Neural Computing

Theory and Practice

Philip D. Wasserman

ANZA Research, Inc.

VNR VAN NOSTRAND REINHOLD
New York

Library of Congress Catalog Card Number 88-34842
ISBN 0-442-20743-3

Printed in the United States of America

Van Nostrand Reinhold
115 Fifth Avenue
New York, New York 10003

Van Nostrand Reinhold International Company Limited
11 New Fetter Lane
London EC4P 4EE, England

Van Nostrand Reinhold
480 La Trobe Street
Melbourne, Victoria 3000, Australia

Macmillan of Canada
Division of Canada Publishing Corporation
164 Commander Boulevard
Agincourt, Ontario M1S 3C7, Canada

16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1

Library of Congress Cataloging-in-Publication Data
Wasserman, Philip D., 1937–
Neural computing : theory and practice / Philip D. Wasserman.
p. cm.
Includes bibliographies and index.
ISBN 0-442-20743-3
1. Neural computers. I. Title.
QA76.5.W353 1989 88-34842
006.3—dc19 CIP

Contents

are labeled as such and references point to more detailed treatments.

This book need not be read from cover to cover. Each chapter is intended to be self-contained, assuming familiarity only with the topics of Chapters 1 and 2. While this implies a certain amount of repetition, most readers should not find it onerous.

Practicality has been a primary objective. If the chapters are studied carefully, it should be possible to implement most of the networks on a general-purpose computer. The reader is urged to do so; no other method will produce the same depth of understanding.

ACKNOWLEDGMENTS

First and foremost, I would like to thank my wife, Sarah, for her encouragement and tolerance during the months I spent in the company of my word processor.

Also, I would like to thank my friends and colleagues, who gave so generously of their time and knowledge, corrected my errors, and created an environment in which ideas developed rapidly. I would like to extend my special appreciation to Dr. Surapol Dasananda, Santa Clara University; Dr. Elizabeth Center, College of Notre Dame; Dr. Peter Rowe, College of Notre Dame; Charles Rockwell, Microlog Corp.; Tom Schwartz, The Schwartz Associates; Dennis Reinhardt, Dair Corp.; Coe Miles-Schlichting; and Douglas Marquardt. Thanks also are due to Kyla Carlson and Nang Cao for their help in preparing the illustrations.

I must, of course, take the blame for any residual errors; they couldn't watch me every minute.

Introduction

WHY ARTIFICIAL NEURAL NETWORKS?

After two decades of near eclipse, interest in artificial neural networks has grown rapidly over the past few years. Professionals from such diverse fields as engineering, philosophy, physiology, and psychology are intrigued by the potential offered by this technology and are seeking applications within their disciplines.

This resurgence of interest has been fired by both theoretical and application successes. Suddenly, it appears possible to apply computation to realms previously restricted to human intelligence; to make machines that learn and remember in ways that bear a striking resemblance to human mental processes; and to give a new and significant meaning to the much-abused term *artificial intelligence*.

CHARACTERISTICS OF ARTIFICIAL NEURAL NETWORKS

Artificial neural networks are biologically inspired; that is, they are composed of elements that perform in a manner that is analogous to the most elementary functions of the biological neuron. These elements are then organized in a way that may (or may not) be related to the anatomy of the brain. Despite this superficial resem-

blance, artificial neural networks exhibit a surprising number of the brain's characteristics. For example, they learn from experience, generalize from previous examples to new ones, and abstract essential characteristics from inputs containing irrelevant data.

Despite these functional similarities, not even the most optimistic advocate will suggest that artificial neural networks will soon duplicate the functions of the human brain. The actual "intelligence" exhibited by the most sophisticated artificial neural networks is below the level of a tapeworm; enthusiasm must be tempered by current reality. It is, however, equally incorrect to ignore the surprisingly brainlike performance of certain artificial neural networks. These abilities, however limited they are today, hint that a deep understanding of human intelligence may lie close at hand, and along with it a host of revolutionary applications.

Learning

Artificial neural networks can modify their behavior in response to their environment. This factor, more than any other, is responsible for the interest they have received. Shown a set of inputs (perhaps with desired outputs), they self-adjust to produce consistent responses. A wide variety of training algorithms has been developed, each with its own strengths and weaknesses. As we point out later in this volume, there are important questions yet to be answered regarding what things a network can be trained to do, and how the training should be performed.

Generalization

Once trained, a network's response can be, to a degree, insensitive to minor variations in its input. This ability to see through noise and distortion to the pattern that lies within is vital to pattern recognition in a real-world environment. Overcoming the literal-mindedness of the conventional computer, it produces a system that can deal with the imperfect world in which we live. It is important to note that the artificial neural network generalizes automatically as a result of its structure, not by using human intelligen nbedded in the form of ad hoc computer programs.

Abstraction

Some artificial neural networks are capable of abstracting the essence of a set of inputs. For example, a network can be trained on a sequence of distorted versions of the letter A. After adequate training, application of such a distorted example will cause the network to produce a perfectly formed letter. In one sense, it has learned to produce something that it has never seen before.

This ability to extract an ideal from imperfect inputs raises interesting philosophical issues; it is reminiscent of the concept of ideals found in Plato's *Republic*. In any case, extracting idealized prototypes is a highly useful ability in humans; it seems that now we may share it with artificial neural networks.

Applicability

Artificial neural networks are not a panacea. They are clearly unsuited to such tasks as calculating the payroll. It appears that they will, however, become the preferred technique for a large class of pattern-recognition tasks that conventional computers do poorly, if at all.

HISTORICAL PERSPECTIVE

Humans have always wondered about their own thoughts. This self-referential inquiry, the mind thinking of itself, may be a uniquely human characteristic. Speculations on the nature of thought abound, ranging from the spiritual to the anatomical. With philosophers and theologians often opposing the opinions of physiologists and anatomists, the questions have been hotly debated to little avail, as the subject is notoriously difficult to study. Those relying on introspection and speculation have arrived at conclusions that lack the rigor demanded by the physical sciences. Experimenters have found the brain and nervous system to be difficult to observe and perplexing in organization. In short, the powerful methods of scientific inquiry that have changed our view of physical reality have been slow in finding application to the understanding of humans themselves.

Neurobiologists and neuroanatomists have made substantial progress. Painstakingly mapping out the structure and function of the human nervous system, they came to understand much of the brain's "wiring," but little of its operation. As their knowledge grew, the complexity was found to be staggering. Hundreds of billions of neurons, each connecting to hundreds or thousands of others, comprise a system that dwarfs our most ambitious dreams of supercomputers. Nevertheless, the brain is gradually yielding its secrets to one of humankind's most sustained and ambitious inquiries.

The improved understanding of the functioning of the neuron and the pattern of its interconnections has allowed researchers to produce mathematical models to test their theories. Experiments can now be conducted on digital computers without involving human or animal subjects, thereby solving many practical and ethical problems. From early work it became apparent that these models not only mimicked functions of the brain, but that they were capable of performing useful functions in their own right. Hence, two mutually reinforcing objectives of neural modeling were defined and remain today: first, to understand the physiological and psychological functioning of the human neural system; and second, to produce computational systems (artificial neural networks) that perform brainlike functions. It is the latter objective that is the major focus of this book.

Along with the progress in neuroanatomy and neurophysiology, psychologists were developing models of human learning. One such model, which has proved most fruitful, was that of D. O. Hebb, who in 1949 proposed a learning law that became the starting point for artificial neural network training algorithms. Augmented today by many other methods, it showed scientists of that era how a network of neurons could exhibit learning behavior.

In the 1950s and 1960s, a group of researchers combined these biological and psychological insights to produce the first artificial neural networks. Initially implemented as electronic circuits, they were later converted to the more flexible medium of computer simulation, the most common realization today. Early successes produced a burst of activity and optimism. Marvin Minsky, Frank Rosenblatt, Bernard Widrow, and others developed networks consisting of a single layer of artificial neurons. Often called perceptrons, they were applied to such diverse problems as weather prediction, electrocardiogram analysis, and artificial vision. It seemed for a time that the key to intelligence had been found; reproducing the human brain was only a matter of constructing a large enough network.

This illusion was soon dispelled. Networks failed to solve problems superficially similar to those they had been successful in solving. These unexplained failures launched a period of intense analysis. Marvin Minsky, carefully applying mathematical techniques, developed rigorous theorems regarding network operation. His research led to the publication of the book *Perceptrons* (Minsky and Papert 1969), in which he and Seymore Papert proved that the single-layer networks then in use were theoretically incapable of solving many simple problems, including the function performed by a simple exclusive-or gate. Nor was Minsky optimistic about the potential for progress:

> The Perceptron has shown itself worthy of study despite (and even because of!) its severe limitations. It has many features that attract attention: its linearity; its intriguing learning theorem; its clear paradigmatic simplicity as a kind of parallel computation. There is no reason to suppose that any of these virtues carry over to the many-layered version. Nevertheless, we consider it to be an important research problem to elucidate (or reject) our intuitive judgment that the extension is sterile.
>
> Perhaps some powerful convergence theorem will be discovered, or some profound reason for the failure to produce an interesting "learning theorem" for the multilayered machine will be found. (pp. 231–32)

Minsky's brilliance, rigor, and prestige gave the book great credibility: its conclusions were unassailable. Discouraged researchers left the field for areas of greater promise, government agencies redirected their funding, and artificial neural networks lapsed into obscurity for nearly two decades.

Nevertheless, a few dedicated scientists such as Teuvo Kohonen, Stephen Grossberg, and James Anderson continued their efforts. Often underfunded and unappreciated, some researchers had difficulty finding publishers; hence, research published during the

1970s and early 1980s is found scattered among a wide variety of journals, some of which are rather obscure. Gradually, a theoretical foundation emerged, upon which the more powerful multilayer networks of today are being constructed. Minsky's appraisal has proven excessively pessimistic; networks are now routinely solving many of the problems that he posed in his book.

In the past few years, theory has been translated into application, and new corporations dedicated to the commercialization of the technology have appeared. There has been an explosive increase in the amount of research activity. With four major conventions in 1987 in the field of artificial neural networks, and over 500 technical papers published, the growth rate has been phenomenal.

The lesson to be learned from this history is found in Clark's law. Propounded by the writer and scientist Arthur C. Clark, it states in effect that if a respected senior scientist says a thing can be done, he or she is almost always correct; if the scientist says it cannot be done, he or she is almost always wrong. The history of science is a chronicle of mistakes and partial truths. Today's dogma becomes tomorrow's rubbish. Unquestioning acceptance of "facts," whatever the source, can cripple scientific inquiry. From one point of view, Minsky's excellent scientific work led to an unfortunate hiatus in the progress of artificial neural networks. There is no doubt, however, that the field had been plagued by unsupported optimism and an inadequate theoretical basis. It may be that the shock provided by *Perceptrons* allowed a period for the necessary maturation of the field.

ARTIFICIAL NEURAL NETWORKS TODAY

There have been many impressive demonstrations of artificial neural network capabilities: a network has been trained to convert text to phonetic representations, which were then converted to speech by other means (Sejnowsky and Rosenberg 1987); another network can recognize handwritten characters (Burr 1987); and a neural network–based image-compression system has been devised (Cottrell, Munro, and Zipser 1987). These all use the backpropagation network, perhaps the most successful of the current algorithms. Backpropagation, invented independently in three

separate research efforts (Werbos 1974; Parker 1982; and Rumelhart, Hinton, and Williams 1986), provides a systematic means for training multilayer networks, thereby overcoming limitations presented by Minsky.

As we point out in the chapters that follow, backpropagation is not without its problems. First, there is no guarantee that the network can be trained in a finite amount of time. Many training efforts fail after consuming large amounts of computer time. When this happens, the training attempt must be repeated—with no certainty that the results will be any better. Also, there is no assurance that the network will train to the best configuration possible. So-called local minima can trap the training algorithm in an inferior solution.

Many other network algorithms have been developed that have specific advantages; several of these are discussed in the chapters that follow. It should be emphasized that none of today's networks represents a panacea; all of them suffer from limitations in their ability to learn and recall.

We are presented with a field having demonstrated performance, unique potential, many limitations, and a host of unanswered questions. It is a situation calling for optimism tempered with caution. Authors tend to publish their successes and give their failures little publicity, thereby creating an impression that may not be realistic. Those seeking venture capital to start new firms must present a convincing projection of substantial accomplishments and profits. There exists, therefore, a substantial danger that artificial neural networks will be oversold before their time, promising performance without the capability for delivery. If this happens, the entire field could suffer a loss of credibility, possibly relapsing into the Dark Ages of the 1970s. Much solid work is required to improve existing networks. New techniques must be developed, existing methods strengthened, and the theoretical foundation broadened before this field can realize its full potential.

PROSPECTS FOR THE FUTURE

Artificial neural networks have been proposed for tasks ranging from battlefield management to minding the baby. Potential applications are those where human intelligence functions ort-

lessly and conventional computation has proven cumbersome or inadequate. This application class is at least as large as that serviced by conventional computation, and the vision arises of artificial neural networks taking their place alongside of conventional computation as an adjunct of equal size and importance. This will happen only if fundamental research yields results at a rapid rate, as today's theoretical foundations are inadequate to support such projections.

Artificial Neural Networks and Expert Systems

The field of artificial intelligence has been dominated in recent years by the logical- and symbol-manipulation disciplines. For example, expert systems have been widely acclaimed and have achieved many notable successes—as well as many failures. Some say that artificial neural networks will replace current artificial intelligence, but there are many indications that the two will coexist and be combined into systems in which each technique performs the tasks for which it is best suited.

This viewpoint is supported by the way that humans operate in the world. Activities requiring rapid responses are governed by pattern recognition. Since actions are produced rapidly and without conscious effort, this mode of operation is essential for the quick, spontaneous responses needed to survive in a hostile environment. Consider the consequences if our ancestors had to reason out the correct response to a leaping carnivore!

When our pattern-recognition system fails to produce an unambiguous interpretation (and when time permits), the matter is referred to the higher mental functions. These may require more information and certainly more time, but the quality of the resulting decisions can be superior.

One can envision an artificial system that mimics this division of labor. An artificial neural network would produce an appropriate response to its environment under most circumstances. Because such networks can indicate the confidence level associated with each decision, it would "know that it did not know," and would refer that case to an expert system for resolution. The decisions

made at this higher level would be concrete and logical, but might require the gathering of additional facts before a conclusion could be reached. The combination of the two systems would be more robust than either acting alone, and it would follow the highly successful model provided by biological evolution.

Reliability Considerations

Before artificial neural networks can be applied where human life or valuable assets are at stake, questions regarding their reliability must be resolved.

Like the humans whose brain structure they mimic, artificial neural networks retain a degree of unpredictability. Unless every possible input is tried, there is no way to be certain of the precise output. In a large network such exhaustive testing is impractical and statistical estimates of performance must suffice. Under some circumstances this is intolerable. For example, what is an acceptable error rate for a network controlling a space defense system? Most people would say that any error is intolerable; it might result in unthinkable death and destruction. This attitude is not changed by the fact that a human in the same situation might also make mistakes.

The problem lies in the expectation that computers are absolutely error free. Because artificial neural networks will sometimes make errors even when they are functioning correctly, many feel that this translates into unreliability, a characteristic we have found unacceptable in our machines.

A related difficulty lies in the inability of traditional artificial neural networks to "explain" how they solve problems. The internal representations that result from training are often so complex as to defy analysis in all but the most trivial cases. This is closely related to our inability to explain how we recognize a person despite differences in distance, angle, illumination, and the passage of years. An expert system can trace back through its own reasoning process so that a human can check it for reasonableness. Incorporation of this ability into artificial neural networks has been reported (Gallant 1988) and its development may have an important effect upon the acceptability of these systems.

SUMMARY

Artificial neural networks represent a major extension of computation. They promise human-made devices that perform functions heretofore reserved for human beings. Dull, repetitive, or dangerous tasks can be performed by machines and entirely new applications will arise as the technology matures.

The theoretical foundations of artificial neural networks are expanding rapidly, but they are currently inadequate to support the more optimistic projections. Viewed historically, theory has developed faster than pessimists had projected and slower than optimists had hoped, a typical situation. Today's surge of interest has set thousands of researchers to work in the field. It is reasonable to expect a rapid increase in our understanding of artificial neural networks leading to improved network paradigms and a host of application opportunities.

References

Burr, D. J. 1987. Experiments with a connectionist text reader. In *Proceedings of the First International Conference on Neural Networks,* eds. M. Caudill and C. Butler, vol. 4, pp. 717–24. San Diego, CA: SOS Printing.

Cottrell, G. W., Munro, P., and Zipser, D. 1987. Image compression by backpropagation: An example of extensional programming. *Advances in cognitive science* (vol. 3). Norwood, NJ: Ablex.

Gallant, S. I. 1988. Connectionist expert systems. *Communications of the ACM* 31:152–69.

Minsky, M., and Papert, S. 1969. *Perceptrons*. Cambridge, MA: MIT Press.

Parker, D. B. 1982. Learning-logic. Invention Report, S81-64, File 1. Office of Technology Licensing, Stanford University.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. 1986. Learning internal representations by error propagation. In *Parallel distributed processing*, vol. 1, pp. 318–62. Cambridge, MA: MIT Press.

Sejnowski, T. J., and Rosenberg, C. R. 1987. Parallel networks that learn to pronounce English text. *Complex Systems* 3:145–68.

Werbos, P. J. 1974. *Beyond regression: New tools for prediction and analysis in the behavioral sciences*. Masters thesis, Harvard University.

1

Fundamentals of Artificial Neural Networks

Artificial neural networks have been developed in a wide variety of configurations. Despite this apparent diversity, network paradigms have a great deal in common. In this chapter, recurring themes are briefly identified and discussed so they will be familiar when encountered later in the book.

Notation and representations presented here have been selected as most representative of current practice (there are no published standards), and are used throughout the book.

THE BIOLOGICAL PROTOTYPE

Artificial neural networks are biologically inspired; that is, researchers are usually thinking about the organization of the brain when considering network configurations and algorithms. At this point the correspondence may end. Knowledge about the brain's overall operation is so limited that there is little to guide those who would emulate it. Hence, network designers must go beyond current biological knowledge, seeking structures that perform useful functions. In many cases, this necessary shift discards biological plausibility; the brain becomes a metaphor; networks are produced that are organically infeasible or require a highly improbable set of assumptions about brain anatomy and functioning.

Despite this tenuous, often nonexistent relationship with biolo-

gy, artificial neural networks continue to evoke comparisons with the brain. Their functions are often reminiscent of human cognition; hence, it is difficult to avoid making the analogy. Unfortunately, such comparisons are not benign; they create unrealistic expectations that inevitably result in disillusionment. Research funding based on false hopes can evaporate in the harsh light of reality as it did in the 1960s, and this promising field could again go into eclipse if restraint is not exercised.

Despite the preceding caveats, it remains profitable to understand something of the mammalian nervous system; it is an entity that successfully performs the tasks to which our artificial systems only aspire. The following discussion is brief; Appendix A provides a more extensive (but by no means complete) treatment of the mammalian nervous system for those who wish to know more about this fascinating subject.

The human nervous system, built of cells called neurons, is of staggering complexity. An estimated 10^{11} neurons participate in perhaps 10^{15} interconnections over transmission paths that may range for a meter or more. Each neuron shares many characteristics with the other cells in the body, but has unique capabilities to receive, process, and transmit electrochemical signals over the neural pathways that comprise the brain's communication system.

Figure 1-1 shows the structure of a pair of typical biological neurons. Dendrites extend from the cell body to other neurons where they receive signals at a connection point called a synapse. On the receiving side of the synapse, these inputs are conducted to the cell body. There they are summed, some inputs tending to excite the cell, others tending to inhibit its firing. When the cumulative excitation in the cell body exceeds a threshold, the cell fires, sending a signal down the axon to other neurons. This basic functional outline has many complexities and exceptions; nevertheless, most artificial neural networks model only these simple characteristics.

THE ARTIFICIAL NEURON

The artificial neuron was designed to mimic the first-order characteristics of the biological neuron. In essence, a set of inputs are

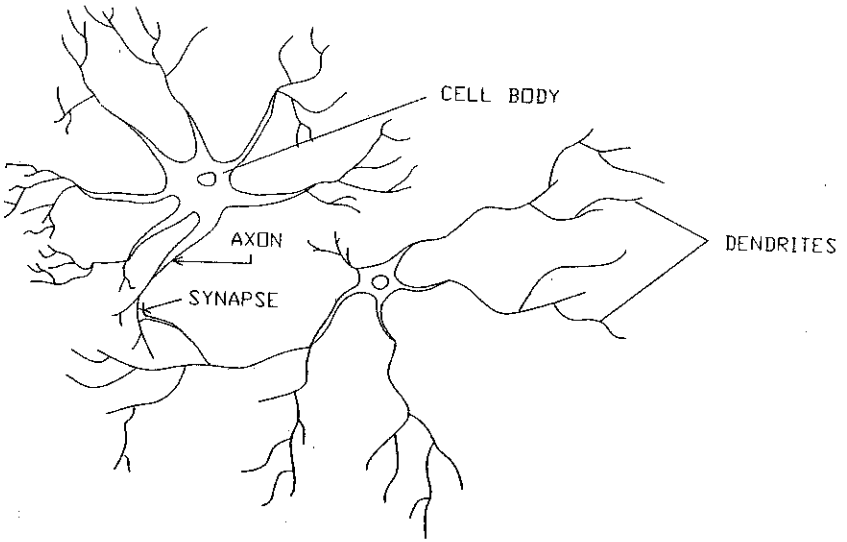
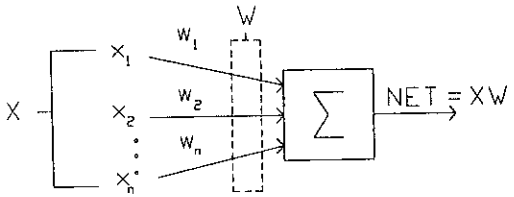


Figure 1-1. Biological Neuron

applied, each representing the output of another neuron. Each input is multiplied by a corresponding weight, analogous to a synaptic strength, and all of the weighted inputs are then summed to determine the activation level of the neuron. Figure 1-2 shows a model that implements this idea. Despite the diversity of network paradigms, nearly all are based upon this configuration. Here, a set of inputs labeled $x_1, x_2, \ldots, x_n$ is applied to the artificial neuron. These inputs, collectively referred to as the vector* $\mathbf{X}$, correspond to the signals into the synapses of a biological neuron. Each signal is multiplied by an associated weight $w_1, w_2, \ldots, w_n$, before it is applied to the summation block, labeled Σ. Each weight corresponds to the "strength" of a single biological synaptic connection. (The set of weights is referred to collectively as the vector $\mathbf{W}$.) The summation block, corresponding roughly to the biologi-

*A few forms of vector notation are used throughout the book. Doing so dramatically simplifies mathematical expressions, thereby preventing the details from obscuring the concepts. Appendix B contains a short tutorial on the vector notation that is used. If your vector skills are rusty, reading this short tutorial now will bring great rewards in speed and depth of comprehension of the material that follows.



$$NET = X_1 W_1 + X_2 W_2 + \ldots + X_n W_n$$

Figure 1-2. Artificial Neuron

cal cell body, adds all of the weighted inputs algebraically, producing an output that we call NET. This may be compactly stated in vector notation as follows:

$$NET = \mathbf{XW}$$

Activation Functions

The NET signal is usually further processed by an activation function F to produce the neuron's output signal, *OUT*. This may be a simple linear function,

$$OUT = K(NET)$$

where K is a constant, a threshold function,

$$OUT = 1 \quad \text{if NET} > T$$
$$OUT = 0 \quad \text{otherwise}$$

where T is a constant threshold value, or a function that more accurately simulates the nonlinear transfer characteristic of the biological neuron and permits more general network functions.

In Figure 1-3 the block labeled F accepts the NET output and produces the signal labeled OUT. If the F processing block compresses the range of NET, so that OUT never exceeds some low

limits regardless of the value of NET, F is called a *squashing function*. The squashing function is often chosen to be the logistic function or "sigmoid" (meaning S-shaped) as shown in Figure 1-4a. This function is expressed mathematically as $F(x) = 1/(1 + e^{-x})$. Thus,

$$OUT = 1/(1 + e^{-NET})$$

By analogy to analog electronic systems, we may think of the activation function as defining a nonlinear gain for the artificial neuron. This gain is calculated by finding the ratio of the change in OUT to a small change in NET. Thus, gain is the slope of the curve at a specific excitation level. It varies from a low value at large negative excitations (the curve is nearly horizontal), to a high value at zero excitation, and it drops back as excitation becomes very large and positive. Grossberg (1973) found that this nonlinear gain characteristic solves the noise-saturation dilemma that he posed; that is, how can the same network handle both small and large signals? Small input signals require high gain through the network if they are to produce usable output; however, a large number of cascaded high-gain stages can saturate the output with the amplified noise (random variations) that is present in any realizable network. Also, large input signals will saturate high-gain stages, again eliminating any usable output. The central high-gain region of the logistic function solves the problem of processing small signals, while its regions of decreasing gain at positive and negative extremes are appropriate for large excitations. In this way, a neuron performs with appropriate gain over a wide range of input levels.

Another commonly used activation function is the hyperbolic

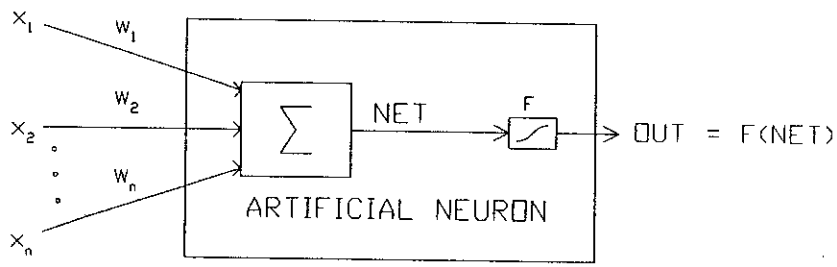


Figure 1-3. Artificial Neuron with Activation Function

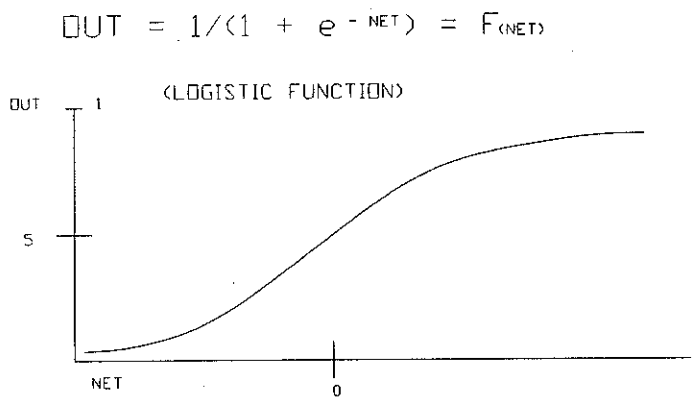


Figure 1-4a. Sigmoidal Logistic Function

tangent. It is similar in shape to the logistic function and is often used by biologists as a mathematical model of nerve-cell activation. Used as an artificial neural network activation function it is expressed as follows:

$$\text{OUT} = \tanh(x)$$

Like the logistic function, the hyperbolic tangent function is S shaped, but is symmetrical about the origin, resulting in OUT having the value 0 when NET is 0 (see Figure 1-4b). Unlike the logistic function, the hyperbolic tangent function has a bipolar value for OUT, a characteristic that has been shown to be beneficial in certain networks (see Chapter 3).

This simple model of the artificial neuron ignores many of the characteristics of its biological counterpart. For example, it does not take into account time delays that affect the dynamics of the system; inputs produce an immediate output. More important, it does not include the effects of synchronism or the frequency-modulation function of the biological neuron, characteristics that some researchers feel to be crucial.

Despite these limitations, networks formed of these neurons exhibit attributes that are strongly reminiscent of the biological system. Perhaps enough of the essential nature of the biological neuron has been captured to produce responses like the biological system, or perhaps the similarity is coincidental; only time and research will tell.

SINGLE-LAYER ARTIFICIAL NEURAL NETWORKS

Although a single neuron can perform certain simple pattern detection functions, the power of neural computation comes from connecting neurons into networks. The simplest network is a group of neurons arranged in a layer as shown on the right side of Figure 1-5. Note that the circular nodes on the left serve only to distribute the inputs; they perform no computation and hence will not be considered to constitute a layer. For this reason, they are shown as circles to distinguish them from the computing neurons, which are shown as squares. The set of inputs **X** has each of its elements connected to each artificial neuron through a separate weight. Early artificial neural networks were no more complex than this. Each neuron simply output a weighted sum of the inputs to the network. Actual artificial and biological networks may have many of the connections deleted, but full connectivity is shown

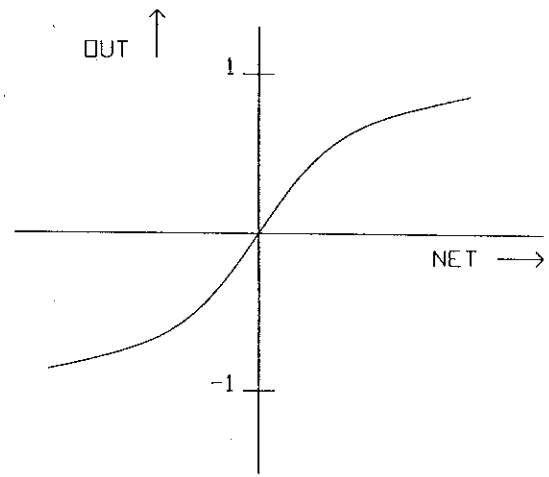


Figure 1-4b. Hyperbolic Tangent Function

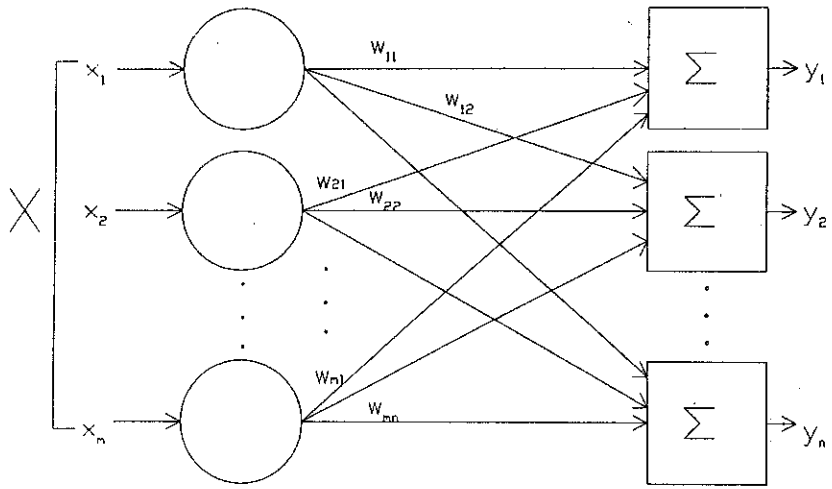


Figure 1-5. Single-Layer Neural Network

for reasons of generality. Also, there may be connections between the outputs and inputs of neurons in a layer; such configurations are treated in Chapter 6.

It is convenient to consider the weights to be elements of a matrix **W**. The dimensions of the matrix are m rows by n columns, where m is the number of inputs and n the number of neurons. For example, the weight connecting the third input to the second neuron would be $w_{3,2}$. In this way it may be seen that calculating the set of neuron NET outputs **N** for a layer is a simple matrix multiplication. Thus $\mathbf{N} = \mathbf{XW}$, where **N** and **X** are row vectors.

MULTILAYER ARTIFICIAL NEURAL NETWORKS

Larger, more complex networks generally offer greater computational capabilities. Although networks have been constructed in every imaginable configuration, arranging neurons in layers mimics the layered structure of certain portions of the brain. These multilayer networks have been proven to have capabilities beyond

those of a single layer (see Chapter 2), and in recent years, algorithms have been developed to train them.

Multilayer networks may be formed by simply cascading a group of single layers; the output of one layer provides the input to the subsequent layer. Figure 1-6 shows such a network, again drawn fully connected.

The Nonlinear Activation Function

Multilayer networks provide no increase in computational power over a single-layer network unless there is a nonlinear activation function between layers. Calculating the output of a layer consists of multiplying the input vector by the first weight matrix, and then (if there is no nonlinear activation function) multiplying the resulting vector by the second weight matrix. This may be expressed as:

$$(\mathbf{XW}_1)\mathbf{W}_2$$

Since matrix multiplication is associative, the terms may be regrouped and written:

$$\mathbf{X}(\mathbf{W}_1\mathbf{W}_2)$$

This shows that a two-layer linear network is exactly equivalent to a single layer having a weight matrix equal to the product of the two weight matrixes. Hence, any multilayer linear network can be replaced by an equivalent one-layer network. In Chapter 2 we point out that single-layer networks are severely limited in their computational capability; thus, the nonlinear activation functions are vital to the expansion of the network's capability beyond that of the single-layer network.

Recurrent Networks

The networks considered up to this point have no feedback connections, that is, connections through weights extending from the outputs of a layer to the inputs of the same or previous layers. This

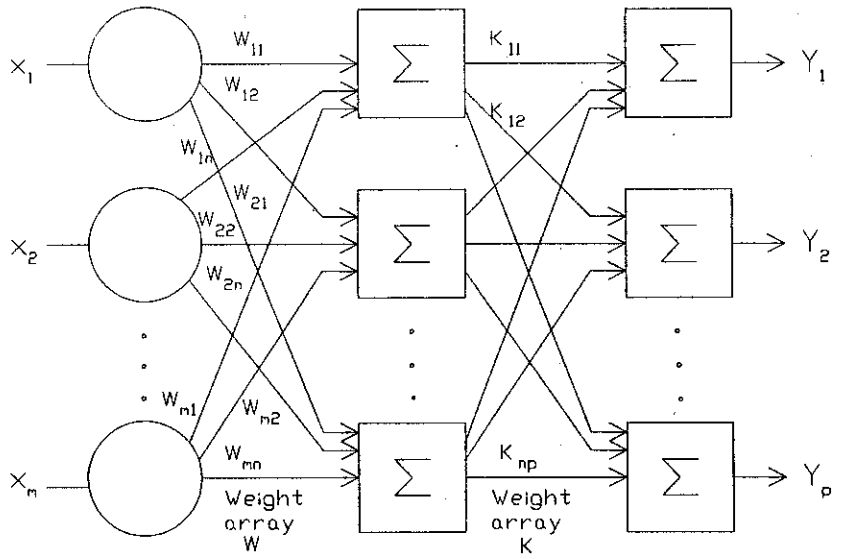


Figure 1-6. Two-Layer Neural Network

special class, called nonrecurrent or feedforward networks, is of considerable interest and is widely applied.

More general networks that do contain feedback connections are said to be recurrent. Nonrecurrent networks have no memory; their output is solely determined by the current inputs and the values of the weights. In some configurations, recurrent networks recirculate previous outputs back to inputs; hence, their output is determined both by their current input and their previous outputs. For this reason recurrent networks can exhibit properties very similar to short-term memory in humans in that the state of the network outputs depends in part upon their previous inputs.

TERMINOLOGY, NOTATION, AND REPRESENTATION OF ARTIFICIAL NEURAL NETWORKS

Unfortunately, there are neither published standards nor general agreement among authors regarding terms, notation, and the graphical representations for artificial neural networks. Identical network paradigms can appear entirely differently when presented by different authors. In this book, the most common and self-descriptive terms have been selected and used consistently.

Terminology

Many authors avoid the term "neuron" when referring to the artificial neuron, recognizing that it is a crude approximation of its biological model. This book uses the terms "neuron," "cell," and "unit" interchangeably as shorthand for "artificial neuron," as these words are succinct and self-explanatory.

Notation: Differential Equations versus Difference Equations

Learning algorithms, like artificial neural networks in general, can be presented in either differential-equation or difference-equation form. The differential-equation representation assumes that the processes are continuous, operating much like a large analog network. Viewing the biological system at a microscopic level, this is not true; the activation level of a biological neuron is determined by the average rate at which it emits discrete action potential pulses down its axon. This average rate is commonly treated as an analog quantity, but it is important to remember the underlying reality.

If one wishes to simulate artificial neural networks on an analog computer, differential-equation representations are highly desirable. However, most work today is being done on digital computers, making the difference-equation form most appropriate, as these equations can be converted easily into computer programs. For this reason, the difference-equation representation is used throughout this volume.

Representation

The literature shows little agreement about the way to count the number of layers in a network. Figure 1-6 shows that a multilayer

network consists of alternating sets of neurons and weights. As previously discussed in connection with Figure 1-5, the input layer does no summation; these neurons serve only as fan-out points to the first set of weights and do not affect the computational capability of the network. For this reason, the first layer is not included in the layer count and a network such as that shown in Figure 1-6 is referred to as a two-layer network, as only two layers are performing the computation. Also, the weights of a layer are assumed to be associated with the neurons that follow them. Therefore, a layer consists of a set of weights and the subsequent neurons that sum the signals they carry.

TRAINING OF ARTIFICIAL NEURAL NETWORKS

Of all of the interesting characteristics of artificial neural networks, none captures the imagination like their ability to learn. Their training shows so many parallels to the intellectual development of human beings that it may seem that we have achieved a fundamental understanding of this process. The euphoria should be tempered with caution; learning in artificial neural networks is limited, and many difficult problems remain to be solved before it can be determined if we are even on the right track. Nevertheless, impressive demonstrations have been performed, such as Sejnowski's NetTalk (see Chapter 3), and many other practical applications are emerging.

Objective of Training

A network is trained so that application of a set of inputs produces the desired (or at least consistent) set of outputs. Each such input (or output) set is referred to as a vector. Training is accomplished by sequentially applying input vectors, while adjusting network weights according to a predetermined procedure. During training, the network weights gradually converge to values such that each input vector produces the desired output vector.

Supervised Training

Training algorithms are categorized as supervised and unsupervised. Supervised training requires the pairing of each input vector with a target vector representing the desired output; together these are called a *training pair*. Usually a network is trained over a number of such training pairs. An input vector is applied, the output of the network is calculated and compared to the corresponding target vector, and the difference (error) is fed back through the network and weights are changed according to an algorithm that tends to minimize the error. The vectors of the training set are applied sequentially, and errors are calculated and weights adjusted for each vector, until the error for the entire training set is at an acceptably low level.

Unsupervised Training

Despite many application successes, supervised training has been criticized as being biologically implausible; it is difficult to conceive of a training mechanism in the brain that compares desired and actual outputs, feeding processed corrections back through the network. If this were the brain's mechanism, where do the desired output patterns come from? How could the brain of an infant accomplish the self-organization that has been proven to exist in early development? Unsupervised training is a far more plausible model of learning in the biological system. Developed by Kohonen (1984) and many others, it requires no target vector for the outputs, and hence, no comparisons to predetermined ideal responses. The training set consists solely of input vectors. The training algorithm modifies network weights to produce output vectors that are consistent; that is, both application of one of the training vectors or application of a vector that is sufficiently similar to it will produce the same pattern of outputs. The training process, therefore, extracts the statistical properties of the training set and groups similar vectors into classes. Applying a vector from a given class to the input will produce a specific output vector, but there is no way to determine prior to training which specific output pattern will be produced by a given input vector class. Hence,

the outputs of such a network must generally be transformed into a comprehensible form subsequent to the training process. This does not represent a serious problem. It is usually a simple matter to identify the input–output relationships established by the network.

Training Algorithms

Most of today's training algorithms have evolved from the concepts of D. O. Hebb (1961). He proposed a model for unsupervised learning in which the synaptic strength (weight) was increased if both the source and destination neuron were activated. In this way, often-used paths in the network are strengthened, and the phenomena of habit and learning through repetition are explained.

An artificial neural network using Hebbian learning will increase its network weights according to the product of the excitation levels of the source and destination neurons. In symbols:

$$w_{ij}(n+1) = w_{ij}(n) + \alpha \, \mathrm{OUT}_i \mathrm{OUT}_j$$

where

$w_{ij}(n)$ = the value of a weight from neuron i to neuron j prior to adjustment
$w_{ij}(n+1)$ = the value of a weight from neuron i to neuron j after adjustment
α = the learning-rate coefficient
OUT_i = the output of neuron i and input to neuron j
OUT_j = the output of neuron j

Networks have been constructed that use Hebbian learning; however, more effective training algorithms have been developed over the past 20 years. In particular the work of Rosenblatt (1962), Widrow (1959), Widrow and Hoff (1960), and many others developed supervised learning algorithms, producing networks that learned a broader range of input patterns, and at higher learning rates, than could be accomplished using simple Hebbian learning.

There are a tremendous variety of training algorithms in use today; a book larger than this one would be required to give this topic a complete treatment. To deal with this diverse subject in an organized if not exhaustive fashion, each of the chapters that follow presents the detailed training algorithms for the paradigm under consideration. In addition, Appendix C gives a general overview that is somewhat wider though not as deep. It presents the historical context of training methods, their general taxonomy, and certain of their advantages and limitations. This will, of necessity, repeat some of the material from the text, but the broadened perspective should justify the repetition.

PROLOGUE

In the chapters that follow some of the most important network configurations and their training algorithms are presented and analyzed. These paradigms represent a cross section of the art, both past and present. If carefully studied, many other paradigms will be seen to be easily understood modifications. New developments are generally evolutionary rather than revolutionary, so understanding the paradigms in this book will enhance one's ability to follow the progress of this rapidly moving field.

The emphasis of the presentation is intuitive and algorithmic rather than mathematical. It is more inclined toward the user of artificial neural networks rather than toward the theorist; therefore, enough information is given to allow the reader to understand the fundamental ideas. Also, one who knows computer programming should be able to implement each of the networks. Detailed derivations and complicated mathematics have been omitted unless they bear directly upon network implementation. For the more analytical reader, references are provided to books and papers that are more rigorous and complete.

References

Grossberg, S. 1973. Contour enhancement, short-term memory, and consistencies in reverberating neural networks. *Studies in Applied Mathematics* 52:217, 257.

Hebb, D. O. 1961. *Organization of behavior*. New York: Science Editions.

Kohonen, T. 1984. *Self-organization and associative memory*. Series in Information Sciences, vol. 8. Berlin: Springer Verlag.

Rosenblatt, F. 1962. *Principles of neurodynamics*. New York: Spartan Books.

Widrow, B. 1959. Adaptive sampled-data systems, a statistical theory of adaptation. *1959 IRE WESCON Convention Record*, part 4, pp. 88–91. New York: Institute of Radio Engineers.

Widrow, B., and Hoff, M. 1960. Adaptive switching circuits. *1960 IRE WESCON Convention Record*, pp. 96–104. New York: Institute of Radio Engineers.

2

Perceptrons

PERCEPTRONS AND THE EARLY DAYS OF ARTIFICIAL NEURAL NETWORKS

The science of artificial neural networks made its first significant appearance in the 1940s. Researchers desiring to duplicate the functions of the human brain developed simple hardware (and later software) models of the biological neuron and its interconnection system. As the neurophysiologists gradually gained an improved understanding of the human neural system, these early attempts were seen to be gross approximations. Still, impressive results were achieved that encouraged further research and resulted in networks of greater sophistication.

McCulloch and Pitts (1943) published the first systematic study of artificial neural networks. In later work (Pitts and McCulloch 1947), they explored network paradigms for pattern recognition despite translation and rotation. Much of their work involved the simple neuron model shown in Figure 2-1. The Σ unit multiplies each input x by a weight w, and sums the weighted inputs. If this sum is greater than a predetermined threshold, the output is one; otherwise it is zero. These systems (and their many variations) collectively have been called *perceptrons*. In general, they consist of a single layer of artificial neurons connected by weights to a set of inputs (see Figure 2-2), although more complicated networks bear the same name.

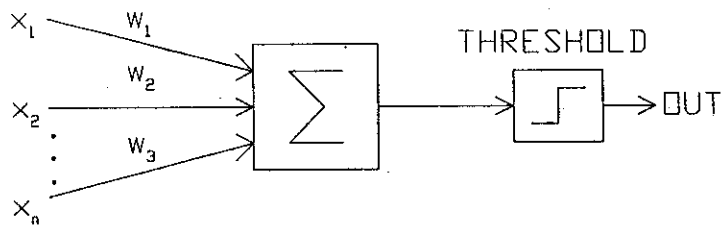


Figure 2-1. Perceptron Neuron

In the 1960s, perceptrons created a great deal of interest and optimism. Rosenblatt (1962) proved a remarkable theorem about perceptron learning (explained below). Widrow (Widrow 1961, 1963; Widrow and Angell 1962; Widrow and Hoff 1960) made a number of convincing demonstrations of perceptron-like systems, and researchers throughout the world were eagerly exploring the potential of these systems. The initial euphoria was replaced by disillusionment as perceptrons were found to fail at certain simple learning tasks. Minsky (Minsky and Papert 1969) analyzed this problem with great rigor and proved that there are severe restrictions on what a single-layer perceptron can represent, and hence, on what it can learn. Because there were no techniques known at that time for training multilayer networks, researchers turned to

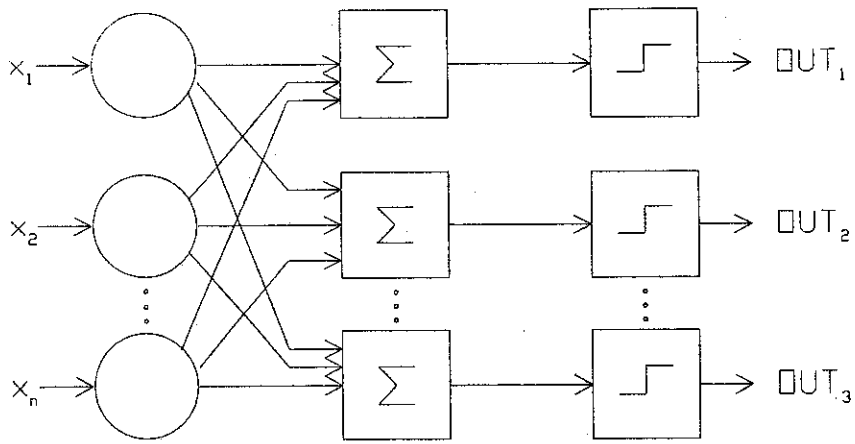


Figure 2-2. Multioutput Perceptron

more promising areas, and artificial neural network research went into near eclipse. The recent discovery of training methods for multilayer networks has, more than any other factor, been responsible for the resurgence of interest and research effort.

Minsky's work may have dampened the ardor of the perceptron enthusiasts, but it provided a period for needed consolidation and development of the underlying theory. It is important to note that Minsky's analysis has not been refuted; it remains an important work and must be studied if the errors of the 1960s are not to be repeated.

Despite the limitations of perceptrons, they have been extensively studied (if not widely used). Their theory is the foundation for many other forms of artificial neural networks and they demonstrate important principles. For these reasons, they are a logical starting point for a study of artificial neural networks.

PERCEPTRON REPRESENTATION

The proof of the perceptron learning theorem (Rosenblatt 1962) demonstrated that a perceptron could learn anything it could represent. It is important to distinguish between representation and learning. Representation refers to the ability of a perceptron (or other network) to simulate a specified function. Learning requires the existence of a systematic procedure for adjusting the network weights to produce that function.

To illustrate the representation problem, suppose we have a set of flash cards bearing the numerals 0 through 9. Suppose also that we have a hypothetical machine that is capable of distinguishing the odd-numbered cards from the even-numbered ones, lighting an indicator on its panel when shown an odd-numbered card (see Figure 2-3). Can such a machine be represented by a perceptron? That is, can a perceptron be constructed and its weights adjusted (regardless of how it is done) so that it has the same discriminatory capability? If so, we say that the perceptron can represent the desired machine. We shall see that the single-layer perceptron is seriously limited in its representational ability; there are many simple machines that the perceptron cannot represent no matter how the weights are adjusted.

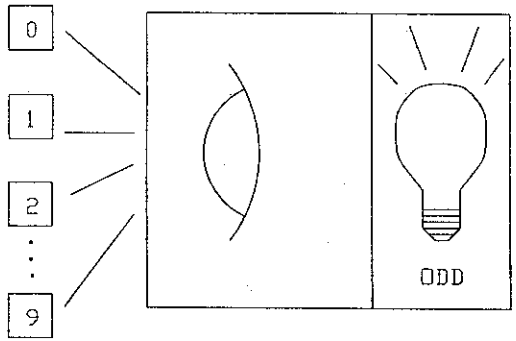


Figure 2-3. Image-Recognition System

Exclusive-Or Problem

One of Minsky's more discouraging results shows that a single-layer perceptron cannot simulate a simple exclusive-or function. This function accepts two inputs that can be only zero or one. It produces an output of one only if either input is one (but not both). The problem can be shown by considering a single-layer, single-neuron system with two inputs as shown in Figure 2-4. Calling one input x and the other y, all of their possible combinations comprise four points on the x–y plane shown in Figure 2-5. For example, the points $x = 0$ and $y = 0$ are labeled as point A_0 in the figure. Table 2-1 shows the desired relationships between inputs and outputs, where those input combinations that should produce a zero output are labeled A_0 and A_1; those producing a one are labeled B_0 and B_1.

In the network of Figure 2-4, function F is a simple threshold producing a zero for OUT when NET is below 0.5 and a one when

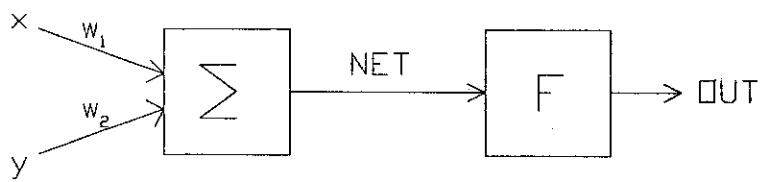


Figure 2-4. Single-Neuron System

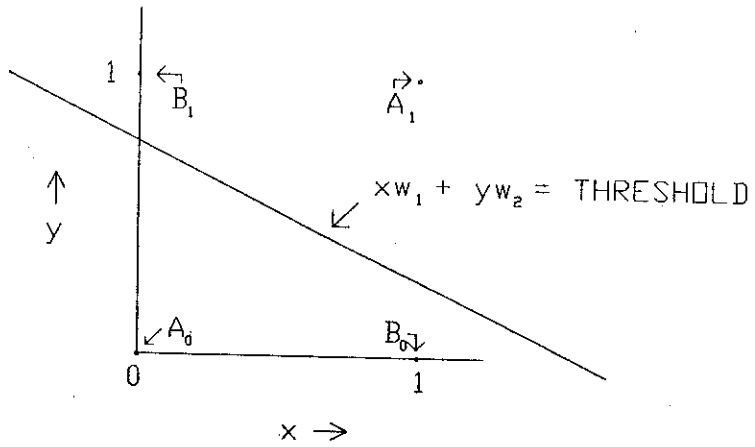


Figure 2-5. Exclusive-Or Problem as Points on the X–Y Plane

it is equal to or above it. The neuron then performs the following calculation:

$$\text{NET} = xw_1 + yw_2 \tag{2-1}$$

No combination of values for two weights w_1 and w_2 will produce the input/output relationship of Table 2-1. To understand this limitation, consider NET to be held constant at the threshold value of 0.5. Equation 2-2 describes the network in this case. This equation is linear in x and y; that is, all values for x and y that satisfy this equation will fall on some straight line on the x–y plane.

$$xw_1 + yw_2 = 0.5 \tag{2-2}$$

Table 2-1. Exclusive-Or Truth Table

Point	x Value	y Value	Desired Output
A_0	0	0	0
B_0	1	0	1
B_1	0	1	1
A_1	1	1	0

Any input values for x and y on this line will produce the threshold value of 0.5 for NET. Input values on one side of the line will produce NET greater than the threshold, hence OUT = 1; values on the other side will produce NET less than the threshold value making OUT = 0. Changing the values of w_1, w_2, and the threshold will change the slope and position of the line. For the network to produce the exclusive-or function of Table 2-1 it is necessary to place the line so that all of the As are on one side and all of the Bs are on the other. Try drawing such a line on Figure 2-5; it cannot be done. This means that no matter what values are assigned to the weights and the threshold, this network is unable to produce the input/output relationship required to represent the exclusive-or function.

Looking at the problem from a slightly different perspective, consider NET to be a surface floating above the x–y plane. Each point on this surface is directly above a corresponding point in the x–y plane by a distance equal to the value of NET at that point. It can be shown that the slope of this NET surface is constant over the entire x–y plane. All points that produce a value of NET equal to the threshold value will project up to a constant level on the NET plane (see Figure 2-6). Clearly, all points on one side of the threshold line will project up to values of NET higher than the threshold and points on the other side must result in lower values of NET. Thus, the threshold line subdivides the x–y plane into two

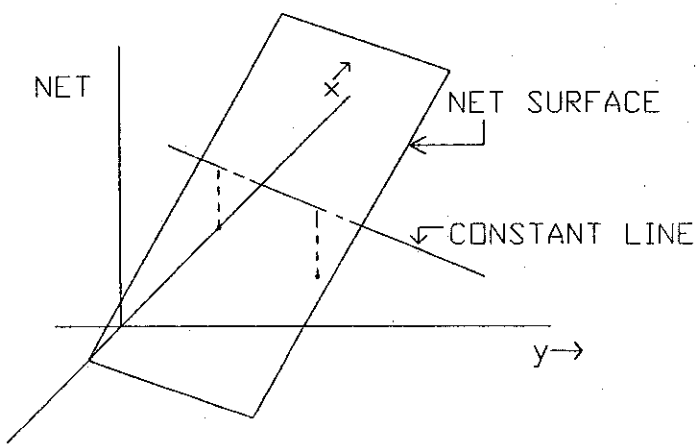


Figure 2-6. Perceptron NET Plane

regions. All points on one side of the threshold line produce a one for OUT; all points on the other side produce a zero.

Linear Separability

We have seen that there is no way to draw a straight line subdividing the x–y plane so that the exclusive-or function is represented. Unfortunately, this is not an isolated example; there exists a large class of functions that cannot be represented by a single-layer network. These functions are said to be linearly inseparable, and they set definite bounds on the capabilities of single-layer networks.

Linear separability limits single-layer networks to classification problems in which the sets of points (corresponding to input values) can be separated geometrically. For our two-input case, the separator is a straight line. For three inputs, the separation is performed by a flat plane cutting through the resulting three-dimensional space. For four or more inputs, visualization breaks down and we must mentally generalize to a space of n dimensions divided by a "hyperplane," a geometrical object that subdivides a space of four or more dimensions.

Because linear separability limits the representational ability of a perceptron, it is important to know if a given function is linearly separable. Unfortunately, there is no simple way to make this determination if the number of variables is large.

A neuron with n binary inputs can have 2^n different input patterns, consisting of ones and zeros. Because each input pattern can produce two different binary outputs, one and zero, there are 2^{2^n} different functions of n variables.

As shown in Table 2-2, the probability of any randomly selected function being linearly separable becomes vanishingly small with even a modest number of variables. For this reason single-layer perceptrons are, in practice, limited to simple problems.

Overcoming the Linear Separability Limitation

By the late 1960s the linear separability problem was well understood. It was also known that this serious representational limita-

Table 2-2. Linearly Separable Functions

n	2^{2^n}	Number of Linearly Separable Functions
1	4	4
2	16	14
3	256	104
4	65,536	1,882
5	4.3×10^9	94,572
6	1.8×10^{19}	5,028,134

Source: R. O. Windner, *Single-stage logic*, Paper presented at the AIEE Fall General Meeting (1960).

tion of single-layer networks could be overcome by adding more layers. For example, two-layer networks may be formed by cascading two single-layer networks. These can perform more general classifications, separating those points that are contained in convex open or closed regions. A convex region is one in which any two points in the region can be joined by a straight line that does not leave the region. A closed region is one in which all points are contained within a boundary (e.g., a circle). An open region has some points that are outside any defined boundary (e.g., the region between two parallel lines). For examples of convex open and closed regions, see Figure 2-7.

To understand the convexity limitation, consider a simple two-layer network with two inputs going to two neurons in the first layer, both feeding a single neuron in layer 2 (see Figure 2-8). Assume that the threshold of the output neuron is set at 0.75 and its weights are both set to 0.5. In this case, an output of one is required from both layer 1 neurons to exceed the threshold and to produce a one on the output. Thus, the output neuron performs a logical "and" function. In Figure 2-8 it is assumed that each neuron in layer 1 subdivides the x–y plane, one producing an output of one for inputs below the upper line and the other producing an output of one for inputs above the lower line. Figure 2-8 shows the result of the double subdivision, where the OUT of the layer 2 neuron is one only over a V-shaped region. Similarly, three neurons can be used in the input layer, further subdividing the plane, creating a triangle-shaped region. By including enough neurons in the input layer, a convex polygon of any desired shape can be formed. Because they are formed by the "and" of regions defined by straight lines, all such polygons are convex, hence only convex regions can be enclosed. Points not comprising a convex region

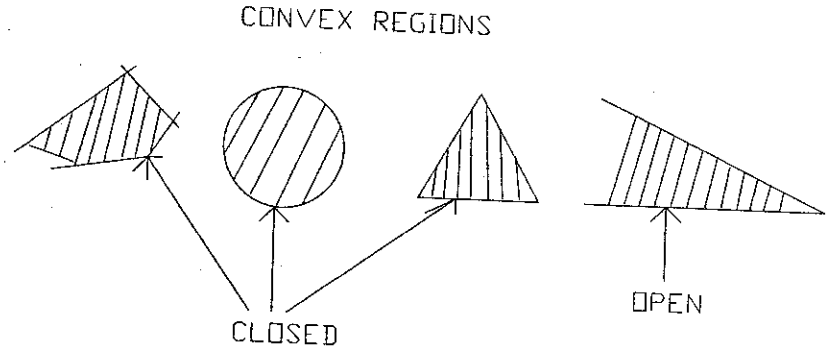


Figure 2-7. Convex Regions, Closed and Open

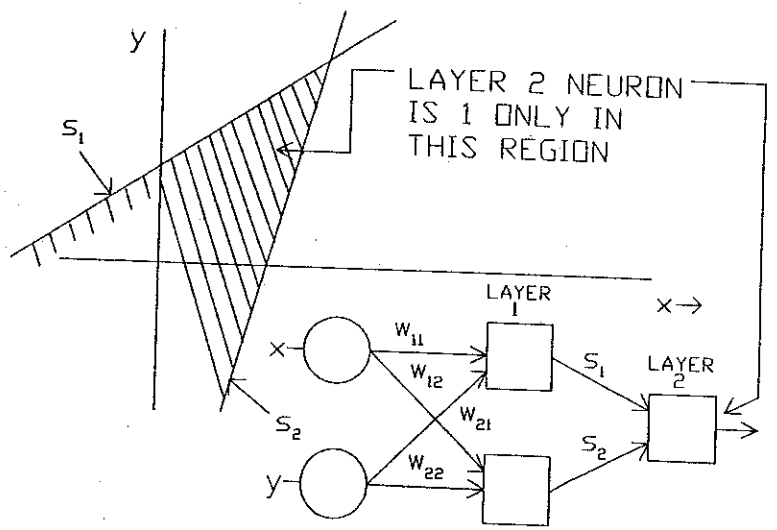


Figure 2-8. Convex Decision Region Produced by Two-Layer Per

cannot be separated from all other points in the plane by a two-layer network.

The layer 2 neuron is not limited to the "and" function; it can produce many other functions if the weights and threshold are suitably chosen. For example, it could be so arranged that either of the two neurons in layer 1 having a one for its OUT level causes the OUT level of the layer 2 neuron to be one, thereby forming a logical "or." There are 16 binary functions of two variables. If the weights and threshold are appropriately selected, a two-input neuron can simulate 14 of them (all but the exclusive-or and exclusive-nor).

Inputs need not be binary. A vector of continuous inputs can represent a point anywhere on the x–y plane. In this case, we are concerned with a network's ability to subdivide the plane into continuous regions rather than separating sets of discrete points. For all functions, however, linear separability shows that the output of a layer 2 neuron is one only over a portion of the x–y plane enclosed by a convex polygon. Thus, to separate regions P and Q, all points in P must be within a convex polygon that contains no points of Q (or vice versa).

A three-layer network is still more general; its classification capability is limited only by the number of artificial neurons and weights. There are no convexity constraints; the layer 3 neuron now receives as input a group of convex polygons, and the logical combination of that need not be convex. Figure 2-9 illustrates a case in which two triangles, A and B, are combined by the function "A and not B," thereby defining a nonconvex region. As neurons and weights are added, the number of sides of the polygons can increase without limit. This makes it possible to enclose a region of any shape to any desired degree of accuracy. In addition, not all of the layer 2 output regions need to intersect. It is possible, therefore, to enclose multiple regions, convex and nonconvex, producing an output of one whenever the input vector is in any of them.

Despite early recognition of the power of multilayer networks, for many years there was no theoretically sound training algorithm for adjusting their weights. In the chapters that follow we explore multilayer training algorithms in detail, but for now it is enough to understand the problem and to realize that research has produced solutions.

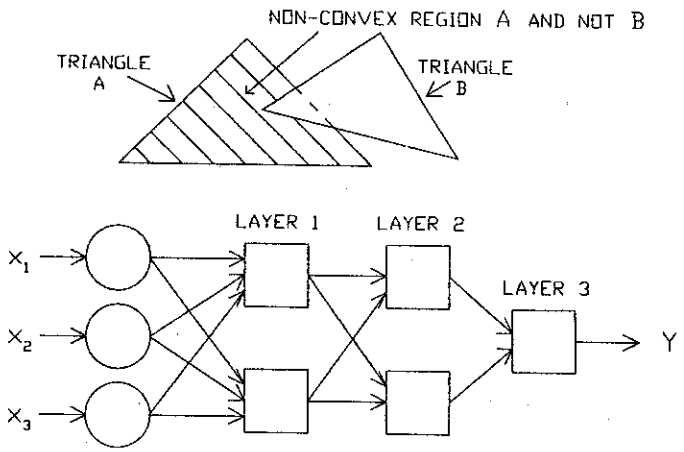


Figure 2-9. Concave Decision Region Formed by Intersection of Two Convex Regions

Storage Efficiency

There are serious questions about the storage efficiency of the perceptron (and other artificial neural networks) relative to conventional computer memory and retrieval methods. For example, it would be possible to store all of the input patterns in a computer's memory along with classification bits. The computer would then search for the desired pattern and respond with its classification. Various well-known strategies could be employed to accelerate the search. If an exact match were not found, nearest-neighbor criteria could be used to return the closest fit.

The number of bits required to store the same information in the perceptron weights can be substantially smaller than the conventional computer memory method, if the nature of the patterns allows a compact representation. However, Minsky (Minsky and Papert 1969) has shown pathological examples in which the number of bits required to represent the weights grows faster than exponentially with the size of the problem. In these cases, memory requirements quickly expand to impractical levels as the problem size increases. If, as he conjectures, this situation is not the exception, perceptrons could often be limited to small problems. How

common are such infeasible pattern sets? This remains an open question that applies to all neural networks. Finding an answer is one of the critical areas for neural network research.

PERCEPTRON LEARNING

The artificial neural network's learning ability is its most intriguing property. Like the biological systems they model, these networks modify themselves as a result of experience to produce a more desirable behavior pattern.

Using the linear-separability criterion it is possible to decide whether or not a single-layer network can represent a desired function. Even if the answer is "Yes," it does us little good if we have no way to find the needed values for weights and thresholds. If the network is to be of practical value, we need a systematic method (an algorithm) for computing the values. Rosenblatt (1962) provided this in the perceptron training algorithm, along with his proof that a perceptron can be trained to any function it can represent.

Learning can be either supervised or unsupervised. Supervised learning requires an external "teacher" that evaluates the behavior of the system and directs the subsequent modifications. Unsupervised learning, which is covered in the chapters that follow, requires no teacher; the network self-organizes to produce the desired changes. Perceptron learning is of the supervised type.

The perceptron training algorithm can be implemented on a digital computer or other electronic hardware and the network becomes, in a sense, self-adjusting. For this reason the act of adjusting the weights is commonly called "training," and the network is said to "learn." Rosenblatt's proof was a major milestone and provided a great impetus to research in the field. Today, in one form or another, elements of the perceptron training algorithm are found in many modern network paradigms.

PERCEPTRON TRAINING ALGORITHM

A perceptron is trained by presenting a set of patterns to its input, one at a time, and adjusting the weights until the desired output occurs for each of them. Suppose that the input patterns are on

flash cards. Each flash card can be marked into squares, and each square can provide an input to the perceptron. If a square has a line through it, its output is one; if not, its output is zero. The set of squares on a card represents the set of ones and zeros presented as inputs to the perceptron. The object is to train the perceptron so that applying a set of inputs representing an odd number always turns the light on, while it remains off for even numbers.

Figure 2-10 shows such a perceptron configuration. Suppose that the vector $\mathbf{X}$ represents the flash card pattern to be recognized. Each component (square) of $\mathbf{X}$, $(x_1, x_2, \ldots, x_n)$, is multiplied by its corresponding component of weight vector $\mathbf{W}$, $(w_1, w_2, \ldots, w_n)$. These products are summed. If the sum exceeds a threshold θ, the output of the neuron Y is one (and the light is on), otherwise it is zero. As we have seen in Chapter 1, this operation may be represented compactly in vector form as $Y = \mathbf{XW}$, followed by the thresholding operation.

To train the network, a pattern $\mathbf{X}$ is applied to the input and the output Y is calculated. If Y is correct, nothing is changed. Howev-

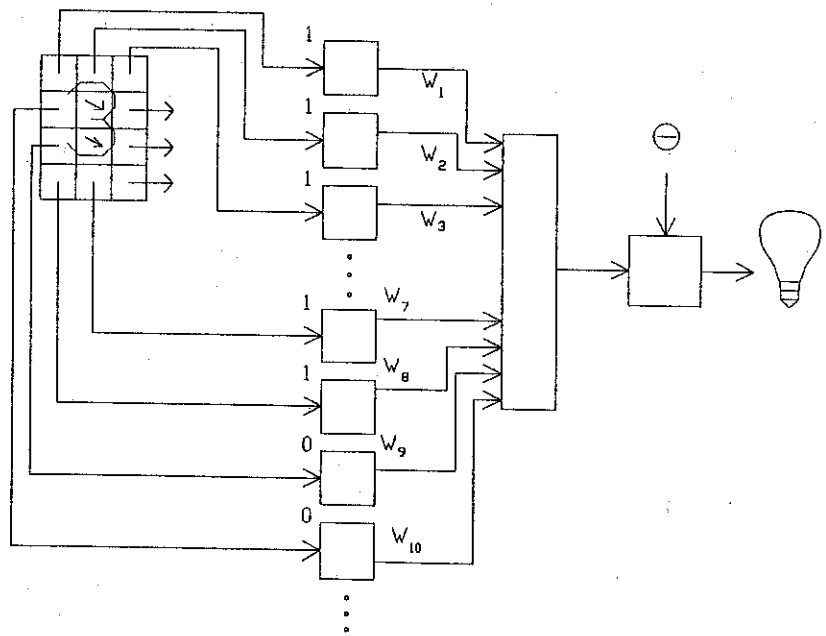


Figure 2-10. Perceptron Image-Recognition Syster

er, if the output is incorrect, the weights connecting to inputs enhancing this erroneous result are modified in value to reduce the error.

To see how this is accomplished, assume that a flash card bearing the number three is input to the system and the output Y is one (indicating odd). Since this is the correct response, no weights are changed. If, however, a flash card with the number four is input to the perceptron and the output Y is one (odd), the weights that connect to inputs that are one must be decreased, as these are tending to produce an incorrect result. Similarly, if a card with the number three produces an output of zero, those weights connecting to inputs that are one must be increased, thereby tending to correct this erroneous condition.

This training method can be summarized:

1. Apply an input pattern and calculate the output Y.
2.
   a. If the output is correct, go to step 1;
   b. If the output is incorrect, and is zero, add each input to its corresponding weight; or
   c. If the output is incorrect and is one, subtract each input from its corresponding weight.
3. Go to step 1.

In a finite number of steps the network will learn to separate the cards into even and odd categories, provided that the sets of figures are linearly separable. That is, for all odd cards the output will be higher than the threshold, and for all even cards it will be below it. Note that this training is global; that is, the network learns over the entire set of cards. This raises questions about how the set should be presented to minimize the training time. Should the set be applied sequentially, over and over, or should cards be selected at random? There is little theory to guide this determination.

The Delta Rule

An important generalization of the perceptron training algorithm, called the delta rule, extends this technique to continuous inputs and outputs. To see how it was developed, note that step 2 of the

perceptron training algorithm may be restated and generalized by introducing a term δ, which is the difference between the desired or target output T and the actual output A. In symbols,

$$\delta = (T - A) \tag{2-3}$$

The case in which $\delta = 0$ corresponds to step 2a, in which the output is correct and nothing is done. Step 2b corresponds to $\delta > 0$, while step 2c corresponds to $\delta < 0$.

In any of these cases, the perceptron training algorithm is satisfied if δ is multiplied by the value of each input x_i and this product is added to the corresponding weight. To generalize this, a "learning rate" coefficient η multiplies the $\delta\, x_i$ product to allow control of the average size of weight changes.

Symbolically,

$$\Delta_i = \eta\, \delta\, x_i \tag{2-4}$$

$$w_i(n+1) = w_i(n) + \Delta_i \tag{2-5}$$

where

Δ_i = the correction associated with the ith input x_i
$w_i(n+1)$ = the value of weight i after adjustment
$w_i(n)$ = the value of weight i before adjustment

The delta rule modifies weights appropriately for target and actual outputs of either polarity and for both continuous and binary inputs and outputs. These characteristics have opened up a wealth of new applications.

Problems with the Perceptron Training Algorithm

It may be difficult to determine if the caveat regarding linear separability is satisfied for the particular training set at hand. Furthermore, in many real-world situations the inputs are often time-varying and may be separable at one time and not at another. Also, there is no statement in the proof of the perceptron learning algorithm that indicates how many steps will be required to train the

network. It is small consolation to know that training will only take a finite number of steps if the time it takes is measured in geological units. Furthermore, there is no proof that the perceptron training algorithm is faster than simply trying all possible adjustments of the weights; in some cases this brute-force approach may be superior.

These questions have never been satisfactorily answered, and they are certainly related to the nature of the set being learned. They are asked in various forms in the chapters that follow, as they apply to other network paradigms. Generally, the answers are no more satisfactory for more modern networks than they are for the perceptrons. These problems represent important areas of current research.

References

McCulloch, W. W., and Pitts, W. 1943. A logical calculus of the ideas imminent in nervous activity. *Bulletin of Mathematical Biophysics* 5:115–33.

Minsky, M. L., and Papert S. 1969. *Perceptrons*. Cambridge, MA: MIT Press.

Pitts, W., and McCulloch, W. W. 1947. How we know universals. *Bulletin of Mathematical Biophysics* 9:127–47.

Rosenblatt, F. 1962. *Principles of neurodynamics*. New York: Spartan Books.

Widrow, B. 1961. *The speed of adaptation in adaptive control systems*, paper #1933-61. American Rocket Society Guidance Control and Navigation Conference.

_____. 1963. A statistical theory of adaptation. *Adaptive control systems*. New York: Pergamon Press.

Widrow, B., and Angell, J. B. 1962. Reliable, trainable networks for computing and control. *Aerospace Engineering* 21:78–123.

Widrow, B., and Hoff, M. E. 1960. Adaptive switching circuits. *1960 IRE WESCON Convention Record*, part 4, pp. 96–104. New York: Institute of Radio Engineers.

3

Backpropagation

INTRODUCTION TO BACKPROPAGATION

For many years there was no theoretically sound algorithm for training multilayer artificial neural networks. Since single-layer networks proved severely limited in what they could represent (hence, in what they could learn), the entire field went into virtual eclipse.

The invention of the backpropagation algorithm has played a large part in the resurgence of interest in artificial neural networks. Backpropagation is a systematic method for training multilayer artificial neural networks. It has a mathematical foundation that is strong if not highly practical. Despite its limitations, backpropagation has dramatically expanded the range of problems to which artificial neural networks can be applied, and it has generated many successful demonstrations of its power.

Backpropagation has an interesting history. Rumelhart, Hinton, and Williams (1986) presented a clear and concise description of the backpropagation algorithm. No sooner was this work published than Parker (1982) was shown to have anticipated Rumelhart's work. Shortly after this, Werbos (1974) was found to have described the method still earlier. Rumelhart and Parker could have saved a great deal of effort if they had been aware of Werbos's work. Although similar duplication of effort is found in virtually every scientific discipline, in artificial neural networks