.3 PERCEPTRON TRAINING ALGORITHM

Now we consider the performance of the perceptron network and are in a position to introduce the perceptron learning rule. This learning rule is an example of supervised training, in which the learning rule is provided with a set of examples of proper network behavior:

$$\{p_1, t_1\}, \{p_2, t_2\}, \ldots, \{p_q, t_q\} \tag{2.2}$$

Here p_i is an input to the network and t_i is the corresponding target output. As each input is applied to the network, the network output is compared with the target. The learning rule then adjusts the weights and biases of the network in order to move the network output closer to the target.

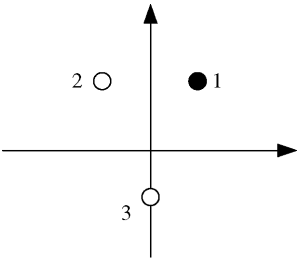

FIGURE 2.8 The test problem.

2.3.1 Test Problem

In our presentation of the perceptron learning rule, we will begin with a simple test problem and will experiment with possible rules to develop some intuition about how the rule should work. The input–target pairs for our test problem are

$$\left\{\mathbf{p}_1 = \begin{bmatrix}1\\2\end{bmatrix}, t_1 = 1\right\}, \quad \left\{\mathbf{p}_2 = \begin{bmatrix}-1\\2\end{bmatrix}, t_2 = 0\right\}, \quad \left\{\mathbf{p}_3 = \begin{bmatrix}0\\-1\end{bmatrix}, t_3 = 0\right\}$$

where $\mathbf{p}_i$ represents the input and t_i represents the corresponding target output. The problem is displayed graphically in Figure 2.8, where the two input vectors whose target is 0 are represented with a light circle ○, and the vector whose target is 1 is represented with a dark circle ●. This is a very simple problem, and we could almost obtain a solution by inspection. This simplicity will help us gain some intuitive understanding of the basic concepts of the perceptron learning rule.

The network for this problem should have two inputs and one output. To simplify our development of the learning rule, we will begin with a network without a bias. The network will then have just two parameters $w_{1,1}$ and $w_{1,2}$, as shown in Figure 2.9.

By removing the bias, we are left with a network whose decision boundary must pass through the origin. We need to ensure that this network is still able to solve the test problem. There must be an allowable decision boundary that can separate the

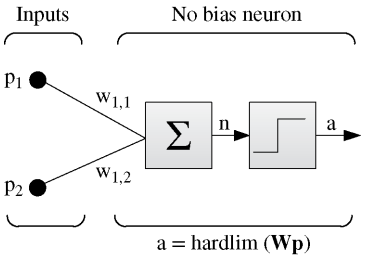

FIGURE 2.9 Test problem network.

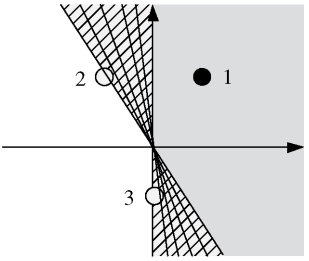

FIGURE 2.10 The boundaries.

vectors p_2 and p_3 from the vector p_1. Figure 2.10 illustrates that there are indeed an infinite number of such boundaries.

Figure 2.11 shows the weight vectors that correspond to the allowable decision boundaries. (Recall that the weight vector is orthogonal to the decision boundary.) We would like a learning rule that will find a weight vector that points to one of these directions. Remember that the length of the weight vector does not matter; only its direction is important.

2.3.2 Constructing Learning Rules

Training begins by assigning some initial values to the network parameters. In this case, we are training a two-input/single-output network without a bias, so we can only initialize its two weights. Here we set the elements of the weight vector ${}_1w$ to the following randomly generated values:

$$ {}_1w^T = \begin{bmatrix} 1.0, & -0.8 \end{bmatrix} \tag{2.3} $$

We will now begin presenting the input vectors to the network and find the corresponding outputs, which we will call ϑ. We begin with p_1:

$$ \begin{aligned} \vartheta &= \text{hardlim}({}_1w^T p_1) = \text{hardlim}\left([1.0 - 0.8] \begin{bmatrix} 1 \\ 2 \end{bmatrix} \right) \\ \vartheta &= \text{hardlim}(-0.6) = 0 \end{aligned} \tag{2.4} $$

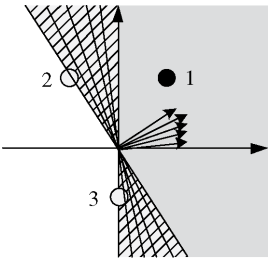

FIGURE 2.11 The weight vectors and boundaries.

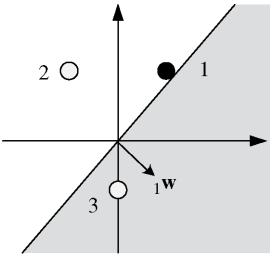

FIGURE 2.12 The classification result for the first input.

The network has not returned the correct value. The network output is 0, while the target response t_1 is 1.

We can see what happened in Figure 2.12. The initial weight vector results in a decision boundary that incorrectly classifies the vector p_1. We need to alter the weight vectors so that it points more toward p_1, so that in the future it has a better chance of classifying it correctly.

One approach would be to set ${}_1w$ equal to p_1. This is simple and would ensure that p_1 was classified properly in the future. Unfortunately, it is easy to construct a problem for which this rule cannot find a solution. Figure 2.13 shows a problem that cannot be solved with the weight vectors pointing directly at either of the two class 1 vectors. If we apply the rule ${}_1w = p$ every time one of these vectors is misclassified, the network's weights will simply oscillate back and forth and will never find a solution.

Another possibility would be to add p_1 to ${}_1w$. Adding p_1 to ${}_1w$ would make ${}_1w$ point more in the direction of p_1. Repeated presentations of p_1 would cause the direction of ${}_1w$ to asymptotically approach the direction of p_1. This rule can be stated:

$$\text{If } t = 1 \text{ and } \vartheta = 0, \text{ then } {}_1w^{new} = {}_1w^{old} + p \tag{2.5}$$

Applying this rule to our test problem results in new values for ${}_1w$:

$${}_1w^{new} = {}_1w^{old} + p_1 = \begin{bmatrix} 1.0 \\ -0.8 \end{bmatrix} + \begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 2.0 \\ 1.2 \end{bmatrix} \tag{2.6}$$

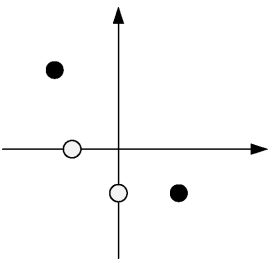

FIGURE 2.13 Another test problem that poses a challenge for setting ${}_1w = p_1$.