ADVANCED METHODS IN NEURAL COMPUTING

Philip D. Wasserman

VNR VAN NOSTRAND REINHOLD
New York

Library of Congress Catalog Card Number 93-12320
ISBN 0-442-00461-3

Printed in the United States of America

Van Nostrand Reinhold
115 Fifth Avenue
New York, NY 10003

International Thomson Publishing
Berkshire House
168–173 High Holborn
London WC1V7AA, England

Thomas Nelson Australia
102 Dodds Street
South Melbourne 3205
Victoria, Australia

Nelson Canada
1120 Birchmount Road
Scarborough, Ontario
M1K 5G4, 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–
Advanced methods in neural computing/Philip D. Wasserman.
p. cm.
Includes bibliographical references and index.
ISBN 0-442-00461-3
1. Neural networks (Computer science) I. Title.
QA78.87.W37 1993
006.3—dc20 93-12320
CIP

Preface

Neural computing techniques are expanding rapidly, both in number and power. Research and application activity have produced important new paradigms, revived interest in some that have seen long use, and resurrected a few that were all but moribund. As a result, tools available to the artificial neural network practitioner have been substantially increased. Finding and evaluating these methods can be a serious problem. With more than 1000 papers published in 1992 alone, it is difficult to sort the important from the trivial, the innovative from the derivative, and the proven from the speculative. This book attempts this separation, providing the reader with a set of useful methods, along with enough theory and applications information to apply them.

Choosing the topics for this volume was difficult; dozens of paradigms with hundreds of variations were considered for inclusion. To select among what seemed at times an excess of riches, the following five major criteria were used:

1. Modernity: The approach must be new, but only in the sense that it has only recently received interest and application. This liberal definition of "new" resulted in the inclusion of recent work as well as some produced in the 1960's that is only now receiving proper appreciation.
2. Utility: Only paradigms proven, or showing exceptional promise to be useful, advantageous, and important were considered. This necessarily excluded many minor improvements and purely theoretical results that may lead to future breakthroughs.

3. Trainability: The method must be capable of some form of learning, either supervised or unsupervised. A set of operations involving a training set must change the function of the network in some desirable way.
4. Generalization: Similar inputs must produce similar responses. This is consistent with human abilities. For example, a young child learns to recognize his/her mother despite variations in lighting, distance, angle, and clothing. While no artificial neural network has duplicated the recognition ability of a human toddler, significant ability to generalize has been demonstrated.
5. Concurrent Operation: It must be possible to make efficient use of a large number of processors; ideally, computational rate would increase linearly with the number of processors employed.

Of these criteria, the last may prove the most significant in the long run. The computational load implied by important, real-time neural network applications exceeds the capacity of today's fastest super computers. Throughput must be increased by orders of magnitude if neural networks are to be applied to important applications such as real-time image processing.

Since the early 1960's, hardware improvements have doubled computation rates every two or three years (at constant cost). We cannot rely on this to continue much longer as fundamental limits are being approached. Throughput of single processors will soon be limited by the speed of light. Thus, in the near future, speed increases must come from efficient use of multiprocessor systems rather than brute force acceleration of single processor hardware.

While multiprocessor hardware is readily available, there are few algorithms that can efficiently employ a large number of concurrent processors. Unfortunately, most algorithms devised over the past several thousand years are inherently serial, perhaps mirroring the sequential nature of human problem solving.

Artificial neural networks promise a solution. Their computation is inherently parallel, modeled after the human brain that performs its functions through the use of about 10^{11} simple, slow processors (neurons) operating concurrently.

It is this parallel computing ability that most clearly distinguishes artificial neural network paradigms from more conventional computational approaches. Indeed, some neural networks are well-known mathematical methods, recast for efficient concurrent computation. Some researchers object to calling these neural networks, citing their lack of biological plausibility; there are no known structures in the brain that perform similar operations. This argument points out a fundamental split between those who are using artificial neural networks to study brain function, versus those who view them as a solution to practical engineering problems. This book will take the latter position (without denigrating the former), due to the inclination of the author and the perceived preponderance of interest.

Inevitably, much important work had to be excluded even though it satisfied all of the criteria for inclusion. For example, some was reluctantly omitted that is promising but not fully developed; perhaps it can be included in future volumes as it matures.

As artificial neural networks emerge from the research labs into real-world applications, a broad range of scientifically trained practitioners need to understand how they can use the new techniques. Many have found this to be difficult, as the literature on artificial neural networks is often rendered opaque by the diversity of terminology, the lack of standard notation, and the use of sophisticated mathematics. While professional mathematicians have learned to deal with this problem, many competent researchers in other fields have found important works to be inaccessible. This situation has slowed the dissemination of knowledge and impeded the application of neural network technology.

This book attempts to bridge the gap. Bringing together a diverse set of important paradigms, its explanations are intended to make them understandable to a broad range of scientific professionals. Notation is standardized; qualitative explanations precede the quantitative, and illustrations are used freely to clarify obscure concepts. Nowhere does the mathematics exceed that provided by an undergraduate degree in the sciences, not out of condescension, but because it is simply unnecessary to support applications-oriented explanations. Throughout, the emphasis is on clarity rather than elegance, explanation rather than proof, and exposition rather than mathematical rigor. Still, there has been no deliberate compromise with accuracy; the reader should have nothing to unlearn when going on to the more specialized and theoretical works listed in the references.

In no way does this emphasis imply that the mathematical rigor and deep theoretical study are unimportant. In fact, the field suffers from the general poverty of rigorous theory, and its progress depends heavily on the success of the theoreticians. For those so inclined, a substantial reference section at the end of each chapter leads to the wealth of research papers that provide the theoretical and mathematical support for the paradigms in the book.

These references were selected for clarity of presentation; there was no attempt to be complete or to determine priority. I apologize in advance to authors who find that their earlier work has been excluded.

The explanations in this book are in algorithmic form. This is consistent with the way that most neural computing is now being done—on general-purpose digital computers. For this reason, difference equations are used instead of differential equations, and explanations tend to be program oriented. It is assumed that the reader wishes to apply artificial neural networks, not just to study them; this book is written to facilitate that objective.

The chapters have been written to be largely self-contained. This makes information on a topic accessible without reading the entire book, or paging back to references in previous chapters. This policy implies repetition; in some cases the same concepts are presented in two or more chapters. It is hoped that the utility of this organization will make the redundancy worthwhile.

Some authors avoid the use of the term "neuron," preferring names such as "unit" for this building block. While this respects the profound simplicity of artificial neurons relative to their biological counterparts, it sacrifices comprehensibility, a major objective of this book. For this reason I shall, without further apology, refer to the computational elements of artificial neural networks as neurons.

While this is not a book for beginners, neither does it assume great expertise in the field of artificial neural networks. If you need a review of fundamentals, I recommend the book *Neutral Computing Theory and Practice*, Van Nostrand Reinhold, 1989; it explains the theory that underlies most of the modern paradigms. Equipped with this background and an eagerness to learn more about neural computing, this book provides the logical next step.

Philip D. Wasserman
Cupertino, CA

Acknowledgments

I would like to express my gratitude to those who have reviewed parts of this book: Dr. Walter J. Freeman, Dr. Donald F. Specht, Dr. Douglas L. Reilly, Dr. Pentti Kanerva, Mr. Mark Jurik, and Mr. Charles Rockwell. Their suggestions have substantially improved the clarity of the book, as did their correction of errors, removal of superfluous material, and expansion of important topics. For any residual errors, I must, of course, take full responsibility.

For the illustrations, proofreading, and general assistance, I would like to thank Mr. Robert Gmelin who took time out from his teaching career to help me with this project. For unparalleled patience, perseverance, and tolerance of my procrastination, I would like to thank my editor, Ms. Dianne Littwin. Without her unflagging support and gentle nudging this book would never have been finished. Finally, I would like to thank my wife, Sarah, who has provided the encouragement and supportive environment that has made this effort possible and worthwhile.

Contents

1

Fundamentals

NEURAL NETWORKS—A UNIFYING PERSPECTIVE

The current interest in artificial neural networks is largely a result of their ability to mimic natural intelligence. Although limited and imperfect, artificial neural networks have, in some applications, performed impressively, in other cases, disappointingly. This mix of failure and success offers the tantalizing suggestion that research will eventually produce artificial systems capable of performing a large percentage of the tasks that now require human intelligence, hence the exponentially increasing growth of neural network research.

As a result of this research, artificial neural networks have been used in a broad range of applications. These include pattern classification, pattern completion, function approximation, optimization, prediction, and automatic control. Furthermore, research has produced a large number of network paradigms, each with its own distinctive name.

Faced with this diversity one might conclude that the field is highly fragmented, consisting of a set of unrelated methods and objectives. Despite appearances, all artificial neural networks perform essentially the same function: They accept a set of inputs (an input vector) and produce a corresponding set of outputs (an output vector), an operation called vector mapping. Likewise, all neural network applications are special cases of vector mapping.

As shown in Figure 1-1, a vector mapper accepts a set of inputs and produces a set of outputs according to some mapping relationship encoded in its

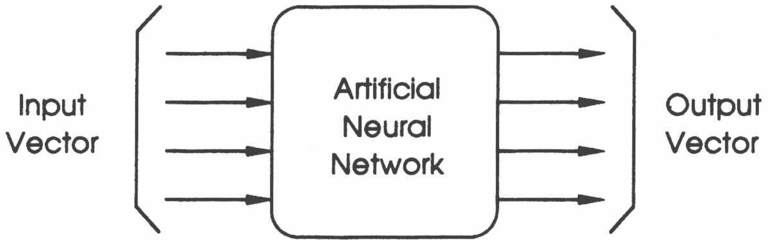

Figure 1-1. A general view of a neural network as a vector mapper.

structure. For example, Figure 1-2 shows a system that maps an input vector with three components—height, weight, and age, into an output vector with two components—life expectancy and insurance premium. Seen from this unifying viewpoint, the various paradigms may be viewed as related approaches, all attempting to solve the same problem.

The nature of the mapping relationship between input and output vectors is defined by the values of free variables (often called weights) within the network. Figure 1-3 shows a configuration where weights (the numbers on

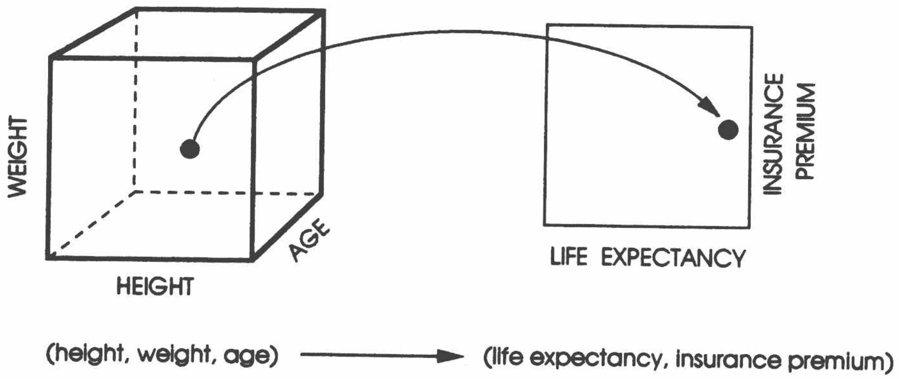

Figure 1-2. Vector mapping accepts one vector (or point) and converts it into another vector (or point).

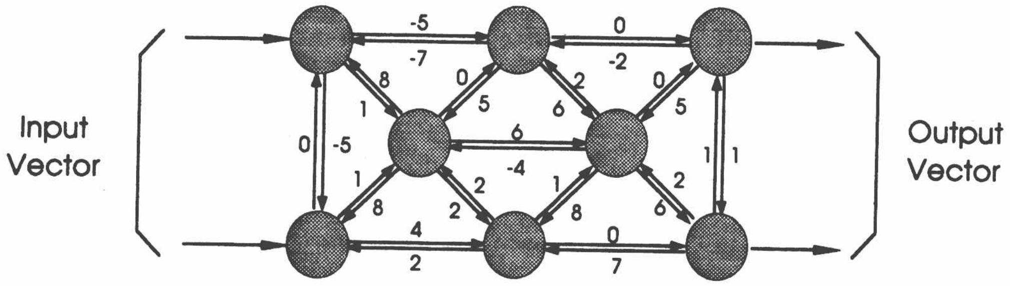

Figure 1-3. Weights in a neural network scale the output value from one processing cell to another.

the arcs) scale the inputs to processing units (circles). Figure 1-4 shows a special case, the feed-forward network, in which the signals flow only from input to output. The mapping relationship between input and output vectors may be *static*, where each application of a given input vector always produces the same output vector, or it may be a *dynamic*, where the output produced depends upon previous, as well as current, inputs and/or outputs. Since feed-forward networks have no memory, they are only capable of implementing static mappings. Adding feedback allows the network to produce dynamic mappings.

Different network paradigms vary greatly in the range of mappings that they can represent. Determining the representational limits for each network type is currently an active area of research. However, recent work on feed-forward networks with one hidden layer (Hornik, Stinchcombe, and White, 1989) has produced a rigorous proof that this functional relationship may be, for all practical purposes, arbitrarily complicated (see Chapter 11). Thus, at least some artificial neural networks are quite general in their vector mapping capability. Without changing internal topology, the same network is capable of producing any functional relationship likely to be encountered by changing its weights.

Vector mapping may be heteroassociative or autoassociative. Heteroassociative mappers are the general case. They produce an output vector that can be different from the input vector. Autoassociative mappers are a subset that yields an output vector that is identical to the input vector on which it was trained. This seems unpromising. However, certain autoassociative paradigms have the ability to produce the desired output vector with a partially incomplete or incorrect input vector. This characteristic has made them useful in pattern completion and noise rejection applications.

Subsequent chapters will present specific artificial neural network paradigms, distinguishing each by its topology, algorithms, benefits, and disadvantages. In the remainder of this chapter the emphasis will be on the underlying principles, thereby providing a unifying framework and a set of terminology that will make it easier to understand the paradigms in this book, as well as other methods encountered elsewhere.

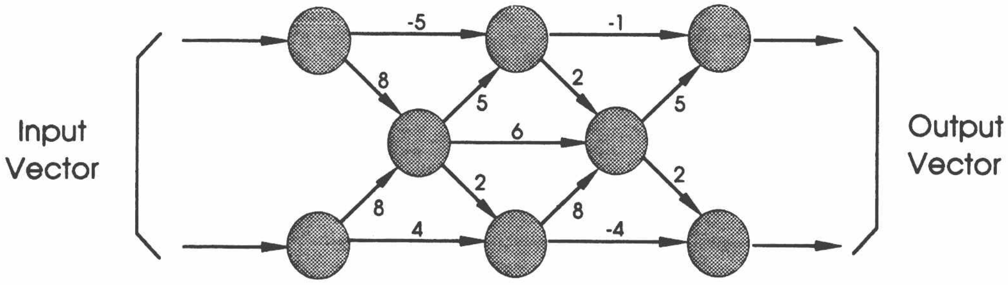

Figure 1-4. In a feed-forward network, weights may be nonzero only in the feed-forward direction.

LEARNING

Artificial neural networks learn from experience. This characteristic, perhaps more than any other, has created the current interest in these methods. In addition to the anthropomorphic implications (that are usually inappropriate), learning offers a powerful alternative to programming.

Learning methods may be broadly grouped as supervised and unsupervised, with a great many paradigms implementing each method.

Supervised Learning

The original Perceptron and, more recently, backpropagation are examples of supervised learning paradigms. In supervised learning, the network is trained on a *training set* consisting of vector pairs. One vector is applied to the input of the network; the other is used as a "target" representing the desired output.

Training is accomplished by adjusting the network weights so as to minimize the difference between the desired and actual network outputs. This process may be an iterative procedure, or weights may be calculated by closed-form equations. Paradigms using the latter form of training may seem to be so far from the biological method that they fail to qualify as an artificial neural network. Nevertheless, such methods are useful, and satisfy a broad definition of artificial neural networks.

In iterative training, application of an input vector causes the network to produce an output vector. This is compared to the target vector, thereby producing an error signal which is then used to modify the network weights. This weight correction may be general, equally applied as a reinforcement to all parts of the network, or it may be specific, with each weight receiving an appropriate adjustment. In either case the weight adjustment is intended to be in a direction that reduces the difference between the output and target vectors. Vectors from the training set are applied to the network repeatedly until the error is at an acceptably low value. If the training process is successful, the network is capable of performing the desired mapping.

Unsupervised Learning

Unsupervised learning, sometimes called self-organization (Kohonen 1988), requires only input vectors to train the network. During the training process the network weights are adjusted so that similar inputs produce similar outputs. This is accomplished by the training algorithm that extracts statistical regularities from the training set, representing them as the values of network weights. Self-organization is reminiscent of the manner in which, in some cases, the human brain modifies its structure under the influence of its experiences without a "teacher."

Applications of unsupervised learning have been limited, however, used in combination with other paradigms they have produced useful results, such as the counterpropagation method (Hecht-Nielsen 1987).

Learning Issues

No network learning paradigm is ideal; all suffer from various limitations and pathologies. For this reason, network training algorithms occupy more research hours than any other aspect of artificial neural networks. Speed, reliability, and generality are important factors in evaluating a training algorithm; improvements are being made in all of these areas.

There are many questions surrounding the learning process. For example:

1. Is the network capable of the desired representation? Does a set of weights exist that will yield the desired mapping?
2. Is the training algorithm capable of adjusting the weights to these values?
3. Will the network train to the best set of weights?
4. Will the network respond correctly to input vectors that are similar, but not identical, to the training vectors? In other words, does the network generalize well enough?
5. Does the training process require a reasonable amount of computation?
6. Is the training set adequate? Does it fully represent the set of input vectors to be encountered in the actual application?

Unfortunately, these questions do not, in general, yet have satisfactory answers. They will be discussed in more detail in Chapter 11, but applying an artificial neural network today requires experience, judgment, and patience. One is often uncertain if a given network, training algorithm, and training set will produce the desired results. Indeed, a certain amount of trial and error seems inevitable with our current state of knowledge. It is fortunate the networks are so robust and the results so striking, otherwise researchers would have long ago directed their efforts toward less ambiguous methods.

GENERALIZATION

The real world suffers from a lack of consistency; two experiences are seldom identical in every detail. Humans accommodate this variability with little effort. For example, we can recognize a friend's voice on the telephone despite a great deal of noise, limited frequency response, and variations in amplitude and pitch.

For a neural network to be useful it must accommodate this variability, producing the correct output vector despite insignificant deviations between the input and test vectors. This ability is called generalization.

Generalization may be quantitatively defined for supervised training if the training set is considered to be randomly selected examples from a specific (but unknown) probability distribution. First, a network is trained on such a set, and the resulting error rate e_1 is measured. Next, a new set of input/output vector pairs is selected from the same distribution, each input vector is applied to the network, the network's response is compared to the desired response (the output vector), and another error rate e_2 is calculated. The $|e_1 - e_2|$ is then a measure of the generalizing ability of the network. More sophisticated methods of measuring generalization are in common use; these will be discussed in Chapter 11.

Generalization as Interpolation

Generalization in neural networks may be viewed as multidimensional interpolation. Poggio and Girosi (1990) use this idea to provide an insightful treatment of artificial neural networks. To see how this idea relates to neural networks, suppose that we have a one-dimensional problem; the input vectors have one component. Suppose also that we are given as a training set the five points plotted in Figure 1-5. We generally assume that these points are samples of some underlying curve. We interpolate between these points so that given intermediate values for x we can determine values for $y = F(x)$ that lie on the underlying (and unknown) curve.

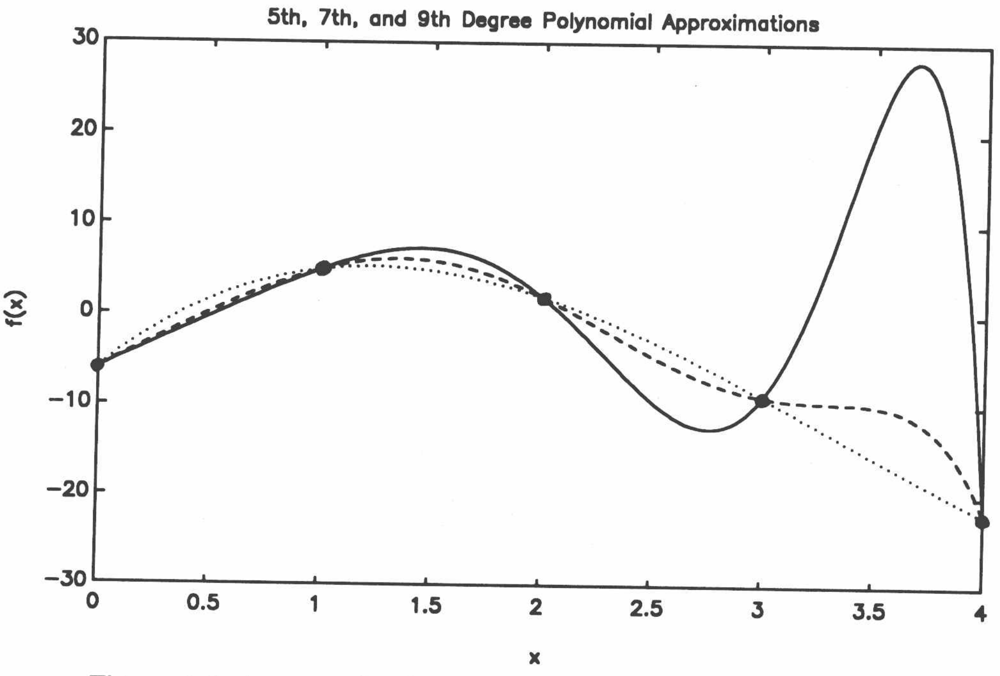

Figure 1-5. An example of smooth continuous interpolation between known points on the X–Y graph.

Three smooth curves have been drawn through the five points, generated by 5th-, 7th-, and 9th-degree polynomial curve fits. Each line constitutes an interpolation, defining a continuum of interpolated points between each of the known points. Since there are three such curves, all of which fit the available information (the training set) perfectly, how can one decide which curve provides "correct" values for intermediate points? In fact, without further information, all three curves are equally valid. We may prefer the smoothest curve based on the lowest degree approximation, however, this is based upon a premise that may be false; we have no certain knowledge of any points other than the five that have been given.

This example demonstrates a fundamental aspect of generalization; that is, with only the data points in a training set, determining other points is an ill-posed problem: There is no unique solution. Therefore, we must supply additional constraints based upon our knowledge of the application. For example, we may choose the 5th-degree fit because we know that the system that generated the points produces smooth curves, or, lacking any other information, we may choose the smoothest curve because we feel that nature favors simple solutions. Such decisions constitute a bias; they restrict the permissible approximations to those involving polynomials with few coefficients. Similarly, we may decide to restrict our neural network topology to the smallest number of weights that produce accurate performance on the training set. While such decisions are often made, we must recognize them as arbitrary and subject to error. In Chapter 11 we develop objective methods for sizing a network to produce good generalization.

Interpolation (generalization) requires an adequate number of points (training set size). Obviously, if there are only a few points (a sparse training set) there is much uncertainty regarding the shape of the curve between points. Much has been written about this problem, and Chapter 11 treats it in detail. It is appropriate here to state merely that there is a close relationship between the number of elements in the input vector (which determines the number of weights in the input layer of the network), the total number of nodes and weights in the network, and the number of vectors required in the training set to generalize with acceptable accuracy.

CLASSIFICATION

Classification, a special case of vector mapping, has an extremely broad range of applications. Here, the network operates to assign each input vector to a category. For example, an input vector might represent the values of the leading economic indicators on a specific date; the two classes might be "Dow Jones Up" and "Dow Jones Down" on the following day.

A classifier may be implemented by modifying a general vector mapping network to produce mutually exclusive binary outputs, namely, an output vector whose components are either 1 or 0. Only a single output (representing category membership) is 1 for a given input vector; all other outputs are 0.

Classification is a central concern in the study of artificial intelligence. An efficient and effective replication of the human's ability to classify patterns would open the door to a host of important applications. These include interpretation of handwriting, connected speech, and visual images. Unfortunately, this goal has been elusive. In most cases humans still perform these tasks far better than any machine devised to date.

The traditional classifiers are either *ad hoc* computer programs or statistical algorithms. Nonstatistical computer programs written for pattern recognition are often "brittle," and easily broken by new data. Such a program will often successfully recognize all examples seen so far, but a new pattern will cause it to fail. This is a consequence of the nature of computer programs; small variations in the input data can produce disproportionately large effects. Thus, it has proven difficult to devise programs that generalize; that produce correct responses to inputs that are similar but not identical to those seen previously. Programmers often find themselves in a frustrating situation: They can correctly classify patterns themselves, but are unable to program a machine to do the task with similar accuracy.

Statistical classifiers have been more successful. Bayesian classifiers and their artificial neural network counterparts are presented in Chapter 3.

A number of artificial neural network paradigms show promise as vector classifiers. Learning from experience rather than being programmed for each problem, they generalize naturally, producing correct answers despite the highly variable, noisy, inconsistent data that is characteristic of real-world problems.

Because of the importance of the classification problem and the power of the neural network approach, a substantial portion of this book will be devoted to the study of this application.

The classification decision is based both upon measurements of the object's characteristics and upon a data base containing information about the characteristics and classifications of similar objects. Therefore, implicit in this process is the collection of data that characterize the statistical properties of the objects being classified.

For example, a lumber mill might wish to automatically separate pieces of pine, spruce, oak, and redwood, putting them into separate bins. This classification could be accomplished by making a set of measurements on the piece, such as color, density, hardness, etc. This set of measurements, expressed numerically, forms a feature vector for that piece, where each measurement is a vector component. The feature vector is compared with a set of recorded feature vectors and their known classifications, those comprising a training set. By some method, a decision is made regarding the correct classification.

Often the classification problem is complicated by the poor quality of the data. Measurements of the sample as well as those in the training set may be noisy and inaccurate. In addition, the training set may be too small to