

A REVIEW OF BACK PROPAGATION NEURAL NETWORKS
AND TRADITIONAL STATISTICAL METHODS

By

GENE BARTON BINNING

Bachelor of Arts in Economics
Bachelor of Arts In Business Administration
Vanderbilt University
Nashville, Tennessee
1975

Masters of Business Administration
University of Oklahoma
Norman, Oklahoma
1977

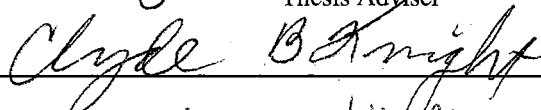
Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
DOCTOR OF EDUCATION
December, 1996

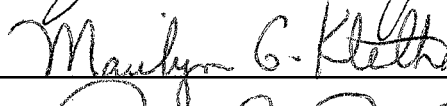
A REVIEW OF BACK PROPAGATION NEURAL NETWORKS
AND TRADITIONAL STATISTICAL METHODS

Thesis Approved:



Thesis Adviser









Dean of the Graduate College

ACKNOWLEDGMENTS

There are many people in my life who have contributed to the completion of this study. First, I am appreciative of my committee, Dr. Clyde Knight, Dr. Ray Sanders, Dr. Megan Kletke, and Dr. David Chance, who have given their time and expertise to help and encourage me to complete this endeavor.

I express sincere gratitude and appreciation to my doctoral committee chairperson Dr. Knight for his confidence and guidance. I thank my dissertation advisor, Dr. Sanders, for his advice, time, and constructive suggestions throughout the completion of the dissertation. I would also like to thank Dr. Chance for sharing his expertise in Neural Network design and for his friendship and tolerance through out the process.

A special thanks goes to Dr. Russell Jones for allowing the use of data he collected for his dissertation, to Dr. Garry Bice for his unanticipated help in narrowing the focus of this project. Thanks to Dr. Hassan Pourbabaee for his help with Chapter III—Non-Parametric Statistics and Backpropagation.

I wish to extend a very special thanks to my entire family and friends for their support throughout my doctoral studies. Thank you to my parents, Gene and Bette Binning; my brother and sister-in-law, Brad and Mary Louise Binning and their children Kelly, Ford, and Mary Ilene; and my sister and brother-in-law, Mark and Barbara Chenier and their children Michael, Megan, and Mason for their support and encouragement.

A special acknowledgment goes to Dr. Janice Williams and Randy Ice for their friendship in the many long drives from Oklahoma City to Stillwater.

Thanks to Bette and Gene Binning for help in proofreading the document, to Wilda Reedy for the final formatting, and to Kay Porter for managing the document in the final stages.

TABLE OF CONTENTS

Chapter	Page
I. INTRODUCTION.....	1
Neural Networks.....	3
Types of Problems Suited to Neural Network Analysis.....	6
Cognitive vs. Non-Cognitive Basis for Neural Network Research.....	8
Categories of Data Analysis.....	10
Academic Status of Neural Network Research.....	10
Definitions of Scholarship.....	11
Method of Research.....	12
Source Materials.....	13
Literature Review and Synthesis.....	13
Problems Suitable for Neural Network Data Analysis.....	13
Nature of the Problem.....	14
Self Learning System Characteristics.....	15
Purpose and Type of the Research.....	16
Research Question.....	17
Definitions and Terms.....	17
Neural Network Related.....	17
Population Distributions.....	25
Organization of the Study.....	27
Limitations of the Study.....	28
II. NEURAL NETWORKS BACKPROPAGATION.....	29
Technology's Impact on Knowledge.....	29
Artificial Intelligence.....	30
Evolution of Neural Networks.....	32
Early Work (pre 1986) Interaction of Individual Neurodes.....	33
Current Work (post 1986) Interaction of a Multidimensional System of Neurodes.....	45
Neurode Operation.....	49
Activation (Transfer) Function.....	48
Other Components of an Activation Function (Squashing, Bias, Gain, etc.).....	52
Types of Neural Network Learning (Adjusting Weights).....	55
Unsupervised Learning.....	55
Correlation Learning.....	57
Error-Correcting Learning (Supervised).....	57
Refinements to Back Propagation.....	67
Probabilistic Neural Networks.....	67
Fuzzy Neural Networks.....	69
Cascading Neural Networks (Systems-Analysis).....	70

Chapter	Page
III. NON-PARAMETRIC STATISTICS AND BACKPROPAGATION.....	74
Neural Networks, Non-Parametric and Probability Concepts.....	74
Inferential and Descriptive Statistics.....	74
Degrees of Freedom.....	81
Sampling Techniques.....	82
Traditional Statistical Classification Methods.....	85
Component Models.....	86
Discriminant Analysis.....	87
Regression.....	89
Bayes Theory.....	90
Gaussian Classifier.....	92
Box-Jenkins.....	92
Decision Tree.....	93
IV. BACK PROPAGATION DESIGN CONSIDERATIONS.....	96
Intelligent Systems Design Methodology.....	96
Objectivity.....	96
Credit Assignment.....	97
Reliability.....	97
Generalization.....	98
Perform Network Design.....	98
Computational Complexity.....	99
Representational Adequacy.....	103
Limitations of Neural Networks.....	106
Variations in Optimum Network Component Design.....	108
Learning by Case or Epoch.....	108
Randomization of Starting Weights.....	109
Optimum Number of Hidden Units.....	110
Optimum Number of Layers.....	117
Overfitting.....	120
Stopped Training.....	122
Models of Neural Networks and their Statistical Equivalents.....	123
V. DATA DESIGN CONSIDERATIONS.....	129
Data Preparation for Neural Networks.....	129
Data Classification.....	129
Data Representation in Neural Networks.....	130
Interpolation Representation.....	133
Minimum Number of Exemplars.....	133
Transformation of Data.....	134
Sequential or Random Presentation.....	137
Adding Noise.....	138
Random Selection vs. Training at the Boundaries.....	139
Neural Network Data Preparation.....	140
Representative Population Size.....	140
Procedures for Setting Up Training.....	141
Encoding <i>a priori</i> Information.....	142

Chapter	Page
Considerations in Training the Network.....	143
Pearson's R^2	143
Benchmarking.....	144
VI. CONCLUSION AND RECOMMENDATIONS.....	145
Problems Suitable for Neural Network (Explanation Stage) Data Analysis.....	146
Intelligent System Design Technology.....	148
Number of Network Nodes.....	149
Sequential or Random Data Selection and Presentation.....	149
Optimal Number of Network Layers.....	150
Overfitting.....	151
Local Minima.....	152
Testing for Generalization.....	152
Using Self-Learning System over Time.....	153
Procedures for Conducting Neural Network Training.....	153
Neural Network Data Preparation.....	154
Use of Binary Data.....	155
Normalization of Data.....	155
Adding Noise to the Training Set.....	156
Conflicting Data.....	156
Recommendations.....	156
Network Modeling.....	157
Data Preparation.....	158
Using Neural Networks in Causal Research.....	159
Summary of Findings.....	160
Recommendations for Future Research.....	160
REFERENCES.....	162

LIST OF FIGURES

Figure	Page
1. Technology and Knowledge Representation.....	30
2. McCulloch-Pitts Neurode.....	34
3. General Form of Single Layer - Single Output Perception.....	36
4. Adaline.....	38
5. Single Layer and the X/OR Problem.....	42
6. Hidden Layer and the X/OR Problem.....	42
7. Simple Neural Network.....	43
8. Neural Network with Hidden Units.....	44
9. Recognition - Representation.....	47
10. Two Layer Neural Network for the X/OR Problem.....	61
11. Probabilistic Neural Network.....	68
12. Cascade Neural Network.....	71
13. Complexity vs. Classifier Fit.....	77
14. Number of Test Cases Needed for Prediction.....	84
15. Idealized Class Separated by Line.....	88
16. Network Layers in Defining Decision Regions.....	115
17. Scatter Diagram, Attendance vs. Reservations.....	120
18. Overfitting, Attendance vs. Reservations.....	121
19. Perceptron (threshold) = Linear Discriminant Function.....	123
20. Simple Linear Perceptron = Multivariate Multiple Linear Regression.....	124
21. Simple Nonlinear Perceptron = Multivariate Multiple Logistic Regression.....	125

Figure	Page
22. Functional Link Network = Polynomial Regression.....	125
23. Multilayer Perceptron = Simple Nonlinear Regression.....	126
24. Maximum Redundancy Analysis.....	127
25. Non-Linear Maximum Redundancy Analysis.....	128
26. Non-Linear Principal Component	128
27. Thermometer Code.....	132

CHAPTER I

INTRODUCTION

Problem solving is a component of the advancement of knowledge. While a person may define a specific problem solving technique for a specific situation, it can be argued that generalized problem solving is an iterative process that involves the basic steps of the gathering and analysis of data, the identification and classification of problems, the development and selection of alternatives to solve the identified problems, and the implementation and monitoring of the chosen alternatives. The step of data analysis has traditionally employed statistical techniques to describe data (descriptive statistics) and to make inferences (inferential statistics). This dissertation discusses the use of backpropagation neural networks as a tool for making inferences about data.

“A neural network is a highly interconnected group of neurons that process information in parallel” (Lawrence, 1991a, p 3). Biological neural networks, such as those found in the brain, may be modeled on computers. Computer based (or artificial) neural networks can be described as an interconnecting set of simple processing units (neurodes that may be either hardware or software based), carrying numeric (as opposed to symbolic) data, in which the relationship between neurodes can be modified by adjusting weights that modify values passed between the connected units. Learning, in an artificial neural network, is a process of adjusting weights such that when a given input data set is presented, the system will produce the appropriate output data. Sarel (1994b) notes that "Artificial neural networks 'learn' in much the same way that many statistical algorithms do estimation . . . If artificial neural networks are intelligent, then many statistical methods must also be considered intelligent" (p 1).

Many different architectures define how computer based neural networks function; there are many different learning algorithms used by computer based neural network systems. A common defining

characteristic of neural networks is that they all learn from examples or associations, rather than from the establishment of rules or mathematical formulas.

This paper is concerned primarily with reviewing the literature related to the methodologies used with artificial neural networks of the backpropagation design used in the analysis of data. When using neural networks as a tool to analyze data, it is useful to think of the neural network as a collection of nodes with weighted connections between the nodes (or neurodes, a term used to differentiate computer based nodes from biological nodes or neurons).

A study describing how neural networks can be used in problem solving and data analysis, specifically how data should be organized to be used with neural networks and how neural networks relate to nonparametric statistics, is appropriate because the use of neural networks represents a new paradigm in the way that knowledge is maintained in computers (Decker and Focardi, 1995; Sarel, 1994b; Caelli, Squire and Wild, 1994; Caudill and Butler, 1992a; Weiss and Kulikowski, 1991). Where traditional computers are programmed with predetermined logical, often sequential steps to perform specific activities, neural networks are trained with examples, to either uncover correlations and similarities, or to filter data. Where standard artificial neural network learning algorithms are designed to be used on massively parallel computers (and may be relatively inefficient when implemented on common serial desktop computers), standard statistics are designed to be used in a relatively sequential methodology. Where artificial neural networks are often designed for situations where data is available in a real-time environment, standard statistical analysis is usually designed to be used with data that is stored and repeatedly accessible (Sarel, 1994b). The data techniques of the past are being overwhelmed because the amount of data available has increased exponentially with the advent of electronic point-of-sale terminals, remote sensing devices, data communication, and lower cost storage (Decker and Focardi, 1995).

The use of neural networks represents, therefore, a new way of analyzing data. Because a paradigm shift means that previous foundations of knowledge are not usable, new foundations must be established. This paper will describe how neural networks can be used in the analysis of data in conjunction with traditional statistics.

Neural Networks

The use of computers to mimic the human cognitive function is a definition of Artificial Intelligence. It can be argued that a step in the evolution of Artificial Intelligence (AI) involved the creation of self-learning systems similar to neural networks -- systems that either chose their rules, or would create new ones based on their analysis of the environment. Neural Networks were originally formulated in an attempt to model the biological operations of the human mind. Where Expert Systems (another form of AI) focus on an automated process that identifies, captures and uses existing heuristic rules developed by expert decision makers in specific domains, Neural Networks are systems that derive rules based on the identification of patterns.

In 1982 Hopfield showed that a network of interconnected switches, used as a simplified model of the brain's neuron organization, could be analyzed mathematically as a physical system. Hopfield's neural network processed information differently from the sequential addition and subtraction of digital computers in that each individual input neurode received signals from other components, added them up, and then chose on the basis of that answer whether to send out a signal of its own.

These different processing mechanisms have lead to differences in the way neural networks and digital computers are given procedures describing how to solve problems. Digital computers are "programmed" using logical steps or specific sequences of actions, with programs and data usually being separate and distinct entities. Neural Networks are "taught" by examining a series of examples, discovering relationships between inputs and outputs, with programs and data indistinguishable from one another.

Lawrence (1991a) provides a cognitive example describing how neural networks learn. In this simplistic example, there are input nodes for "large," "round," "orange," "yellow," and "vegetable." The outputs are "pumpkin" and "zucchini." When the network is first initialized, it is possible that the system will initially decide that a large, not-round, yellow vegetable is a pumpkin. Lawrence continues with the example:

Let's suppose our untrained network initially decides that a large, round, orange vegetable is a zucchini. The training example has the correct data and states that the vegetable is actually a pumpkin. The neural network simulation software looks at the correct output and realizes that its guess was wrong. The software makes changes to its internal connections so that the next time it sees the same inputs, it will more likely produce the correct answer. The connections are adjusted so that the inputs are associated more strongly with the pumpkin output and less strongly with the zucchini output. This training is repeated for a set of examples until the network learns the correct answers. Once the network is trained using pre-selected inputs and outputs, you can run it on new input information (without any supplied outputs) and have it recognize, generalize, or predict answers for you (p. 6).

Kimmel (1991) suggests that neural networks typically attempt to discover relationships among inputs, a form of inductive reasoning. Neural networks use mathematical techniques to uncover relationships between variables, with its analysis concluding when the resulting model can accurately predict the relationship of inputs. According to Kimmel:

The program (or shell), given a set of empirical evidence (data), is going to try to "guess" a relationship among the data elements. It will continue to tinker with its guess until the guess becomes fairly accurate in predicting the evidence. The resulting "network" can be used as a model for predicting the behavior of the phenomenon in the future.

The result is a new way to apply computing power. In the past, programmers translated known mathematical models into programs so that models could be executed at high speed. The development of the electronic neural networks allows the computer be used to build models -- and allow phenomena be examined for which no mathematical relationships previously could be discerned. That is why neural networks are not "programmed" they are "trained" on a given set of data (p 56).

To further expand on Kimmel's example, applying it to the realm of computerized data processing, classic digital computer systems store two types of separate but related knowledge:

- * programs—a listing of instructions, based on sequential logic, that tell the digital computer how to operate.
- * data—a listing of the values of variables used by programs

In the traditional computing paradigm, programs and data work in concert with each other to provide the digital computer with the intelligence needed to carry out tasks in specific knowledge areas. To determine why a digital computer is performing a specific activity; one examines the program listing and reviews the logic described therein. To verify the results presented by a digital computer, assuming the data itself is correct, one evaluates the logic of the program that is using the data.

In contrast, Neural Networks are information processing systems that do not operate in the traditional sequential manner. Neural Networks are parallel in nature, patterned after the way biological neural cells operate, and can be described as consisting of a number of interconnected nodes called neurodes. Numeric values are passed between neurodes in the same way that electric signals are passed between biological neurons. The strength of the signal is modified by weights associated with that connection—weights represent the strength of the relationship between neurodes and are when taken as a whole a form of knowledge representation. A neurode may receive many input signals either from other neurodes or from the outside world. Each neurode produces only one output signal. This output signal corresponds to the axon of a biological neuron and may be connected as weighted inputs to a large number of other neurons. Some of these outputs may terminate outside the network and generate control or response patterns.

Traditional statistics, when performed by a computer, can be validated independently by manual calculations. By viewing the program code, it can be verified that statistical programs are using the same statistical logic that would be used were calculations to be done manually. However, since neural networks combine both program and data in the weights connecting the network's neurodes, a direct examination of the weights will not intuitively provide an understanding of how the computer based neural network arrived at the information being provided. Part of the purpose of this dissertation will be an attempt to bridge the gap of understanding between traditional statistics and neural networks.

From the standpoints of the mechanics of a computer based neural network, at a minimum system will have an input and an output layer, and may contain one or more intermediate (or hidden) layers. The input layer consists of several nodes, each receiving a specific category of data (ex., temperature or distance) and can be characterized as independent variables. The output layer consists of one or more nodes, that represent the result of the network's interaction and can be characterized as dependent variables. The weights that connect the nodes of the input layer, through any hidden layers, to the output layer, represent relationships between the independent and dependent variables. The objective of the neural network is to iteratively adjust the weights between the nodes such that when a set of data is presented at the input layer (a case of the independent variables), a corresponding response is generated at the output layer (the correct value of the associated dependent variable.) In effect, a neural network attempts to map inputs to outputs, and assumes that learning, parameter estimation and information processing are inherently parallel, and that no additional knowledge is needed about the system under analysis (Caelli et al., 1994).

Types of Problems Suited to Neural

Network Analysis

Sarel (1994b) notes that artificial neural networks are used in three main ways:

1. as models of biological nervous systems and models of cognitive intelligence
2. as real-time adaptive signal processors or controllers for use in applications such as robots
3. as data analytic methods

This paper focuses on neural network use as a data analysis method, but other areas will be discussed to provide background information. Neural networks are popular techniques used in forecasting (Decker and Focardi, 1995). Historical data are used in training the neural network to identify patterns. Once the network has learned the patterns, the most recent historical data are used to predict the future. Klimasauskas, (1989) used NeuralWare software to construct a multilayer network and to predict the level of Standard and Poor (S&P) 500. Using three years' of data, he trained the network with the previous ten weeks as input. Klimasauskas reported that neural networks outperformed 4-week and

10-week moving average methods. On the whole, the method is able to predict the next week's closing S&P price, with an average accuracy of 5%, and correctly predict the trend 61% of the time.

Neural Networks are also good at classification or cluster analysis (Decker and Focardi, 1995). Classification would use a supervised learning with historical data for such tasks as targeting direct mail, fraud detection or bankruptcy prediction where patterns from previous known cases would be extracted from data by the neural network as used to predict future cases. Cluster analysis would use unsupervised learning to group data so that all members of a group are similar according to some characteristic(s), and is useful in such activities as setting insurance tariffs or customer segmentation.

From a more general perspective, neural networks are good at solving several types of problems. Pattern recognition, filtering, generalization, and prediction are a few examples to discuss. A pattern may consist of visual images, numeric information, or symbolic data. A neural network is good at recognizing patterns, even when the information contains noisy, ambiguous, or distorted data. In a related activity to pattern recognition, neural networks are also good at identifying patterns and removing noise. The process of eliminating noisy, ambiguous or distorted data from patterns is called filtering.

From a problem solving standpoint, generalization is "the ability to draw conclusions about something that hasn't been seen before, when the information given is different from, but similar to, information seen before" (Lawrence, 1991a, p 15). Many problems have the ability of being solved based on rules or formulas that can be easily identified and placed in expert systems. Because neural networks learn by example and not by predefined rules, they can be used to solve complicated, nonlinear problems, in which heuristic rules are not available.

Because of neural networks pattern recognition and generalization abilities, they can be used in the prediction of trends. This prediction of trends is a special case of pattern recognition in which time is a part of the input pattern. Based on a training set of data, the neural network is often able to generalize, recognizing patterns that we might not have explicit rules or formulas to describe. From the network's standpoint, it does not necessarily know that the pattern it is analyzing is a future event, it just knows that, for example, given specific environmental conditions, the S&P 500 is likely to be at a specific level.

From an unsupervised learning standpoint, neural networks can be trained to, for example, recognize three-dimensional objects using sonar and echolocation (Dror, Zagaeski, and Moss, 1995)

Lawrence (1991a) suggested that because of their nature, neural networks are not good at mathematical precision, serial logic, deduction or logical thinking. In contrast, rule based expert systems are very good at deduction and serial logic, but bad at generalization and error tolerance. Neural networks would not be good at learning operations that involve discrete steps. For example, while a neural network could recognize whether a list of ingredients is likely to make a good cake, it would be unlikely to be able to describe the steps used in making the cake.

Cognitive vs. Non-Cognitive Basis for

Neural Network Research

Current interest in Neural Networks originates primarily from the area of Cognitive Psychology through the work of David Rumelhart, Geoffrey Hinton and Ronald Williams at the University of California at San Diego (Kimmel, 1991). Often they and other colleagues are collectively called the Parallel Distributed Processing group (or the PDP group), after the title of a series of books of the same name that they edited beginning in 1986. Early pioneers laying foundations for neural networks include Hebb (1949) who first modeled learning processes on the computer. The general concept of mathematically modeling neurons was put forward by Rosenblatt (1958). However, research in the field significantly slowed during the mid 1960 when Marvin Minsky and Seymour Papert (1988, expanded edition), showed that the Neural models being built at that time could not solve the “exclusive/or” problem. The work of Rumelhart, Hinton and Williams (and others of the PDP group) sparked renewed interest when they showed that the use of a nonlinear transfer function and the use of multiple layers would correct the problems pointed out by Minsky and Papert.

The use of neural networks can be loosely broken into two distinct areas. One area consists of those seeking to use neural networks as a modeling tool to explain human cognitive behavior. The other area consists of those using neural networks as an engineering tool for analysis of data. With those using neural networks as a modeling tool, there are several different academic areas of interest. One group is

focusing on using neural networks in modeling brain functions in an attempt to understanding how learning and thinking are accomplished. For example, Norris (1990) used neural networks to help validate possible psychological explanations for idiot savants (individuals with limited general intellectual abilities, who also display exceptional skills in one narrow area of cognitive performance). In another area Plunkett and Marchman (1991) have used neural networks to help validate conflicting theories of how children learn language. This group uses neural networks as a cognitive research tool to verify philological theories of how the brain functions. This cognitive psychology group is concerned that the neural network input/output rules and learning algorithms used in computers also mimic biological functions of neurons in the brain.

With those using neural networks as an engineering tool, there are several different areas of interest. Researchers in this area seem to view neural networks as a new statistical tool, analogous to multiple regression, Fourier analysis, or discriminate analysis (Lapedes and Farber, 1987). Neural networks are good at pattern matching activities. The pattern matching ability of neural networks can be used in very diverse applications. These applications include cognitive applications such as language acquisition (Plaut and Hinton, 1987; Bottou, Soulie, Blanchet & Lienard, 1990; Kammerer and Kupper, 1990; Plunkett and Marchman, 1991). In addition to the cognitive applications, neural Networks are also used other fields such as signal processing (Lapedes and Farber, 1987), image processing -- especially those that use Fourier methods (Silverman and Noetzel, 1991), image compression (Rosenfield and Kak, 1982; Barsley and Sloan, 1988; Mougeot, Azencott & Angeniol, 1991), discriminate analysis (Rosenblatt, 1958; Silverman and Noetzel, 1990; Webb and Lowe, 1990; Yao, Freeman, Burke & Yang, 1991; Barschdorff, Monostori, Ndenge & Wostenkujler, 1991), multiple regression analysis (Stone, 1986; White, 1990; Chance, MacLin, and Lykins, 1993), financial analysis and forecasting (Stein, 1991); and statistical process control (Pugh, 1989; 1991). From a statistical standpoint, this group's concern is not that the networks model human neurological functions, but that the neural network learning algorithms are efficient, robust, and consistent.

Categories of Data Analysis

Much of this discussion is based on Kerlinger (1986). The discussion concerning problems suitable for Neural Network data analysis is based both on the objectives of the research being conducted, and on a definition of the word theory. From a scientific perspective, a theory is defined as “a set of interrelated constructs (concepts), definitions, and propositions that present a systematic view of phenomena by specifying relations among variables, with the purpose of explaining and predicting the phenomena.” (Kerlinger, 1985, p 9). From a layman’s perspective, a theory might be described as a plausible explanation of how something works, a rule-of-thumb. When looking at theory from a scientific standpoint, the usefulness of a theory is based upon its ability to successfully predict; “very nature of the theory lies in its explanation of observed phenomena” (Kerlinger, 1985, p 9). When looking at theory from a layperson standpoint, the usefulness of a theory is based on its ability to be successfully implemented. Each definition of theory implies different methods for use in practical applications.

Academic Status of Neural Network Research

Roger Ratcliff (1990) suggests that we are currently in a transition between stages of a process that introduces new ideas to academia. With regards at least to the cognitive focus of Neural Networks, Ratcliff suggests that we are reaching maturity in the first stage called exploration. During this exploration stage, the criteria used to evaluate an application is necessarily loose because one is interested in gathering as much data as possible in as many different circumstances as possible, from a large range of models, exploring in which domains a model is best suited. During this first stage, the emphasis is on accounting for major findings within various domains of knowledge. Ratcliff suggests that we need to enter a second stage where we expand our focus by explaining the reasons for the failures, examining the limitations of the various models, and identifying phenomena that are not explained by these models. During this second stage, models compete with each other within their theoretical framework, and are evaluated according to more stringent criteria. The foci in the second stage changes from one of "let's try this model with that variation and see what happens" to one of "what are the strengths and weaknesses of this model."

Definitions of Scholarship

Boyer (1990) has argued that the American professoriate is in the middle of a societal paradigm shift that will require an expanding definition of scholarship. It is suggested that where the original purpose of the professoriate was to expand the frontiers of knowledge (pure research), that society is now expecting the professoriate to perform scholarly activities. According to Boyer there are four components to scholarship, the Scholarship of Discovery, the Scholarship of Integration, the Scholarship of Application, and the Scholarship of Teaching. Boyer defines four components to scholarship:

- * Scholarship of Discovery—the advancement of knowledge, the confrontation and investigation of the unknown, the seeking of understanding for its own sake (traditional basic research).
- * Scholarship of Integration—giving meaning to isolated facts, linking concepts across disciplines in an authenticating process, synthesizing knowledge by placing specialties in larger context, educating nonspecialists
- * Scholarship of Application—applying knowledge to consequential problems, determining how knowledge can be helpful to individuals and institutions
- * Scholarship of Teaching— identifying pedagogical procedures that directly relate to the subject being taught, the education and enticement of future scholars, the transformation, extending, and transmitting of knowledge to others

The scholarship of discovery often uses analytical processes. A part of the process of establishing that a theory is an expression of reality, is being able to express the theory in mathematical terms, and showing that no part of the quantitative description contradicts what we observe in the real world. Van Dalen (1979) notes that sometimes the theory comes first (deduction) and a mathematical model is built to predict how a phenomenon will react given the environment. Then, evidence is gathered and compared to the model to support or refute the theoretical relationships. Other times, when analyzing a phenomenon, certain relationships are observed (induction), which leads to the creation of a mathematical model, which leads to the development of a theory to explain the model.

Aristotle developed a deductive argument (syllogism) to provide a means of testing predictions or conclusions. A valid syllogism contains three parts: a major premise (or statement of reality) and a minor premise are used to provide evidence for the third part, a conclusion. The premises are stated in a way that relate them to the conclusion such that if the premises are true, then it follows that the conclusion is also true. Deductive reasoning assumes an existing body of knowledge (premises), and extends that body of knowledge by deducing new relationships between premises. Additionally, the conclusion of a deductive argument does not extend beyond what is already known and stated in the premises.

The existing body of knowledge is often initially established using inductive reasoning. In inductive reasoning, one makes observations of specific relationships in reality, and general conclusions are drawn from these specific observations. Van Dalen (1979) suggests that these general conclusions that are arrived at by induction may be used as premises for deductive inferences. Additionally, it is through inductive reasoning that the body of knowledge is extended by arriving at conclusions of varying degrees of probability.

This paper is based on aspects of the Scholarship of Integration (it looks at how neural networks are used in a variety of disciplines) and the Scholarship of Application (how can what has been learned be used to solve specific problems.)

Method of Research

The focus of this study is methodological insofar as it seeks answers to the problem of objective evaluation of research data using a neural network as a quasi statistical technique. Procedures for methodological research are similar to that of historical research. Methodological Research as defined by Kerlinger (1986) “is controlled investigation of the theoretical and applied aspects of measurement, mathematics and statistics, and ways of obtaining and analyzing data” (p 622).

Historical Research, as defined by Key (1990) is “employed by researchers who are interested in reporting events and/or conditions that occurred in the past. An attempt is made to establish facts in order to arrive at conclusions concerning past events or to predict future events” (p. 118). Insofar as this study could also be classified as Scholarship of Application (Boyer, 1990), where basic knowledge previously

discovered in one academic area is applied to another academic area, this study is similar to historical research.

Van Dalen (1979) seems to allude to historical research being able to be used as a pattern for the scholarship of application by noting that historical research can be used to “appraise the past facts encountered in everyday life” (p 350). Van Dalen suggests that the following steps be used in a historical study; formulating the problem, collecting the source materials, criticizing source materials, formulating hypotheses to explain events or conditions, and interpreting and reporting the findings.

Source Materials

Source materials for this study are articles published in research journals. In answering the three basic questions of the study, a review of literature was conducted and a synthesis of concepts performed. The review of literature was based on literature review searches from Dissertation Abstracts, ERIC, PsychLit, and Business Periodical's Index. Key word searches on the Internet using Lycos, AltaVista, and Excite were also performed.

Literature Review and Synthesis

Database searches using ERIC, PsychLit, the Wilsonline series, and Dissertation Abstracts were made using the keywords of “Neural Network” and “back propagation.” Several hundred “hits” were examined for insights in applying neural networks to the data analyses problem. The Review of Literature section contains summaries of articles from various disciplines relating to the use of back propagation neural networks. The purpose of the review was to take concepts of data analysis and interpretation that were found useful in one academic area and apply them to solving business problems.

Problems Suitable for Neural Network Data Analysis

It is suggested that the first step in deciding if to use a neural network for data analysis is to examine the problem and determine what traditional statistical methods would be appropriate. Neural networks tend to use pattern matching techniques to discover similarities in data. While neural networks

may be used to analyze many different types of problems, when using networks as a tool for data analysis the problem must be able to be described by an analytical equation. Based on the Review of Literature, back propagation neural networks as a data analysis tool have been found to be at least equivalent or sometimes better than traditional statistics in the following cases

- Fourier methods (Kohonen, 1984; Lapedes and Farber, 1987)
- Nonlinear transfer function models (Lapedes and Farber, 1987)
- Discriminant analysis (Rosenblatt, 1958; Lippmann, 1987; Silverman and Noetzel, 1990; Webb and Lowe, 1990; Yao, Freeman, Burke & Yang, 1991; Barschdorff, Monostori, Ndenge & Wostenkujler, 1991) without the need to make parametric assumptions concerning the underlying population distribution (Weiss and Kulikowski, 1991)
- Regression (Stone 1986; Webb and Lowe, 1990) without the need for parametric assumptions of the population (White, 1990)
- Box-Jenkins Techniques (Tang, de Almeida, and Fishwick, 1990; Varfis and Versino 1990; Sharda and Patil 1990; Reynolds, Mellichamp and Smith, 1995)
- Maximum Redundancy Analysis (Sarle, 1995b)
- Principal Component Analysis (Kohonen, 1984)
- Financial analysis and forecasting (Stein, 1991)
- Statistical process control (Pugh, 1989; 1991)

Once it has been determined that neural networks are appropriate for the problem under study, the next step is to choose the network architecture—how neurodes will be interconnected.

Nature of the Problem

Traditional statistical techniques used as tools for decision making are logic oriented and work well when programmed on digital computers. Allman (1989) suggests that neural networks are both better in some areas as predictors of the human mind, and more useful as an aid in decision making. The problem is that because neural networks are a new technology, there is confusion surrounding how data should be organized as inputs to neural networks, and uncertainty in the interpretation of both measures of

significance and how these measures relate to classic statistics, leading to a lack of confidence in the use of neural networks for non-cognitive research.

Self Learning System Characteristics.

When analyzing artificial neural networks, Sarle (1994b) notes that it is important to distinguish between neural network *models* and the *algorithms* used in learning (adjusting interconnecting weights between neurodes). Sarle suggests that many neural network models are "similar or identical to popular statistical techniques such as generalized linear models, polynomial regression, nonparametric regression and discriminant analysis, projection pursuit regression, principal component and cluster analysis" (p. 1). However, Sarle also notes that many neural network researchers are engineers, neurophysiologists, or computer scientists "who know little about statistics and nonlinear optimization" (p. 1).

Kolen and Goel (1991), drawing on the earlier work of Samuel (1967) and Minsky and Papert (1988) suggest that an information processing system capable of learning must have several interrelated abilities. White (1990) added a fifth dimension. Abilities of learning systems include:

1. Representational Adequacy (internal representation of what was learned)
2. Credit Assignment (able to identify why an incorrect response was given)
3. Generalization (the ability to take what is learned from specific examples and generalize to generic abstractions)
4. Computational Complexity (learn in a reasonable amount of time)
5. Reliability (ability to yield the same basic results with the same examples trained on neural networks under the same conditions)

Van Dalen (1979) suggests that when determining confidence in a statistical research tool, validity, reliability, objectivity, and suitability should be considered. Many aspects of statistical validity are covered by Kolen & Goel's definition of generalizability, and many aspects of suitability are covered by the definition of representational adequacy. However, the topics of reliability and objectivity do not seem to be addressed by any of the suggested learning system evaluation criteria of Kolen & Goel. It is therefore suggested that the criteria of reliability (ability to yield the same results when the same

exemplars are trained on networks under the same conditions) and objectivity (ability to obtain the same results when computer program is run on different machines).

Roy, Govil and Miranda (1995) define the problem in terms of the need for a Robust and Efficient Learning theory, and list the following criteria:

- 1) Perform network design task: learning methods must be able to design an appropriate network for a given problem without the need for external input in the form of a predesigned structure
- 2) Robustness in learning: should not have local minima problems, problems of oscillation, and catastrophic forgetting.
- 3) Quickness in learning
- 4) Efficiency in learning: when provided with a finite number of training examples, should be computationally efficient
- 5) Generalization in learning: must be able to generalize reasonably well so only a small amount of network resources would be needed.

A problem identified as a result of this study was that there is a need to assess procedures for organizing data for use with neural networks in modeling business problems. As Sarel (1994b) noted "Few published works provide much insight into the relationship between statistics and neural networks" (p 1).

Purpose and Type of the Research

The purpose of this study was to assess literature with regards to the use of back propagation based neural networks using supervised learning design for data analysis. Definitions of Scholarship are first presented to provide a framework for the analysis. The methodology described in this chapter was designed to address the following questions concerned with using neural network in solving business problems based on the Review of Literature of the previous chapter.

1. What are the characteristics of a problem that make it suitable for investigation using neural networks? (Problems Suitable for Neural Network Data Analysis)

2. Given the problem, how should the neural network model be designed? (Intelligent System Design Methodology)
3. Given the problem and chosen network design, how should data be prepared for presentation to a neural network for modeling? (Neural Network Data Preparation)

The research falls under Scholarship of Application as defined by Boyer (1990). The type of research is Methodological Research as defined by Kerlinger (1986) using a Historical Research framework. Literature concerning the application of back propagation neural networks from many academic areas was reviewed to see if application techniques valid for one academic discipline could be applied in the business area. Based on the information presented in the Review Literature, conclusions are drawn on how to prepare data for model analysis using back propagation neural networks and apply them to solving business problems.

Research Question

This study attempts to answer the following research question:

Can Back propagation Neural Networks be used in Research to establish causality between variables?

Definitions and Terms

Neural Network Related

Artificial Intelligence (AI): a term coined in 1956 by John McCarthy as the theme of a Dartmouth College Conference to describe computers with the ability to mimic or duplicate functions of the brain (Stair, 1992). The concept of artificial Intelligence has evolved to include the concept of what computers can “do” that mimic human functions (Hutchinson and Sawyer, 1992). What AI “does” includes:

Robotics -- an automatic device that performs functions normally ascribed to humans (unintelligent robots), or that operates with what appears to be almost human intelligence (perceptron robots).

Natural Language Processing -- to enable the computer to communicate with the user in the user's native language, or translating text from one language to another.

Expert System -- solves problems that require expertise to understand. Weiss and Kulikowski (1991) suggest that expert systems are designed to capture an expert's knowledge by codifying "rules of thumb" in situations where the complexity of the problem makes automatic learning difficult. Because many experts do not logically step through their decision making process (they make decisions automatically based on based on the information at hand), this process of knowledge acquisition is facilitated by knowledge engineers who are trained in capturing this knowledge in the form of a set of decision rules or other representational elements. This knowledge is then made available to non-experts for their use in solving similar problems.

Virtual Reality -- allows the user to experience being inside a computer-generated environment that some call cyberspace. Rucker, Sirius, and Mu, (1992) define Virtual Reality as a technology that creates the illusion of being immersed in an artificial world (synthetic experience), or of being present in a remote location in the physical world (remote control). The highest development of Interactive Multi-media technology creates a simulation that becomes a virtual reality.

Back propagation: a method that enables a neural network to learn to discriminate between classes of patterns that are not linearly separable (Rumelhart, Hinton, and Williams, 1986). Based on the Least Mean Square rule (also called the Delta Rule), back propagation operates in a two step sequence to optimize interconnecting weights in a way that minimizes sum of the errors squared. For each example presented to a neural network there is a "forward pass through the network to determine the network's current options, followed by a backward pass to determine, based on the difference between these [calculated] outputs and the correct ones, how [interconnecting] weights should be changed" (Smith, 1993, p 23). It is estimated that 80% of current neural network projects use back propagation because they are "simple to implement, solve most problems correctly, and are 'unfuzzy' about how they are used" (Caudill and Butler, 1992, p 173)

Cluster Analysis: the statistical equivalent to the neural network concepts of competitive learning or adaptive vector quantization (Sarle, 1994b).

Data: facts, images, or sounds that are the raw materials used to be processed by information systems in the creation of information and knowledge (Capron, 1995)

Data Reduction: the statistical equivalent to the neural network concepts of unsupervised learning, encoding, or autoassociation (Sarle, 1994b).

Discriminant Analysis or Regression: a statistical tool that is the equivalent to the neural network concepts of supervised learning or heteroassociation (Sarle, 1994b).

Error Function or Cost Function: the neural network equivalent to the statistical concept of an estimation criterion (Sarle, 1994b).

Error Surface: a graphical representation of the error of a model with respects to actual data. In a linear model ($y = b_0 + b_1x$) an error surface would be represented by a three-dimensional model with the “height of each point on this error surface represents the error on a certain set of examples of one particular linear model—the model whose parameters correspond to that point’s b_0 and b_1 coordinates. The optimal weights are the pair corresponding to the lowest point on the error surface. The best model is the one that produces the least error” (Smith, 1993, p 60). In statistical terms, the error term is called a residual.

Estimation: the statistical equivalent to a neural network concept of training, learning, adaptation or self-organization (Sarle, 1994b).

Estimation Criterion: the statistical equivalent to the neural network concepts of error function or cost function (Sarle, 1994b).

Features: the neural network equivalent to the statistical concept of variables (Sarle, 1994b).

Generalization: concerning neural networks and in terms of useful model responses, “after some amount of training, to novel (unlearned) inputs” (Peterson and Hartman, 1989, p 481). (See also Self Learning Systems)

Gradient Descent: a procedure for modifying a network of weights by evaluating a point’s current position in *error space*. At a point in error space, the slope of the error surface is computed. Weights are

changed in the direction in which the error surface has the steepest decrease. Using new weights, calculate the new point in error space and repeat the process iteratively until the values of the weights approach the lowest point in the error surface. (Smith, 1993).

Input: the neural network equivalent in statistics to an independent variable (Sarle, 1994b).

Information: Data organized in a way that is useful to an individual in their decision making process (Capron, 1995).

Interactions: the statistical concept equivalent to a neural network's higher-order neurons (Sarle, 1994b).

Knowledge: a combination of instincts, ideas, rules, and procedures that guide actions and decisions (Miller, 1985). One purpose of Artificial Intelligence is to increase the efficiency and effectiveness of knowledge development.

Learning: with respect to neural networks, learning is a process “designed to find the coefficients or weights that provide the best fit between the mapping function and data consisting of examples of the target function” (Smith, 1993, p 59). (See also Self Learning Systems). The neural network term used describe the concept of statistical estimation (Sarle, 1994b).

Supervised Learning: A control system in which the difference between a desired state (defined with *a priori* information, usually given to the system as a training set) and a calculated state is iteratively measured as an error, which is used in a feedback loop to modify the algorithm used in calculation. The procedure is repeated until the error reaches a specified level. Learning is therefore the memorization of a given sequence of controls (Borghese & Arbib 1995).

Reinforcement Learning: With no *a priori* information available, learning is a system's self-discovery of a sequence of algorithms and states that will accomplish a specified task. The *credit assignment* problem is the function used to describe which sequence increases or decreases the overall performance of the system. Reinforcement learning approaches include back propagation through time, minimum principle, and dynamic programming (Borghese & Arbib 1995).

Local Minima: a lumpy error surface may have several valley's that are not the lowest in the global error surface. These valley's are called local minima. While an interesting theoretical problem, in

practice “there is almost no risk that a model’s performance would be seriously impaired because the system was trapped in local minimum that could not be avoided by adding more hidden units” (Smith, 1993, p 62).

Mapping: the result of an equation produced by statistical modeling whereby a dependent variable is mapped in the space of the independent variables. The mapping of a linear equation in two dimensions (one independent and one dependent variable) produces a straight line. The mapping of a non-linear equation in two dimensions produces a curved line. The mapping of a linear equation in three dimensions (two independent and one dependent variable) produces a plane. Mapping of any number of variables can occur, but can not be visualized beyond three dimensions. (Smith, 1993)

Monotonic Learning: occurs when an agent may not learn any knowledge that contradicts what it already knows. Monotonic learning greatly simplifies artificial learning systems in environments assumed to be constant. An agent is said to learn non-monotonically when it replaces old knowledge with new knowledge when it believes there is sufficient reason to do so (Wray, Phillips, Rogers, and Walsh, 1994).

Neural Network: “a highly interconnected group of neurons that process information in parallel” (Lawrence, 1991a, p 3).

Neurode: the processing element of neural networks which are the computer equivalent to biological neural cells or neurons in the brain. Each neurode typically receives many signals over its incoming connections; some from other neurodes, others from the outside world. Each neurode produces one output signal which may be transmitted to several other neurodes, or to an end device (Caudill & Butler, 1992). All neurodes fundamentally operate in the same manner; they sum weight-adjusted inputs, possibly add a bias value, and finally pass the results through a transfer (or activation) function to produce an output that is offered for other neurodes to use (Kempka, 1994a).

Noise: One of two primary sources of modeling error. “Noise is the effect of missing (or inaccurate) information about the world” (Smith, 1993, 17). The concept of noise includes inaccuracies in the data and in the measuring instrument. Inaccuracies may also be due to the fact that the independent variables do not contain all the information needed to determine the dependent variable. Noise is not an

inherent randomness, lack of causality, or inaccuracy in the mapping function (which may not have the same form as the target function.)

Observation: the statistical equivalent to the neural network concepts of patterns or training pairs (Sarle, 1994b).

Output: the neural network equivalent to a statistical predicted value (Sarle, 1994b).

Overfitting: because neural networks are universal approximators in most cases, given enough time and enough nodes, a neural network can model any data set to any degree of accuracy required. However, we usually want neural network models to learn something about the past that can be generalized into the future. Overfitting is said to occur when the network “models noise in addition to the underlying function” (Smith, 1993, p 113) thereby making the model too specific for generalized prediction.

Patterns or training pairs: the neural network equivalent to the statistical concept of observations (Sarle, 1994b).

Parameter estimates: are the statistical equivalent to the neural network's (synaptic) weights (Sarle, 1994b).

Residual: a statistical term equivalent to errors in neural networks (Sarle, 1994b).

Regression or Discriminant Analysis: a statistical tool that is the equivalent to a type of neural network, supervised learning or heteroassociation (Sarle, 1994b).

Self Learning Systems describe “how” a class of AI applications perform their functions. One of the attractive features of a Neural Network is that it is a self learning system. Kolen and Goel (1991), drawing on the earlier work of Samuel (1967) and Minsky and Papers (1988) suggest that an information processing system capable of learning must have several interrelated abilities of representational adequacy, credit assignment, generalization, and computational complexity. Van Dalen (1979) suggests that when determining confidence in a statistical research tool, reliability, objectivity should be also be considered. Roy, Govil and Miranda (1995) suggest that quickness in learning and the ability to automatically perform network design should be considered. (See also Learning)

Representational Adequacy: the system must be able to internally represent what it knows and what is needed to be learned. This concept could be expanded to include the statistical concepts of suitability (Van Dalen, 1979) -- the system must be capable of analyzing the problem under question. This concept seems to be similar to concept-learnability issues suggested by Haussler (1989) and Kolen & Goel (1991).

Credit Assignment: once implemented, when incorrect performance has been identified, the system must be able to identify its structural components that are responsible for incorrect performance, and must be able to make modifications so that correct performance can be achieved.

Perform Network Design: distantly related to credit assignment, a network learning method should be able to design an appropriate network for a given problem without the need for external input or predefined structures (Roy, Govil and Miranda, 1995)

Generalization: what the system learns from specific examples must be generalizable to general abstractions that are needed to perform specific tasks. Generalization directly concerns itself with the statistical concept of external validity. Because of the training rather than programming nature of neural networks, one could also argue that generalization is also concerned with internal and instrument validity. Kempka (1994b) suggests generalization is the ability to "infer from incomplete data or the success of acting upon data not included during training" (p 42). Similar to the Generalized Learning of Roy, Govil and Miranda (1995).

Computational Complexity: the system must be computationally efficient so that learning can occur in a reasonable amount of time (Kolen and Goel, 1991). Similar to the Efficiency in learning of Roy, Govil and Miranda (1995).

Reliability: the system must be able to yield the same basic results with the same exemplars are trained on networks under the same conditions. White (1990) called this statistical consistency in his evaluation of neural networks performing nonparametric regression functions. A procedure is consistent if the approximation error of the forecast approaches zero as the sample size approaches infinity. Procedures that are not consistent will make errors in classification,

recognition, or forecasting. Procedures that are consistent will only make errors due to the inherent randomness or fuzziness of the true relation between independent and dependent variables. Similar to the Robustness in Learning of Roy, Govil and Miranda (1995)

Objectivity: the system must yield the same findings regardless of the equipment used to run the computer program, or who administers or records the findings (Van Dalen, 1979).

Statistical Modeling: the development of equations that approximate the general pattern of a relationship of a dependent variable with one or more independent variables. Modeling techniques, such as linear and non-linear regression, develop mapping functions that approximate the shape of the real world data. (Smith, 1993)

Squashing function: a type of activation function that is designed to give outputs that are bounded, often 0 to 1 or -1 to +1. (Sarle, 1994b)

Supervised Learning: Uses both independent (network input) and dependent (network output) data as a unit, the network seeks to "learn" the relationships between the patterns described by the variables, such that when an input pattern is presented the appropriate output patterns is generated by the network. Based on these cases, generalizations are formed.

Target or Training Variable: a neural network equivalent to a statistical dependent variable (Sarle, 1994b).

Theory: "a set of interrelated constructs (concepts), definitions, and propositions that present a systematic view of phenomena by specifying relations among variables, with the purpose of explaining and predicting the phenomena." (Kerlinger, 1985, p 9)

Training: a neural network concept of learning that is the equivalent to statistical estimation (Sarle, 1994b).

Universal Approximator: A mapping function that is flexible enough to take any form the data requires. A model that is mathematically proven to approximate any function to any desired degree of accuracy. Multi layered feed forward Neural Networks using back propagation have been proven to be universal approximators (Hornik, Stinchcombe, and White. 1990).

However, Wray and Green (1995) suggest that because networks are simulated on computers inherently have rounding errors in computations, neural networks when implemented on computers may not be universal approximators.

Unsupervised Learning: The use of dependent data for both input and output in a network. With unsupervised learning, data patterns are discovered based on logical characterizations of the regularities (Decker and Focardi, 1995)

Variable: a statistical concept that is the equivalent to a neural network feature. Independent variables are neural network inputs, predicted values are called outputs, and dependent variables are called neural network targets or training values (Sarle, 1994b).

Population Distributions

A list of all possible outcomes of an event (A) and their corresponding outcomes is a probability distribution. Many events may be modeled on distribution functions.

The *normal distribution* uses estimates of the population's mean and variance to create a bell shaped curve that has been shown to model many different types of continuous random variables. The *standard normal distribution* is defined as a normal distribution with a mean of zero and a variance of one. A *standardized normal random variable* (Z) is one that has been subtracted from the mean and divided by the standard deviation of the population.

$$\frac{1}{\sqrt{2\pi\sigma}} \exp \left[-\frac{1}{2\sigma^2} (x - \mu)^2 \right] \quad (1-1)$$

where: μ = mean of population
 σ = standard deviation of population
 x = sample observation
 Z = standardized normal variable

$$Z = \frac{X - \mu}{\sigma} \quad (1-2)$$

The *binomial distribution* is a symmetric distribution in which the elements are discrete, and may be split into two classes. These classes may be called success or failure. Assuming a probability of 50%, as population samples exceed 100, the binomial distribution approaches the normal distribution.

$$P(X) = \frac{n!}{X!(n-X)!} p^X (1-p)^{n-X} \quad (1-3)$$

where:
 n = the number of trials
 p = the probability of success on a single trial
 X = the number of successes in the n trials
 μ = binomial population mean
 σ = binomial population standard deviation

$$\mu = np \quad (1-4)$$

$$\sigma = \sqrt{np(1-p)} \quad (1-5)$$

The *Poisson distribution* is a nonsymmetric distribution in which the mean equals the variance.

The Poisson distribution has been found to closely model events that occur randomly over a period of time. It has been found useful in modeling the number of times a particular event occurs over a period of time, or the number of times an event occurs over a specific area or volume (such as a manufacturing defect).

$$P(X) = \frac{\lambda^X e^{-\lambda}}{X!} \quad (1-6)$$

where: λ = average number of successes per unit of time
 e = base of the natural logarithmic system (2.71828...)
 X = designated number of successes

note: both the mean and the variance of the poisson random variable are equal to λ .

The *exponential distribution* is a continuous distribution. Where the Poisson is a distribution representing the number of arrivals over a specific period of time, the exponential distribution models the time between successive arrivals. The exponential distribution has been found useful in modeling situations in which subjects wait in a line.

$$P(X) = e^{-(1/\lambda)} \quad (1-7)$$

where: $1/\lambda$ = time between successes

e = base of the natural logarithmic system (2.71828...)

note: the exponential random variable's mean and standard deviation equals $1/\lambda$

The *t-distribution* is similar to the normal distribution in that it is symmetric to zero, however, its shape is dependent specifically on the *degrees of freedom* (given the result, the number of sample values that are free to vary to obtain the result), or generally on the number of items in the sample. When one substitutes the sample variance for the population variance in normalizing items, the sample normalized items follow the t-distribution, rather than the normal distribution. For sample sizes greater than 30, there is little difference between the t-distribution and the normal distribution. The t-distribution is often used in establishing confidence intervals and for determining whether differences in the means of two samples are probably based on natural variations in sampling, or whether there is a real difference. The t-distribution is used when the population is normally distributed, the mean of the population is known,

and the standard deviation of the population is unknown and replaced by the standard deviation of the sample. Given these requirements, the t-distribution is used as a method of testing and constructing confidence intervals for the mean of sample populations.

The *chi-square distribution* (c^2) is a nonsymmetrical, skewed right distribution that is derived by sampling the distributions of the sum of squares of independent, normally distributed variables, that is dependent only on the degrees of freedom of the sample. c^2 is used to construct confidence intervals or perform hypothesis testing on the standard deviations or variances of sample populations. Keppel (1982, p 46) suggests that degrees of freedom (df) can be thought of as:

$$df = \left[\begin{array}{c} \text{number of} \\ \text{independent} \\ \text{observations} \end{array} \right] - \left[\begin{array}{c} \text{number of} \\ \text{restraints} \end{array} \right] \quad \text{or} \quad \left[\begin{array}{c} \text{number of} \\ \text{independent} \\ \text{observations} \end{array} \right] - \left[\begin{array}{c} \text{number of} \\ \text{population} \\ \text{estimates} \end{array} \right] \quad (1-8)$$

The *F-distribution* is a nonsymmetrical, skewed right distribution that resembles chi-square. The F-distribution is used to test normally distributed populations with the equal variances to see if their means are equal or different. The F-statistic is used in the social sciences to determine the effect of an intervention on the population. Typically, a sample of the total population is divided into groups and at least one group is exposed to some treatment or experience that is designed to make the treated group behave differently from the non-treated group. The F-statistic is used to test whether any differences between the groups are due to random error or due to the treatment. The difference between actual behavior and average behavior is called error. Total error can be defined as consisting of error attributed to the intervention (variance between the sample means) and error that is randomly generated (variance within the samples). The F ratio can be defined as:

$$F = \frac{\text{variance between the sample means}}{\text{variance within the samples}} \quad \text{or} \quad \frac{\text{random error} + \text{treatment effect}}{\text{random error}} \quad (1-9)$$

Organization of the Study

Chapter I introduces the study, presents the problem, purposes and objectives, assumptions, limitations. This section also includes definitions to clarify terms used in this study. Chapter II describes the literature concerning the historical development of neural networks in general as well as some

refinements to the back propagation design. Chapter III describes literature concerning non-parametric statistics and back propagation. Chapter IV describes literature concerning back propagation design considerations. Chapter V describes literature concerning data preparation and design considerations for back propagation neural networks. Chapter VI provides a summary, conclusions and recommendations.

Limitations of the Study

The limitations of this study are:

1. The focus of the study is limited to feed forward neural networks using back propagation algorithms for learning (weight adjustment), and supervised learning designs.
2. The study is limited to examining neural networks used as data analysis tools and not as either models of biological systems or real-time adaptive signal processors or controllers.
3. The recommendations have not been tested as a unit.

CHAPTER II

NETURAL NETWORKS BACK PROPAGATION

The Review of Literature is divided into several basic components: a review of the Evolution of Neural Networks, a review of Non-Parametric Statistical Concepts, Traditional Statistical Classification Methods and their Relation to Neural Networks, a review of generalized Neural Network Learning techniques and specific back propagation based techniques that could be used in data analysis, and a review existing literature describing how data should be prepared for use with back propagation neural networks.

Technology's Impact on Knowledge

Computers are changing the way we deal with knowledge. The technologies used by systems to store and disseminate knowledge is evolving. A knowledge system may be described by two dimensions, how knowledge is stored and how knowledge is controlled (Kroenke, 1989). Knowledge has traditionally been stored alpha-numerically, in books stored in libraries. Technology is making computer storage of knowledge viable in databases accessible electronically. Knowledge has traditionally been controlled by humans, who select books to read and manually scanning pages of text, typically sequentially. Technology is making computer searches of databases viable, allowing for information to be both searched for and presented in nonsequential methods. Figure 1 presents a model describing a relationship between the way knowledge is stored and how the knowledge is controlled.

Books and publications tend to store knowledge sequentially. Hypertext, interactive multi-media and virtual reality tend to store knowledge in non-sequential formats. Computers have traditionally focused on the storage and manipulation of data and information (data organized in a

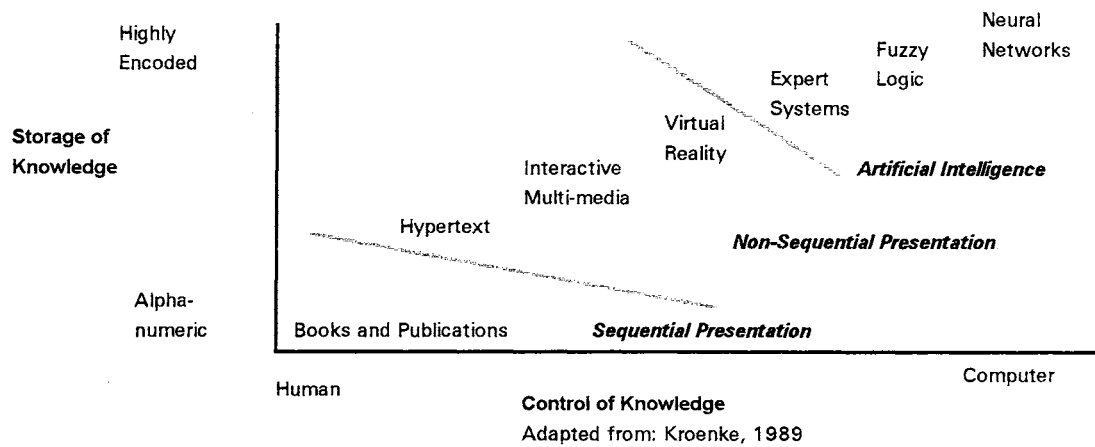


Figure 1. Technology and Knowledge Representation

way that makes it useful in decision making). The components of artificial intelligence are being used to interact with knowledge (a combination of instincts, ideas, rules and procedures that guide actions and decisions) in ways we are just beginning to be demonstrated.

Artificial Intelligence

While one might think of neural networks as a component of artificial intelligence, there is evidence that the concept has not yet been generally accepted by academia. A non-statistically based survey of eleven popular introductory Information Systems texts showed that, while all had sections on Artificial Intelligence, eight did not mention Neural Networks (Alter, 1992; Hutchinson and Sawyer, 1992; Schulethes, Sumner and Bock, 1992; Stair, 1992; Ahituv and Neumann, 1990; Lucas, 1990; McLeod, 1990; Senn, 1990). Only three of the reviewed texts mentioned neural networks, with discussions ranging from a few sentences to an entire section (Laudon and Laudon, 1992; O'Brian, 1990; Parker, 1989). To understand possible reasons for this omission, we will first look at the general topic of Artificial Intelligence.

Artificial Intelligence (AI) is a term that was coined in 1956 by John McCarthy. The term was used for the theme of a Dartmouth College conference to describe computers with the ability to mimic or

duplicate the functions of the human brain (Stair, 1992). Today, some say that AI is commonly defined as "the effort to develop computer-based systems (both hardware and software) that behave as humans" (Laudon and Laudon, 1991, p 662). Others suggest that existing definitions of AI are contradictory with some experts saying that "AI is the science of making machines do things that would require intelligence if done by a person. Others state that if we can imagine a computer that can collect, assemble, choose among, understand, perceive, and know, then we have artificial intelligence" (Hutchinson and Sawyer, 1992, p 486).

One reason that Neural Networks may not be specifically recognized as being a primary component of Artificial Intelligence is that many define Artificial Intelligence by "what it does." Neural networks do not fall neatly into a "to do" classification of robotics, natural language processing, expert systems, or virtual reality (Hutchinson and Sawyer, 1992). However, neural networks can be used to facilitate all of these "to do" classifications, but not in the traditional ways. To do classifications of AI include Robotics, Natural Language Processing, Expert Systems and Virtual Reality.

Ahituv and Neumann (1990) suggest that the whole area of artificial intelligence can be subdivided into two main branches, the engineering—analysis approach and the biological modeling approach, each with divergent objectives. In the engineering—analysis approach, the objective is to create a system that accomplishes tasks efficiently and reliably, regardless of whether the methods used are similar to those used by human biological and/or cognitive functions. Examples of this approach are pattern recognition tasks (reading characters or voice recognition), translating texts from one language to another, composing music, or optimizing network distributions.

In the biological modeling approach, the objective is to try to gain an understanding of the inside mechanisms of a real-life biological system, thereby enabling one to explain and predict behavior. Specifically, cognitive psychologists saw neural networks as a tool for modeling such brain functions as automatic response, recognition, learning, problem solving, etc. From a power and validity standpoint, when researchers use neural networks as a validation tool for various psychological and learning theories, it is important that the computer based neural network mimic biological functions. To the

extent that computer based models do not mimic biological functions, one may argue that any differences uncovered are the result of these mechanical differences and not the intervention of the experiment.

Interest in the concept of neural networks can be traced to the works of McCulloch and Pitts (1943) and Rosenblatt (1958). Interest reached a peak until 1969 when Minsky and Papert (Expanded 1988 edition) identified a major weakness in existing neural network designs (they could not model the X/OR problem, described in the section: *Hidden Layers, the Solution to the X/OR Problem*). The X/OR problem was not solved until McClelland and Rumelhart (1986) lead a group of cognitive psychology researchers in an attempt to refine neural networks as a model of biological functions.

Currently, within the connectionist community (those interested in neural networks), there seems to be two developing groups (Allman, 1989). The biological modeling group, primarily based in cognitive psychology and neuro-physiology, has as an objective, the modeling of biological neurodes on the computer. The engineering—analysis groups look at neural networks as another tool, similar to traditional statistics, that may be used in both data and systems analysis. Within the engineering analysis group, interest in artificial neural networks can be grouped by those who use networks either as real-time adaptive signal processors or controllers, or as data analysis tools (Sarel, 1994b).

Each group has differing objectives for the design and use of neural networks. To the biological modeling group, it is very important from a validity standpoint that the neural networks model biological systems. To the engineering—analysis group, what is important is neural networks (any design of neural network) can be used as a tool in analyzing and modeling systems of any kind. A focus of this paper will be the engineering “to do” approach rather than the modeling approach to neural networks. We will look at how neural networks may be used in data analysis for business related problems.

Evolution of Neural Networks

Early research in Neural Networks focused on modeling an individual biological neurode on a digital computer (Kimmel, 1991). Biological neurodes interact with other neurodes across synapses. An activated neurode releases neural transmitter chemicals that in turn stimulate a target neurode. Several activated neurodes may be stimulating either a single target neurode or multiple target neurodes. When

the stimulation of the target neurode reaches a critical level, the target neurode is activated, releasing neural transmitters to other target neurode(s).

In describing the developmental evolution of neural networks, it is useful to categorize work before 1969 (focusing on the interaction of individual neurodes) from work after 1986 (focusing on the interaction of a multidimensional system of neurodes.) The year 1969 is significant because that is the year Minsky and Papert published their book describing what neural networks could and could not do. The book was a consequential event for two interrelated reasons. First Minsky and Papert identified a primary problem with single level neural networks of the time—they could not solve the exclusive/or problem (X/OR)—when there is a choice between two alternatives and there is an equal preference for either one or the other, but not both. Second, they suggested that this inability to solve the exclusive/or problem was a major stumbling block in developing neural networks as a form of artificial intelligence. Work on neural networks significantly slowed until 1986 when cognitive psychologists Rumelhart, Hinton and Williams, and others collectively known as the PDP Research Group at the University of California - San Diego, showed how multi layered neural networks could solve the exclusive/or problem. Additionally, in 1986 Patricia Churchland, also at San Diego, provided a philosophical basis for neural networks in her book Neurophilosophy: Towards a Unified Science of the Mind-Brain.

Early Work (pre 1986) Interaction of Individual Neurodes

In 1936 Turing proposed the architecture of a logical machine that was the foundation for modern Automata Theory. Automation represents a formalization of rules for computation, studied as a branch of mathematics, and is the foundation for building digital computers. The Turing machine considered time to be discrete elements in which the computer performed tasks in a logical sequence.

McCulloch and Pitts in 1943 suggested that neurodes in the brain could be described as working like on/off switches. They offered a mathematical description of these processes and concluded that the brain worked like a Turing machine. The McCulloch and Pitts model of the neurode (Figure 2) has become the standard model used today. The input values are adjusted by weights (w_i) that are typically

between -1 and 1. The adjusted inputs are then (in its simplest form) combined to produce an output value. The learning of a neural network is accomplished by iteratively adjusting the input weights until an output value (or goal) is achieved.

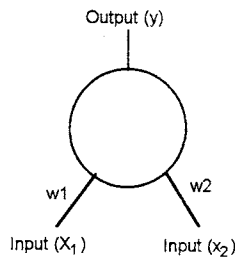


Figure 2. McCulloch-Pitts Neurode

In 1945 von Neumann established the basic design principles for digital computers. The design was for a single central processing unit for each computer that executed one processing command at a time in a serial manor. Later, von Neumann also proposed the design for a digital computer that stored memory programs in memory. Previously, a computer was programmed by making hand adjustments to the components of the system.

In 1956 Marvin Minsky, John McCarthy, Nathaniel Rochester and Claude Shannon organized what is recognized to be the first international conference to discuss the potential use of computers in the simulation of learning and intelligence. It was for this conference, sponsored by the Rockefeller Foundation, that John McCarthy coined the term Artificial Intelligence, which researchers then used to describe the new academic field that included neural computing. (Stair, 1992).

Perceptron. In 1958 Rosenblatt took the McCulloch-Pitts neurode and added to it an algorithm that would adjust the weights based on the success of predicting how to separate patterns into two categories. The model, called a *perceptron*, became the first trainable neural network.

The simplest perception is a device that decides whether a case belongs to one of two classes. It is the neural network equivalent to a linear discriminate. The general form of the single output unit perceptron is known as the threshold logic unit or Adaline (Widrow and Hoff, 1960) discussed in the next section. The objective of a perceptron based neural network is to determine (or learn) the proper weights that will produce the least error in determining the classification of presented cases. This is the same objective as we find with the statistical tool of discriminant analysis (discussed later.)

The perceptron uses an error-correcting procedure that attempts to totally eliminate all misclassifications. The output of the perceptron is either 0 or 1 and is either correct or not correct, nothing in-between. The consequence of modeling the perceptron's output this way is that the classes in the population must be linearly separable for the network to converge (determine weights that produce no errors). Many real world applications, such as problems requiring optimization, and problems involving growth, do not have populations that are linearly separable.

$$w_{j \text{ new}} = w_{j \text{ old}} + \beta y x_j \quad (2-1)$$

where: x_j = input i
 w_j = weight of input i
 y = output of neurode
 $\beta = \begin{cases} +1 & \text{if } y \text{ is correct} \\ -1 & \text{if } y \text{ is incorrect} \end{cases}$

For example, when looking at Figure 3, The number of inputs (x_n) corresponds to the number of variables for a specific application. The individual inputs are multiplied by weights (W_n). The sum of the weighted inputs are added to a constant (Q). The activation function of Widrow and Hoff's perceptron is a function that produces an output of either 0 or 1 to indicate membership in one of two classes. When the sum is greater than 0, the neuroid is said to be activated, and the output is 1. When the sum is less than 0, the output is 0.

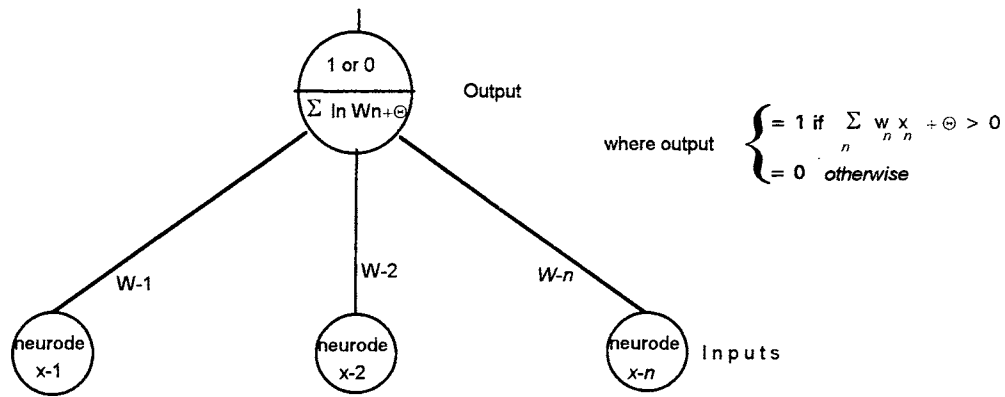


Figure 3. General Form of Single Layer - Single Output Perception

The constant (Θ) in a neural network's output node is called the threshold or bias. As with discriminant analysis, it geometrically represents the x intercepts of a discriminant line. In the operation of the network, it represents a hurdle rate, above which the sum of the weighted inputs must exceed for the neurode to fire, or present a 1 as an output.

Other activation functions to determine the value of y (the output) include:

$$\text{Step Function: } y = \begin{cases} 1, & \text{if } x > 0 \\ 0, & \text{if } x \leq 0 \end{cases} \quad (2-2)$$

$$\text{Perceptron: } y = \begin{cases} x, & \text{if } x > 0 \\ 0, & \text{if } x \leq 0 \end{cases} \quad (2-5)$$

$$\text{Signum: } y = \begin{cases} 1, & \text{if } x > 0 \\ 0, & \text{if } x \leq 0 \end{cases} \quad (2-3)$$

$$\text{Exponential: } y = e^x \quad (2-6)$$

$$\text{Signum0: } y = \begin{cases} 1, & \text{if } x > 0 \\ 0, & \text{if } x = 0 \\ -1, & \text{if } x < 0 \end{cases} \quad (2-4)$$

where: y = output
 x = input to node

Source: NeuralWare Advanced Reference Guide, 1993

A perceptron with a linear activation function is the equivalent of a linear regression model; with multiple inputs and outputs the equivalent to multivariate multiple linear regression. A perceptron with a non-linear (or logistic) activation function is the equivalent of a non-linear regression model. A perceptron with a threshold activation function is a linear discriminant function (Sarle, 1994b).

The perceptron has a very simple algorithm for finding optimum weights (or learning). Based on an initial random setting of the weights (usually between 0 and 1), each case containing both features and their correct classification is presented to the perceptron. Errors (variations between the perceptron

classification calculation and the true calculation) are corrected by adjusting weights after each inaccurate output. If there is an error in one of the cases, all cases are presented again. This epoch of case presentations continues until no errors are made.

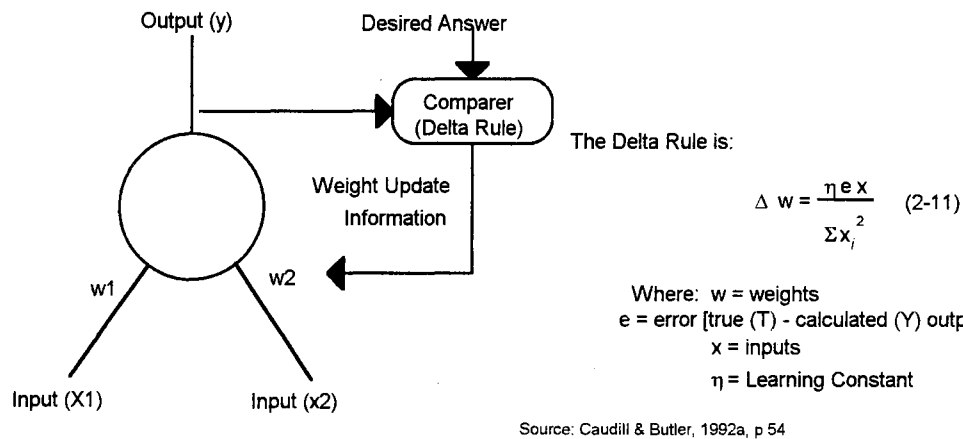
When an adjustment is necessary, the new weights will be calculated by adding to the old weight an adjustment factor. In the classic perceptron, adjustments to the weights and to the bias (q) are calculated based on the errors. The adjustment factor for each weight is calculated by multiplying the input by the difference between the true answer with the old calculated answer.

$$\begin{array}{ll}
 w_i(t+1) = w_i(t) + Dw_i(t) & (2-7) \quad \text{where: } w_i = \text{weight for Input } i \\
 Dw_i(t) = (T - Y) I_i & (2-8) \quad T = \text{true or correct answer} \\
 Q(t+1) = Q(t) + DQ(t) & (2-9) \quad I_i = \text{Input of Characteristic for case } i \\
 DQ(t) = (T - Y) & (2-10) \quad Y = \text{perceptron output} \\
 & \quad Q = \text{Bias}
 \end{array}$$

The advantage to the perceptron's learning algorithm is that it, unlike classic discriminant analysis, requires no assumptions concerning the normality of the population's distribution. The perceptron based neural network is nonparametric in nature (Widrow and Hoff, 1960; Rumelhart, Hinton, Williams, 1986a; White, 1990; Weiss and Kulikowski, 1991).

The neural network Rosenblatt used was a grid of 400 photocells (inputs) arranged to resemble the eye's retina. These photocells were randomly connected to 512 neurodes, that were in turn connected to a set of output units. When a letter of the alphabet (e.g., s or v) was displayed to the photocell array, a set of photocells would activate. These photocells would then activate the neurode like units, which would intern activate an output unit that indicated whether the pattern fell into an s or a v category. By manually adjusting weights between neurodes, Rosenblatt was able to get his network to recognize all letters of the alphabet.

Adaline. In 1960, Widrow and Hoff modified Rosenblatt's algorithm and created a machine they called *adaline* (Figure 4), that used a network of adaptable neurodes to recognize patterns. The adaline was the first practicable example of what is known as supervised learning, training a network based on known correct responses. The adaline provided the framework for automated telephone switching equipment. The network's weights were modified using what has become known as the Delta Rule, or the Least Mean Square Rule (discussed in section: *Least Mean Square Rule*).



Source: Caudill and Butler, 1992a, p. 54.

Figure 4. Adaline

The classic delta rule looked at the response of the perceptron. For a specific case (pattern of inputs $x_1, x_2 \dots$), the calculated output of the perceptron is measured with the known correct value. If the case's output is different from the true value, the delta rule is initiated, the weights are adjusted, and the next case is tried. If the next case's calculated output matches the correct value, then the next case is tried. If the next case's calculated output is different from the true value, then the delta rule is initiated, etc.

Neural Networks Modeling Biological Activities. By the middle part of the 20th century, it was thought that biological learning was the result of varying the strengths of synaptic connections. Early researchers built models of these synaptic connections on computers using mathematical functions to emulate the stimulation of neurodes. In 1949 Hebb proposed that if one neurode sends many signals that are stimulating another neurode, then the synaptic connection between the two neurodes is strengthened. The conclusion to this theory is that with every new incident experienced, the brain slightly rewires itself. This Hebbian learning has a psychological counterpart in conditioned response described by Pavlov. As researchers started implementing Hebb's law on computers, they

found several problems. For example, the classic formulation does not specify how much learning should increase or decrease, nor how two neurodes interact.

In 1969, Grossberg provided a mathematical theory of classical conditioning model described by Pavlov. Grossberg defined a neo-Hebbian learning model consisting of two sets of dynamic differential equations, "one governing the activity change of an arbitrary network at a given instant in time and the other governing the weight changes of an arbitrary connection in the network at any instant in time" (Caudill and Butler, 1992a, p 61). Grossberg's model provided some elementary decay characteristics.

$$\Delta w_j = \alpha_j (\eta_j x_j - \delta_j w_j) \quad (2-12)$$

where:

- x = input patterns
- w = weights
- η = learning rate
- δ = decay rate parameter

Grossberg also suggested the use of a sigmoid output function and contributed to the Cohen-Grossberg theorem which suggests that the response to any external input, a neural networks' weights will eventually converge to an equilibrium.

Clark and Rvishanker (1990) proved a convergence theorem for Grossberg's learning rule when the learning rate converges. They also showed that the Grossberg learning rule computed the probabilistic centroid of the training set.

Teuvo Kohonen of Helsinki Technical University in Finland made two important contributions to the field of neural computing in the 1970's. One of Kohonen's contributions concerns adaptive and associative learning (an extension of Hebbian learning), in which rules of network weight modification depend only on the previous weight values and the values of neurode input and outputs. These adaptation rules include memories defined by an auto-correlation matrix. Special cases of these rules include the Linear Associator and the Brain-State-in-a-Box.

The second of Kohonen's contributions is the principle of competitive learning (also called Self-Organizing Maps) in which neurode connecting weights are determined by a competition with other connectors, with the winner adapting itself to respond more strongly to a stimulus. Competitive learning is a non-supervised form of learning, with weight modifications determined only by input to the network. Competitive learning is an attempt to model the lateral interactions of neighboring

groups of neurodes in the brain. (Competitive learning is discussed further in this dissertation's section on *Types of Neural Network Learning*.)

Mathematical Foundation of Computer Based Neural Networks. In 1982, Hopfield provided the mathematical theory that describes the operation of computer based neural networks. The process of training neural networks involves modifying the interconnecting weights between network neural nodes. Previously, these weights were modified largely on a trial and error basis. Hopfield suggested that the process of optimizing interconnecting weights be thought of as a system whose energy is represented by the error found when comparing patterns at the input and output nodes. According to Hopfield, as the system moves from high energy (little correlation between input and output nodes) to low energy (high correlation between inputs and outputs), it did so not according to fixed rules of logic, but gradually settling on the right combination of weights that minimizes energy (or error).

One significant consequence of Hopfield's concept is that it suggests that a digital computer and a neural network "think" in very different ways. A typical computer and its programs operate in a very sequential, methodical, hierarchical, and logical way, passing information from one stage to another in the process. Information is received at the beginning of the sequence, processed, and passed along to the next higher stage, where it is processed and compared with other information and passed on to still higher stages. By the time the information has sequenced its way through the hierarchical chain, all the available information has been examined and a conclusion is reached. This type of hierarchical information processing is well suited for tasks involving precise details and exact answers, such as bookkeeping or high-level mathematics.

However, in a neural network information flows simultaneously, back and forth between different elements of the system. An analogy to the way neural networks work would be a town meeting where each person informs others how they feel about a topic. As the discussion continues, people will change the strength of how they feel about the topic. Gradually the group reaches a decision that minimizes the groups dissatisfaction, through either consensus or a form of majority rule. This type of freewheeling information processing seems to be better at pattern matching or "optimizing

problems where incoming information is often slightly inaccurate or incomplete and a choice must be made from many alternatives" (Allman, 1989, p 94).

Boltzman machine of Hinton and Sejnowski (1984) was based on Hopfield's suggestion that when a network operates, it seeks to find the relationships that represent the lowest energy level of that network. This design became the first self-learning neural network.

Hidden Layers, the Solution to the X/OR Problem. The exclusive/or (X/OR) problem arises when there is a choice between two alternatives, and there is equal preference for one or the other, but not both. From a neural network standpoint, Minsky and Papert (1988 expanded edition) showed that it was mathematically impossible for a single-layer network to be trained to simulate the problem. Allman (1989) explained the X/OR problem by describing a situation where the happiness of a young man (Patrick) at a party was contingent on only one of his two girl friends (Pam or Sue) were present. If both Pam and Sue were at the party, Patrick would be miserable. In designing a neural network to model the problem, the inputs (Pam and Sue) would have a value of 1 if they were at the party or 0 if they were away from the party. The output (Patrick) would have a value of 1 if he were happy or 0 if he were unhappy. Patrick's value would be calculated based on values of the input and weights connecting the inputs to the output node. In terms of the Allman's example, what Minsky and Papers showed what that it was impossible to calculate the correct value for Patrick (unhappy) when both of his girlfriends were attending the party (see Figure 5).

Neural networks using a single hidden layer can solve the X/OR problem. For example, when either Pam or Sue are present (input are either 0 and 1 or 1 and 0), then Patrick is happy (output = 1). When both Pam and Sue are present (inputs are 1 and 1) then Patrick is unhappy (output = 0), as shown in Figure 6. In fact, adding hidden layers solves most of the problems Minsky and Paper identified in their 1969 analysis of the state of neural networks.

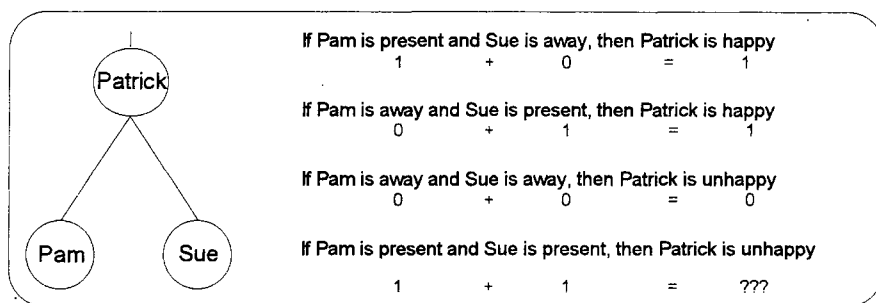


Figure 5. Single Layer and the X/OR Problem

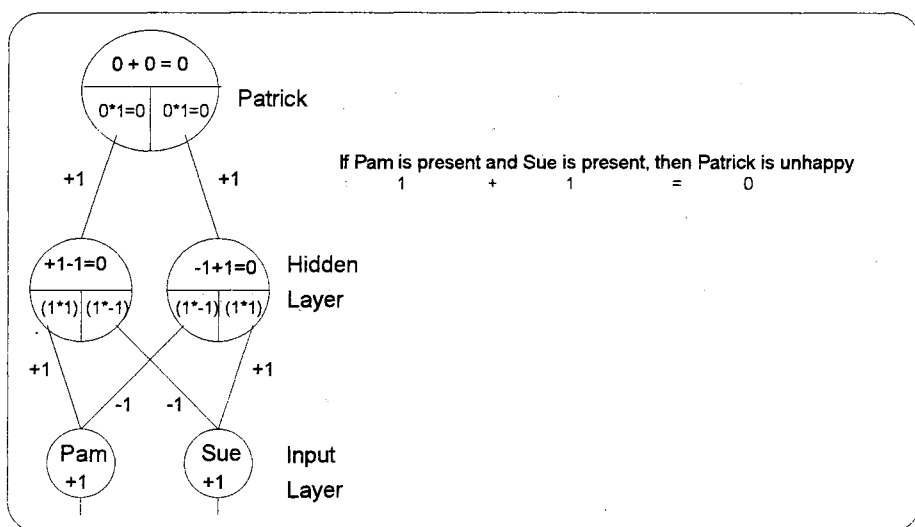


Figure 6. Hidden Layer and the X/OR Problem

Hidden units are an important advancement because they allow neural networks to form representations of real world systems—representations that can then be used to make complex decisions (Allman, 1989). For example, assume a situation where John is an investment advisor who makes recommendations on investment decisions based on various economic indicators. Figure 7 is a representation of how inputs (economic indicators) and outputs (investment decisions) may be related.

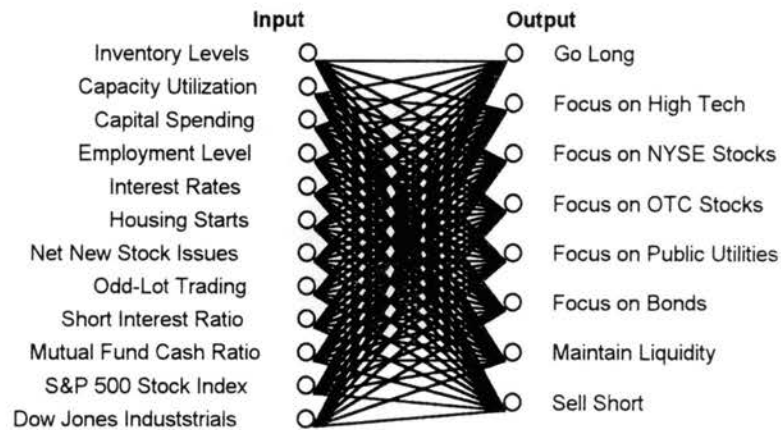


Figure 7. Simple Neural Network

One could also construct a network that uses hidden units between the input and output. (See Figure 8). Note that the Simple Neural Network uses many more connections than the network with hidden units. Also note that the hidden units each represent something in the outside world. While in reality one may not specifically be able to design hidden to perform specific tasks, in this case we can see that they become representations of a Bull Market, a Neutral Market, or a Bear Market.

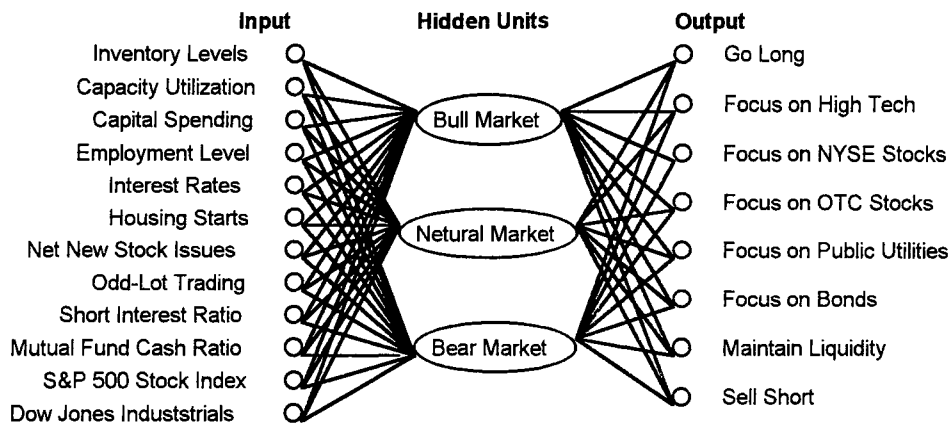


Figure 8. Neural Network with Hidden Units

This ability to form implicit representations gives neural networks their power. Hidden units allow for networks to represent high level concepts, in addition to being able to perform more complex computations. McClelland suggests that "They represent what has to be represented to solve a problem. They represent meanings." (Allman, 1989, p 126) The meanings they represent are different kinds of symmetry in the patterns that the networks are representing. When a neural network is training, it uses as many hidden units as necessary to model the different kinds of symmetry of what is being modeled. If the network does not need a particular hidden unit, its weights tend toward zero and it effectively is eliminated from the network. (Allman, 1989, 127)

In another example of the power of hidden layers, Hinton (1987) used neural networks to identify relationships between two separate families, one English and the other Italian. He trained the network by listing examples of relationships within the families, such as Sophia's mother is Lucia, or Lucia's mother is Maria, or James's spouse is Victoria. Once the networks were trained, it was able to recall relationships, and make inferences about relationship it had seen. For example, it could infer that Sophia is Maria's granddaughter, without the explicit relationship being stated.

Even though the training provided no information to distinguish the two families, a person's sex, or relative age. But because in the family trees, Italians are only related to other Italians, and English are

related to other English, the network was able to make these distinctions. After training, hidden units were examined, it was noticed that the pattern of activation across the units could be found to represent these attributes.

Current Work (post 1986) Interaction of a Multidimensional System of Neurodes

The year 1986 proved to be a watershed year for work in neural networks because of work done at the University of California - San Diego. The spark for renewed interest in neural networks did not come from the statistical/mathematical fields, but in the areas of philosophy and cognitive psychology. While Parker and Werbos (1985) independently introduced an algorithm similar to back propagation (a back propagation neural network learning algorithm) at about the same time, the paper seems to have been largely ignored. In 1987, Lapedes and Farber working at Los Alamos, began a foundation for using back propagation neural networks as a statistical tool by showing that back propagation could be used to model nonlinear transfer functions, and could predict chaotic behavior in systems better than traditional statistical systems.

Philosophical Foundation of Neural Networks. In 1986 Churchland provided a philosophical basis for the use of neural networks. She did that by "questioning whether a philosopher could investigate the qualities of the mind without paying attention to the mechanics of the brain" (Allman, 1989, p 40). The study of philosophy can be divided into Metaphysics (the study of what is real), Epistemology (the study of what is truth), Ethics (the study of the right behavior) and Aesthetics (the study of what is beautiful, and good). Typically, philosophers have approached epistemology as an exercise of pure logic and reason by searching for the *a priori* foundations of human knowledge, unconcerned about the biological function of the human brain. This approach has led to philosophy's general approach of modeling the mind's thought processes by using a system of symbols. These symbols are manipulated by rules of logic that are assumed to represent the basic thought processes that are common to all humanity. It therefore followed that if the mind works by manipulating universal symbols with a universal set of rules, then one could focus on examining those rules without regard to the biological functions of the

brain. Logical Empiricism studied these symbols and rules, and provided a theoretical framework that could be used in an attempt to program computers that would mimic human thought processes.

However, it can be argued that Logical Empiricism is a failed philosophical doctrine because there is evidence that the mind does not work with symbols that are manipulated by logical rules. According to Paul Churchland, "If there were a distinction between software and hardware in the brain, then the approach was a good one... But there isn't this clear dissection, so the basic assumption of the whole approach is screwy" (Allman, 1989, p 52).

From a cognitive psychology standpoint, this realization is similar to what Kuhn (1970) calls a paradigm shift. According to Patricia Churchland:

A neurobiological understanding of the mind will cause us to rethink our old presumptions about who we are: what it is to have a self, what it is to have a soul, what it is to be responsible, to think, to introspect, and to have free will. It may have tremendous implications for morals, too... Our decision making is much more complicated and messy and sophisticated -- and powerful -- than logic... Our decision making may turn out to be much more like the way neural networks function: The neurodes in the network interact with each other, and the system as a whole evolves to an answer. Then introspectively, we say to ourselves: "I've decided." (Allman, 1989, p 55)

Psychological - Cognitive Foundation of Neural Networks. Interest in today's neural networks increased in 1986 both when Churchland provided a philosophical basis for neural networks and when psychologists Rumelhart, Hinton and Williams showed that a multi layered neural network could solve the exclusive/or problem. It was also in 1986 that psychologists McClelland and Rumelhart edited a two-volume book entitled *Parallel Distributed Processing: Explorations in the Microstructure of Cognition* that some think would have become the definitive book on the subject if technology in the area were not moving so fast.

James McClelland, a cognitive psychologist, provided a key in understanding how a neural network could mimic cognitive recognition and representation processes. McClelland was interested in how words are recognized. According to traditional logical empiricism theory, it was thought that one

recognizes letters first, then put the letters together to recognize a word. The problem McClelland identified with that theory can be described by reading looking at Figure 9. The first letter may be either an R or a P. The second letter may be an E or a F. The third letter could be a B or a D. In spite of the ambiguity in letters, most people can recognize the word as being RED, contrary to traditional theory.



Source: Allman, 1989, p 102

Source: Altman, 1989, p. 102.

Figure 9. Recognition - Representation

McClelland suggested that the mind works instead by having a mechanism that indicates its degree of confidence in the different possibilities of each letter, and sends that information to another level that indicates its degree of confidence in the different word combinations that could be made from the possible letters. In our mind, a word detector for RED is activated because of its high degree of confidence.

Rumelhart, trained in mathematics, was working in the cognitive area of determining how people use memory packages to fill in missing information about stories they hear. For example, when a tourist in Manhattan asks a police officer "Do you know where the Empire State Building is?" Why does the police officer usually not just answer "Yes," but also gives directions? Rumelhart suggested this behavior indicates a mechanism that uses a degree of confidence in determining the real question being asked, and lead to a theory of filling in missing information.

McClelland's concept of cognitive levels of thinking lead to the concept of having neural networks constructed in layers, which enabled neural networks to perform the exclusive/or problem

identified by Minsky and Papert. These intermediate layers of a neural network are called hidden layers because they receive no direct input, and provide no direct output to the outside world.

Neural Networks as a Statistical Tool. Kohonen (1984) suggest that discriminant analysis using Fourier methods is comparable to linear neural networks. In 1987, Lapedes and Farber working at Los Alamos, showed that hierarchical neural networks using back propagation (see section: *Error-Correcting Learning [Supervised]*) could be used to model nonlinear transfer functions, and could predict chaotic behavior in systems better than statistical systems. They showed the Least Mean Squares algorithm (see section: *Least Mean Squares - Delta Rule*) "works for the Garbor Polynomial Method with virtually no change in implementation" (p 7), and suggest that for large systems, neural networks may prove to be a better alternative. Their paper, written in the context of signal processing, identified the ability of neural networks to infer algorithms and to perform generalizations as being "nothing more than real values function interpolation" (p 23). They also showed the process of training a back propagation hierarchical neural network is "in essence, constructing a discrete Fourier series... with the number of adjustable frequencies determined by the number of neurodes in the hidden layer" (p 26).

According to a mathematical theorem proved by Andrie Komogorov in the late 1950's (and restated for neural networks by Robert Hecht-Nelson in the late 1980's) "the network will always eventually figure out how to make perfect forecasts of the data on which the neural network is being trained" (Coats and Fant, 1991, p 12).

Neural Networks as a Universal Approximator. Hornik, Stinchcombe, and White (1990) demonstrated that a hierarchical neural network using back propagation could be used to approximate any mathematical function, and that the "it is not the specific choice of the activation function, but rather the multilayer feedforward architecture itself which gives neural networks the potential of being universal learning machines" (p 252). They also showed that neural networks can be used to discriminate nonlinear separable classes and that they have universal approximator properties.

However, Wray and Green (1995) suggest that because of rounding error and other limitations of digital computers, neural networks run on computers may be Best Approximators rather than universal

approximators. This suggests that not all problems may be solvable when neural networks are implemented on digital computers.

Gaynier and Downs (1995) have noted that networks with non-monotonic transfer functions tend to have infinite Vapnik-Chervonenkis (VC) dimensions which means that they can not be expected to arrive at a general solution to a problem when presented with only a training set of the total population. They note that networks with high VC dimensions may perform well on training data, but perform poorly on test data. Testing data is usually a component of cross-wise validation techniques.

Neurode Operation

All neurodes fundamentally operate in the same manner; they sum weight-adjusted inputs, possibly add a bias value, and finally pass the results through a transfer (or activation) function to produce an output that is offered for other neurodes to use. After the weighted sums of the inputs are computed at each neurode, the activation (or transfer) function is used to activate the neurode's output to a non-zero value, transferring the internally generated sum to an output value.

Cichocko and Unbehauen (1992) suggest that the proper choice of criterion that should be used in the selection of an activation function depends on partly on the specific applications and greatly on the distribution of errors in the sample data.

Activation (Transfer) Function

Activation functions evaluate the inputs to a neurode and produce a single output of the neurode. Activation functions are also used to introduce non-linearity into the network; a feature that provides neural networks a modeling advantage over traditional linear techniques. There are four basic types of activation functions classically used in neurodes: Logistic or Sigmoidal, Linear, Linear Threshold, and Hard Limiter or Binary.

Hard Limiter: (or threshold) a binary type function that is either all on or all off. The Hopfield net, the McCulloch Pitts neurode, and the perceptron all include a hard limiter activation. It provides a basic nonlinear function required to obtain complex behaviors expected in neural networks.

Values are either 0 or 1. Will provide the fastest training times in networks designed to operate logically, such as an X/OR problem (Kempka, 1994b).

A perceptron with a threshold activation function is the equivalent in statistics to a discriminant function. If there is only one output in the system, the network is known as an Adaline (Sarel, 1994b).

Linear: where the output is the sum of the inputs, merely passing the weighted-sum value to the next level. Kempka (1994a) suggests that network output neurodes should exclusively use the linear function, due to the linear function's mathematical properties. A linear function is defined as:

$$y = \sum x \quad (2-18) \quad \text{where: } \begin{array}{l} y = \text{output} \\ x = \text{input to node} \end{array}$$

Linear activation functions should be found exclusively at the output neurodes of a neural network due to the mathematical properties of linear functions. Sarel (1994) suggests that if the output variables are assumed to be unbounded, a linear activation is appropriate. However, if the variables are positive but have no upper bounds, an exponential output activation function is recommended.

A perceptron with a linear activation function is the equivalent to a linear regression model. A network with multiple inputs and outputs is the equivalent to a multivariate multiple linear regression (Sarle, 1994b).

In a multi-layered neural network, the output layer should contain a linear activation function. Blum and Li (1991) have constructed a proof that a neural network with two hidden layers and a linear output are universal approximators.

Logistic: The logistic activation function is the de facto standard used in the clear majority of neural network systems (Caudill and Butler, 1992a; Smith 1993; Kempka, 1994a). The logistic function is sigmodal, continuous differentiable function that provides the non-linear characteristic that provides neural networks their unique abilities. The logistic function transforms values to either 0 and 1 or -1 and +1.

$$y = \frac{1}{1 + e^{-x}} \quad \text{for values between 0 \& 1} \quad (2-13) \quad \text{where: } y = \text{output}$$

$$y = \tanh(x) \quad \text{for values between -1 \& 1} \quad (2-14) \quad \text{where: } x = \text{input to node}$$

The logistic function may not be the best choice when run-time efficiency is an issue (Kempka, 1994a).

A sigmodal logistic function is defined as:

$$y = (1 + e^{-x * g})^{-1} \quad (2-15) \quad \text{where: } y = \text{output} \\ g = \text{gain} \\ x = \text{input to node}$$

Ito (1991) proved that a sigmoid function can, under certain circumstances, become universal approximators. Sontag and Sussmann (1991) showed that with linearly separable data, neural networks with no hidden neurodes, using back propagation gradient descent with sigmodal activation functions will find a global minimum from any initial weight configuration in a finite time.

Klimasauskas (1991a) suggests that sigmodal functions work best when the problem involves learning about “average” behavior. However, if learning involves “deviations” from the average it is suggested that Hyperbolic Tangent should be used.

The Hyperbolic Tangent is a continuous, monotonic mapping of the sum of the inputs (similar to Sigmoid) into a value between -1.0 and 1.0. The increased magnitude and a much greater slope give a different error space from the related logistic function.

$$i = x * g \quad (2-16) \quad \text{where: } y = \text{output}$$

$$y = \frac{e^i - e^{-i}}{e^i + e^{-i}} \quad (2-17) \quad \text{where: } g = \text{gain} \\ x = \text{input to node} \\ i = \text{modified input characteristic}$$

Klimasauskas (1991a) suggests that hyperbolic tangent functions work best when the problem involves identifying “exceptional” situations.

A perceptron with a non-linear (or logistic) activation function is the statistical equivalent to a non-linear regression model (Sarle, 1994b).

Linear Threshold: a type of hybrid function that is very similar to a logistic function and can be used as a replacement to estimate a sigmodal function when computer processing time is at a premium (Kempka, 1994a). In terms of generalization, the linear threshold will perform about 10% less than a logistic function, but may be a better choice when offset with its reduced computer

overhead. Because of its constant slope, the linear threshold may be a better choice than logistic when output values are expected to be in the range of .5 (or the center of the sigmoid). However, when using linear threshold, the possibility of a zero slope will stop back propagation and require an algorithm other than gradient descent (such as MLO) (Kempka, 1994b).

Other Components of an Activation Function

(Squashing, Bias, Gain, etc.)

Squashing: transforming an output of unlimited range to an output of a specific range, generally either 0 to 1 or -1 to +1. The choice of range will influence the training of any neural network. Differences in magnitude (one vs. two units) and the steepness of the slope of the function will give different error surfaces.

Bias: One way of looking at neural networks is to think of them as modeling a hyperplane through N-dimensional space. The weights between neurodes determine where the hyperplane is in input space, with the plane passing through the origin of the hyperspace as defined by the inputs. A *bias* (Q) is a mathematical term that allows the plain to be moved somewhere else. If a neural network has many neurodes in a layer, all neurodes share the same input space; without a bias term, all would also be constrained to pass through the origin (Prechelt, 1995).

In mathematical terms, *bias* (Q) can be thought of as the point at which the a function crosses the y axes. For example, an activation function of the form $y = f(Q + gx_1)$ with a *bias* of 0, the logistic function with a maximum of 1 and a minimum of 0, would crosses the y axes at .5. By increasing the bias weight from 0 to 1, the function is shifted to the left, thereby increasing (or raising) the y-intercept.

When used in neural network activation functions, bias may be used to set hurdle rates for neurode activation. See also the gain, which may be used to change the slope of the function, thereby also effecting the function's y-intercept. In the operation of the network, the constant (Q) is a bias, or threshold, represents a hurdle rate, above which the sum of the weighted inputs must exceed for the neurode to fire, (present a 1 as an output). In the case of a single neurode with one input and one

weight, the bias is the equivalent to the y-intercept in an equation of a line. Kempka (1994a) notes that “bias values add a clear benefit to training, and using them increases the training efficiency” (p 35). It is also noted that in the case of networks with a large number of neurodes and no assigned bias, a separate neurode may evolve into a bias device.. When used with the Hyperbolic Tangent Activation function (with values between -1.0 and +1.0), bias is recommended if the training set contains an instance in which an important input vector is zero when a non-zero output is needed.

Gain: In mathematical terms, *gain* (g) can be thought of as a weight that adjusts the slope of a function. For example, consider an activation function of the form $y = f(Q + gx_1)$ with a *gain* (g) of 1. If the function is a logistic function with a maximum of 1 and a minimum of 0, increasing the gain weight from 1 to 2 would create a line more closely tracks the y axes with the slope of the curve at the boundary changing from .25 to .5. Decreasing the gain from 1 to .1 would flatten out the line.

Adaptive Gain: Where Gain is defined as a multiplicative adjustment to the inputs of a node, and its implementation can be thought of as a continuous modulator of the error propagated to the weights during training. Adaptive gain is an indication of how much the node participates in representing the input and the extent to which the node participates in learning.

The gain of a node in a hierarchical back propagation neural network is a multiplicative constant that either amplifies or attenuates the inputs to a node. The concept of gain in activation functions was first introduced independently by Movellan (1987) and Tawel (1989). Kruschke and Movellan (1991) have shown that learning is enhanced using gradient descent and an adaptive gain factor. An additional benefit of gain is that it, in effect, normalizes weight vectors of neurodes. (See the following section on normalization.) Adaptive gain is used in the place of a momentum factor in back propagation gradient descent calculations. When gain is used, it can be calculated using the

chain rule of the gradient descent procedure by modifying equations, as shown below:

$$y = g_j(\sum w_j x_j) \quad (2-19)$$

$$\Delta g_j = -\gamma \sum_j g_j s_{jj} \quad (2-20)$$

$$\gamma(\tau) = \frac{-k \sum_{i=0}^{\tau} \varpi \Delta \text{RMSE}(\tau-1)}{\text{RMSE}(\tau)} \quad (2-21)$$

where: y = output
 g = gain
 w = weights
 x = input to node
 γ = competition rate
 s = lateral connection strength
 k = competition rate modulator constant
 ϖ = exponential average constant
 RMSE = square root of the mean squared error

In the above equations, k is a non-negative constant of proportionality called the competition rate modulator (typically around 2), and represents the degree of competition to be used in evaluating hidden nodes. A value of k that is too large will result in a network that is overgeneralized, and will typically operate with high error rates. The other constant, v , is a positive constant governing the weighting of the exponential average (typically set about 0.7)

Kruschke and Movellan (1991) suggest that adaptive gain enhances the learning process by modifying the magnitude, and not the direction, of a weight change. Early in the training process when a successful direction has not been found, weights and gain factors are small. Once a successful direction is found, weights and gain increase.

Adaptive gain can also be used to eliminate hidden layer neurodes from participating in the neural network. Kruschke and Movellan (1991) have suggested that "a given hidden node participates in representing the input if and only if its activation changes for some change of input." (p 276). It is also suggested that gain may be thought of as the reciprocal of a Boltzman machines temperature parameter.

For those interested that neural networks model human biological functions, there is evidence suggesting that the nervous system has mechanisms that modulate the neural response function in a manner similar to gain. Servan-Schreiber, Printz, and Cohen (1990) proposed that a gain parameter helps explain the effect of biogenic monoamines, a type of neurotransmitter associated with neural responsiveness modulation. They successfully used the gain parameter in back propagation networks to simulate phenomena associated with catecholamine manipulations.

Types of Neural Network Learning (Adjusting Weights)

The concept of learning is a very broad topic. Learning systems are often defined in terms of classification or prediction. From a cognitive standpoint, classification is often associated with pattern recognition. From a statistical standpoint, a classification problem is sometimes called a prediction problem. From a computer system standpoint, classification and prediction is sometimes called concept learning. From a neural network standpoint, learning is the process of adjusting the weights to connecting neurodes that make up a system that describes a natural phenomenon.

From an empirical standpoint, the goal of learning is to "extract a decision rule from sample data that will be applicable to new data. A typical learning system is designed to work with some general model, such as a decision tree, a discriminant function or a neural net.

'Learning' consists of choosing or adapting parameters within the model structure that work best on the samples at hand and others like them." (Weiss and Kulikowski, 1991, p 4)

From a neural network standpoint, there are several different ways of classifying learning rules. Matheus and Hohensee (1982) suggests that there are three basic classes of learning rules, Error-correcting rules, correctional rules, and unsupervised learning rules.

Unsupervised Learning

Unsupervised learning occurs as a self-adaptation process to detect regularities in the input space without direct feedback from a teacher or supervisor, with the resulting network called an auto-associative network. Weiss and Kulikowski (1991) suggest that the goal here is to identify clusters of patterns in the data that are similar. The assumption is that these similar clusters will be categories or classes that may be used in solving problems. The ability of neural networks to solve and find solutions using unsupervised learning is limited to the degree that the data are mutually exclusive. If the data are not mutually exclusive, structure (or constraints) will need to be added into the basic classification representation. Establishing this structure usually requires insight and an understanding of the application area.

Sarle (1994b) suggests that the goal of most forms of unsupervised learning is "to construct feature variables from which the observed variables, which are really both input and target variables, can be predicted" (p 6). From a neural network standpoint, this means that dependent data is used for both network inputs and network outputs. When the data is presented as an input, the network (which has at least one hidden layer with a smaller number of neurodes than is in the input and output layers) seeks to extract patterns that will recreate the input pattern as an output.

Networks using this type of learning paradigm are also called self-organizing. Competitive learning is a variation of unsupervised learning that is used in Kohonen feature maps. This concept of competitive learning is based on characteristics of the brain called lateral inhibition. A Kohonen network uses neurodes connected in the layers above and below, and also connections to other neurodes within the layer. Within each layer, the connections are positive when connecting to neurodes close by, and negative when connection to neurodes farther away. These internal connections within layers tend to create a competition between neurodes within the layer whose purpose is to determine which neurode within the layer has the strongest response and eliminate all other neurodes within the layer.

Each neurode acts independently, without a global measure of a layer maximum performance. Once the Kohonen layer has stabilized, the output from the layer is a +1 from the winning neurode, and no response from others in the layer. The training rule used by the network can be expressed by:

$$\Delta w_{j,new} = \beta (x_j - w_{j,old}) \quad (2-22) \quad \text{where: } \begin{array}{l} \beta = \text{learning constant or gain} \\ x_j = \text{input signal} \\ w_j = \text{weight} \end{array}$$

A Kohonen feature map is very versatile, either used on its own, or existing as a layer within a larger network. When being used on its own, the feature map is a two-layer network in which the input layer is fully connected to the Kohonen layer, distributing the entire input pattern to each of the neurodes in the Kohonen layer. The Kohonen layer neurodes act as the output layer and transmit their outputs to the outside world. A critical part of the Kohonen layer's self-organizing property is a result of a large number of additional connections that link the neurodes within the layer to each other.

When using unsupervised learning techniques one should measure the homogeneity of the population (Decker and Focardi, 1995). A group is homogeneous if all members are similar according to some characteristic.

Correlation Learning

Correlation learning rules are those in which individual weight changes are solely based on the levels of activity between connected units. Initially, connections between units are randomly set, and then modified as the network comes closer to solving the problem. The closer to the proper solution, the closer the units are to a workable arrangement. Models proposed by Hebb (1949) and Hopfield (1982) are non-statistical networks that are examples of correlation learning (Sarle, 1994b).

A variant of correlation learning is called reinforcement learning, where an external source indicates whether the response to an input is good or bad (Sutton and Barto, 1981).

Error-Correcting Learning (Supervised)

Error-Correcting rules (also called supervised learning rules) depend on external feedback about the desired signals of the output units. Examples are models developed by Widrow and Hoff (1960) and Rumelhart, Hinton, and Williams (1986a). With supervised learning, the data is presented in the form of samples observations or cases, with a corresponding correct classification designation. The resulting network is called a hetero-associative network. If the input data is the same as the output data, then the resulting network is called an auto-associative network. The neural network adjusts its internal weights in an attempt to find a general way of relating specific patterns of observations to the specified correct class. The process of the neural network adjusting its internal weights in an attempt to find a generalized relationship between the cases and classes is called training or learning. The objective of the learning task is to find and identify some pattern that is not too specific between the observations and their corresponding classes. The goal of the neural network is to predict new observations, not to discriminate based on old cases.

There are several techniques that can be used by the neural network to adjust internal weights in an attempt to identify a pattern between observations and corresponding classes. Reinforcement learning, first modeled on the computer by Hebb (1949), occurs when two neurodes contribute to a correct response, and the weight between the neurodes is increased. Back Propagation is a supervised learning technique independently developed by Rumelhart Hinton, and Williams (1985) and Parker and Werbos (1985). Back propagation could be thought of as an extension of reinforcement learning where in addition to information about how close it is to solving the task, the network is also given information about errors for use in adjusting the connections between layers of neurodes to improve performance.

Back propagation tends to be much faster at training multilayered neural networks than unsupervised or reinforcement methods. However, it is doubtful that the brain actually uses a form of back propagation in learning. And it seems to be on this point that there is a division in how neural networks should be used—either as a model of human cognitive function or as an engineering tool for statistical analysis.

There are many techniques that are being used in the Neural Network's "guessing" process. Back propagation is one of these techniques. In back propagation, both input factors and the actual results are presented to the network. A connection matrix, that describes the relationship between input and output variables, is iteratively adjusted until the predicted results are close to the historic results. Kimmel (1991) suggests that this type of supervised learning using back propagation can be used to predict almost any phenomenon provided there is enough input data linked to their corresponding results.

Least Mean Square Learning (Delta Rule) To overcome the requirement of using linearly separable data with perceptions, Widrow and Hoff (1960) developed an algorithm to describe how weights should be updated. Their algorithm is now sometimes called the Least Mean Square rule, and sometimes called the Delta Rule. The classic Least Mean Square method does not limit the output to being either 0 or 1; the activated output may fall in a continuous range between 0 and 1.

$$\text{Output} = \sum_i w_i I_i + Q > 0 \quad (2-23) \quad \text{where: } w_i = \text{weight for Input } i$$

$$I_i = \text{Input of Characteristic for case } i$$

$$Q = \text{Bias}$$

Instead of using an error-correcting procedure where error correction is based on the absolute correctness of the calculated output to the true output, the Least Mean Square method tries to minimize the difference between the calculated output with the true output.

$$\text{Error} = \sum_p (T_p - Y_p)^2 \quad (2-24) \quad \text{where: } T = \text{true or correct answer}$$

$$Y = \text{perceptron output}$$

$$p = \text{specific input pattern}$$

The goal of the Least Mean Square procedure is to minimize the average squared distance from the true answer to the calculated answer. The training procedure is the same as for the perceptron.

While both the perceptron and the Least Mean Square procedure attempt to derive a linear separator from the data, Least Mean Square does not focus on directly reducing the error rate. Instead it focuses on reducing the distance between the calculated output and the true answer. This means that the Least Mean Square does not require data to be linearly separable or mutually exclusive.

Cickocki and Unbehauen (1992) suggest that the proper choice of function depends on the specific applications and the distribution of error. The standard least squares criterion is optimal for a Gaussian distribution. With outliers, a reweighted least squares or least absolute value criterion can be used.

The iterative technique used to update weights is known as gradient descent. The weights are updated in the direction (positive or negative) that we expect to reduce error. A typical problem of the gradient descent method is that it may oscillate around minimums and not converge. Another problem is that there may be local minimums (not the true global minimum) in the error space, into which it may converge. Convergence to the true minimum is guaranteed only when a very small value for the learning rate (h) is used. The downside is that the smaller the learning rate, the longer it will take for convergence.

White (1990) has shown that hierarchical neural network with one hidden layer and one output, using back propagation has the statistical property of consistency which means that as network experience accumulates, the probability of network approximation errors tend toward zero. White

suggested that this property could be generalized to other more complex neural network designs. This meets the self-learning systems' fifth requirement of Repeatability discussed in the first chapter (page 18).

Back propagation. The development of multi-layered networks to solve the exclusive/or problem also lead to a generalized form of Least Mean Square training called back propagation. The method was first presented by Rumelhart, Hinton, and Williams (1986a) in a book on Parallel Distributed Processing, and later in a paper published in Nature (1986b). They noted that the problem with the classic perceptron was that, while there could be feature analyzers between the input and output units, they were not true hidden units because their input connections were fixed by hand. Therefore, no learning could take place.

With single layer perceptron networks, once the weighted sum was computed, the activation of the unit was discrete and determined by the neurode's bias value—if the sum exceeded the threshold, the output was 1, otherwise the output was 0. With single layer networks using Least Mean Square, activation still required a positive number, but the output was continuous and linear. Multi-layered networks allow for the modeling of non-linear phenomena, and provide for multiple inputs of neurodes (Figure 10). To take full advantage of a multi-layered network's capabilities, the neurode's activation (also called “transfer”) function needs to be a non-linear, continuously differentiable function. A commonly used activation function is a logistic model. A logistic activation function would be expressed as in the equations below.

$$N_j = \sum w_{ij} Y_i + Q_j \quad (2-25) \quad \text{where: } w_{ij} = \text{weight for Input } ij$$

$$Y_j = \frac{1}{1 + e^{-N_j}} \quad (2-26) \quad \begin{array}{l} Q_j = \text{Bias of neurode } j \\ Y_j = \text{output (activation) of neurode} \\ N_j = \text{Net Input, the sum of all inputs to neurode } j \end{array}$$

Basically, the input value to the neurode is the sum of the weighted outputs from the neurodes on the lower level, plus the bias factor. For cases using a logistic activation (or transfer) function, the output of the neurode is computed by applying a logistic function to the net input value.

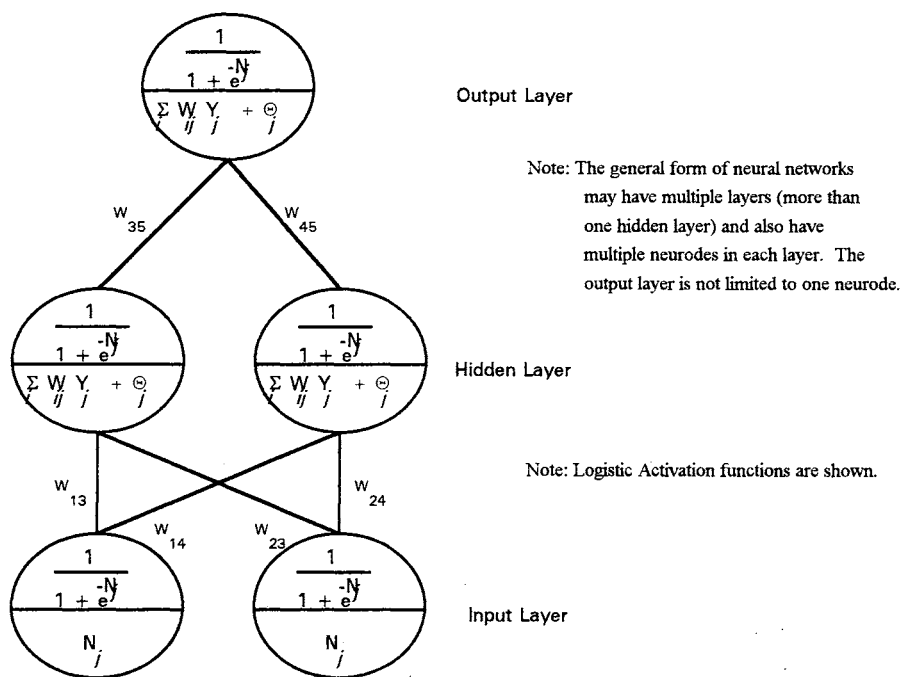


Figure 10. Two Layer Neural Network for the X/OR Problem

The training procedure involves making adjustments to the weights (W_{ij}) connecting each neurode to the bias (Q_j) of each neurode in an attempt to minimize global error. In back propagation, these adjustments are based on a proportional the product of the learning rate (h),

$$\text{Outer}_j = Y_j (1 - Y_j) (T_j - Y_j) \quad (2-27) \quad \text{where:} \quad \begin{array}{l} Y_j = \text{Calculated output for } j \\ T_j = \text{True output for } j \end{array}$$

$$\text{Errder}_j = Y_j (1 - Y_j) (T_j - Y_j) \left(\sum_k \text{Errder}_k w_{jk} \right) \quad (2-28) \quad \begin{array}{l} w_{jk} = \text{weights from inputs to } j \\ \text{Outer} = \text{Output unit error derivative} \end{array}$$

$$Dw_{ij}(t+1) = w_{ij}(t) + h * \text{Errder}_j * Y_j \quad (2-29) \quad \begin{array}{l} \text{Errder} = \text{Hidden unit error derivative} \\ h = \text{Learning Rate Constant} \end{array}$$

$$DQ_j(t=1) = Q_j + h * \text{Errder}_j \quad (2-30) \quad Q_j = \text{Bias}$$

an error derivative, and the net input to the neurode. The calculation of the error derivative is based on the difference between the calculated and true values, and differs slightly for the output neurodes and the hidden neurodes.

Back propagation is a recursive, two step procedure that provides a form of feedback to the network during the learning phase. In the first step, all current weights are propagated (or used to

calculate neurode outputs) from the input layer, through the hidden layer(s) and to the output layer. In the second step, working backwards, the error derivatives are calculated and adjustments are made to the interconnecting weights and the neurodes' bias.

In the classic back propagation model of Rumelhart, Hinton, and Williams (1986b), direct connections between the input and output layers were forbidden. There could be any number of intermediate layers, which may or may not be fully connected. The calculations for units within a layer were determined in parallel, with calculations for different layers set sequentially, starting at the bottom and working upwards. Weights were adjusted to minimize total error using a gradient descent algorithm.

This type of learning is also known as the Delta Rule. Popular extensions of the Delta Rule as used by the NeuralWare Neural Network computer program include the Cumulative Delta Rule (which accumulates weight changes over several examples) and the Normalized Cumulative Delta Rule (Klimasauskas, 1991a). A third extension, the Delta-Bar-Delta is discussed in a later section of this paper.

Gori and Tesi (1992) showed that classic gradient descent back propagation (weights updated in a batch mode) leads to an absolute minimum if the network has only one hidden layer, if the output layer has as many neurodes as there are classes in the population, and if the network is fully connected.

Webb and Lowe (1990) suggest that the discriminatory ability of hierarchical neural networks is partly due to the activation function of the feed-forward network performing a specific nonlinear transformation of the input data in a way that should make discrimination easier.

Caudill (1992) notes that back propagation is the preferred method used in neural networks. She speculates that there are two reasons for this preference. First, back propagation is based on least mean square error techniques, a technique with which many people are already familiar and which allow for relatively simple mathematical proofs. The other advantage is that it allows for supervised learning, which means that the designer can dictate results, and measure the network's performance against the desired results.

Cichocko and Unbehauen (1992) suggest that the standard least squares criterion is optimal for a Gaussian distribution of the noise, it is noted that the assumption of a Gaussian error distribution is frequently unrealistic because data errors usually are not isolated to a single source. There may be instrument errors, modeling errors, sampling errors and human errors. Additionally, data may contain large errors (outliers) or wild (spiky) noise. In order to reduce the influence of the outliers, the more robust iteratively reweighted least squares technique can be used. In the presence of outliers (and/or wild noise) an alternative approach is to use the least absolute value criterion.

Learning Rates (h). As a network goes through its learning cycles, intuitively we would expect the perceptron to get closer than answer and to make less drastic revisions to weights. The learning rate parameter (h) is a correction factor that modifies calculated weight corrections such that $(h) * (Dw_i(t))$. The selection of a learning rate will have significant impact on the neural network. With back propagation, on one hand, the smaller the learning rate, the slower the progress in finding an optimum solution. On the other hand, the larger the learning rate, the more oscillations there are and the greater the possibility of skipping over a minimum solution. One popular form for the learning rate (h) is to make it inversely proportional to the time spent on learning (1/t). Weiss and Kulikowski (1991, p 101) suggest a learning rate of $h = 0.5$ as being typical.

Kung and Hwang (1988) in applying the algebraic projection analysis to back propagation, developed a formula to estimate the optimum learning rate:

$$\eta = \frac{72}{P + 1} \quad (2-31) \quad \text{where: } P = \text{the number of hidden neurodes}$$

The estimation of the learning rate assumes that the algebraic projection relaxation parameter (λ) is set to one.

In analyzing the effect of the learning rate Ratcliff (1990) trained a neural network with four vectors, A, B, C, and D using various learning rates and trials. Ratcliff found that for low learning rates and few learning trials, the system learns to reproduce an average or prototype of the four cases. With large number of trials and/or high learning rates, the network trained to one of the four patterns not an

average prototype. Ratcliff also found that high learning rates gave rise to extreme values of activation values (near either 0 or 1).

Tollenaere (1990) suggests that there is an optimal learning rate for which learning is fast and the procedure remains stable. for a 10-10-10 network $h = .25$ has been found to be optimal, for a 10-5-2 network, $h = 1.0$ has found to be optimal.

Reyneri and Filippi (1991) suggest an algorithm that may be used to find the optimal learning parameter that is based on the number of inputs, and may be different for various layers of the network.

$$\eta_k = \frac{2\rho^2(N_k + 1)}{y_{\max}^2 x_{\max}^2} \quad (2-32)$$

where: η = learning rate
 ρ = normalized steepness of activation function
 N = number of input units
 x_{\max} = maximum input value to any neurode in layer
 y_{\max} = maximum output of any neurode in layer

Reyneri and Filippi (1991) note that the performance of back propagation depends heavily on the value of the learning parameter, especially when inputs have varying dimensions. Almeida and Silva (1990) note that continuous adaptation of the value of the learning rate during the learning phase usually increases performance.

Momentum (∂) Momentum is a mathematical term added to the weight update algorithm that includes a percentage of the previous weight adjustment in the current weight adjustment. Back propagation tries to minimize the mean squared error of the system by using a technique called gradient descent to move down an error curve. The error curve of a data set is not necessarily smooth, and may contain minor peaks and valleys into which the system may become trapped. Momentum is a mathematical device used to keep the network moving on the downhill error surface, even when localized minimums are reached.

When momentum is added to the learning procedure, weights are revised by combining part of the indicated new weight revision, $Dw_{ij}(t+1)$, with part of the previous weight revision $Dw_{ij}(t)$. This makes the equation similar to an exponential smoothing form of statistical forecasting. Note that a momentum of $\partial = 0$ will eliminate the term from the equation. Weiss and Kulikowski (1991, p 101)

suggest a momentum term of $\partial = 0.9$ as being typical.

$$Dw_{ij}(t+1) = w_{ij}(t) + (h * \text{Errder}_j * Y_j) + (\partial * Dw_{ij}(t)) \quad (2-33)$$

Because of the learning algorithm used in back propagation, it is possible for the total error of a training set to stop decreasing and stalls at some value higher than the acceptable level. The network is then said to be stuck in a local minimum. Caudill and Butler (1992a) provide an explanation on how a momentum term helps avoid local minimums:

To see the effect of the momentum term on the weight change, suppose the first term suddenly became zero. In that case, the new weight change is just a times the previous weight change. In other words, the weight vector continues to move in the same direction as it moved in the last time it was changed, so the momentum term does act just like a physical momentum in that it tends to keep the weight vector moving in the same direction unless forcefully nudged in a different direction. This means that... the change to the weight vector has a term that tends to keep the weight vector moving so that it does not get trapped easily in a local minimum.

(pp. 197-198)

Tollenaere (1990) suggests that there is an inverse relationship between an optimum learning rate and momentum. Some have reported not being able to train networks with momentum factors (∂) > 0.5, while others report success with $\partial = 0.9$. Tollenaere suggests that the contradictory claims could be explained by the different learning rates used. The following suggestions concerning momentum are made:

- * the optimum learning rate (h) decreases as momentum (∂) increases.
- * using little momentum results in a wider distribution of learning times
- * the use of momentum speeds up learning—in most cases a speedup factor of 2 to 3 can be expected
- * the use of very high momentum does not result in instability, provided that step sizes are sufficiently small.

Networks without hidden layers tend to use the delta rule with a momentum term. Networks with hidden layers tend to use back propagation (Vogl, Mangis, Rigler, Zink and Alkin, 1988).

Delta-Bar-Delta Rule. A modification to Rumelhart et al. (1986a) back propagation rule was developed by Jacobs (1988), called the delta-bar-delta rule. Under Rumelhart's model, weights were updated layer by layer, with each neurode in a layer obtaining the same weight modification. Jacobs proposed a series of heuristics for increasing the rate of convergence

- * every weight should have its own individual step size
- * step sizes should be allowed to vary over time to accommodate error surfaces that have different properties along different regions
- * when the derivative of a weight possesses the same sign for consecutive steps, the learning rate for the weight should be increased
- * when the derivative of a weight possesses a different sign for consecutive steps, the learning rate for the weight should be decreased

Tveter (1991) describes the basic implications of the Delta-Bar-Delta rule on the learning rate, and also describes minor changes to the weight change and output layer activation function.

Learning Rate (η) is modified when slope of error function is:

a) negative for two iterations $\eta = \eta + \kappa$ (2-34)

where: ϕ = decay rate (between zero & one)

κ = increase constant

b) positive for one iteration $\eta = \eta * \phi$ (2-35)

η = learning rate

The effect of Jacobs' delta-bar-delta rule is that learning rates decrease exponentially when it appears that a minimum error region has been reached (positive slope of error function), and increase linearly when one is on a steeper region of the error space and learning is still taking place (negative slope of error function). Using this rule, the probability of skipping over a global minimum is significantly decreased.

Tveter (1991) reports some success with the Delta-Bar-Delta rule. However, he also reports that when using the X/OR problem as a benchmark, the technique provided the worst results when compared with standard back propagation.

Devos and Orban (1988) independently developed a self-adapting back propagation strategy that conforms to Jacobs' delta-bar-delta rule. Tollenaere (1990) added momentum to the self-adapting back propagation strategy and called it SuperSAB.

Refinements to Back Propagation

Three refinements to traditional back propagation are presented. Cascading Neural Networks represent an attempt at applying systems-analysis problem solving techniques to neural network formulation (Hillman, 1991). Probabilistic neural networks basically look at using *a priori* knowledge in the construction of a network. Fuzzy neural networks use Fuzzy logic, in which the *law of the excluded middle* is relaxed.

Probabilistic Neural Networks

The probabilistic neural network (Figure 11) is a partially connected neural network, first designed by Specht (1990), that attempts to approximate Bayesian decisions under conditions of imperfectly known probabilities. The probabilistic neural network categorizes patterns by estimating their probability distribution functions and classifies patterns in a way that minimizes expected risk. The probabilistic neural network consists of four layers. The first layer (the input layer) is fully connected to the next layer and distributes patterns to every neurode in that layer. The second layer (the pattern layer) contains one neurode for each pattern represented in the training set and is fully connected to the input layer. The pattern layer neurodes performs a weighted sum of incoming signals and applies a nonlinear activation to determine the neurodes output. The pattern layer is not fully connected to the third layer, sending only one signal to that layer, and weights connecting the pattern layer to the next layer are fixed at 1.0. The third layer (the summation layer) contains as many neurodes as there are categories and merely adds the outputs from all the pattern layer neurodes. The pattern neurodes are connected to their corresponding category neurode in the summation layer. The fourth layer (the output layer) receives totals from the summation layer and generates the network's category choice.

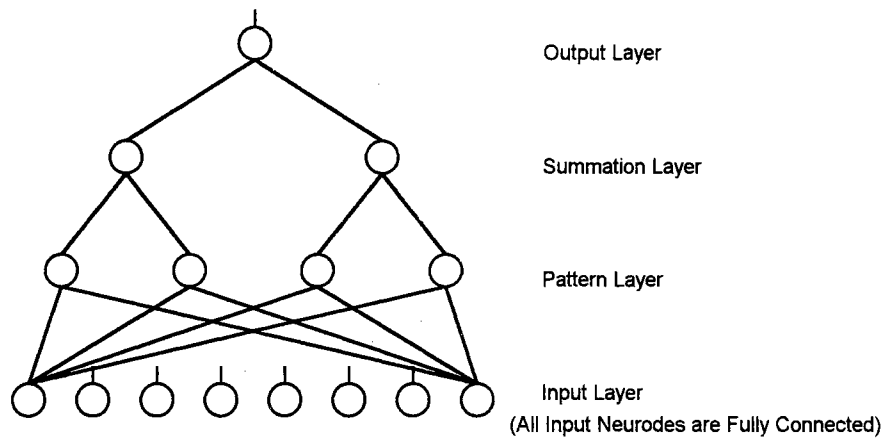


Figure 11. Probabilistic Neural Network

The probabilistic neural network uses an exponential activation (transfer) function rather than the traditional sigmoid activation function used by back propagation. 2-49

$$y_j = \exp \left[\frac{\sum x_j - 1}{s^2} \right] \quad (2-48)$$

where: y = output

x_j = net weighted input to neurode

s = smoothing constant (establishes the non-linear shape of the decision surface) typically between 0.5 and 10.0

"The reason this exponential activation function is used is that it is a simplification of an estimator of a Bayesian decision surface. Using a Bayesian estimator function in the pattern layer neurodes of the probabilistic neural network provides the network with a near-Bayesian performance in categorizing patterns." (Caudill and Baker, 1992, p 223)

The activation (transfer) function in the output layer contains the *a priori* likelihood of a pattern being classified in a particular category, and the loss associated with misidentifying a pattern.

Fuzzy Neural Networks

A discussion of Fuzzy Neural Networks needs to begin with an understanding of its underlying conceptual foundation, Fuzzy Logic. In a changing environment, Fuzzy Logic can be used to get a further handle on uncertainty. Kosko and Isaka (1993) note that traditional digital computers manipulate precise facts that are either true or false. Humans, on the other hand, can "reason" using uncertainties or value judgments (i. e., too cold or very beautiful). Fuzzy logic helps computers manipulate uncertain information in situations that can not be described by specific algorithms by using commonsense rules. Where possible, rules will be described by experts. However, when no expert can give rules, fuzzy logic systems learn rules by observing how the systems are regulated.

When using traditional logic, the *law of the excluded middle* applies—an object either belongs to a set or it does not belong to a set; there is no middle or gray area, and an object can not belong to both sets. However, with fuzzy logic, the law of the excluded middle is broken to some degree—items may belong to more than one set. Were the boundaries of the standard sets are exact, the boundaries of a fuzzy set are curved, and this curvature may create partial contradictions. For example, for categories of warm or cold, a temperature of F55 may be classified in one or the other, depending on the person, the location of the temperature, or the season of the year. To a person in San Diego, F55 would probably seem cold any time of the year. However, to a person in Nome, Alaska, a temperature of F55 might seem very warm in January. Note that there is a certain probability that the F55 would be classified in one or the other categories, and that its classification is not based on the number itself, but how a person perceives the temperature measurement. Because people perceive things through their experience, and in context, different groups of people will categorize measurements differently. Nevertheless, the probability that a measurement will be categorized in a specific category can be determined. Then, a fuzzy set contains not members, but probabilities.

Fuzzy logic is then useful whenever measurements are imprecise, or their interpretation depends on context or opinion. Fuzzy logic is a different way of expressing reality that uses perception in the place of objective measurements. Fuzzy logic conveys a different type of information than has been traditionally used in establishing the foundations for many logical and mathematical procedures.

Fuzzy logic looks at classifying patterns, not in terms of certainties, but in terms of probabilities. The membership of a fuzzy set depends, not on the actual pattern being classified, but the pattern's relationship with its environment (on context or on human opinion). Fuzzy sets uses not objective measurements, but seeks to capture human perceptions. With fuzzy logic, the law of the excluded middle is broken to some degree, allowing items to belong to more than one set.

Fuzzy logic is typically applied to neural networks in one of two ways. Input may be preprocessed with a "fuzzifier" or output may be converted into a series of fuzzy categories using probabilities.

Cascading Neural Networks (Systems-Analysis)

A variation on the approach of starting with zero hidden units, called Cascade Learning, was developed in 1990 by Scott Fahlman and Christian Lebiere of Carnegie Mellon University (NeuralWare, 1993, p. NC-74). The model consists of logistic activation functions in the interior neurodes and linear activated output neurodes (Kempka, 1994b).

A Cascade neural network (Figure 12) uses an algorithm that both accomplishes training and also dynamically builds the network architecture. Initially, only the number of input and output units are specified, there are no hidden units. During training, hidden units are added layer by layer, with a single hidden unit in each layer. The inputs are directly connected to the outputs, initially without hidden nodes. After each iteration, a hidden node is added to the network with each hidden node receiving inputs from all prior hidden nodes as well as all input nodes.

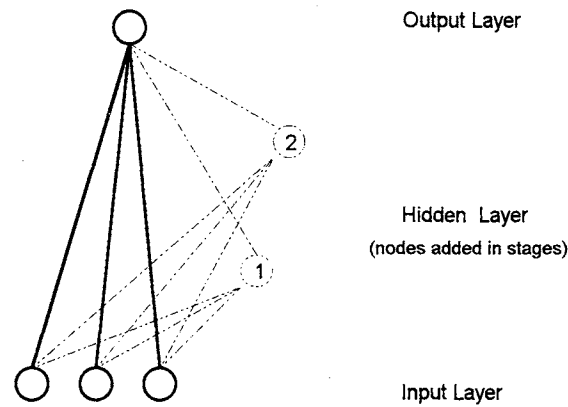


Figure 12. Cascade Neural Network

With each hidden unit added, the network is retrained in a two phase process. In the first phase, only the new hidden unit is trained in such a way that its output covaries with the errors made by the current network. Fahlman's (1988) quickprop algorithm is used in this phase. Once accomplished, the new hidden unit's weights are frozen. In the second phase, the all weights to the output node (including the weight from the new hidden unit) are optimized to maximize the correlation of its output with the error signal previously existing in the network (or minimize network error) according to the formula:

$$C = S | S (y_p - y) (e_{op} - e_o) | \quad (2-50)$$

Because the new hidden node is trained to covary with the existing network's error, its addition should decrease total network error. Gradient ascent, rather than descent, is used because the objective is to maximize covariance to find the optimal hidden node weights. (Smith, 1993)

Hillman (1991) expanded the concept of cascade learning at the network level to a system of problem solving that uses neural networks as a tool for one of the problem solving steps. While traditional neural networks are useful in classification and process control type problems, suggests that this approach may prove limiting in systems-analysis problems. Hillman defined systems-analysis problems as being those that attempt to extrapolate conclusions that attempt to describe the nature or

behavior of the total system from a limited set of data characterized as what, who, where, when and how. Hillman goes on further:

Many systems-analysis problems try to derive a “big picture” from a relatively small set of data, which might tell you who is involved, when and where it happened, what kind it is, or how it took place. It is up to the observer to “fit” the data into some pattern that makes sense. This process is often done by determining what parts or who is available and fitting the parts together to determine function at subsystem and system levels (p 47).

When analyzing a problem from a systems basis, Hillman describes three steps: data collection (gathering facts), assessment (map what is known about a problem to a solution), and augmentation (identify data that would increase confidence in the assessment). Data collection is accomplished by traditional methods. Augmentation is typically based on rule-based systems that are designed to identify data based on belief values, correlating the proposed solution with other data that may improve the solution’s validity. Assessment lends itself to the use of multiple cascading neural networks of varying degrees of complexity.

Because knowledge domains tend to be large and complex, they are often decomposed into smaller subcomponents that are easier to manage. The process is similar to large expert systems using subgoals within rule subsets, or using the concept of children or in inheritance hierarchies. When using cascading neural networks, the problem is divided into relatively self-contained subsystems that are easier to model and test. Once the subsystem are trained, the larger system is then tested. Outputs can then be traced into other networks.

When deciding whether to use the systems approach of cascading neural networks, Hillman (1991, p 50) offers the following rules:

Rule 1: If the entire input and output data set represents similar kinds of data such as a signal intensity level, a single neural network representation may be adequate.

Rule 2: If input and output data represent a domain’s physically or functionally discernible parts, consider breaking the larger network into two or more networks... according to the following criteria:

- * The domain knowledge should decompose into logical, subordinate chunks of knowledge

- * Each chunk should be complex (such as three or more neural-net input variables mapped to one or more output)

- * The mapping of a single network (inputs to outputs) is based solely on that network

The integration of the subsystems requires neural network software capable of performing the tasks, bridges between the subnetworks in the cascade, and a control mechanism to manage the execution of neural networks to process data. Hillman (1991) suggests using expert systems for pre- and post-processing data for the neural networks.

Smith (1993) suggests that cascade networks train very rapidly. However, the covariance function tends to have many local minima and the “network is inclined to produce input-output mapping with very sharp transitions or boundaries” (p 188). Smith concludes that Cascade networks are better at discriminating between classes that do not overlap rather than for estimating continuously varying quantities.

CHAPTER III

NON-PARAMETRIC STATISTICS AND BACK PROPAGATION

The Review of Literature is divided into several basic components: a review of the Evolution of Neural Networks, a review of Non-Parametric Statistical Concepts, Traditional Statistical Classification Methods and their Relation to Neural Networks, a review of generalized Neural Network Learning techniques and specific back propagation based techniques that could be used in data analysis, and a review existing literature describing how data should be prepared for use with back propagation neural networks.

Neural Networks, Non-Parametric and Probability Concepts

The following presents statistical concepts, and shows how back propagation neural networks relate to the statistical concepts. "Statistics is the science comprising rules and procedures for collecting, describing, analyzing, and interpreting numerical data" (Kvanli, Guynes, and Pavur, 1989, p 1).

Inferential and Descriptive Statistics

Statistics can be divided into two parts: descriptive statistics and inferential statistics. The process of collecting and describing sample data is called descriptive statistics. The process of drawing conclusions about a population based on the results of a sample is called inferential statistics. The basic measures that describe data are the measures of central tendency (mean, median, midrange, and mode), measures of dispersion (range, mean absolute deviation, variance, standard deviation, and coefficient of variation), measures of position (percentile, Z score, skewness, and kurtosis), measures of relation (coefficient of correlation, rank-order coefficient of correlation, coefficient of multiple

correlation). One infers based on an analysis of differences or significance (F statistic, T statistic, analysis of variance), multivariate analysis (multiple regression, canonical correlation, discriminant analysis, factor analysis, path analysis, analysis of covariance structures).

In other words, statistics is concerned with the analysis of data. From a neural network standpoint, the concept of statistical inference can be translated to means learning to generalize from a noisy set of data (Prechelt, 1995). While those neural networks designed to model biological systems have little to do with statistics, most neural networks can learn to generalize noisy data in ways that are similar to statistical methods.

Standard Error The overall objective of a Neural Network is to modify its internal weights to represent relationships learned from samples, and generalize those relationships to new cases. The performance of a Neural Network is usually measured in terms of the error rate, or the ratio of the number of errors to the number of cases. Weiss and Kulikowski (1991) define the *true error rate* as "the error rate of the classifier [neural network] on an asymptotically large number of new cases that converge in the limit to the actual population distribution" (p 17). The true error rate may be estimated by the error rate calculated from the sample.

$$E = \frac{\text{number of errors}}{n} \quad (3-1) \quad \text{where: } E = \text{error rate} \\ n = \text{number of cases}$$

The error rate described above can also be thought of as an apparent error rate, based on the sample cases. To assure that the apparent error rate is an approximation of the true error rate, cases should be selected randomly. This means that cases "should not be preselected in any way. This means that the investigator should not make any decisions about selecting representative samples" (p 26). However Hommertzheim et. al. (1991) note that a neural network needs to have a sufficient number of cases identifying boundary conditions for learning to occur.

The apparent error rate may or may not be representative of the true error rate of the total population. One reason why the apparent and true error rates may be different is that in training the neural network to classify sample data with few errors, the relationships developed may be too specific to the sample data, and may not be generalizable to the total population. In these cases, the neural

network is said to have been overfitted, overspecialized or overtrained to the training cases, and the error rate of the samples is not a good approximator of the true error rate.

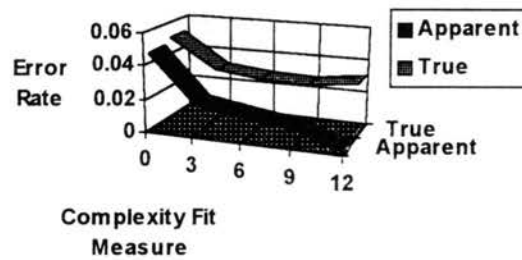
Another reason why the apparent and true error rates may be different is that there are too few sample sets of cases to provide a good representation of the total population. The accuracy of the estimated error rate in estimating the true error rate is based on the number of cases in the sample in relation to the total population. A measure of how far the error rate will deviate from the true error rate is the Standard Error or Standard Deviation. The standard error is derived from the binomial distribution, and can be expressed as:

$$SE = \sqrt{\frac{E(1-E)}{n}} \quad (3-2) \quad \text{Where: SE = Standard Error}$$

In the above equation, E is the error rate based on n randomly drawn independent test cases. We shall see that neural networks learn how to classify cases by dividing a population sample into test sets and training sets.

Error rates are useful in estimating the performance of neural networks on future data, assuming that the samples used to train the network will be representative of future samples. Error rate estimation techniques are also useful in optimizing learning systems by providing an objective basis of comparison between neural networks using different structures. However, while one may be tempted to choose a neural network that has the smallest error rate without overtraining the network to the data, one should also remember that the more complex the learning system design, the more degrees of freedom there are to contend with, and the more divergence between the apparent and true errors. In Figure 13, when we reach a complexity measure of 12, we are overtraining the network.

Sarel (1994) suggests that for linear models, the MSE (mean squared error, defined as the ratio of the Sum of the Squared Errors and the Degrees of Freedom for Error) is an unbiased estimate of error variance. He also suggests that for nonlinear models, the MSE is biased, but not as biased as the average squared error.



Source: Weiss and Kulikowski, 1991, p 42

Figure 13. Complexity vs. Classifier Fit

The Coefficient of Determination (Pearson's R Coefficient) or R^2 statistic is often used to assess the accuracy of a model's prediction. The statistic is basically a ratio of explained variation and total variation. When applied to neural networks, Burke (1993) uses the following formula:

$$R^2 = 1 - \frac{(n - 1) \text{SEE}}{(n - p - 1) \text{SST}} \quad (3-3)$$

Where: SSE = Sum of the Squared Errors
 SST = Sum of the Squared Deviations from the Mean
 n = Number of Training Set Examples
 p = number of input nodes (parameters)
 (or number of independent variables)
 $0 \leq R^2 \leq 1$

Correlation Coefficient. The process of postulating relationships between variables begins with calculating correlation coefficients for variables to give an indication of the strength of the relationship between them. Correlation plots may be used to give insight to the type of relationship (quadratic, exponential, etc.) High correlations between a rejected input variable and a model's errors may indicate additional explanatory power in the variable. When there are two highly correlated variables, it is often desirable to eliminate one of them, thereby reducing the weights of the neural network.

The correlation coefficient (r) measures strength of the relationships between two variables;

which can be shown by

$$\rho = \frac{\sum \left[\frac{(X - \mu_x)}{\sigma_x} \left(\frac{Y - \mu_y}{\sigma_y} \right) \right]}{n} \quad (3-4)$$

where: X = independent variable observation
 Y = dependent variable observation
 n = number of observations (sample size)
 μ = mean of observations
 σ = standard deviation of observations

While correlation is not causation, and a non linear relationship may exist between items.

Covariance If two (or more) independent variables are correlated with each other, the predictability of the model is brought into question. Correlated inputs make models (either neural network or traditional statistical models) more sensitive to any statistical peculiarities of the sample data. Testing covariance using neural networks is problematic because back propagation is able to model complex relationships between variables. “The only way to measure the contribution that an additional variable would make—beyond the contribution of a previously selected set of variables—is to train two networks, one with and one without the new variable” (Smith, 1993, p 144). The basic tool for examining covariation between two variables is crosstabulation. The chi-square test, which is based on crosstabulation, is a useful statistical tool, but the level of significance is set arbitrarily.

t-Statistic A *t*-test is used for many things, among them to compare the means of two groups. If two sample means are far enough apart, the *t*-test will yield a significant difference, suggesting that the populations from which the samples were taken probably have different means. From a neural network standpoint, *t*-tests have been used to estimate whether or not a variable should be included in a network. Least squares regression can be used to test linear significance of a candidate variable. Stein (1993) suggests that any variable with a student's *t*-statistic exceeding 1.98 indicate that the variable is significant for large samples.

Probability

When focusing on inferential statistics, one typically wishes to infer something about a population based on a sample of that population. To make an inference that the sample is

representative of the population, we use probability to describe the accuracy of the sample results.

When we deal with uncertainty, we use probability to help describe that uncertainty.

Kerlinger (1986) suggests that there are two broad definitions of probability, *a priori* and *posteriori*, based on the knowledge one initially has concerning the population. The *a priori* definition, first formulated by Simon Laplace around 1795, defined the probability of an event as "the number of favorable cases divided by the total number of (equally possible) cases" (Kerlinger, 1986, p 90), and assumed that we can determine the probabilities of events before the event is empirically analyzed. The posterior definition suggests that "in an actual series of tests, probability is the ratio of the number of times an event occurs to the total number of trials" (p 90), and suggests that probability can be determined only after an event is empirically analyzed. Kerlinger suggests that both definitions of probability complement each other and both are needed in the pursuit of knowledge.

Using either *a priori* and *posteriori* definitions of probability, to calculate the probability of any outcome, one needs to first determine the total number of possible outcomes. The *sample space* is the set of all possible outcomes. Sample space under an *a priori* assumption is the universe of all possible outcomes. Sample space under a *posteriori* assumption includes the possible outcomes of an experiment being analyzed. The components of a sample space are called *elements* or *sample points*.

Probability is always a non-negative number, and the probability of all points in a sample space must add up to one. In other words, the probability for the sample space is one, and the probabilities for the various classes or events that make up the sample space add up to one. Probability is expressed as a weight that is assigned to each class or event that make up the sample space, and must add up to one.

Sometimes we may wish to study the relation between two separate events, we wish to know the probability of two separate events occurring at the same time. "A compound event is the co-occurrence of two or more single (or compound) events" (Kerlinger, 1986, p 96) In analyzing the relationship between separate events, we may wonder if the occurrence of event A precludes or otherwise influences the occurrence of event B, or if they are unrelated? If events are exhaustive, then

the occurrence of each event uses up all points in the sample space. Events are mutually exclusive when they are disjoint, or when a specific event has no points in common with other events.

$$p(A \cup B) = p(A) + p(B) \quad (3-5) \quad \text{where: } p(A \cup B) = \text{probability of either A or B}$$

Independent Events Events are said to be *independent* when the probability of events occurring at the same time is equal to the probability of one event times the probability of other events.

$$p(A \cap B) = p(A) * p(B) \quad (3-6) \quad \text{where: } p(A \cap B) = \text{probability of A and B}$$

In much of the world, the relationship between events is not statistically independent. Much research in social sciences is based on finding the causal relationships between events. When events are not independent of each other, the analysis of probability must compensate for this bias.

Conditional probability is a term used to describe probabilities that are conditional on prior knowledge or facts. The formula for conditional probability involving two events is:

$$p(A|B) = \frac{p(A \cap B)}{p(B)} \quad (3-7) \quad \text{where: } p(A|B) = \text{probability of A given B}$$

In effect, conditional probability alters the sample space. Through knowledge, the sample space has been decreased to the space of event B. With independence, the conditional probability of A given B is the probability of A. Without independence, the relationship becomes more complex.

When dealing with *a priori* probability assumptions, independence requires that the sample being used is representative of the total population. Specifically, from a research standpoint, representative means "that the sample has approximately the characteristics of the population relevant to the research in question" (Kerlinger, 1986, p 111).

Linear Independence Linear independence can be described as a situation in which there is no combination of positive or negative real numbers that any two vectors (a one-dimensional array) can be multiplied by to produce a third vector. In other words, any set of vectors for which at least one cannot be written as a linear combination of the others is linearly independent. In one, two, and three dimensional space, vectors that form linearly independent sets lie at right angles to one another. Because we have a hard time visualizing in more than three dimensions, a more general term is needed to convey the concept of right angle. In spaces of any dimensionally, linearly independent vectors are

said to be orthogonal. The concept of the inner, or dot product of two vectors provides a natural way to define orthogonality.

Representative Sample A technique that helps in the assurance that a sample is representative of a population is called random sampling. *Random sampling* is a method of drawing a sample of a population such that all possible samples have the same probability of being selected. (Kerlinger, 1986, p 110) There are two variations on random sampling. One variation is *sampling with replacement* where the selected item is put back in the population and may be selected again. The other variation is *sampling without replacement*, where an item may be selected only once.

Representative Population Size A primary question to be answered is how large does a sample need to be to be assured that the sample is representative of the total population.

The *central limit theorem* says that when using variables drawn randomly from a population, the resulting sample mean has a normal population distribution with a mean of m and standard deviation of s / \sqrt{n} . "This is true for any sample size, n , if the underlying population is normally distributed, and is approximately true for large sample sizes (generally $n > 30$) obtained from any population" (Kvanli, Guynes, and Pavur, 1989, p 212).

Degrees of Freedom

When analyzing data, one needs to be cognizant of bias that may be introduced as a part of the analysis. Hanke and Reitsch (1992) define the term *degrees of freedom* by noting that it is used to indicate the number of data items that are free of each other in the sense that they cannot be deduced from each other and can therefore carry unique pieces of information. For example, suppose the following four statements are made:

- | | |
|---------------------------------|------------------------------|
| * I am thinking of the number 5 | * The sum of the two numbers |
| * I am thinking of the number 7 | I am thinking of is 12 |

At first glance there are three pieces of information presented here. However, if any two of these three statements are known, the other one can be deduced. It could be said that there

are only two unique pieces of information in the three statements above, or to use the statistical term, there are only two degrees of freedom. (p 15)

When studying a sample of the total population, a calculation of means and standard deviations may be biased to the extent that the sample mean and standard deviation is different from the total population mean and standard deviation. The degrees of freedom may be used to help adjust for that situation.

Sampling Techniques

When the available sample size is limited, resampling methods can be used to provide better estimation of the true error rate. We will discuss three related techniques that may be used, depending on the sample size.

Train-and-Test Sample Size The requirements for any unbiased sample error is that the sample data are drawn randomly from the parent population. Because one of our objectives with neural networks is to determine classification parameters based on test cases, and use the parameters to classify new cases. If the samples are biased, or not representative of the total population, one will not be able to generalize the sample based parameters to the general population.

An additional complication is brought about because many times we do not have an unlimited population from which to draw our samples. In fact, in the real world, one usually uses a sample from a single population, not a sample from all possible populations. Additionally, to test the proficiency of the classification scheme developed by the neural network, we need to divide the sample into two different sets, one for training the network and the other for testing. The training set is used to design the classification parameters. We can assume that the cases in the test set will be representative of new cases, and can therefore use the test cases as a measure of the sample error. The error rate of the trained neural network on the test case is the test sample error rate.

Baum and Haussler (1989) have written concerning the relation of the network size to the number of training examples that need to be presented to assure statistical validity. They summarize their results in a rule of thumb that says the appropriate number of training examples is approximately the number of weights between nodes of the network, times the inverse of the accuracy parameter e .

For example, if we desired an accuracy level of 95%, which corresponds to $e = 0.05$, we would need 100/5 or 20 times as many training examples as there are weights in the network.

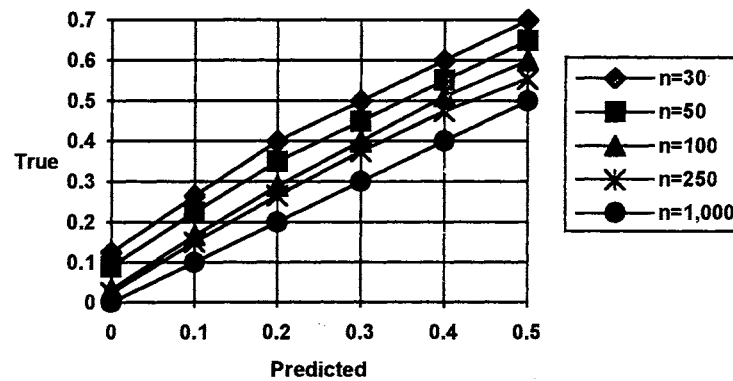
Smith (1993) suggests that “back propagation does not require a larger sample size than linear regression” (p 136). As a practical matter, as the sample size decreases the number of hidden nodes can be decreased. A network with no hidden nodes would approximate a linear regression model with “the same number of degrees of freedom, [and would be] no more inclined to overfit than the linear model” (p 136). The accuracy of both neural network and regression models is determined by the noise in the data and the extent to which the target function approximates the underlying causal relationships.

The test size used for the train-and-test paradigm needs to be large enough for the test cases to be a good estimator of the true error, and the training size needs to be large enough to provide the neural network with a representative sample of the population to ascertain classification criteria. The usual proportions are to divide the sample population, 2/3 into a training set and 1/3 into a testing set, provided that once the test cases exceed 1,000, a greater number can be allocated to the training set (Weiss and Kulikowski, 1991, p 30).

Based on Figure 14, the training set error approaches the actual error when the training set cases exceed 1,000. Based on a 2/3 training, 1/3 test partition, we would need 3,000 cases in our sample population for an unbiased error estimate.

Assuming that there are sufficient samples for statistical validation, Smith (1993) suggests that “on balance, increasing sample size does not increase training time. Larger samples require fewer training epochs” (p 136).

Leaving-One-Out A special case of Cross-Validation that is very computationally intensive. For a given sample size, n , the neural network is trained on $(n-1)$ cases, and tested on the remaining case. This process is repeated n times, each time using a different case for the test item. The error rate is the average of errors on the single test cases. Although Leave-One-Out can be used with sample sizes numbering in the 100s, because of its computationally intensity Weiss and Kulikowski (1991) suggest that should be used for sample sizes of less than 100 cases.



Source: Weiss and Kulikowski, 1991, p 29

Figure 14. Number of Test Cases Needed for Prediction

Cross-Validation In a k -fold Cross-Validation (Stone, 1974), cases are randomly divided into k mutually exclusive testing partitions of approximately equal size. For a given epoch of network training, a test partition is selected and the remaining cases are used for training the network. Once the train-and-test epoch is completed for one set of data, a different testing partition is selected as the test set. The new set of remaining cases is used for training the network. The process is repeated for k epochs. The average error rates over all k epochs are the cross-validated error rate. Weiss and Kulikowski (1991) suggest that a 10-fold cross-validation is adequate and accurate for most cases of greater than 100 cases.

Cross Validation has an additional advantage when considering the problem of overgeneralization. Cross validation may be used as a tool to indicate the optimum number of hidden nodes. A portion of the data is trained using networks with different numbers of hidden layers. Each network is run on a validation test data set. The alternative with the lowest test error is assumed to be the optimum alternative. Validation test errors sequences that do not increase are an indication that the original number of hidden units are insufficient for generalization to occur and that additional hidden nodes are needed.

Cross validation may also be used to indicate if training should be stopped before convergence is reached. Periodically the network is stopped and tested using a validation test data set. As the network learns, the error on the test set should decrease. When overgeneralization is said to occur when the test error increases.

There are no statistical proofs that cross-validation will produce optimal results. However, Smith (1993) notes that:

There is no guarantee that cross-validation will produce the optimal model. The smaller the validation sample and the higher the noise level, the more likely it is that cross-validation will fall short of this ideal. Despite this theoretical risk, cross-validation has been utilized for a long time in statistics and is a valuable practical tool. It is all the more valuable because there is little alternative (p 120).

Traditional Statistical Classification Methods and Their Relation to Neural Networks

Many neural networks can learn to generalize from noisy data in ways that are similar to traditional statistical methods. For example (Prechelt, 1995):

- Feedforward nets with no hidden layers are basically generalized linear models
- Feedforward nets with multiple hidden layer are closely related to nonlinear regression and discrimination models (the methods commonly used for fitting nonlinear models, such as Levenbert-Marquardt and conjugate gradient algorithms, can also be used to train feedforward neural networks.
- Probabilistic neural nets (discussed later) are identical to kernel discriminant analysis
- Kohonen nets for adaptive vector quantization are very similar to k-means cluster analysis
- Hebbian learning is closely related to component analysis
- Reinforcement learning is treated in the operations research literature as being similar to Markov decision processes

In order to understand how neural networks might be used as a statistical tool, we will first review selected statistical tools. Many of the operations of a back propagation neural network using gradient descent are similar to standard statistical methods. In this section, we will identify some of those similarities.

Moving Average - Exponential Smoothing

A moving average is a series of averages of n values, each average being placed at the chronological end (or at the chronological center for a centered moving average) of these consecutive values. Moving averages are often used to remove the effects of random variation on series data. In a weighted moving average, each element of the series is assigned a weight (typically the most recent items have the largest weight) in the averaging process.

Exponential Smoothing is a special case of a weighted moving average used in forecasting. It is a procedure for continually revising an estimate in light of the most recent experiences, and is of the form:

$$\text{New Forecast} = \alpha * (\text{new observation}) + (1 - \alpha) * (\text{Old Forecast}) \quad (3-8a) \quad \text{where: } \alpha = \text{smoothing constant } (0 < \alpha < 1)$$

$$\text{or} \\ = * (\text{Old Forecast}) + * [(\text{new observation}) - (\text{Old Forecast})] \quad (3-8b)$$

The old forecast can be thought of as a weighted average of all past observations with weights that decrease exponentially with the age of the data. The degree of decrease is determined by α , with values close to 1 decreasing more.

Component Models

The goal of the component model is to first decompose time series data by isolating the components that characterize the data. Next, future values are projected based on the individual components, assuming the components will repeat themselves in the future. Finally the individual component forecasts are combined for a forecast. The general mathematical representation for the component model is:

$$X_t = f(T_t, C_t, S_t, I_t, E_t) \quad (3-9)$$

X is the time series at period t. T is the trend component, describing long-term behavior. C is the cyclic component and may be defined as either a smooth fluctuation around the trend, or as being dependent on another factor (ex. plumbing sales are dependent on housing starts and therefore cycle with the construction industry). S is the seasonal component that predictably recurs within fixed intervals, usually a year. I is the Irregular component that can be identified, but is not expected to be repeated (ex. no sales due to strike). E is the residual or random component.

Since each component has a specific definition, the behavior of the original series can be explained by combining the effects of each component. The functional relationship of some time series may be additive, others may be multiplicative.

Discriminant Analysis

In discriminant analysis, an equation is used to separate or discriminate cases into classes (Figure 15). When the equation used is linear in form, it is said to be a linear discriminant. For problems with two characteristics, a line will be sufficient to discriminate between classes. For problems with three characteristics, the discriminant equation will be that of a plane.

For more complex situations, nonlinear discriminant (a curve, rather than a linear line) and piecewise linear discriminant (multiple lines or planes) may be used to separate classes. In situation where classes are not mutually exclusive (they overlap) there will always be some error in trying to separate cases into appropriate classes.

The general form of the linear discriminant equation is given below, where e_d are vectors of observed features, d is the number of features, and w_d are constants (weights) that must be estimated. Note that in a two feature analysis, w_0 is a constant that is graphically represented by the x intercept of the discriminant line, such that when the result is positive, the case is classified in one group, when negative, classified in the other.

$$w_1e_1 + w_2e_2 \dots + w_d e_d - w_0 \quad (3-10) \quad \text{where: } w_i = \text{weights} \\ e_i = \text{vector of characteristics}$$

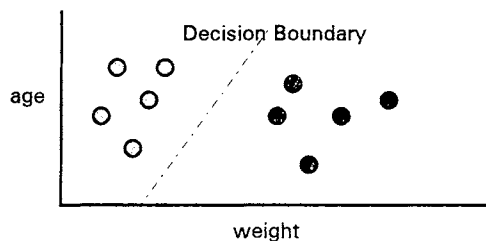


Figure 15. Idealized Class Separated by Line

A linear discriminant can be thought of as the weighted sum of the values of a cases' characteristics. If each class has the same variance and is normally distributed, then the analysis can be in the form of a single threshold rule. A threshold value or score is calculated. If a cases' weighted score is above the threshold, it is classified in one group. If the score is below a threshold, the case is classified in the other group. The task of discriminant analysis is to determine the weights to be used in the scoring equation.

However, if the variances of the two classes are different, more than one threshold will be needed. Additional problems will surface if the actual population is not normally distributed or if the variables are not continuous (a normal distribution implies that the variable is continuous over the variable's limits).

It has been recognized for that feed forward neural network have very similar structures to discriminant function based pattern classifiers (Lippmann, 1987). The blocks used in discriminant functions are similar to nodes in a feed forward network. According to Weiss and Kulikowski (1991, p 82) many neural networks turn out to be variations of piecewise linear classifiers without the need to make parametric assumptions concerning the underlying population distribution.

Silverman and Noetzel (1990) suggest that neural networks are superior to discriminant analysis in retrospective accuracy. Networks tend to be somewhat superior to discriminant analysis in terms of recognition of unknown cases. In their experiment, they noted that predictive accuracy seems

to have plateaued or decreased after 2000 passes, while retrospective accuracy increased indefinitely. They suggest that as retrospective accuracy increases, the ability to classify unknowns decreases, suggesting that the network becomes more and more like the nearest neighbor classifier with individual cases are being memorized by the network.

Neural networks have been successfully applied in business applications in discriminate analysis. Kimoto, Asakawa, Yoda and Takeoka (1990) produced a neural network that would suggest timing for buying and selling of stocks. Odom and Sharda (1990) used neural networks to predict bankruptcies from various financial indicators. Yoon and Swales (1991) used neural networks to predict the stock performance for various companies.

Webb and Lowe (1990), in analyzing a neural network with nonlinear input activation (transfer) functions and linear activation functions for other layers, proved that a nonlinear activation function allowed a back propagation neural networks to perform nonlinear discriminant analysis. The nonlinear activation function acts as a nonlinear transformation of the input data. "In the special case of a totally linear network with one hidden layer, the solution which maximizes the network discriminant function also maximizes the ratio of determinants of the between-class and total covariance matrices." (p 374)

Regression

Regression analysis is used to test the relationship between a dependent variable and one or more independent variables. Least squared regression attempts to do what a neural network using back propagation attempts to do, minimize the sum of the squared error. In fact, when comparing a back propagation network, with no hidden units, and linear activation (or transform) function, with least squares regression, one will find approximately the same (coefficients or) weights for a given data set (Burke, 1993).

Regression procedures can be categorized as being either univariate or multivariate. One of the most common forms of univariate regression is in time series forecasting with time as the independent variable. Univariate autoregressive time series forecasting makes no attempt at

discovering underlying causal factors -- its use as a forecasting tool rests with the primary assumption that the past is a good predictor of the future. Univariate methods are appropriate when the objective of the analysis is to forecast rather than analyze the causal effects affecting the data. Other types of univariate statistical procedures include exponential smoothing, component models, and the Box-Jenkins autoregressive integrated moving average (ARIMA) process.

Gallinari, Thiria, and Fogelman-Soulie (1988) showed that for a neural network performing a one-from-N classification, the weights used in the optimum network design which minimize total mean squared error of the output error, also maximized the ration of determinants of the between-class and total covariance matrices.

In attempting to analyze how neural networks worked, Webb and Lowe (1990) simplified the model of Rumelhart et al. (1986) by assuming a neural network with nonlinear input transformation and linear hidden layer transformation functions. In doing so, they showed how this modified neural network had equivalent results to those used in linear regression theory. They concluded that the discriminatory ability of neural networks stem from the input layer's nonlinear transformation of data into a form that makes linear discrimination easier in later layers of the network. They also concluded that the final linear transformation used in the network "could be replaced by a more sophisticated statistical pattern analysis method" (p 370).

White (1990) in his analysis of the statistical consistency of back propagation neural networks also established the nonparametric regression capability of the neural network model. Stone (1986) has shown that multiple regression analysis is similar to hierarchical neural networks.

Bayes Theory

Many times one will need to calculate the probability of an event based on its original probability and new information. Bayes Theory provides a procedure for incorporating *a priori* information as well as for accounting for new information. Bayes Theory is also the basis for statistical pattern recognition, and "provides a unified approach to the well known classification algorithms, including linear and quadratic discriminant analysis (Patuwo, Hu, and Hung, 1993, 827).

A disadvantage of Bayes theory is that one must have information of the prior distributions for the parameters. When the distributions are not known or can not be expressed algebraically, nonparametric methods such as k -nearest-neighbor or linear programming models must be used.

The base of Bayes Theory is the Bayesian Inversion Formula:

$$P(x|y) = \frac{P(y|x) P(x)}{P(y)} \quad (3-11)$$

which says that for an event x [with a known probability $P(x)$], the probability of "event x given event y " can be computed by factoring in the likelihood of y 's occurring, given that x has occurred, and the overall likelihood that y will occur at all. The Bayes formula also provides the basis of a method of categorizing patterns. In this interpretation, x is interpreted as a possible category into which a pattern might be placed, and y is interpreted as the pattern itself. A decision function can be associated with each possible category (all values of x). Generally, the decision functions are simple relations of $P(x|y)$ in which the potential loss if an incorrect decision is made is factored into the probabilities. If the loss is the same no matter what error occurs, the decision functions reduces to $P(x|y)$.

The problem with Bayesian analysis is that real world problems rarely have perfectly known probabilities. As best we can hope to estimate or approximate such Bayesian decisions. The analysis is simplified if a Parametric model of normalcy is assumed. The normal or Gaussian distribution has been the basis of a large part of statistical theory, not only because it models many natural event distributions exactly, but even more because it is particularly good at representing the properties of aggregated and cumulative phenomena.

Weiss and Kulikowski (1991) suggest that from the perspective of minimizing average error rates all learning systems can be seen as approximations of the Bayes rule.

The simple empirical implementation of the Bayes rule is equivalent to the direct lookup of a new case in a table containing a huge sample of cases, each with the correct class noted. The minimum error answer is then simply the class that occurs most frequently for the given pattern.... The objective of the learning system is to find a set of rules that covers the cases of a particular class without covering the cases in the other classes, or at least minimizing the

number of errors that are made. The hope is that not all features need to obtain coverage. For example, we might be given a sample of cases with two classes: the class of wealthy people and those that are not. We also might be given data with many features, such as age, sex, income, net worth, height, and weight. If the true classifications were assigned strictly on the basis of net worth above \$1,000,000, then a learning system could readily cover all the cases using this single feature. While the Bayes rule would find the correct answer, all the useless features would also be considered in the lookup table. (Weiss and Kulikowski, 1991, p 116)

Thacker and Mayhew (1990) suggest that hierarchical neural networks using the least squares method of learning and normalized data is equivalent to the method used in the Bayesian classifier. Patuwo, et. al. (1993) suggest that neural networks are comparable to Bayesian methods in terms of classification rates in training samples but not in test samples.

Gaussian Classifier

Barschdorff, Monostori, Ndenge, and Wostenkujler (1991) suggest that the most frequently used traditional Parametric pattern recognition technique is a derivation of the Bayes classifier for Gaussian data. This derivation is called the Gaussian Classifier. The Gaussian classifier can be used with features that are non-Gaussian or even discrete valued, and the covariance matrix and mean vectors are usually estimated as averages taken over the sample set.

Yau and Manry (1990) showed that the Gaussian classifier is isomorphic to the Sigma-Pi two layer neural network (Rumelhart et al., 1986a).

Box-Jenkins

Tang, de Almelda, and Fishwick (1990) reported on forecasting using Box-Jenkins methods vs. back propagation neural networks. The analysis involved airline passenger data, domestic car sales, and foreign car sales. They found that where Box-Jenkins does not work well with short series of input data, neural networks worked reasonably well, and considered this an advantage of neural networks. As the number of forecast periods increased, the relative performance of neural networks

increased -- Box-Jenkins tends to nicely reproduce the details of the original series, while neural networks tend to generalize, ignoring the details. As the number of input variables increased, the forecasting performance of the neural network improved.

Varfis and Versino (1990) applied neural networks and Box-Jenkins models using two series of data, and found neural networks to compare favorably. Sharda and Patil (1990) compared neural networks with the Box-Jenkins model and found neural networks to compare favorably. Reynolds, Mellichamp and Smith (1995) showed that a three-layer back propagation neural network is a better method of identifying Box-Jenkins models for univariate seasonal time series.

Decision Tree

A decision tree consists of nodes and branches. Each node represents a decision point, with the starting node called the root node. In a binary decision tree, a decision of true or false will direct the tree to branch either to the right (false) or to the left (true). At the end of the decision tree, a decision has been made at the terminal node. If the decision tree can be thought of as a collection of rules, then the terminal node will correspond to a specific decision rule. Non-binary decision trees will have more than two branches leaving a node. All paths in a decision tree are mutually exclusive. For any new case, there is one and only one path that will lead to a satisfactory conclusion.

An inductive logic process is used to optimize the structure of a decision tree. Using this method, nodes are split into two or more classes on either a random or an arbitrary basis. One way the node becomes a terminal node is when all data points are members of one class. The node may also be classified as terminal when the number of cases assigned fall below some minimum threshold and the node is assigned the class having the greatest frequency. As the number of nodes increase, the apparent classification error approaches zero.

Weiss and Kulikowski (1991, p 121) note that there are several functions that can be used to determine whether a node should be split, and that no overwhelming case can be made for any single function. The best function seems to be based on the expected nature of the decision tree. The underlying concept for most decision tree evaluation functions is based on evaluating at a specific

node, the likelihood of the split producing a better tree, without testing possible splits of successor nodes. The objective of these evaluation functions is to reduce the degree of randomness of classes within the current node. Two popular evaluation functions are the entropy and the gini functions.

$$\text{entropy } i = - \sum p_j \log p_j \quad (3-12) \quad \text{where: } p = \text{probability of each class}$$

$$\text{gini } i = 1 - \sum p_j^2 \quad (3-13) \quad \begin{array}{l} j = \text{classes within the node} \\ i = \text{impurity} \end{array}$$

Using either the entropy or gini functions for evaluating the impurity of a node, one calculates the reduction in impurity based on the expected value of the cases from the parent node that take that branch. The impurities of all immediate branches of a parent node are summed, and the reduction is found by subtracting the impurities of the children branch nodes from the impurities of the parent node.

$$D_i = i_n - \sum p_k i_k \quad (3-14) \quad \text{where: } p = \text{probability of each class} \\ i_n = \text{impurity in current node}$$

Many different splits are examined, and the number of splits of the parent node is the one with the greatest reduction in impurities. Chi-square can be used to compare the distribution of classes at the node with the classes at the resultant split node. When the statistical test indicates that further splitting will not be significant, then the process is stopped. A problem with this approach is that it does not look ahead beyond the current step.

The inherent weakness in this method is that the use of continuous data is problematic (how does one know where the best split should be made?), splits for every variable in the sample will need to be considered. Categorical variables work well with binary trees and may either be automatically mapped into true or false splits, or split into one attribute specific group with all others in another group.

Because the induction process typically continues until all terminal nodes contain only members from the sample of a single class, the apparent error rate will approach zero. This tends to lead to overtraining, and the actual error rate for new data may be significantly different from the apparent error rate for test data.

To help overcome this overtraining problem, pruning techniques are used to remove branches from a fully expanded tree. Working backwards, the weakest branches are removed until no weak branch remains. There are several heuristics for the pruning of decision trees.

CHAPTER IV

BACK PROPAGATION DESIGN CONSIDERATIONS

The Review of Literature will be divided into several basic components: a review of the Evolution of Neural Networks, a review of Non-Parametric Statistical Concepts, Traditional Statistical Classification Methods and their Relation to Neural Networks, a review of generalized Neural Network Learning techniques and specific back propagation based techniques that could be used in data analysis, and a review existing literature describing how data should be prepared for use with back propagation neural networks.

Intelligent Systems Design Methodology

As discussed in Chapters one and two, drawing on the works of Samuel (1967), Van Dalen (1979), Minsky and Papert (1988), Kolen and Goel (1991), and Roy, Govil and Miranda (1995), one can identify major characteristics that describe Self-Learning Systems such as Neural Networks when used in experimental design. The characteristics include Objectivity, Credit Assignment, Reliability, Generalization, Computational Complexity, and Representational Adequacy. In designing experiments using Neural Networks, one should be cognizant of the characteristics.

Objectivity

The system must yield the same findings regardless of the equipment used to run the computer program, or who administers or records the findings (Van Dalen, 1979). This is a concern especially when developing systems based on custom algorithm using UNIX type operating systems. It is common knowledge that UNIX operating systems are not identical across various computer equipment platforms.

Additionally the various software houses (Novel, Microsoft, Santa Cruse Operating Systems, IBM, etc.) each have versions of UNIX that are not 100% compatible with each other.

The problem of objectivity is not of great concern when using commercial available software, such as NeuralWare. Commercial organizations have a vested interest in writing software at a level that will operate identically on different platforms.

Credit Assignment

Once implemented, when incorrect performance has been identified, the system must be able to identify its structural components that are responsible for incorrect performance, and must be able to make modifications so that correct performance can be achieved. This characteristic is typically a feature of the Neural Network systems man-machine interface.

Reliability

The system must be able to yield the same basic results with the same exemplars are trained on networks under the same conditions. White (1990) called this statistical consistency in his evaluation of neural networks performing nonparametric regression functions. Roy, Govil and Miranda (1995) called this characteristic Robustness of Learning. A procedure is consistent if the approximation error of the forecast approaches zero as the sample size approaches infinity. Procedures that are not consistent will make errors in classification, recognition, or forecasting. Procedures that are consistent will only make errors due to the inherent randomness of fuzziness of the true relation between independent and dependent variables. A back propagation neural network, given the same initial weights assigned to the neurode connections, will produce the same results (adjusted for computer rounding error) each time the network is run. However, when neural network is initiated with random weight settings within a set range, for a given stopping criteria, the network may converge on local optimum point that produces a slightly different result.

Generalization

Peterson and Hartman (1989) defined generalization concerning neural networks as "the response of a network, after some amount of training, to novel (unlearned) inputs" (p 481).

Generalization is what the system learns from specific examples must be generalizable to general abstractions that are needed to perform specific tasks. Generalization directly concerns itself with the statistical concept of external validity. Because of the training rather than programming nature of neural networks, one could also argue that generalization is also concerned with internal and instrument validity. Kempka (1994b) suggests generalization is the ability to "infer from incomplete data or the success of acting upon data not included during training" (p 42).

The ability of a neural network to generalize is sensitive to the number of hidden units in the network. If the number is too small, then the network is overconstrained and there is not enough structure for all the needed abstractions. The number is too great, then the network tends to memorize the features of a specific data set (Chandrasekaran and Goel, 1988; Kruschke and Movellan, 1991). Peterson and Hartman (1989) suggest two basic ways to test generalization:

1. Continuous Learning - uses the entire training set. Each time the network is presented with a case, it is first tested for generalization on that pattern. The disadvantage here it that there is not a clear distinction between learning and generalization.

2. Fixed Training - Cases are separated into learning and training sets. After learning the training set, the network is tested on a set of cases that the network has not seen before.

Perform Network Design

Distantly related to the concept of credit assignment, a network learning method should be able to design an appropriate network for a given problem, without the need for external input or predefined structures (Roy, Govil and Miranda, 1995). To the extent that network design may not be totally performed by the learning method, the characteristic of computational complexity becomes increasingly important.

Computational Complexity

The system must be computationally efficient so that learning can occur in a reasonable amount of time (Kolen and Goel, 1991). Roy Govil and Miranda (1995) called this characteristic the Efficiency in Learning. When evaluating computational complexity, considerations should be made in the areas of the Epoch size, the initial random weight settings, the activation function, and the learning (weight change) algorithm:

Epoch Size. Back propagation error derivatives may be calculated either on a case by case basis or on an epochic basis. In the epoch method, the error derivative is the sum of the error derivatives for each case presented. Weiss and Kulikowski (1991, p 99) have found that the case by case approach yields superior results in terms of small mean squared errors. Determining epoch size can be an iterative process, depending in part on the amount of noise in the data set. Klimasauskas (1991a) recommends the following procedure:

1. Pick an initial value (16 seems to be a good number)
2. Train the network for a pre-established number (perhaps 10,000) of iterations
3. Test the network for accuracy, or calculate R^2 .
4. Repeat the process for additional epoch sizes, graph the results. The point at which the network peaked is the optimum size for the network and data set.

Initial Weight Settings. Significant differences in learning rates can be caused by the initial random selection of the connection weights. They also noticed a sharp increase in the non convergence rate with weights greater than two, and at weights near the origin. Based on their results, they also concluded that the learning rate (h) and momentum (a) have little effect on the percentage of non convergent initial weight states. (Kolen and Pollack, 1990; Kolen and Goel, 1991).

When setting initial weights, Smith (1993) suggests that if *a priori* knowledge is not available,

then an initial midrange node output is best. Smith recommends the following in setting weights:

1. Hidden Layer Nodes: set to small random numbers distributed evenly around zero, with the average absolute value of the weights set by the following formula:

$$|w| = \frac{\zeta}{i} \quad (4-1) \quad \text{where: } |w| = \text{average absolute value of weights}$$

i = number of system neurodes
 ζ = maximum input value

2. Output-Layer Node: set one half with 1 and the other half with -1; the weighted sum of weights should be close to zero.

Ye, Wang, and Robert (1990) suggest that partially connected neural networks should be used where *a priori* knowledge is available. Partially connected networks may be constructed to extract time-varying features.

Activation Function. The activation function (sometimes called the transfer function) is the part of the neurode that produces an output. The choice of a non-linear activation function in hidden neurodes will allow the network to model non-linear functions. The multi-layered structure of a back propagation network (and not the choice of activation function) is the feature that allows the network to be a universal approximator (Hornik, Stinchcombe, and White, 1990).

In general, all output neurodes should contain linear activation functions (Kempka, 1994a). The choice of hidden unit activation functions depends partly on the distribution of errors in the sample data and partly on the specific application (Cichocko and Unbehauen, 1992). For given traditional statistical design techniques, Sarle (1994b) suggests following network designs and activation functions:

- Linear Discriminant Function - Single Layer with a threshold activation function at the output layer
- Non-linear Principal Component - Three hidden layers with the first and third hidden layers containing more units than there are input units with non-linear activation, and the second hidden layer containing fewer units than there are inputs with linear activation. Linear activation on the output layer.
- Multivariate Linear Regression - Single Layer with linear activation function at the output layer

- **Multivariate non-Linear Regression** - Either Single Layer with non-linear activation function at the output layer (usually logistic), or multi-layered with non-linear activation on the hidden neurodes and linear activation on the output neurodes. Note: multi-layered will tend to give better generalizations are universal approximators.
- **Linear Maximum Redundancy Analysis**- One hidden layer with fewer units than there are inputs, linear activation on both hidden and output layers.
- **Non-linear Maximum Redundancy Analysis** - Two hidden layers with the first hidden layer containing more units than there are input units with non-linear activation, and the second hidden layer containing fewer units than there are inputs with linear activation. Linear activation on the output layer.

Considerations in selecting the appropriate hidden unit activation function include:

Squashing: transforming an output of unlimited range to an output of a specific range, generally either 0 to 1 or -1 to +1. The choice of range will influence the training of any neural network.

Differences in magnitude (one vs. two units) and the steepness of the slope of the function will give different error surfaces. Using a -1 to +1 range may generate unwanted zero values unless bias is also used.

Bias: In mathematical terms, *bias* (Q) can be thought of as the point at which the a function crosses the y axes. In neural networks bias may be used to set hurdle rates for neurode activation. (See gain, which may be used to change the slope of the activation function)

Gain: In mathematical terms, *gain* (g) can be thought of as a weight that adjusts the slope of a function. If the activation function is a logistic function with a maximum of 1 and a minimum of 0, increasing the gain weight from 1 to 2 would create a line more closely tracks the y axes with the slope of the curve at the boundary changing from .25 to .5. Decreasing the gain from 1 to .1 would flatten out the line. An additional benefit of gain is that it, in effect, normalizes weight vectors of neurodes. Adaptive gain is an algorithm that speeds learning by automatically changing values as the slope of the error curve changes and can be used in the place of a momentum (α) factor in back propagation gradient descent

calculations. Adaptive gain can also be used to eliminate unnecessary hidden layer neurodes from a neural network (Kruschke and Movellan, 1991).

Learning (Weight Change) Algorithm. The algorithm used to describe how weights should be updated was developed by Widrow and Hoff (1960), and is now sometimes called the Least Mean Square rule, and sometimes called the Delta Rule. Using the algorithm does not require linearly separable data. The development of multi-layered networks to solve the exclusive/or problem also lead to a generalized form of Least Mean Square training called back propagation, first presented by Rumelhart, Hinton, and Williams (1986a). Parameters that affect back propagation include:

Learning Rates (h). The learning rate parameter (h) is a correction factor that modifies calculated weight corrections such that $(h) * (Dw_{i(t)})$. With back propagation, the smaller the learning rate, the slower the progress in finding an optimum solution. The larger the learning rate, the more oscillations there are and the greater the possibility of skipping over a minimum solution. Weiss and Kulikowski (1991, p 101) suggest a learning rate of $h = 0.5$ as being typical. Tollenaere (1990) suggests that there is an optimal learning rate for which learning is fast and the procedure remains stable. for a 10-10 network $h = .25$ has been found to be optimal, for a 10-5-2 network, $h = 1.0$ has found to be optimal. Reyneri and Filippi (1991) suggest an algorithm that may be used to find the optimal learning parameter that is based on the number of inputs, and may be different for various layers of the network. Kung and Hwang (1988) in applying the algebraic projection analysis to back propagation, developed a formula to estimate the optimum learning rate:

$$\eta = \frac{72}{P + 1} \quad (4-2) \quad \text{where: } P = \text{the number of hidden neurodes}$$

Momentum (a). Momentum is a mathematical term added to the weight update algorithm that includes a percentage of the previous weight adjustment in the current weight adjustment; it is a mathematical device used to keep the network moving on the downhill error surface, even when localized minimums are reached. Tollenaere (1990) suggests that there is an inverse relationship between an optimum learning rate and momentum. Some have reported not being able to train networks with

momentum factors ($\alpha > 0.5$, while others report success with $\alpha = 0.9$. Tollenaere suggests that the contradictory claims could be explained by the different learning rates used. The following suggestions concerning momentum are made:

- the optimum learning rate (h) decreases as momentum (α) increases.
- the use of very high momentum does not result in instability, provided that step sizes are sufficiently small.

Representational Adequacy

Representational Adequacy can be defined as the ability of the system to internally represent what it known and what it is to learned. The concept also includes the statistical concept of suitability - the system must be capable of analyzing the problem under question.

Stein (1993) has suggested that the most important decision one makes in approaching a modeling project is in choosing the independent variables assumed to effect the dependent variable. After a preliminary list of candidate variables has been chosen, standard statistical analysis can be used to narrow the field. Stein suggests that the “goal of any modeler should be parsimony: to find the simplest explanation of the facts using the fewest variables” (p 44).

One may view the problem of representational adequacy in three perspectives: Statistical Sufficiency, Logical Suitability, and Data Representation. One looks at representational adequacy from the perspective of the structure of the neural network—is the structure of the neural network conducive to solving the problem under question? The other looks at representational adequacy from the perspective the training sets—is the data presented to the network representative of the problem under question?

Statistical Sufficiency. A back propagation hierarchical neural network is discrete Fourier series with the number of adjustable frequencies determined by the number of neurodes in the hidden layer (Lapedes and Farber, 1987). According to a mathematical theorem proved by Andrie Komogorov in the late 1950's (and restated for neural networks by Robert Hecht-Nelson in the late 1980's) “the network will always eventually figure out how to make perfect forecasts of the data on which the neural network is

being trained” (Coats and Fant, 1991, p 12). Hornik, Stinchcombe, and White (1990) demonstrated that a hierarchical neural network using back propagation could be used to approximate any mathematical function. Ito (1991) proved that a sigmoid function can, under certain circumstances, become universal approximators.

The perceptron based neural network is nonparametric in nature (Widrow and Hoff, 1960; Rumelhart, Hinton, Williams, 1986a; White, 1990; Weiss and Kulikowski, 1991). The Least Mean Square (or the Delta) rule that is the foundation of back propagation does not require data to be linearly separable or mutually exclusive (Widrow and Hoff, 1960). Neural networks seem to be variations of piecewise linear classifiers without the need to make parametric assumptions concerning the underlying population distribution. (Weiss and Kulikowski, 1991). Webb and Lowe (1990), in analyzing a neural network with nonlinear input activation functions and linear activation functions for other layers, proved that a nonlinear activation function allowed a back propagation neural networks to perform nonlinear discriminant analysis. White (1990) in his analysis of the statistical consistency of back propagation neural networks also established the nonparametric regression capability of the neural network model. Stone (1986) has shown that multiple regression analysis is similar to hierarchical neural networks. Webb and Lowe (1990) assuming a neural network with nonlinear input transformation and linear hidden layer transformation functions showed this had equivalent results to those used in linear regression theory. Least squared regression attempts to do what a neural network using back propagation attempts to do, minimize the sum of the squared error (Burke 1993).

Logical Suitability. The system must be able to internally represent what it knows and what is needed to be learned. This concept could be expanded to include the statistical concepts of suitability (Van Dalen, 1979)—the system must be capable of analyzing the problem under question. This concept seems to be similar to concept-learnability issues suggested by Haussler (1989) and Kolen and Goel (1991).

Multi-layered feedforward Neural Networks using back propagation have been proven to be universal approximators (Hornik, Stinchcombe, and White, 1990). Many researchers have shown that a neural network with a single hidden layer (two layer) can arbitrarily approximate any polynomial function

(Hornik, Stinchcombe and White, 1990, Kammerer and Kupper (1990), Baldi and Hornik (1989).

Funahashi (1989) showed that any continuous mapping can be approximated by a back propagation neural network with at least one hidden layer with sigmoid output functions. However, from a problem solving standpoint, a single hidden layer may not be enough to define a problem space. Blum and Li (1991) have constructed a proof that a neural network with two hidden layers and a linear output are universal approximators. They theorize that the first layer will divide the classification region by lines and the second layer will identify regions separated by the lines; if a task is of the type that may be solved by a single layer neural network, adding additional layers will not provide additional discriminatory power.

From a discriminant analysis standpoint, a single layer neural network can form two decision regions separated by a hyperplane. A two layer network, with one layer of hidden units, typically form open or closed convex decision surfaces, and thus can separate the inputs of the exclusive/or problem. The decision regions of a two layer neural network are formed from the intersections of regions defined by each node of the second layer. They note that a three layer neural network is required to approximate complex decision regions. (Barschdorff, Monostori, Ndenge, and Wostenkujler, 1991)

Kolen and Goel (1991) suggest that traditionally constructed neural networks are not capable of performing high level abstractions. They suggest that without the high-level abstractions, a network can only learned to mimic the situation--specific responses. The network could not generalize to the high-level abstractions. They concluded that high-level abstractions can be a major source of power for learning in back propagation neural networks. They suggest that much of the success of neural networks is due to the "programming" of these high-level abstractions in the networks in "compiled" form, either by arbitrarily setting initial weights, or by not fully connecting the network. These abstractions are useful because they provide a domain model to the network, which decomposes the learning space into smaller and simpler spaces, and guides the network in navigating these spaces.

Data Representation. One major consideration when deciding how data should be represented to a neural network is the consideration of whether to describe information as one of a set of exclusively categorized items (a nondistributed representation or a class of variables), or as a set of qualities that may be used to describe several different items (a distributed representation or quantitative variables). In

neural networks, the representation of discrete class type variables may be significantly different from the representation of quantitative variables. Because a reasonably sized network can store only a few unique patterns, neural networks tend to do best with distributed information.

Nominal Numbers - are labels that happen to use numbers as identifiers, they do not reflect any specific sequence or order. Crooks (1992) recommends that an input neurode be assigned for every possible value. For example, if there are three cities Tulsa, Enid, and Dallas, each city would be assigned a specific input neurode, and the input sequence for Enid would be 0 1 0.

Class variables can be problematic because networks can not generalize between classes (Smith, 1993). One solution is to group data by class, and randomly select training cases within each class.

Ordinal Numbers - Ordinal numbers imply a ranking or an ordering, but not magnitude. With ordinal numbers, one input neurode is usually needed per class of information. However, when there are several options within a class of data, a *thermometer code* may be a useful way of representing the data.

Interval (Continuous) Variables have values that can be any number within a range. Networks tend to be biased towards large magnetite numbers. To overcome this bias, inputs are typically scaled (or normalized) so that all inputs correspond to roughly the same input values, typically between 0 and 1 or -1 and +1. Assuming no *a priori* knowledge, dependent variable data should be scaled to equal variances; a variable with twice the variance of another variable has approximately twice the influence on training (Sarel, 1994).

In general, non-linear data should be transformed to linear data (Smith, 1993). While a back propagation neural network with non-linear activation function may model non-linear data, with non-linear data there is an increased risk of overfitting that would prevent the network from generalizing to new data.

Limitations of Neural Networks

Kolen and Goel (1991) suggest three interrelated issues that are limitations to the use of neural

networks as a form of artificial intelligence:

1. Back Propagation neural networks have a poor capacity in representing knowledge in that the network has difficulty in learning higher-order abstractions.
2. Neural networks make relatively little use of domain knowledge that exists in reality.
3. Neural networks do not reflect the structure of the task they address. Back propagation as a learning tool is task-neutral that does not make use of constraints that may be beneficial in decomposing the learning space into smaller and simpler regions.

Representational Adequacy Limitations. Kolen and Goel (1991) suggest three interrelated issues that are limitations to the use of neural networks as a form of artificial intelligence:

1. Back Propagation neural networks have a poor capacity in representing knowledge in that the network has difficulty in learning higher-order abstractions.
2. Neural networks make relatively little use of domain knowledge that exists in reality.
3. Neural networks do not reflect the structure of the task they address. Back propagation as a learning tool is task-neutral that does not make use of constraints that may be beneficial in decomposing the learning space into smaller and simpler regions.

From the standpoint of artificial intelligence, the problems being addressed seem to be related to the problem of Representational Adequacy: The system must be able to internally represent what it known and what it is to learned. The idea also includes the statistical idea of suitability -- the system must be capable of analyzing the problem under question.

One may view the problem of representational adequacy in two perspectives. One looks at representational adequacy from the perspective of the structure of the neural network -- is the structure of the neural network conducive to solving the problem under question? The other looks at representational adequacy from the perspective the training sets—is the data presented to the network representative of the problem under question? We will discuss aspects of each question in sequence, after which we will discuss statistical measures that may be used to compare representational adequacy between networks.

Is the structure of the neural network conducive to solving the problem under question?

Some problems are not representable in traditional neural network form -- structured learning, etc. Is the data presented to the network representative of the problem under question?

Variations in Optimum Network Component Design

Setting up a neural network is, at this time, largely a trial and error task. The following are conclusions others have drawn from their neural network research concerning how best to set up a neural network.

Learning by Case or Epoch

Back propagation error derivatives may be calculated either on a case by case basis or on an epoch basis—after all error derivatives have been computed for all cases presented. In the epoch method, the error derivative is the sum of the error derivatives for each case presented. Weiss and Kulikowski (1991, p 99) have found that the case by case approach yields superior results in terms of small mean squared errors.

There are two schools of thought in choosing an epoch size. If the epoch size is set to the size of the training set, the RMS error will be an indicator whether the network is more often right than wrong. In this case, a RMS error of 0.5 or greater indicates that the network is not learning. As the number of output units increase, the level of significance hurdle point decreases according to the formula: (Garavaglia, 1993).

$$E = \frac{1}{n} \quad (4-3) \quad \text{where: } n = \text{number of output neurodes}$$

Determining epoch size can also be an iterative process, depending in part on the amount of noise in the data set. Klimasauskas (1991a) recommends the following procedure:

1. Pick an initial value (16 seems to be a good number)
2. Train the network for a pre-established number (perhaps 10,000) of iterations

3. Test the network for accuracy, or calculate R^2 .
4. Repeat the process for additional epoch sizes, graph the results. The point at which the network peaked is the optimum size for the network and data set.

Randomization of Starting Weights

Symmetry Breaking was first described by Rumelhart, Hinton and Williams (1986a) and refers to the fact that if weights start with equal values in a network, modifications to those weights will be the same because hidden units will all receive the same error signal back from the output layer. To overcome the problem, small random unequal values are assigned as the initial values to the weights.

Neural networks typically begin with randomized settings for interconnecting weights. (Weiss and Kulikowski, 1991, p 100) suggest that the random numbers typically range from -.5 to .5. Repeating the training of a set of data with different starting points will typically produce slightly different answers, owing to starting at a different place in the error space, and to a stopping rule that is satisfied with error reaches a pre-specified minimum number, not zero.

Kolen and Goel (1991) conducted a simulation of the X/OR problem using the standard back propagation algorithms of Rumelhart et al. (1986a). In analyzing the simulations, they noticed significant differences in learning rates and concluded that the differences lie in the initial random selection of the connection weights. They also noticed a sharp increase in the non convergence rate with weights greater than two, and at weights near the origin. Based on their results, they also concluded that the learning rate (h) and momentum (a) have little effect on the percentage of non convergent initial weight states. Kolen and Pollack (1990) report on additional experimental results in support of this conclusion.

When setting initial weights, Smith (1993) suggests that when using a sigmoid activation function, if *a priori* knowledge is not available, then an initial midrange node output is best. Smith recommends the following in setting weights:

1. Set the hidden node weights (and the weights on direct links from inputs to outputs, if they exist) to small random numbers distributed evenly around zero. "If there are 1000 inputs and the

largest input value is 1, then the absolute value of each initial weights should be less than 0.01”

(Smith, 1993, p 96).

2. Initialize half of each output-node’s weights with values of 1 and the other half with -1; if there is an odd number, initialize the bias weight at zero. While we would want the weighted sum of weights close to zero, small weights would mean small derivatives, which would mean small weight changes and a long training time.

Sarel (1994a) notes that while generalization can be improved in linear models with small weights, there can be an adverse effect in Neural Networks using sigmodal activation functions. If the weights feeding a sigmodal activation function are sufficiently small and the activation region for the training cases lie in the central, almost linear, region of the activation function curve, the neurode may behave as if it has a linear activation function. If a network contains many small weights, the effective number of hidden units may be much less than the actual number of hidden units.

Optimum Number of Hidden Units

The optimum number of hidden units that should be used in a neural network is a matter of usually determined by trial and error. Chandrasekaran and Goel (1988) note that the ability of a neural network to generalize is sensitive to the number of hidden units in the network. If the number is too small, then the network is overconstrained and there is not enough structure for all the needed abstractions. Kruschke and Movellan (1991) suggest that the reason for improved generalization in reducing the number of hidden units is intuitively clear: "A small hidden layer forces the input patterns to be mapped through a low-dimensional space, enforcing proximities between hidden-layer representations that were not necessarily present in the input-pattern representation. Only the differences between patterns that are most important for decreasing error will be preserved as large distances between hidden layer patterns" (p 276).

The complexity of a network can be varied by changing the number of hidden units and the number of hidden layers. Sarle (1994b) notes that with a small number of hidden units, the network is similar to a parametric regression model. With a moderate number of hidden units, the network is similar

to a projection pursuit regression. As the number of hidden nodes increase, the network's ability to model ever increasingly complex functions also increase.

Neurodes of a hidden layer tend to have specific logical functions that can often be correlated to real world causal identifiers. Kung and Hwang (1988) suggest that the optimum number of hidden neurodes should be one less than the patterns in the population. Volper and Hampon (1990) suggest that when modeling quadratic equations, $d(d+1)/2$ cross terms as well as the linear terms are distinct features represented by hidden nodes.

There are many approaches that can be used in determining the optimum number of hidden units. Fujita (1991) proposed the Orthogonal Complement Method as a means to optimize the number of internal units in a binary network. It has been suggested that the optimum number of hidden notes can be calculated by $(c \log n)$.

Garavaglia (1993) suggests that by examining the weights of individual neurodes, one can estimate significance -- with weights above 0.5 indicating class membership. In networks with multiple neurodes in the output layer, the unit with the highest value represents the class predicted by the network. If one of the hidden layer neurodes contains a value close to 1.0, and others have values close to 0.0, one can conclude that the network has achieved a high level of discrimination among classes. If values of the output units and the percentage of correct classifications do not decrease together monotonically, a higher degree of nonlinearity is indicated.

Smith (1993) suggests that functions that approximate reoccurring wave functions can be optimally modeled using networks with five hidden nodes.

There have been two basic dynamic procedures suggested for determining optimum node number. Either start with zero hidden units and expand, or start with many hidden units and contract.

Start With Zero Hidden Units. Another approach by Weiss and Kulikowski (1991) is to start with zero hidden units and gradually add hidden units. With a single layer of hidden units, this provides a direct route to finding the complexity fit in terms of numbers of hidden units.

For a solution to be consistent with an accurate analysis, the following patterns should emerge:

- * Error distances on the training sets, which are usually correlated with the error rates, should decrease with the addition of hidden units until the distance reaches (almost) zero.
- * Error rates on the test set should follow the classic pattern of first decreasing, then flattening out, and finally increasing again.
- * Error distances on replication of the training session, but with different initial random starting states, should be reasonably close (as measured by standard deviations from the mean).

If inconsistencies are noted, it is probably an indication of local minima, and it may be necessary to revise learning parameters. This is particularly true if the inconsistencies remain even after the replicated runs are averages or the best training solution is selected. (p 106-107).

Network Pruning. Sietsma and Dow (1991) have been studying the question of effective network size. They note that if a network can be trained with a small number of units in the first processing layer, these units must be extracting features of the classes which can be interpreted by higher layers. To have many units in a layer can allow a network to become overspecific, approximating a look-up table, particularly in the extreme where the number of units in the first processing layer is equal to the number of cases in the training set. It is suggested that one way of producing a network would be to train the network with minimum number of hidden units, and then add extra layers, having the network relearning the task.

Sietsma and Dow (1991) proposed a methodology for determining optimum network size based on pruning. Their method has two stages of pruning. Stage one is removing hidden units that can be considered not contributing to the solution. Stage two is removing units that give information not required at the next layer. Noncontributing units are those which either have a relatively constant output, or have outputs that are either positively or negatively redundant. Unnecessary units are those that are independent of other units in the layer but give information not required at the next layer. However, it should be noted that removing units which contribute unnecessary information to the next layer may lead

to the output of the next layer being linearly inseparable. In other words, removing a unit may be the equivalent of creating an exclusive/or problem, which can not be done in a single layer.

When a layer has been pruned, the remaining outputs are tested for linear separability with respect to the classes imposed by the next layer. If outputs are not linearly separable, a small sub-network is trained to take the outputs of the pruned layer as inputs and reproduce the outputs of the layer above. This small sub-network enables the building of narrow, many-layered networks.

Conflicting Data. When two or more similar input patterns have entirely different outputs, a condition of conflicting data is said to exist. A neural network has difficulty learning when not enough fields are included in the model to distinguish adequately between two output categories. Versaggi (1995) suggests that conflicting data should not be eliminated, but the network model should be reexamined, increasing the number of inputs as required to eliminate the problem.

Optimum Number of Layers

The number of hidden layers that should be used in a neural network seems to be a problem specific. There are several proofs discussed later that show neural networks with one hidden layer and enough hidden layers are universal approximators. However, from a problem solving standpoint, a single hidden layer may not be enough to define a problem space.

Single Hidden Layer. Many researchers have shown that a neural network with a single hidden layer can arbitrarily approximate any polynomial function (Hornik, Stinchcombe and White, 1990, Kammerer and Kupper (1990), Baldi and Hornik (1989). These networks can also approximate functions that are not differentiable in the classical sense, but possess only a generalized derivative. Funahashi (1989) showed that any continuous mapping can be approximated by a back propagation neural network with at least one hidden layer with sigmoid output functions. Silverman and Noetzel (1990) suggest that networks with one hidden layer are best at generalization or identifying previously unknown items, and that two hidden layers work better at classifying previously known items.

White (1990) has shown that hierarchical neural network with one hidden layer and one output, using back propagation have the statistical property of consistency which means that as network experience accumulates, the probability of network approximation errors tends toward zero. White suggested that this property could be generalized to other more complex neural network designs. This meets the fifth requirement of Repeatability discussed in the first chapter.

Multiple Hidden Layers. Multi-layered networks allow for the modeling of non-linear phenomena, and provide for multiple inputs of neurodes. To take full advantage of a multi-layered network's capabilities, the neurode's activation (transfer) function needs to be a non-linear, continuously differentiable function. Two commonly used activation functions are known as logistic and sigmodal.

Blum and Li (1991) have constructed a proof that a neural network with two hidden layers and a linear output are universal approximators. They theorize that the first layer will divide the classification region by lines and the second layer will identify regions separated by the lines.

If a task is of the type that may be solved by a single layer neural network, adding additional layers will not provide additional discriminatory power. Adding additional layers when not needed may even prove to be disadvantageous because of the additional parameters that need adjusting. A two layer network can carry out both the task of identity (symmetric case) and selection (antisymmetric case.)

Multiple layers in neural networks could be beneficial in at least two contexts. The different layers of a neural network could represent descriptions of different layers of abstractions. In another form of representation, the first layer might be used for feature recognition and the next layers for recognition of groups of features, and other layers providing object representation. Thacker and Mayhew (1990) suggest that feedback mechanisms may be used to enforce contextual descriptions. When classifying order independent patterns, it suggested that a complete epoch be completed before passing information to the next layer so that a neurode is not overwhelmed by noise. If information is passed before the epoch is completed, a winner-take-all situation may develop that conveys no information about how well the pattern is described by a neurode in preference to any other.

From a discriminant analysis standpoint, a single layer neural network, as exemplified by Rosenblatt's (1958) perceptron, can form two decision regions separated by a hyperplane. A two layer

network, with one layer of hidden units, typically form open or closed convex decision surfaces, and thus can separate the inputs of the exclusive/or problem. As explained by Barschdorff, Monostori, Ndenge, and Wostenkujler (1991), the decision regions of a two layer neural network are formed from the intersections of regions defined by each node of the second layer (Figure 16). They note that a three layer neural network is required to approximate complex decision regions.

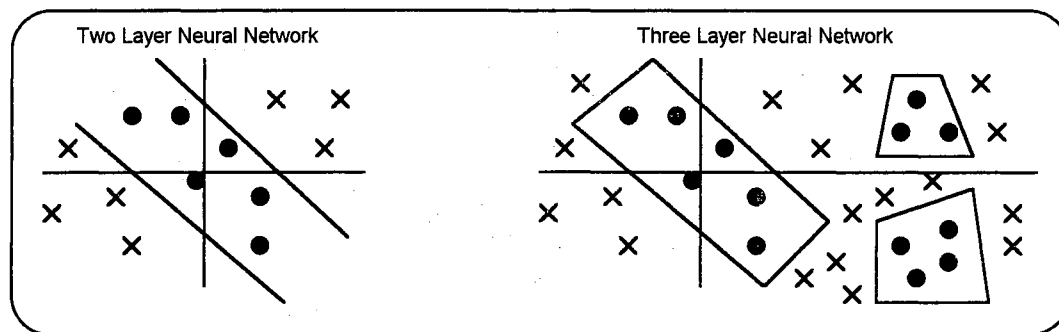


Figure 16. Network Layers in Defining Decision Regions

Geometrically, a two-layer network, with one layer of hidden units, can implement most decision surfaces, and can closely approximate any decision surface. A three-layer network, with two layers of hidden units, can implement any separating decision surface with sufficient hidden units are represented in the two layers. Thus, while it may be more convenient and efficient to specify more than one layer of hidden units, additional layers do not add any representational powers to the discrimination. (Weiss and Kulikowski, 1991, p 94)

There is empirical evidence that generalization to novel input patterns is improved by using hidden layers with small number of nodes. However, if one uses too few nodes, then the network has

trouble learning. Adaptive gain can be used OT make hidden nodes compete for the right to participate in the network representation of the data.

Thacker and Mayhew (1990) suggest that one reason that hierarchical neural networks have been shown to be well suited at pattern recognition is that network learning algorithms are "in most respects identical to the nearest-neighbor classifiers, such as k-means clustering, used in conventional pattern recognition" (p 291). Barschdorff and Bothe (1991) introduced a neural network model based on the nearest neighbor classification. Their network had three layers, with hidden layer nodes performing quadratic features, and simple nodes performing logical/or operations in the output layer.

Language Recognition and Layers. Ye, Wang, and Robert (1990) suggest that speech recognition problems cannot be solved by general purpose hierarchical back propagation neural networks because the network structure is not well suited to processing higher order information. As a solution to the problem, they propose using partially connected neural networks where *a priori* knowledge is available. Partially connected networks may be constructed to extract time-varying features

Plunkett and Marchman (1991) performed neural network language acquisition research in the form of how regular and irregular past tense forms are acquired. They noted that there seemed to be two distinct mechanisms involved:

1. the abstraction of family resemblance clusters of phonologically similarity
2. higher-level lexical representations manipulated by the past tense rule regardless of their lower-level phonological character.

They show that previous work in the area (Rumelhart et al., 1986a) incorporated only the first mechanism, and that the single-layer network that they used can not solve the overall mapping problem described above. In applying a multi-layered network to the problem, Plunkett and Marchman (1991) noted that networks learned the mapping of regular verbs better than the mapping of irregular verbs. They suggest that the difference in learning can be understood in terms of the degree to which a set of examples constitutes a single homogeneous class, with irregular verbs being thought of one of several

subclasses. When training different classes there is a "clear competition effect that is not directly predictable from the ability of the network to perform the mappings in isolation" (p 64). From a training standpoint, class size plays an important role in determining the degree to which other subclass characteristics leak into the target class, and therefore the performance of the network. They concluded that "In general, the larger the irregular class, the greater the likelihood of an irregular stem [verb] resembling and thus interfering with the mapping of a regular stem" (p 74).

Kammerer and Kupper (1990), researching language recognition, note that from a neural network standpoint, there are two basic approaches to classification that may be used. One approach is direct classification, where a single large network is used to discriminate one class from all other classes. Another approach is hierarchical classification, where the first level of the hierarchy consists of networks which are able to distinguish pairs of classes, and subsequent levels select the proper class based on outputs of lower levels.

Direct classification schemes require no assumptions about the partitioning of the problem into subtasks. If the problem allows for subtasks, it is assumed that multiple layers will be able to learn to assign certain subtasks to different layers. Hierarchical classification allows one to design parts of a whole system independently of each other. Kammerer and Kupper (1990) suggest a hierarchical classification scheme that involves two specific task levels. The first level is one of identifying patterns with a sub-network trained for pairwise class discrimination. The next level is one of selecting patterns for membership in specific classes based on the input pattern.

Complex Task Modeling (Idiot Savant). Dennis Norris (1990) attempted to construct a neural network that would model an idiot savant's ability to calculate on what day of the week any given date would fall. The network used to model this ability contained inputs of 31 day units, 12 month units, and 15 year units. There were 509 hidden units and seven output units, one for each day of the week. After 1000 iterations, the network could only accurately predict 73% of the test dates, and 17% of new dates. With a chance probability of 14.3% in guessing new dates, it would seem that the neural network was, in effect, trying to construct a look-up table.

To explain the failure, Norris noted that the task could be divided into three sections: learning about days, learning about months, and learning about years. Norris then devised a three step algorithm that could be used to predict the day of the week. For example, if the task was to find the day of the week that March 10, 1975 fell, one would:

1. From a base year (say 1951) determine the day of the week that the specific date would fall in the first month. For example, January 10, 1951 was a Wednesday.
2. From the output of step one, determine a date offset based on the specific month had the year been the base year. For example, the March offset for 1951 is three days.
3. Determine a date offset taking into account the difference between the base year (1951) and the target year (1975), and add that to the offsets calculated in steps one and two. For March 1975, the offset is two days. Therefore the total offset is five days (three days from step 2, and two days from step 3). Add the five day offset to the Wednesday calculated in step 1, and the March 10, 1975 fell on a Monday.

Norris notes that while a multilayered neural network could, in theory, solve the problem by allocating a separate layer to each stage, there is no guarantee that back propagation would be able to learn the task in a trial-and-error procedure. "The real problem with back propagation is that it has no way of imposing structure on the last learning. There is no way that back propagation can discover that a [multi-step task] is best performed by breaking the task into steps" (p 282).

When Norris created three subnetworks that were individually trained on the subtasks, the network performed substantially better. The subnetwork for the first two tasks performed perfectly. The third layer was able to match training set 90% of the time, with the majority of the errors being linked to leap years. Based on his analysis, Norris concluded that part of the problem was that there were relatively few examples of leap years in the 15 year training set. From the network's standpoint, the addition of an extra day on one month once every four years, did not seem that important.

Norris noted that the important human contribution to the learning effort was not the understanding of the detailed structure of the date task, but was telling the network that "its input changed from experiences of dates in a month to experiences of months in the year" (p 286). Once the

network was shown this feature, regularities in mapping between dates in a month and days of the week were preserved. These regularities were then used in the task of mapping between dates in the year and days in the week.

Norris also noted that although the network knows about days and dates, "it has no explicit knowledge of calendrical information. The net has no explicit representation of facts about leap years, or how often the calendar repeats itself. It is not simply that the net does not have the ability to interrogate its own internal representations, but rather that the net has not had to encode such information directly in order to perform the task" (p 286).

High Level Abstractions (Tic-Tac-Toe Simulation). Kolen and Goel (1991) suggest that traditionally constructed neural networks are not capable of performing high level abstractions. They come to this conclusion after reviewing an earlier report by Rumelhart, Smolensky, McClelland and Hinton (1986) concerning their success at training a neural network to play tic-tac-toe. Kolen and Goel concluded that the success of Rumelhart et al. (1986a) was due to the inadvertent preprogramming of high level abstractions. The preprogramming was done through organizing a large number of hidden units in a way that they correspond to the eight ways in which an entire row, column or diagonal can be captured. Kolen and Goel devised a variation of the original problem in which symbolic abstractions were not preprogrammed. As the opponent, a symbolic algorithm was used that, because of the nature of the heuristic, could after a sequence of moves, be beaten. The neural network won one game out of 10,000. The number of hidden units did not seem to have any influence on the ability of the neural network to learn the game.

Kolen and Goel (1991) draw two conclusions from the results of their attempts to model Tic-Tac-Toe games, taken together with the experimental results of Rumelhart et al. (1986a). First, without the high-level abstractions, their network only learned to mimic the situation--specific responses. The network could not generalize to the high-level abstractions needed to win against the opponent. The success of a similar study by Rumelhart et al. (1986a) is linked to their inadvertently giving the needed abstractions to the network in its formulation. With the high-level abstractions, the network appears to have learned the strategy for playing and winning tic-tac-toe. Therefore, one can conclude that the

content of what is learned by the method of back propagation is strongly dependent on the initial abstractions presented in the network.

The second conclusion is that these high-level abstractions are a major source of power for learning in back propagation neural networks. They suggest that much of the success of neural networks is due to the "programming" of these high-level abstractions in the networks in "compiled" form, either by arbitrarily setting initial weights, or by not fully connecting the network. These abstractions are useful because they provide a domain model to the network, which decomposes the learning space into smaller and simpler spaces, and guides the network in navigating these spaces.

Overfitting

Because back propagation neural networks are universal approximators in most cases, given enough time and enough hidden nodes, we can train a neural network to fit any data set. At this point one may question whether the neural network model is useful as a real world representation. Smith (1993) provides an example concerning a graph relating the number of people attending a dinner club to the number of reservations (Figure 17) for each of the last 10 years:

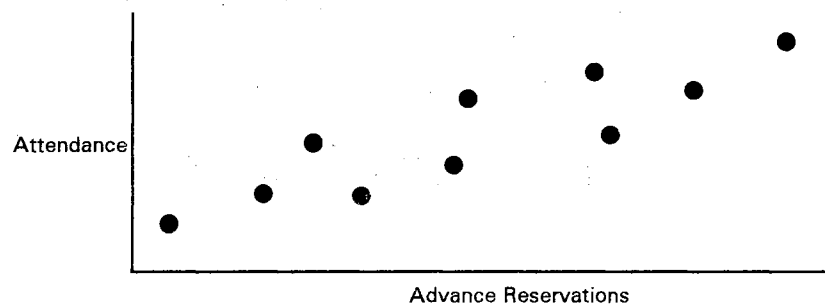


Figure 17. Scatter Diagram, Attendance vs. Reservations

We would expect the relationship of attendance to reservations to be roughly linear (Figure 17), and the data presented could be approximated by a straight line. Attendance usually runs 85% of reservations, but is higher in some years and lower in others. Reasons for the variation away from a straight line might include weather or competition from other special events - causal factors not taken into account in the model. Variations also may be due to inaccurate information or mistakes in recording data.

However, if one uses 10 years of data to train a neural network with 16 hidden nodes, the network will converge after 450,000 epochs of training. A graphical representation of the trained network output is in Figure 18.

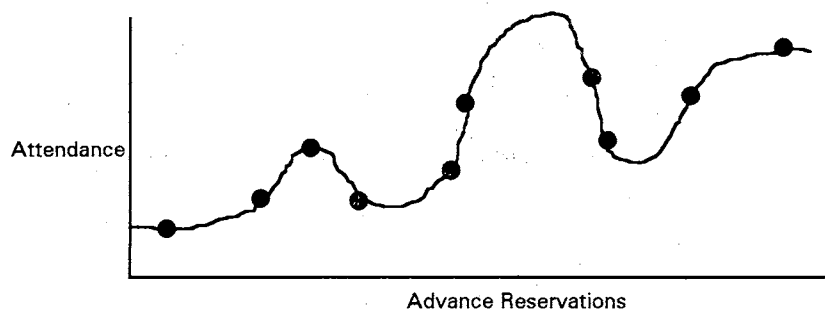


Figure 18. Overfitting, Attendance vs. Reservations

Chances are that this mapping of a network will not be a good predictor of attendance given the number of reservations. Although the model fits the training data with minimum error, it probably will not generalize very well. A network with these characteristics is said to be *overfitted* by modeling noise specific to the situation in addition to actual causal factors. Smith (1993) notes that the problem of overfitting may be addressed in one of three ways:

- * reducing the number of hidden nodes, thereby limiting the number of factors that may be mapped
- * discouraging the network from using large weights
- * limiting the number of training epochs

Sarel (1994c) notes that the ability of artificial neural networks to generalize is similar to that of other statistical models and suggests that networks do well at interpolation with adequate data available for sufficient training. However, because of the non-linear nature of artificial neural networks, extrapolation is risky.

Stopped Training

Sarle (1995c) suggests that the most popular method of preventing overfitting of the data by an artificial neural network is a method called stopped training, or early stopping, or optimal stopping. The stopped training method has the following steps:

1. Divide the available data into two separate data sets, one for training and the other for validation
2. Use a large number of hidden units
3. Use small random initial values
4. Use a small learning rate
5. Compute the validation error periodically during training
6. Stop training when validation error increases

Sarle (1995c) notes that this stopped training method with a sufficiently small learning rate is similar to ridge regression, has an advantage of being fast, and can be successfully applied to networks with interconnecting weights far exceeding the number of observations (a greater than 300 to one ratio). A disadvantage to this procedure is that usual statistical theory does not apply. Sarel also notes that even though it is widely accepted, there has been very little research on the subject.

Models of Neural Networks and their Statistical Equivalents

Sarle (1994b) notes that the efficiency of neural networks largely depends on the learning algorithms used in training the networks (adjusting interconnecting weights) and suggests that in many cases standard statistical techniques may be more efficient. One reason that standard learning algorithms are inefficient is that they are designed to be used on massively parallel computers, but are in fact usually implemented on serial computers. Another reason for inefficiency is that neural networks are designed for a real-time environment with transient data, where standard statistics are designed for stored data that is repeatedly accessible. However, apart from the efficiency issue, different models of neural networks (neurode activation functions, hidden units, etc.) have different statistical properties. The rest of this section is adapted from Sarle's (1994b) analysis of the structure of neural networks and their statistical equivalents.

Single Layer Networks A simple perceptron calculated a the sum of the inputs and then an activation function is applied to the net inputs to produce an output. When the activation function is a threshold type function, the network acts as a discriminant function, as show in Figure 19. If there is only one input, the system is called an *Adaline*.

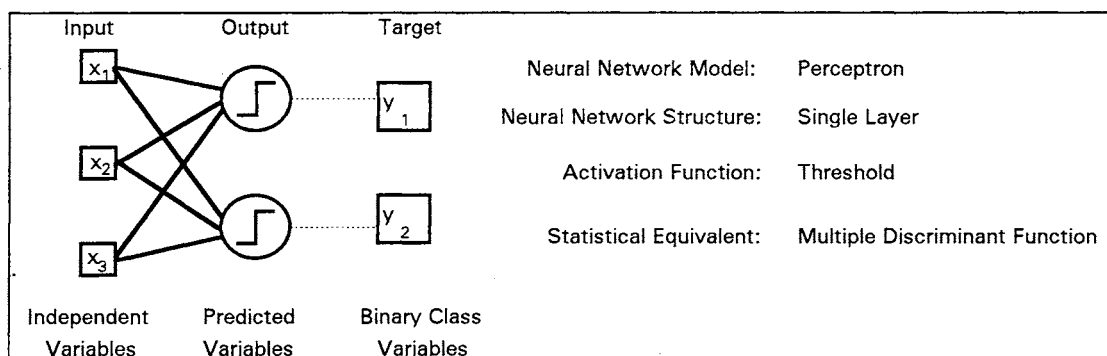


Figure 19. Perceptron (threshold) = Linear Discriminant Function

A simple perceptron with a linear activation function using a back propagation learning algorithm seeks to minimize the sum of the squared errors, which is the equivalent to a linear regression model. With multiple inputs and outputs, the model may be of a multivariate multiple linear regression, as in Figure 20.

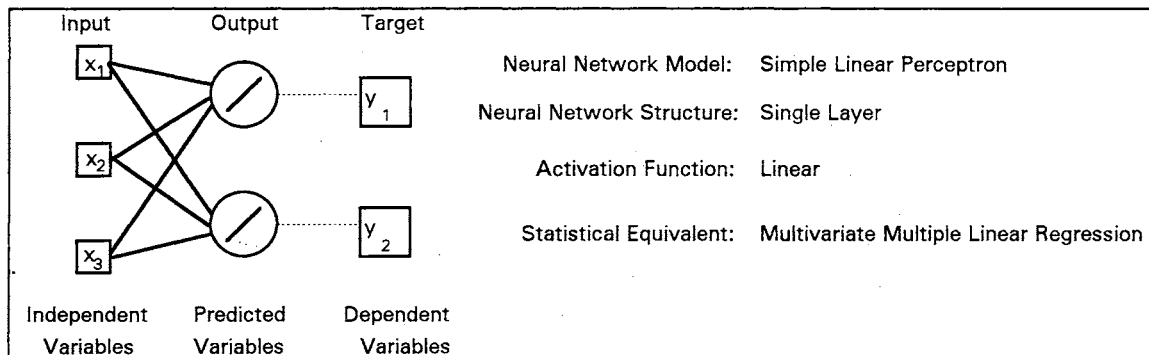


Figure 20. Simple Linear Perceptron = Multivariate Multiple Linear Regression

A simple perceptron using a non-linear activation function (usually, but not necessarily logistic) may produce a bounded output. Bounded activation functions are sometimes called squashing functions and have ranges of either 0 to 1 or -1 to +1. A perceptron with a logistic activation function is the equivalent to a logistic regression function, as in Figure 21.

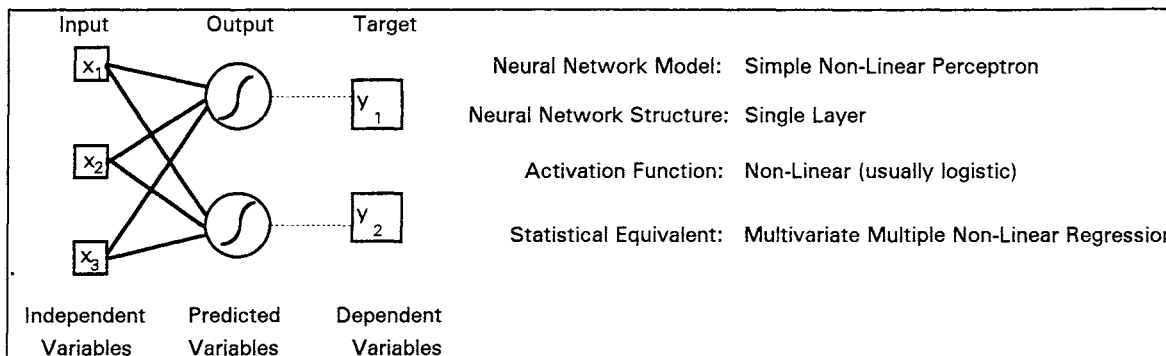


Figure 21. Simple Nonlinear Perceptron = Multivariate Multiple Logistic Regression

Multiple Layer Networks. With the addition of multiple layers with non-linear activation functions, the network model has nonlinear parameters. The complexity of a network model can be varied by the number of either layers or hidden units or both. When a hidden layer's activation function is a polynomial term and the input weights are a constant 1, the model is said to be a Functional Link Network, as in Figure 22, and are used primarily in image processing tasks.

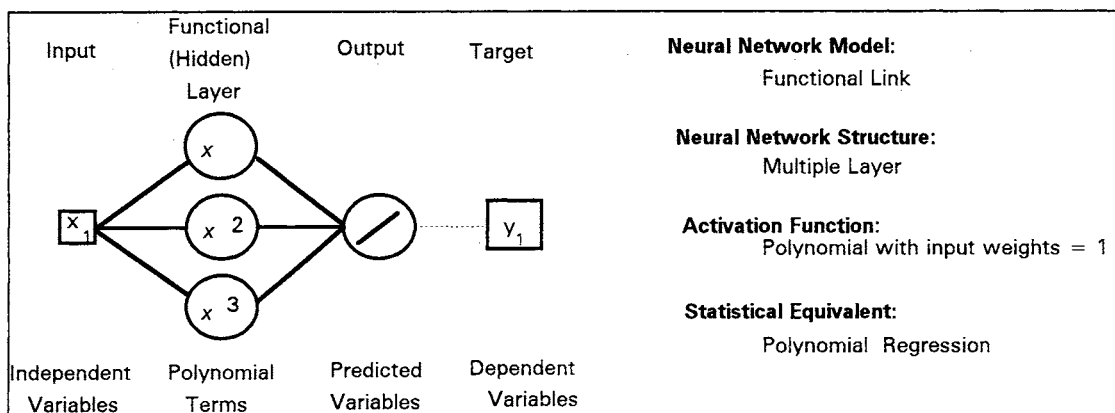


Figure 22. - Functional Link Network = Polynomial Regression

A multilayered perceptron with a non-linear activation function on the hidden units and a linear activation function on the output units are the equivalent of non-linear regression models as shown in Figure 23. A multilayered system can have any number of inputs, outputs, and hidden units, as required to approximate virtually any function to any degree of accuracy—these systems are universal approximators (Hornik, Stinchcombe, and White; 1990).

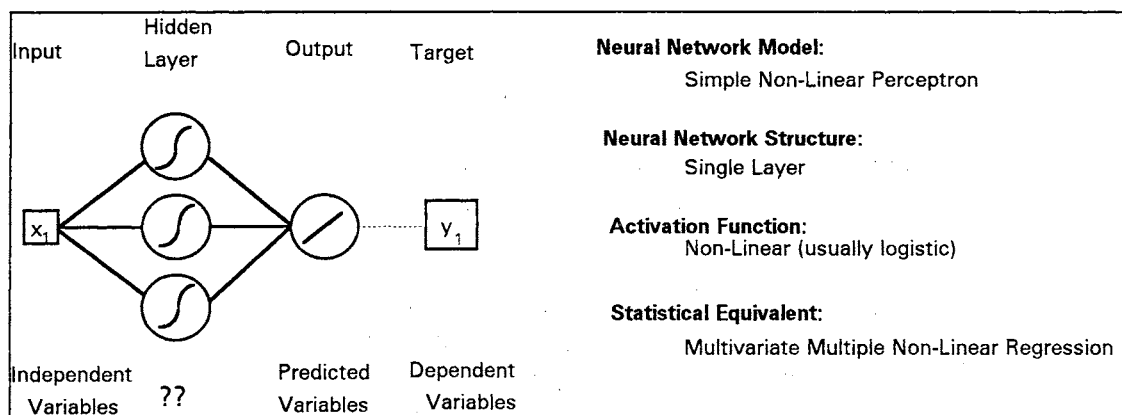


Figure 23. Multilayer Perceptron = Simple Nonlinear Regression

A variation is to allow direct connection from the input to the output layer, which is similar to the statistical concepts of main effects.

When the hidden layer uses a linear activation function, and the hidden layer has fewer units than the input layer, then the hidden layer acts as a bottleneck that accomplishes a dimensional reduction, as seen in Figure 24. This dimensional reduction is a form of generalization.

Principal component analysis is a form of dimension reduction that uses unsupervised learning in that the dependent variable is both input and output to the system (sometimes called autoassociation or encoding networks).

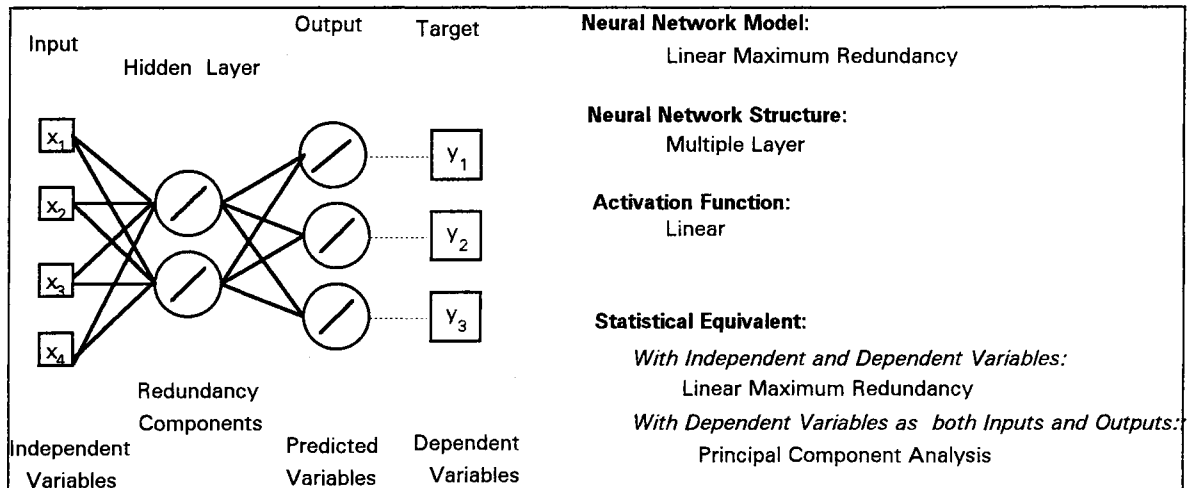


Figure 24. Maximum Redundancy Analysis

A nonlinear implementation of Maximum Redundancy Analysis may be created by adding an additional hidden layer, between the input and redundancy components, with a non-linear activation function, as seen in Figure 25.

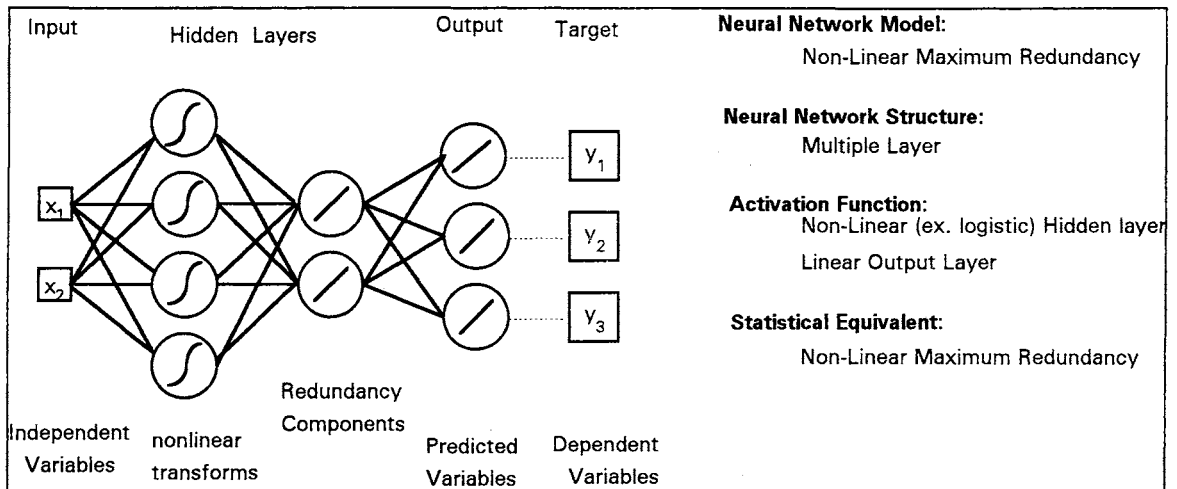


Figure 25. Non-Linear Maximum Redundancy Analysis

A non-linear equivalent to principal component analysis may be generated by adding an additional non-linear hidden layer between the output layer and the Redundancy Component layer, as in Figure 26.

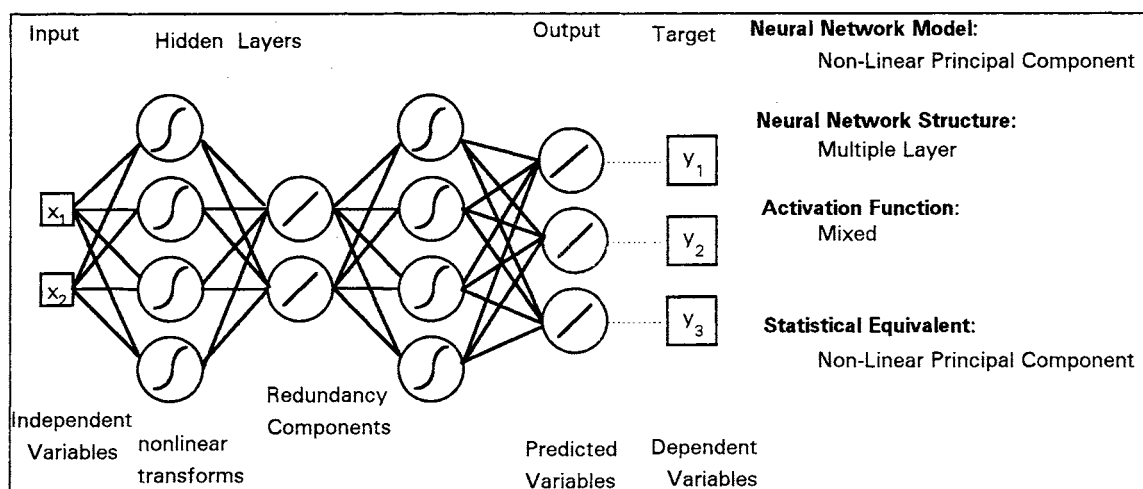


Figure 26. Non-Linear Principal Component

CHAPTER V

DATA DESIGN CONSIDERATIONS

The Review of Literature will be divided into several basic components: a review of the Evolution of Neural Networks, a review of Non-Parametric Statistical Concepts, Traditional Statistical Classification Methods and their Relation to Neural Networks, a review of generalized Neural Network Learning techniques and specific back propagation based techniques that could be used in data analysis, and a review existing literature describing how data should be prepared for use with back propagation neural networks.

Data Preparation for Neural Networks

This paper has discussed several types of neural networks (feedforward, feedback, competition, etc.), several types of learning algorithms used (Hebbian, Back Propagation, etc.), different activation functions that may be used (sigmoid, linear, etc.), and a variety of other modifications to the network. The options used largely depend on the application. Regardless of the model used, how the data represents cases being used to train the neural network will have a significant role in the ability of the neural network to model the environment. While neural networks can be used as a new way of analyzing data, many of the traditional statistical concepts are still relevant when making judgments concerning validity, consistency, and representative sampling.

Data Classification

Data can be classified as either *discrete* (has gaps between possible values) or *continuous* (any value over a specific range is possible). Data can also be classified according to their level of measurement, such as: *nominal* (labels such as male-female, cat-dog, or the percentage of an item in

categories), *ordinal* (arranged in an order from best to worst, but with no statement as to the magnitude of difference), *interval* (difference between values does have meaning), and *ratio* (interval data with a definite zero starting point). The difference between interval and ratio data is often determined by the scale being used. For example temperature measured in Kelvin units would be ratio (there is no temperature below 0^oK), but temperature measured in Fahrenheit would be interval. In another way of looking at the difference, four acres is twice the land area of two acres, therefore this measure is ratio.

Data Representation in Neural Networks

The way data are represented at the input nodes of a neural network has a major impact on the training of the network and of the ability of the resulting model to both discriminate and generalize. One major consideration when deciding how data should be represented to a neural network is the consideration of whether to describe information as one of a set of exclusively categorized items (a nondistributed representation or a class of variables), or as a set of qualities that may be used to describe several different items (a distributed representation or quantitative variables). In neural networks, the representation of discrete class type variables may be significantly different from the representation of quantitative variables. Because a reasonably sized network can store only a few unique patterns, neural networks tend to do best with distributed information. As described by Lawrence (1991b, p 36)

Information is deemed distributed when the qualities defining unique patterns are spread out over many pieces of information. For example, a purple object can be thought of as being half red and half blue. By using three primary color inputs (red, blue, and yellow), many possible color combinations can be represented without adding neurodes.

Data used by neural networks may be either continuous or binary in nature. Binary numbers in nature would be represented by nominal numbers and may represent class type variables. Continuous, quantitative numbers in nature would be represented by Interval (Real) and Ordinal Numbers.

Nominal Numbers - are labels that happen to use numbers as identifiers, they do not reflect any specific sequence or order. Crooks (1992) recommends that an input neurode be assigned for every possible value. For example, if there are three cities (Tulsa = 1, Enid = 2, Dallas = 3) being represented

in a survey by their respective numbers, the neural network would be designed such that there were three specific input neurodes to represent each city (other input neurodes would represent other information. In the series, Tulsa would be represented by 1 0 0, Enid by 0 1 0, etc. When there are many possible values, Crooks suggests combining categories, such as East, West, North and South for the states of the United States. By combining data into categories, one reduces the weights (and degrees of freedom) in the network, thereby increasing its power.

Class variables can be problematic because networks can not generalize between classes. Smith (1993, p 154) makes the following recommendations concerning the use of classes in neural networks:

First, classes that do not have a statistically reliable number of examples (given the noise level in the data) should probably be consolidated rather than represented separately. A small class can be consolidated with a larger class that it resembles in some relevant way.

Second, as the number of variables and the number of values per variable increase, the possible combinations of values increase exponentially—and the sample size needed to achieve a given level of accuracy increases at roughly the same rate.

A related problem (discussed later) concerns procedures for random selection of data vs. manual selection to assure that there are sufficient training cases at the boarder between two groups for the network to establish a clear demarcation. One solution is to group data by class, and randomly select training cases within each class.

Ordinal Numbers - Ordinal numbers imply a ranking or an ordering, but not magnitude. One can not sensibly perform arithmetic operations with ordinal numbers and produce meaningful results. With ordinal numbers, one input neurode is usually needed per class of information. However, when there are several options within a class of data, a *thermometer code* (Figure 27) may be a useful way of representing the data. When used in a neural network, a thermometer code is a set of binary input nodes, that as a group represent multiple states of nature. Smith (1993) uses the example of coding the results of a horse race by recording Win, Place, Show or out of the money.

	Node		
	1	2	3
Win	●	●	●
Place	●	●	○
Show	●	○	○
Out	○	○	○

Figure 27. Thermometer Code

This type of coding does not allow for a horse finishing in 2.5 place. Thermometer coding represents what ordinal variables mean, no more and no less. This type of coding allows for incremental credit to be assigned to the win position. Smith (1993) notes that this type of coding also facilitates appropriate discrimination and generation for ordinal variables.

Interval (Continuous) Variables have values that can be any number within a range. Interval numbers are continuous and able to determine not only which one of two number is greater, but also by how much. One may sensibly perform arithmetic operations with Interval numbers and produce meaningful results.

Neural networks pay attention to the absolute magnitude of interval and ordinal numbers. Because of the nature of the way networks handle neurode weights, fluctuations in nodes caused by values with small magnitudes (range 1 to 10) tend to be swamped by fluctuations in nodes caused by values with large magnitudes (range 1 to 100,000,000). To overcome this problem, inputs are typically scaled (or normalized) so that all inputs correspond to roughly the same input values, typically between zero and one.

Interpolation Representation

With interval (continuous) numbers, one input neurode is usually needed per class of information. However, to the extent that it is known that relationships are not linear, and that modeling requires several hidden nodes, these factors may also limit the ability of the network to generalize. In these cases interpolation representation may be used (Hutchinson, 1987). When using interpolation representation, multiple nodes are used to represent the various ranges of the input variable. For example, when looking at a range from -30 to 50, we might use four input variables to represent specific numbers: node 1 for -20, node 2 for 0, node 3 for 20, and node 4 for 40. When the independent variable is equal to the number represented by the node, the nodes value is 1. If the variable is 18, for example, it is represented by values of .1 for node 2 (because 18 is one-tenth of the way from 20 to 0) and .9 for node 3 (because 18 is nine-tenths of the way from 0 to 20).

Smith (1993) suggests that interpolation representation is attractive in neural networks for continuous variables because, since no precision in the value of the variable is lost, it permits whatever discrimination that is appropriate. Because ranges can be treated separately, it may permit a greater degree of generalization. Additionally, the use of interpolation representation to localize the effects of continuous value variables may decrease training time. "When the effect of each example is only local, the network doesn't have to seek a global solution.... It is not unusual for an interpolation representation to reduce the number of training epochs of training by two orders of magnitude" (p 157).

Minimum Number of Exemplars

The minimum number of observations pairs (exemplars) in a neural network depends on the architecture of the network. The more complex the architecture, the greater the number of exemplars needed to adequately train the network. The number of weights and biases of the network's neurodes and the number of hidden layers determines the number of undetermined parameters (or degrees of freedom) associated with the neural network. Equation 2-49 describes the optimum number of exemplars required for a neural network with one hidden layer.

$$N = \xi [J(I+1) + K(J+1)] \quad (5-1)$$

where: N = minimum number of exemplars
I = number of input nodes
J = number of hidden layer nodes
K = number of output nodes
 ξ = overdetermination constant

A neural network with one input node (I), one output node (K), four nodes on a single hidden layer (J) and an overdetermination constant (ξ) of 1.2, would need a minimum of 16 (exactly 15.6) exemplars (or observations) to adequately train the network. Were this network to be trained with less than 16 exemplars, the network would be susceptible to overtraining.

The overdetermination constant is generally a number between 1 and 2. With an overdetermination constant less than one, the number of exemplars is less than the networks degrees of freedom. With a overdetermination constant of 1, the equation would indicate the minimum number of exemplars necessary for the neural network to adequately capture the basic characteristics of the data with the recognition that the network is susceptible to overtraining (when the network memorizes the data and the results are not necessarily generalizable). With an overdetermination constant of 2 or greater, the model is said to be overdetermined and therefore not susceptible to overtraining. Carpenter and Hoffman (1995) suggest that having an overdetermination constant between 1.2 and 1.5 will usually provide the neural network with enough training pairs to develop a good approximation of the phenomenon being modeled without the network being susceptible to overtraining.

Transformation of Data

Neural networks not only pay attention to the magnitude of variables, but also to the variability of the values. In general data for neural networks are to either make relationships appear linear (logarithm, exponential, etc.) or to compensate for magnitude differences (normalization). Sarel (1994) suggests that the standardization of data tends to improve the training of neural networks by ensuring that certain parameters involved in initialization and termination of training are appropriate. He also notes that for linear models, standardization will produce results that are invariant or equivariant under

certain conditions. Considerations regarding standardizing data include:

- If the output layer activation function assumes a bounded range (0 to 1 or -1 to +1), the data for the dependent variables should fall in the same range, and an output layer bounded activation function should be used. (If dependent variables are not bounded, an unbounded activation function, such as exponential, should be used.)
- Dependent variables should be scaled to values less than the bounded range (ex. .1 and .9 for a 0 to 1 range -- traditional training methods can be very slow in giving good predictions using values of 0 and 1.)
- Multiple dependent variables that are measured in noncomparable units (such as miles and feet) must be standardized
- assuming no *a priori* knowledge, dependent variable data should be scaled to equal variances; a variable with twice the variance of another variable has approximately twice the influence on training
- Scaling dependent data should not be performed if the results are to be interpreted in the same units as the original units of measure—scale estimates apply to the standardized values, not the original values.
- Standardizing inputs has less effect than standardizing outputs—a change in scale of an input can be absorbed by a reciprocal change in corresponding weights and biases. However, scaling inputs can affect initialization and some training methods. With back propagation, a properly chosen Momentum (α) factor will overcome minor errors in selecting proper scaling factors.

Normalization. Normalizing the data will make information independent of the units of measure. Crooks (1993) notes that “for continuous values, this method not only maps inputs to roughly equivalent magnitudes, but it tends to give equal value to variations of equal rareness, regardless of the absolute magnitude of the valuation” (p 40).

Stein (1993) notes that most neural networks perform better with normally distributed data. In general data with skewness coefficients between -0.5 and $+0.5$ and kurtosis coefficients between -1.0 and $+1.0$ are considered normally distributed.

Weiss and Kulikowski (1991) suggest that the relative scale of variables may influence network performance. Some inputs may assume values measures in the thousands, while others may all be small integers. They found performance can often be improved by normalizing the data before training so that, for example, all input variables fall within the range from 0 to 1.

Reyneri and Filippi (1991) found that the results of training neural networks can be influenced by the magnitude of variables used. This is especially true in digital computer systems, where magnitude differences between variables can cause either overflow problems in computer registers, or rounding errors. They found that by normalizing data, performance of the learning rule improved. They report improvement of back propagation learning by normalizing inputs to a layer based on the number of neurodes in that layer.

Sutton and Barto (1981) suggest that one advantage of normalization is that it maintains the weight values within bounds. Kruschke (1989) found that normalization can also be used to facilitate comparison between nodes. Reyneri and Filippi (1991) report that adding a normalization factor to an activation function improves network performance.

Yao, Freeman, Burke and Yang (1991) suggest that prototype patterns can be created for training purposes. They suggest two approaches, a statistical and geometric approach. The statistical approach is used if the features are normally distributed and the number of training cases are large enough to determine the means and the standard deviations, the following be used for normalizing data:

1. Normalizing: for all training cases, find the mean and the standard deviation of each feature.

Then all data is converted into normalized data based (often called z-scores) on the transformation:

$$z = (x - \text{mean}) / (\text{standard deviation})$$

2. Averaging: find the center points for each of the classes into which cases belong.
3. Binarization: based on the normalized data, cases are classified into appropriate classes

A nonparametric procedure, called the geometric approach, can be used that basically eliminates the first step of normalization. These prototype patterns can then be used as centroid for a direct classification network.

Crooks (1993) notes two precautions that must be recognized with working with z-scores:

1. The mean and standard deviation used is an estimate of the population mean and standard deviation. Different samples from the same population may show different sample mean and standard deviations. This may result in significant differences when more than one training set is used.
2. Some neural network neurodes are not designed to accept negative numbers. Some are not designed to accept numbers greater than 1.0. The process of normalizing data may produce either numbers greater than 1.0 or negative numbers.

Transformations for Linearity. Logarithmic transformations may be used to make data appear relatively linear. While a neural network is perfectly capable of modeling non-linear data provided enough hidden nodes exist, realistically, it is "almost certain that before the network found these statistical anomalies in the data it would have found many other features that do not generalize" (Smith, 1993, p 143). In data preparation we have two choices. One may use *a priori* information, and when indicated transform the data, thereby speeding up the process of convergence. As an alternative, one can let the neural network model non-linear information recognizing that there would be an increased risk of overfitting that would prevent the network from generalizing to new data.

Sequential or Random Presentation

Cases may be presented to the neural network in a sequential or random manor. In random presentation, the order of cases presented is changed for each epoch. Weiss and Kulikowski (1991, p 99) have found that for non-time sensitive data, random presentations tend to produce better results.

Ratcliff (1991) noted that "items encoded into groups are significantly more resistant to forgetting than items learned individually" (p 291).

Adding Noise

Non-linear systems, whose output is not proportional to the input -- no output below a certain threshold, have been found to benefit from stochastic resonance, or random noise. The degree to which a system can benefit depends heavily on the signal-to-noise ratio. If too little noise is added, the system is not effected. If too much noise is added, the noise overwhelms the system.

Stochastic resonance has been shown to exist in biological sensing neurodes of crayfish and other squids (Moss and Wiesenfeld, 1995). It has also been found that slightly noisy data will tend to decrease the network's training time. Caudill (1991a) suggests that this benefit of noisy data is due to noisy data forcing the neural network to generalize relationships rather than memorize a particular combination of input patterns. If the network memorizes a training set, it may be able to replicate the input and output relationships for the specific data set, but not be able to predict with data with which it was not trained. Adding noise will also tend to make the network more robust to noisy inputs after training.

There are two basic ways noise can be added to the training set. For non-time sensitive data one may randomize the order the data is presented to the network, ensuring that the network never sees the same input pattern more than once. For time sensitive data, the actual input data may be slightly modified by adding (or subtracting) a small value.

Baba (1989) proposed a method of adding noise to the initial weights of Rumelhart et al. (1986) back propagation method for finding optimum weights in a multi layered neural network. Based on modifications to the Random Optimization Method of Matyas (1965), under certain conditions Baba proposed added a Gaussian random vector to the input vector of a neurode. It was noted that by adding this Gaussian noise, the network reached optimum weights faster than back propagation without the noise. It was also noted that the selection of the learning rate (h) and the variance to the Gaussian distribution had a significant effect on the learning speed.

Sietsma and Dow (1991) have found that adding noise to the training set has been shown to improve the performance of neural networks. Noise seems to cause more hidden units to be used in the network. They also found cases where a neural network with the fewest hidden units needed to

classifying the training set degraded the ability of the network to generalize, indicating that in some situations, networks with more hidden units generalize better than networks with few hidden units. Sietsma and Down concluded that "adding noise to the training set created more stable, robust networks which use more of the available units. Networks rarely use all units independently. Training with noise has the surprising effect of improving the ability of the network to correctly classify phases which were not in the training set, as well as improving the classification of noise corrupted inputs." (p 77).

Random Selection vs. Training at the Boundaries

Hommertzheim, Huffman, and Sabuncuoglu (1991) looked at using neural networks to fly an aircraft simulation of the pure pursuit maneuver against a target. It was thought that neural networks could be used to simulate pilot response and that the network could be embedded in an analytical simulation model. The training set chooses did not have many observations of the plane close to the target. When the pursuer was close to converging on the target, there were no examples for the network to have learned from, and the network model failed.

One conclusion Hommertzheim et al. drew from this study was that "Users cannot blindly train the network with a set of training examples without thinking about the characteristics of their problem. Very careful consideration must be given to determine the training set or the resulting network may not be trained for the problem that the user thinks for which it is trained for (p 347). They also made three suggestions concerning training set selection: (p 351)

1. sufficient cases should be included in the training set to capture the input-output transformation that is the subject of the study. Randomly generated training sets tend to produce superior results. While carefully selected non-random sets may work better in some cases, it is easy to bias the network if the training set is not properly selected.
2. functions with discontinuities and steep gradients are difficult to model with the neural network. When modeling higher order functions, these problems may not be obvious.
3. the number of connections should be large enough to properly model the mapping, but not so large that it cannot generalize.

Neural Network Data Preparation

Representative Population Size

A primary question to be answered is how large does a sample need to be to be assured that the sample is representative of the total population.

Train-and-Test Sample Size. The requirements for any unbiased sample error is that the sample data are drawn randomly from the parent population. Because one of our objectives with neural networks is to determine classification parameters based on test cases, and use the parameters to classify new cases. If the samples are biased, or not representative of the total population, one will not be able to generalize the sample based parameters to the general population.

The usual proportions are to divide the sample population, $2/3$ into a training set and $1/3$ into a testing set, provided that once the test cases exceed 1,000, a greater number can be allocated to the training set (Weiss and Kulikowski, 1991, p 30).

Cross-Validation In a k -fold Cross-Validation (Stone, 1974), cases are randomly divided into k mutually exclusive testing partitions of approximately equal size. For a given epoch of network training, a test partition is selected and the remaining cases are used for training the network. Once the train-and-test epoch is completed for one set of data, a different testing partition is selected as the test set. The new set of remaining cases are used for training the network. The process is repeated for k epochs. The average error rates over all k epochs is the cross-validated error rate. Weiss and Kulikowski (1991) suggest that a 10-fold cross-validation is adequate and accurate for most cases of greater than 100 cases.

Cross Validation has an additional advantage when considering the problem of overgeneralization. Cross validation may be used as a tool to indicate the optimum number of hidden nodes or hidden layers by comparing the results of a validation test on a trained network—the alternative with the lowest test error is assumed to be the optimum alternative. Validation test errors sequences that do not increase are an indication that the original number of hidden units are insufficient for generalization to occur and that additional hidden nodes are needed.

Cross validation may also be used to indicate if training should be stopped before convergence is reached. Periodically the network is stopped and tested using a validation test data set. As the network learns, the error on the test set should decrease. Overgeneralization is said to occur when the test error again increases.

While there are no statistical proofs that cross-validation will produce optimal results, there are few other practical tools that may be used (Smith, 1993).

Leaving-One-Out. A special case of Cross-Validation that is very computationally intensive. For a given sample size, n , the neural network is trained on $(n-1)$ cases, and tested on the remaining case. This process is repeated n times, each time using a different case for the test item. The error rate is the average of errors on the single test cases. Although Leave-One-Out can be used with sample sizes numbering in the 100s, because of its computationally intensity Weiss and Kulikowski (1991) suggest that should be used for sample sizes of less than 100 cases.

Procedures for Setting Up Training

Weiss and Kulikowski (1991, p 106) suggest the following procedures be used to ensure proper learning and to avoid local minima.

1. Replicate the training sessions several times on the complete sample. If results are inconsistent, it is likely that a local minimum has been found, and one should decrease the learning rate and/or increase the momentum.
2. Replicate the training sessions several times with test and training sets in an attempt to determine the optimum number of hidden neurodes and layers. In general, the larger the number of hidden units, the longer for training, and the greater the chance of overtraining the neural network. For an equal number of hidden units, two layers will train faster than one layer because of the fewer weights.
3. Once satisfactory training conditions have been established for the complete data, resampling techniques and the Train-and-Test paradigm should be used. For most cases a 10-fold cross-validation will provide reasonable results. Each cycle of the cross-validation training

sets be trained at least five times using different starting states to avoid concerns about local minima and the effects of random starting variables. Resampling must be repeated for each size network tested.

Encoding *a priori* Information

As a possible solution to the problem of Representational Adequacy, Joerding and Meador (1991) suggest that it may be beneficial to encode *a priori* information in feed forward neural networks. They note three basic benefits in doing this (p 848)

1. it may prevent the network from making impossible or implausible predictions
2. it may improve the ability of the network to generalize arising from the constraints themselves, independent of the informational content. Examples can be seen from Kruschke (1989) who used dynamic distributed bottlenecks.
3. it may increase the efficiency of training.

Joerding and Meador (1991) suggested two general approaches for incorporating *a priori* knowledge into specific network constraints:

1. Architecture Constraint - determines a class of feedforward network that satisfy some hypothesis regardless of weight values and independent of training. When using this constraint, since every variation on the network produces a network that satisfies the desired condition, no additional training procedures need to be considered.
2. Weight Constraint - training algorithms are modified by changing the associated output error function in such a way that it optimizes a subset of the desired hypothesis. This occurs when the initial network does not satisfy the desired hypothesis, so constraints are imposed during training.

Joerding and Meador (1991) suggest that prior constraints may be needed in situations of monotonicity and concavity. A monotonic function is one whose slope does not change sign, and a concave (or convex) function has a slope that decreases (increases) as the function argument increases. Economists use concave (convex) functions to describe production functions, diminishing returns, etc.

A quasi-convexity curve is similar to a growth curve, convex to a point, and then concave. They also note that:

In certain circumstances it is desirable for a feedforward network to approximate both a function and its gradient. For a firm maximizing profits, the gradient of the production function equals the relative price of the inputs. In a control problem, it can be crucial that the network closely approximate both the position and the velocity of the process. The Hornik, Stinchcombe, and White (1990) results show that feedforward networks are good candidates for this problem since they can approximate both the surface and the gradient of continuous functions to any desired degree of accuracy. If data on the surface and the gradient are available for training then there is a network architecture that uses all this data and yet has more weights to be estimated. (p 854)

Considerations in Training the Network

Validation of the results of a neural network simulation is necessary for many reasons. As the number of connections and nodes in a network increase, the ability of the network to model the data increases, but the number of degrees-of-freedom also increases. Part of the process of validating results involves analyzing the tradeoffs between modeling ability and repeatability. The following are useful tools:

Pearson's R²

As discussed in the Review of Literature Section, the Pearson's R Coefficient (R²) statistic is often used to assess the accuracy of a model's prediction. The statistic is basically a ratio of explained variation and total variation. When applied to neural networks, Burke (1993) uses the following formula:

$$R^2 = 1 - \frac{SSE (n - 1)}{SST (n - p - 1)} \quad (5-2)$$

Where:

- SSE = Sum of the Squared Errors
- SST = Sum of the Squared Deviations from the Mean
- n = Number of Training Set Examples
- p = number of input nodes (parameters)

Benchmarking

Benchmarking is used with neural networks to judge the efficiency of various alternative network designs. In general, the same data is run under different network configurations to see which design has the lowest error or fastest time to convergence. Tollenaere (1990) suggests that at present, there does not seem to be any consensus on how to benchmark the performance of various training algorithms. Some researchers base their training algorithms purely on theoretical analysis, making nontrivial assumptions about the shape of the error space (Jacobs, 1988). Others run a few simulations on a small number of problems, and subsequently draw their conclusions (Schmidhuber, 1988). Both number and relevance of the benchmark problems, and so the validity of the conclusions are questionable.

Even if the most frequently used problems are indeed valid benchmarks, the fact that every author seems to use different parameter values or different variants of the basic algorithm, often without mentioning what values actually have been used, makes it very difficult and even dangerous to compare one's results with others. (Tollenaere, 1990, p 572)

Tollenaere (1990) also suggests that while there is not any consensus on how to benchmark the performance of various types of training algorithms, there are two basic categories being used. Some benchmarks involve using purely theoretical data for training sets, making major assumptions concerning population characteristics. Others run a few simulations on a small number of problems, subsequently drawing conclusions from the results.

A commonly used theoretical algorithm used is the X/OR parity with a network of one output node that indicates whether an input pattern is even or odd. Tollenaere suggests that if the goal of most neural networks is to develop training algorithms that classify patterns and generalize tasks, these parity problems may not be best suited for testing. Parity problems typically have input patterns that differ by only one bit, and tend to be not generalizable. It is suggested that the an auto-association problem in which the input is the same as the output are the best suited for back propagation.

CHAPTER VI

CONCLUSION AND RECOMMENDATIONS

The purpose of this study was to assess the literature with regards to the use of backpropagation based neural networks using supervised learning design for data analysis. Definitions of Scholarship are first presented to provide a framework for the analysis.

We have identified a paradigm shift in the way neural network analysis is viewed. We noted that Ratcliff (1990) suggested that the analysis of Neural Networks should advance from an *exploration* stage focusing on sifting through data to find relationships, to an *explanation* stage focusing on competing theoretical frameworks and explaining reasons for experimental failures. Ratcliff seems to be suggesting that neural network analysis needs to evolve from an inductive approach to a deductive framework for analysis.

In accomplishing an *explanation* stage analysis using neural networks, it has been suggested that Aristotle's deductive argument (a syllogism) be used as a foundation framework for deductive analysis. There are three major components to a syllogism:

- 1) Major Premise - a statement of reality.
- 2) Minor Premise - a observation of new facts that are assumed to support a statement of reality.
- 3) Conclusion - the suggestion of a theoretical basis for a new relationship between existing statement of facts and a new understanding between existing and proposed statements of reality. The premises are stated in such a way that if the Major and Minor Premises are true, then it follows that the conclusion is also true.

The experimental design is established to test the validity of the minor premise and its relationship to the major premise.

The methodology described in this chapter are designed to address the following questions concerned with using neural network in solving business problems based the use of Neural Networks in an explanation stage analysis. The analysis follows the Review of Literature of the previous chapter:

1. What are the characteristics of a problem that make it suitable for investigation using neural networks? (Problems Suitable for Neural Network Data Analysis)
2. Given the problem, how should the neural network model be designed? (Intelligent System Design Methodology)
3. Given the problem and chosen network design, how should data be prepared for presentation to a neural network for modeling? (Neural Network Data Preparation)
4. How should one proceed with the analysis? (Recommendations)

Each of the questions will be addressed separately. Within each section previously discussed methodologies will be summarized.

Problems Suitable for Neural Network

(Explanation Stage) Data Analysis

We have seen that at a fundamental level, neural networks are good at predicting future events based on past patterns. Additionally, neural networks are good approximators, if not actual imitators of non-linear regressions, discriminate analysis, and other statistical techniques. However, because of the “black box” mechanisms of neural networks they are not in-and-of themselves especially good at establishing causality. A neural network with several inputs and several outputs may exhibit and predict a relationship between variables, but may not necessarily establish the validity of why that relationship exists. Hence, the methodology used in the research method becomes important in explaining results.

For the purposes of this discussion we will categorize the task of data analysis in two broad categories: 1) Causal Research Analysis—the analysis of data to uncover new cause and effect relationships that may extend the leading edge of an academic discipline’s knowledge frontier; and 2) Operational Decision Investigation—the use of data to identify patterns that may be useful in identifying the best course of action in an uncertain environment. The idea for this categorization may

be traced to Conant's discussions in *Science and Common Sense* (1951) where Causal Research Analysis would be close to Conant's definition of Science, and Operational Decision Analysis would be close to his definition of Common Sense. This type of categorization is important because each category has different purposes in the analysis of data and different assumptions used in the preparation of data for analysis.

Causal Research Analysis is similar to the traditional scientific methods of research which focuses on an understanding of a theoretical solution to a research problem. The theoretical structures (or schema) are built upon congruent facts. These structures are systematically tested for internal consistency, and external validity to be sure that there is no contradiction between fact and theory that may invalidate the theoretical structure. These theories are tested and re-tested in laboratory and field conditions designed to substantiate the cause and effect relationships the theory assumes to exist. Through the use of controlled experiments, one attempts to systematically rule out variables that are not part of a hypothesized theory as possible causes of a phenomenon under investigation. When attempting to establish causality, the experimental process used is extremely important—a hypothesis is developed and then the deduced implications of the hypothesis are tested. The focus of Causal Research Analysis is on things that may be observed and tested; the fundamental legitimacy of the results are based upon the processes used to arrive at the conclusions. The basic aim of research is the development of a generalized theoretical explanation for the way things are.

Operational Decision Investigation is similar to the development of a systemic rule-of-thumb that works within an existing paradigm. While this type of decision investigation uses theory, assumptions are made concerning internal consistency and external validity that may be valid under some conditions, and may be invalid under other conditions. The testing of these rules-of-thumb occurs in a selective fashion, with one being concerned with "what works" and what "does not work" in specific situations. Stereotyping would be an example of an Operational Decision Investigation tool that would not be considered a valid Causal Research Analysis tool—one tends to accept preconceptions as a component of the existing paradigm. These theories or rules-of-thumb are often expressed in metaphysical terms that are difficult to test. The fundamental legitimacy of Operational Decision

Investigation is based upon the accuracy of the results rather than the processes used to develop the results. The basic aim of decision analysis is the development of processes to predict future events. The primary focus of Operational Decision Investigation is to uncover relationships (which may or may not be causal) by looking for patterns that will suggest the outcome of specific behaviors. When attempting to identify potential areas of causality, data is examined to see if patterns exist, and a hypothesis is presented to explain relationships (this only suggests and does not verify causality).

Neural Networks may be a good choice in an analytical tool when dealing with Operational Decision Investigation problems where results are the primary focus. When using neural networks as a tool for Operational Decision Investigation problems, an inductive reasoning approach may be legitimately used, since it is the result and not the cause that is important. However, because the process of analysis is of major importance in Causal Research Analysis, Neural Networks must be used with caution. Because of a neural network's black-box approach to data analysis, when using neural networks to establish causality, a deductive approach must be used. This means that the problem must be defined explicitly in terms of a model upon which the neural network will be constructed to test the hypothesis presented.

Intelligent System Design Methodology

When using Neural Networks in the process of either an operational or causal analysis, several *a priori* implementation decisions need to be made with regards to how the neural network model should be constructed and how the network should interact with the data. Some of these network design decisions may in effect encode *a priori* information that may bias the results. In an operational analysis, the *a priori* information may not be a significant factor. However, in a causal analysis, unless the *a priori* information is explicitly stated in the description of the problem, results are tainted.

The following are several neural network design parameters that should be examined carefully and an assessment made as to their significance before the analysis of data has started:

Number of Network Nodes

The greater the number of nodes, the greater the number of connections between the nodes. We have seen that the number of connections between nodes in a neural network can be related to degrees-of-freedom, which in turn effect the assumption of statistical validity of the neural model. Baum and Haussler (1989) suggest a rule-of-thumb that says the appropriate number of training examples is approximately the number of weights between nodes of the network, times the inverse of the accuracy parameter ϵ . For example, if we desired an accuracy level of 95%, which corresponds to $\epsilon = 0.05$, we would need 100/5 or 20 times as many training examples as there are weights in the network.

Sequential or Random Data

Selection and Presentation

Traditional statistics suggests that, unless there is a compelling reason otherwise, a random selection of exemplars is desired to guard against bias and to help insure generalizability. Logically, the use of time series data represents one compelling reason for an ordered presentation of data, but the selection of the sequenced data may still be random within each time frame. Weiss and Kulikowski (1991, p 99) have found that for non-time sensitive data, random presentations tend to produce better results.

However, in situations where neural networks are used with conflicting data, Ratcliff (1991) noted that "items encoded into groups are significantly more resistant to forgetting than items learned individually" (p 291), suggesting that this difference is due to what amounts to structural differences in how the network is organized. It would seem that Ratcliff is suggesting that by encoding items into groups, because of the way back propagation assigns weights, the network trains as if it were a quasi probabilistic network. This would also suggest that for complex problems, when using sequential presentation, multiple layers on the network would perform better than networks with a single hidden layer.

We have seen that Hommertzheim, Huffman, and Sabuncuoglu (1991) have suggested that "Users cannot blindly train the network with a set of training examples without thinking about the characteristics of their problem. Very careful consideration must be given to determine the training set or the resulting network may not be trained for the problem that the user thinks that it is trained for." (p 347). They also made three suggestions concerning training set selection: (p 351)

1. sufficient cases should be included in the training set to capture the input-output transformation that is the subject of the study. Randomly generated training sets tend to produce superior results. While carefully selected non-random sets may work better in some cases, it is easy to bias the network if the training set is not properly selected.
2. functions with discontinuities and steep gradients are difficult to model with the neural network. When modeling higher order functions, these problems may not be obvious.
3. the number of connections should be large enough to properly model the mapping, but not so large that it cannot generalize.

While the Hommertzheim, Huffman, and Sabuncuoglu technique may be acceptable in situations of Operational Decision Investigation, there are clear problems of bias when being applied to Causal Research Analysis situations. For causal analysis, it is suggested that stratified data sets be used to establish an adequate number of exemplars at the decision boundary areas. This means that the decision problem will need to be stated in such a way that these boundary areas are identified *a priori*, and that sufficient logical support are provided for this identification. To avoid the problem of bias, a random selection of exemplars within the stratified data set should be conducted. If a non-random presentation of exemplars is required, a justification should be provided before the analysis is conducted.

Optimum Number of Network Layers

The optimum number of hidden layers that should be used in a neural network seems to be problem specific and therefore of concern in causal research analysis. There are several proofs that suggest various constructions of neural networks are universal approximators to varying degrees (Blum

and Li, 1991; Hornik, Stinchcombe and White, 1990; Kammerer and Kupper, 1990; Baldi and Hornik, 1989); and suggest that neural networks should be constructed with as few hidden layers as possible.

However, multiple layers in a neural network are required to approximate complex decision regions (Weiss and Kulikowski, 1991). In addition, Kolen and Goel (1991) have shown that traditional neural networks are not capable of performing high level abstractions. This means that neural networks can learn to mimic situation-specific responses, but have difficulty generalizing those responses. Since most data analysis used sample data, the problem of generalizability is of concern (discussed further in the section on Testing for Generalizing).

Kolen and Goel also have shown how using *a priori* information, a neural network may be constructed to perform high level abstractions through a form of pre-compiling the network design to match a pre-conceived notion of reality. While this technique may be acceptable in situations of Operational Decision Investigation, there are clear problems of bias when being applied to Causal Research Analysis situations.

In situations of causal analysis, it is therefore suggested that when using a neural network with more than one hidden layer, especially if the layers are partially connected, that extensive analysis be devoted to a justification for the *a priori* design and how it related to the problem definition. It is suggested that lack of attention to this explanation may lead to unexpected bias and an inability to generalize the results from a sample to the whole population.

Overfitting

Overfitting is a problem specific to the nature of neural networks that is generally not a concern when using traditional statistics. Overfitting is a problem that is of concern for both operational and causal analysis methodologies. Overfitting is said to occur when the neural network tries to account for random events in a systematic way. Because back propagation neural networks are universal approximators in most cases, given enough time and enough hidden nodes, a network may be trained to fit any data set. Overfitting occurs when a model fits the training data with minimum error, but will not fit newly provided data from the same source—there is poor generalization.

Sarle (1995c) suggests a stopped training method to prevent overfitting of data. In the stopped training method, neural network training is periodically stopped (and network weights frozen) so that the network may be tested with a different data set. Validation errors are measured. Under normal network training, validation errors should decrease as the network better models the data set. Training is stopped with validation errors start increasing, suggesting that overfitting is occurring. A disadvantage to this procedure when applying neural networks to Causal Research Analysis situations is that traditional statistical theory does not apply.

Local Minima

The problem of Local Minima is another problem specific to the nature of neural networks. Local Minima is a problem that is of concern for both operational and causal analysis methodologies. A neural network is said to have reached an optimum state when error space defined by the network model is at a minimum—when random adjustments to weights connecting network nodes have little effect on the output of the network. As an analogy, one may visualize the result of the network model of all possible combinations of the connecting weights as a mountain range with peaks and valleys. The objective is to find the combination of weights that produces the lowest valley (least error, most accurate representation). A Local Minima is said to occur when the neural network model is in a valley, but the valley is not as low as one over the next range. A network modeling a localized minima may adequately model the sample data, but may not generalize to the general population.

Testing for Generalization

It has been suggested that there may be occasions where neural networks require a sequential presentation of exemplars, and more than one hidden layer in the network design. The problem of overfitting has also been identified as a possible obstacle to generalization. These situations may result in unexpected bias and an inability to generalize the results to a total population. We have seen that Peterson and Hartman (1989) define generalization with regards to neural networks as "the response of a

network, after some amount of training, to novel (unlearned) inputs" (p 481). They go on to suggest two basic ways to test generalization:

1. Continuous Learning - uses the entire training set. Each time the network is presented with a case, it is first tested for generalization on that pattern. The disadvantage here is that there is not a clear distinction between learning and generalization.
2. Fixed Training - Cases are separated into learning and training sets. After learning the training set, the network is tested on a set of cases that the network has not seen before.

Using Self-Learning System over Time

As a self learning system, neural networks may have an advantage over traditional statistical systems. When using traditional statistics, there is a basic assumption that the causal events of the past will be the same causal events in the future—the objective of the statistical study is to understand those causal events within the event paradigm. The choice of tests of neural network generalization depends partly on the event paradigm. Fixed training cases are appropriate when it is assumed that there are no paradigm shifts, i.e., that the causal events of the past will be the same in both the present and the future. Because of a Neural Network's pattern matching ability, they may be more useful than traditional statistics in identifying changing causal relationships caused by paradigm shifts in the external environment.

Procedures for Conducting Neural

Network Training

When have seen that Weiss and Kulikowski (1991, p 106) suggest the following procedures be used to ensure proper learning, help insure generalization, and to avoid local minima.

1. Replicate the training sessions several times on the complete sample. If results are inconsistent, it is likely that a local minimum has been found, and one should decrease the learning rate and/or increase the momentum.

2. Replicate the training sessions several times with test and training sets in an attempt to determine the optimum number of hidden neurodes and layers. In general, the larger the number of hidden units, the longer the training, and the greater the chance of overtraining the neural network. For an equal number of hidden units, two layers will train faster than one layer because of the fewer weights.
3. Once satisfactory training conditions have been established for the complete data, resampling techniques and the Train-and-Test paradigm should be used. For most cases a 10-fold cross-validation will provide reasonable results. Each cycle of the cross-validation training sets should be trained at least five times using different starting states to avoid concerns about local minima and the effects of random starting variables. Resampling must be repeated for each size network tested.

Neural Network Data Preparation

When analyzing data, the use of Neural Networks establish a different set of problems than the use of traditional statistics. For example, we have seen that Neural Networks work best when there are many exemplars in boundary areas (the more exemplars the less fuzzy the decision area.) However, one of the principles of the process of research is the elimination of bias through the random sampling of exemplars. Another problem in the implementation of Neural Networks involves degrees of freedom – the more network nodes used, the better the modeling ability of the neural network, the more the degrees of freedom, and the larger the data set required to compensate for the increase in degrees of freedom.

When using Neural Networks in the process of analysis, several *a priori* implementation decisions need to be made concerning how data is to be prepared for use by the neural network tool. Because of the binary nature of Neural Networks, for optimum performance, data format requirements for Neural Network tools are different from data format requirements for traditional statistical tools. Within the Review of Literature is a review of concepts of Back propagation based Neural Network data analysis and interpretation that were found useful in a wide variety of academic disciplines. Considering the information presented in the Review of Literature, conclusions are drawn on how to

prepare data for model analysis using back propagation neural networks and apply them to solving business problems.

Use of Binary Data

In neural networks the representation of discrete class type variables may be significantly different from the representation of quantitative variables. Because the power of a neural network is its ability to recognize patterns, and because a reasonably sized network can store only a few patterns, neural networks tend to do better when quantitative data is transformed into distributed data (a set of qualities that may be used to represent several different items). For example, with nominal numbers, Crooks (1992) suggests that an input node should be assigned for every possible value. Since ordinal numbers imply a ranking or ordering, not a magnitude, a thermometer code type of classification is a useful way of representing data. Since Neural Networks have a bias towards large numbers, Sarel, (1994) suggests that interval (continuous) variables should be normalized or scaled. Smith (1993) suggests that non-linear data should be transformed to linear data.

A disadvantage of using binary data representations is that high precision requires a large number of input nodes, which leads to a large number of connections and degrees of freedom. For research data sets not containing a large number of observations (where degrees of freedom may even exceed the number of observation) the validity of the results may be seriously questioned.

Normalization of Data

Some inputs may assume values measured in the thousands, while others may all be small integers. Several have found performance can often be improved by normalizing the data before training so that, for example, all input variables fall within the range from 0 to 1. (Sutton and Barto, 1981; Kruschke, 1989; Weiss and Kulikowski, 1991; Reyneri and Filippi, 1991)

Adding Noise to the Training Set

Many have found that adding noise to the training set has been shown to improve the performance of modeling systems in general, (Matyas, 1965) and neural networks in particular (Baba, 1989; Sietsma and Dow, 1991). It has been noted that the selection of the learning rate (η) and the variance to the Gaussian distribution had a significant effect on the learning speed. Noise seems to cause more hidden units to be used in the network. Networks rarely use all units independently. By using more of the available units, the neural network will be more stable and robust network. Training with noise has the effect of improving the ability of the network to correctly classify phases that were not in the training set, as well as improving the classification of noise corrupted inputs. Training without noise in a network with many hidden units will tend to produce non-generalizable results.

While this technique may be acceptable in situations of Operational Decision Investigation, there are clear problems of bias when being applied to Causal Research Analysis situations. In causal research models, the practice of adding noise to the training set should be clearly stated and included in as a fundamental component of the problem formulation.

Conflicting Data

When two or more input patterns have entirely different outputs, a condition of conflicting data is said to exist. Versaggi (1995) suggests that conflicting data should not be eliminated, but that the number of inputs should be increased to account for the problem. However, the greater the number of inputs, the more the network will pattern match specific examples, rather than generalize.

Recommendations

The nature of academic research (as opposed to practical applications) is the domain of Causal Research Analysis. When using Neural Networks in the process of analysis, several *a priori* implementation decisions need to be made with regards to how the neural network model should be constructed (network modeling) as well as how the network should interact with the data and how the

data should be presented to the network (data preparation). We will make recommendations in each of these areas, as well as for the general experiment.

Network Modeling

Because a principal of Causal Research Analysis is that the appropriate method should be followed to avoid the possibility of bias, the use of *a priori* decisions in Causal Research Analysis situations should be used with caution. Because the benefit of an Operational Decision Investigation is the success of the prediction, the use of *a priori* decisions in these non-scientific areas of inquiry are allowed, to the extent that they are successful.

Neural Networks are good at pattern recognition and pattern matching. While Neural Networks are good at recognizing relationships, they are not particularly useful in situations where causality needs to be either determined or verified. Because Neural Networks perform better using either binary or nominal numbers, the use of neural networks in statistical analysis requires much data preparation to transform data into a format that is usable by the neural network.

Because of the black-box nature of the operation of neural networks, when they are being used for establishing causality, deductive research methodologies should be used. Additionally, *a priori* conditions need to be examined for potential threats to bias and generalizability assumptions.

Specifically:

- 1) for the equivalent of an .05 level of confidence, there needs to be about 20 times more training exemplars than there are weights connecting nodes in the network.
- 2) the problem identification should include a discussion of how data may be stratified around theoretical decision points, with sufficient exemplars provided to provide the network a clear demarcation of the decision space.
- 3) the problem identification should also include a theoretical discussion of reasons for conflicting data, and such conflicting data should be presented in a stratified format.

4) problem identification should be approached from the standpoint of identifying complex decision regions or areas where the performance of high level abstractions are required; these should be modeled *a priori* using multiple hidden layers and/or partially connected network nodes.

5) design methodology should include steps to assure overfitting and local minima problems do not compromise generalizability of the results.

Data Preparation

It may therefore be suggested that the advisability of using Neural Networks for analysis depends in part on the purpose of the analysis and the amount of data available. Many Causal Research Analysis situations are limited by both the amount of data available and the form in which the data is available. Because effective use of neural networks in modeling often requires multiple nodes, which also increase the degrees of freedom. To the extent that binary or nominal number representations also increases the number of input nodes, the degrees of freedom also increase. The degree of freedom issue tends to suggest that Neural Networks may require more data than traditional statistics to maintain an equal level of statistical validity.

An Operational Decision Investigation, on the other hand, does not have the strict methodological requirements of a Causal Research Analysis. Many of the statistical tools in time series analysis, such as moving average and linear trends, do not require the establishment of causality. Additionally, many organizations maintain large historical databases (knowledge warehousing) that will provide sufficiently large amounts of data to overcome the degrees of freedom problems of smaller databases.

These factors tend to suggest that neural networks as a stand-alone analysis tool will be best suited for Operational Decision Investigation situations and have limited use in Causal Research Analysis situations where the establishment of cause and effect relationships are important. In a research environment, Neural Networks may be useful in areas where inductive reasoning is accepted, typically new areas where massive amounts of data needs to be sifted and the determination of a possible

relationships with events are more important than the determination of a causal link between a variable and an event.

Specifically, with data preparation, *a priori* conditions need to be examined for potential threats to bias and generalizability assumptions:

- 1) where possible, binary data should be used; recognizing that there may be a trade off between the use of binary data and degrees-of-freedom of the network. Where possible, thermometer type codes should be used to classify data representations
- 2) interval (continuous) data should be normalized, non-linear data should be transformed into linear data.
- 3) noise should be added to the training set, but a detailed analysis should also be presented *a priori* in the description of the experimental design.

Using Neural Networks in Causal Research

Because of the fundamental differences between traditional statistics and neural network as tools for research, it is suggested that, to avoid the appearance of bias, many of the existing generally accepted protocols currently used with traditional academic experimental design and methodology may need modifications when used with neural networks. To the extent that academic research requires a statement of causation, deductive methodologies should be used.

Causal research using Neural Networks should only be done using deductive methodologies that will also assure that the results are generalizable. Because neural networks recognize patterns and predict relationships between data objects similar to parametric statistics,

Because neural networks have non-traditional threats to bias and generalization assumptions, (sequential vs. random data selection and presentation, number of hidden layers, uncollected nodes, overfitting, and local minima, to name a few) it is highly recommended that the test and train paradigm be repeated more extensively on new data sets to assure generalization. The new data set should not have been used in any previous analysis endeavors.

Summary of Findings

This study attempts to answer the following question:

Can Back propagation Neural Networks be used in Research to establish causality between variables?

While the use of Back propagation Neural Networks have inherent problems in establishing causality, it is suggested, by using the experimental design methodologies that are based upon deductive reasoning as described in this section, that Back propagation Neural Networks can be used as a tool to establish causality in some cases. However, clear hypothesis and comprehensive experimental design structures need to be established before analyzing the results of a network to avoid threats to assumptions of bias and generalizability.

Recommendations for Future Research

Because of the fundamental differences between traditional statistics and neural network as tools for research, it is suggested that many of the existing generally accepted protocols currently used with traditional academic experimental design and methodology may need modifications when used with neural networks. Specifically additional research needs to be performed in areas that may introduce bias in the results—procedures need to be developed for dealing with:

- 1) outliers and other errors in the data set. Because neural networks are significantly influenced by variance and magnitude of the data set, outliers may have a larger effect in neural networks than when using traditional statistics. To avoid bias, traditional statistics suggests that outliers not be removed from data sets. When using neural networks, it may be necessary to use other logical assurances that bias is not introduced and remove the outliers. The nature of these logical assurances needs to be developed.

- 2) random selection of data exemplars. The self-learning nature of Neural Networks involves pattern recognition at the boundaries. For the network to be able to model changes of state, a significant number of exemplars need to be available to model the boundary areas. With small data sets, this may mean that an insignificant number of exemplars may be available at the boundary, leaving the modeling

of the boundary area fuzzy and unpredictable. Methodologies that will allow for a significant number of boundary exemplars to be specified while maintaining an unbiased data sample need to be developed.

Recommendations in the Network Modeling and Data Preparation sections are based on a synthesis of experimental conclusions across several disciplines. It is not know how robust these recommendations are when taken as a complete unit with respects to threats to bias, validity, and generalizability assumptions.

REFERENCES

- Adams, J. L. (1979). *Conceptual Blockbusting*, 2nd ed. W. W. Norton & Co., New York.
- Ahituv, N., & Neumann, S. (1990). *Principles of Information Systems for Management*, Third Edition. Dubuque, IA: Wm. C. Brown Publishers.
- Allman, W. F. (1989). *Apprentices of Wonder: Inside the Neural Network Revolution*. New York: Bantam Books
- Almedia, L. B., & Silva, F. M. (1990, February) Acceleration techniques for the back propagation algorithm. In Lecture Notes in Computer Science, Volume 1: Parallel Architectures and Languages Europe, G. Goos and J. Hartmanis, Eds. New York: Springer-Verlag, 412, 110-119.
- Alter, S. (1992). *Information Systems: A Management Perspective*. Reading, MA: Addison-Wesley Publishing Co.
- Baba, N. (1989). A new approach for finding the global momentum error function of the neural networks. Neural Networks, 2(5) 367-373.
- Baldi, P., & Hornik, K. (1989). Networks, 2(1), 53-58.
- Barschdorff, D, & Bothe, A (1991). Adaptive condensed nearest neighbor analysis using neural and statistical classifiers: A Comparison. Fourth International Conference on Neural Networks and Applications, November 4-8, as quoted in Barschdorff, Monostori, Ndenge, and Wostenkujler (1991)
- Barschdorff, D.; Monostori, L.; Ndenge, A. F.; & Wostenkujler, G. W. (1991) Multiprocessor Systems for Connectionist Diagnosis of Technical Processes. Computers in Industry, 17(2,3), 131-145.
- Barsley, M. F.; & Sloan, A. D. (1988). A better way to compress images. Byte, 215-223. As quoted in Maugeot, et al., 1991.
- Baum, E. B., & Haussler, D. (1989). What Size Net gives Valid Generalization? Neural Computation, 1, 151-160.
- Blum, E. K., & Li, L. K. (1991). Approximation Theory and Feedforward Networks. Neural Networks, 4, 511-515.
- Borghese, N. A. & Arbib, M. A. (1995). General Temporal Sequences Using Local Dynamic Programming. Neural Networks, 8(1), 39-54.
- Bottou, L.; Soulie, F. F.; Blanchet, P.; & Lienard, J. S. (1990). Speaker - Independent Isolated Digit Recognition: Multilayer Perceptrons vs. Dynamic Time Warping. Neural Networks, 3 453-465.
- Boyer, E. L. (1990). Scholarship Reconsidered: Priorities of the Professoriate. Princeton, NJ: The Carnegie Foundation for the Advancement of Teaching.

- Burke, L.. (1993, March/April). Assessing a Neural Network Validation Procedures. PC AI. 20-24.
- Caelli, T. M.; Squire, D. M.; & Wild, T. P. J. (1994). Model-Based Neural Networks. Neural Works, 6: 613-625.
- Capron, H. L. (1995). Essentials of Computing, Second Edition. Redwood City, CA: The Benjamin/Cummings Publishing Company, Inc.
- Carpenter, W. C., & Hoffman, M. E. (1995, March). Training Backprop Neural Networks. AI Expert, 10(3), 30-33.
- Caudill, M. (1991a, January). Neural Network Training TIPS and Techniques. AI Expert, 56-61
- Caudill, M. (1991b, July). Avoiding the Great Back propagation Trap. AI Expert., 29-35.
- Caudill, M. (1992). The View from Now. AI Expert, June, 24-31.
- Caudill, M., & Butler, C. (1992a). Understanding Neural Networks, Volume 1: Basic Networks. Cambridge, MA: Bradford Book.
- Caudill, M., & Butler, C. (1992b). Understanding Neural Networks, Volume 2: Advanced Networks. Cambridge, MA: Bradford Book
- Chance, D. C.; MacLin, O. H.; & Lykins, S. (1993, March 23). Multiple Regression and Artificial Neural Networks Compared to a Multivariate Nearest Neighbor Technique. Presented at the Graduate Colloquium, University of Central Oklahoma.
- Chandrasekaran, B., & Goel, A. (1988, May-June). From numbers to symbols to knowledge structures: Artificial intelligence perspectives on the classification task. IEEE transactions on System, Man and Cybernetics, 18(3), 415-424. In Kolen & Goel.
- Churchland, P. S. (1986). Neurophilosophy: Towards a Unified Science of the Mind-Brain. Cambridge, MA: MIT Press. As quoted in Allman (1989).
- Cichocko, A., & Unbehauen, R. (1992, February). Neural Networks for Solving Systems of Linear Equations and Related Problems. IEEE Transactions on Circuits and Systems - I Fundamental Theory and Applications, 39(2), 124-138.
- Conant, J. (1951). Science and Common Sense. New Haven: Yale University Press, as found in Kerlinger, F. N. (1986). Foundations of Behavioral Research, Third Edition. Ft. Worth: Hold, Rinehart, and Winston, Inc. (4)
- Coats, P. K.; & Fant, L. F. (1991-92, Winter). A Neural Network Approach to Forecasting Financial Distress. The Journal of Business Forecasting, 10(4), 9-12.
- Crooks, T. (July, 1992). Care and Feeding of Neural Networks. AI Expert. 36-41.
- Decker, K. M., & Focardi, S. (1995). Technology Overview: A Report on Data Mining, CSC TR-95-02. SCSC-ETH, Swiss Scientific Computing Center. Downloaded 8-8-95 from: <ftp.cscs.ch/pub/CSCS/techreports/1995/CSCS-TR-95-02.ps.gz>

Denker, J. Schwartz, D., Wittner, B., Sola, S., Hopfield, J. Howard, R., & Jackei, L (1989). Automatic learning, Rule extraction, and generalization. Complex Systems, 1, 877-922.

Devos, M. R., & Orban, G. A. (1988). Self adaptive back propagation. Proceedings NeuroNimes 1988. E. Z., Naterre, France. As quoted in Tollenae (1990).

Dror, I. E.; Zagaeski, M.; & Moss, C. F. (1995). Three-Dimensional Target Recognition via Sonar: A Neural Network Model. Neural Networks, 8(1), 149-160.

Fahlman, S. E. (1988). Faster-learning variations on back-propagation: An Empirical Study. Proceedings of the 1988 Connectionist Models Summer School, 00. 38-51. As quoted in Springer, P. L., and Gulati, S. (1995). Parallelizing the Cascade-Correlation Algorithm using Time Warp. Neural Networks, 8(4) 571-577.

Funahashi, K. (1989). On the approximate realization of continuous mapping by neural networks. Neural Networks, 2, 183-192. In Hommertzhaim, Huffman, and Sabuncuogly (1991)

Gardner, H. (xxx). The Mind's New Science: A History of the Cognitive Revolution. As quoted in Allman (1989)

Gaynier, R. J., & Downs, T. (1995). Sinusoidal and Monotonic Transfer Functions: Implications for VC Dimension. Neural Networks, 8(6), 901-904.

Guiver, J. P., & Klimasauskas, C. C. (1991, July/August). Applying Neural Networks, Part IV: Improving Performance. PC AI, 34-41.

Gori, M., & Tesi, A. (1992, January). On the Problem of Local Minima in Back propagation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 14(1), 76-85.

Grossberg, S. (1969). On learning and energy-entropy dependence in recurrent and nonrecurrent signed networks. Journal of Statistical Physics, 48, 105-132.

Hanke, J. E., & Reitsch, A. G. (1992). Business Forecasting, 4th Edition. Needham Heights, MA: Allyn and Bacon.

Haussler, (1989). Generalizing the PAC model for neural net and other learning applications. (UC Santa Cruz Computer Research Laboratory Technical Report UCSC-CRL-89-30). In White (1990).

Hebb, D. O. (1949). Organization of Behavior. New York: Wiley.

Hillman, D. (1991). Knowledge Systems Based on Cascading Neural Nets. AI Expert, 46-53.

Hinton, G. (1987, December). Learning translation invariant recognition in a massively parallel network. In Lecture Notes in Computer Science. Volume 1: Parallel Architectures and Languages Europe. G. Goos and J. Hartmanis, Eds. New York: Springer-Verlag.

Hommertzhaim, D.; Huffman, J.; & Sabuncuoglu, I. (1991). Training an Artificial Neural Network the Pure Pursuit Maneuver. Computers and Operations Research, 18(4), 343-353.

Hopfield, J. J. (1982, April). Neural networks and physical systems with emergent collective computational abilities, Proc. Natl. Acad. Sci., 81:3088-92. As quoted in Allman, (1989).

Hornik, K.; Stinchcombe, M.; & White, H. (1990). Universal Approximation of an Unknown Mapping and its Derivatives Using Multilayer Feedforward Networks, Neural Networks, 3, 551-560.

Hornik, K. (1991). Approximation Capabilities of Multilayer Feedforward Networks. Neural Networks, 4, 251-257.

Hutchinson, W. (1987). Personal communication as quoted in Smith, Murray (1993, p 157). Neural Networks for Statistical Modeling. New York: Van Nostrand Reinhold.

Hutchinson, S. E., & Sawyer, S. C. (1992). Computers: The User Perspective. Homewood, IL: Irwin.

Ito, Y. (1991). Approximation of Functions on a Compact Set by Finite Sums of a Sigmoid Function without Scaling. Neural Networks, 4, 817-826.

Jacobs, R. A. (1988). Increased rates of convergence through learning rate application. Neural Networks, 1(4), 295-307. As quoted in Silverman & Noetzel (1990) and Tollenaere (1990).

Joerding, W. H., & Meador, J. L. (1991). Encoding A Priori Information in Feedforward Networks. Neural Networks, 4, 847-856.

Kammerer, B. R., & Kupper, W. A. (1990). Experiments for Isolated--Word Recognition with Single- and Two-Layer Perceptrons. Neural Networks, 3, 693-706.

Kempka, A. A. (1994a, June). Activating Neural Networks: Part I. AI Expert. (6), 33-37.

Kempka, A. A. (1994b). Activating Neural Networks: Part II. AI Expert. August 9(7), 42-49.

Kerlinger, F. N. (1986). Foundations of Behavioral Research, Third Edition. Ft. Worth: Hold, Rinehart, and Winston, Inc.

Key, J. P. (1990). Research Design. Stillwater, Oklahoma, Oklahoma State University.

Kimmel, D. (1991, October). Neural Nets inch toward mainstream. System 3x/400, 19(10), White Plains, NY: IBM, 53-58.

Kimoto, T.; Asakawa, K; Yoda, M.; & Takeoka, M. (1990). Stock Market Prediction System with Modular Neural Networks. Proceedings of the International Joint Conference on Neural Networks, 1, 1-6. As quoted in Cnang (1992, p 3).

Klimasauskas, C. C. (1989, Spring). Neural Networks: A New Technology for Information Processing. Data Base, 21-23. as quoted in Cnang (1992, p 3).

Klimasauskas, C. C. (1991a, May/June). Applying Neural Networks, Part III: Training a Neural Network. PC AI, 20-24.

Klimasauskas, C. C. (1991b, May/June). Applying Neural Networks, Part V: Integrating a Trained Network into an Application. 36-41.

Klimasauskas, C. C. (1991c). Applying Neural Networks, Part VI: Special Topics. PC AI, May/June, 46-49.

Kohonen, T. (1984). Self-organizing and associative memory (2nd ed). Berlin: Springer-Verlag. As quoted in Silverman and Noetzel, 1990.

Kolen, J. F., & Goel, A. K. (1991, March-April). Learning in Parallel Distributed Processing Networks: Computational Complexity and Information Content. IEEE Transactions on Systems, Man, and Cybernetics, 21(2), 359-367.

Kosko, B., & Isaka, S. (1993, July). Fuzzy Logic. Scientific American, 76-81.

Kroenke, D. M. (1989). Management Information Systems. New York: Mitchell McGraw Hill

Kruschke, J. K. (1989). Creating local and distributed bottlenecks in hidden layers of back propagation networks. In Proceedings 1988 Connectionist Models Summer School. D. Touretzky, G. Hinton, and T. Sejnowski, eds. San Mateo, CA: Morgan Kaufmann, pp. 120-126.

Kruschke, J. K., & Movellan, J. R. (1991, January/February). Benefits of Gain: Speeded Learning and Minimal Hidden Layers in Back-Propagation Networks. IEEE Transactions on Systems, Man, and Cybernetics, 21(1) 273-280.

Kuhn, Thomas S. (1970). The Structure of Scientific Revolutions. Chicago: The University of Chicago Press.

Kung, S. Y., & Hwang, J. N. (1988). An algebraic Projection Analysis for Optimal Hidden Units Size and Learning Rates in Back-Propagation Learning. Proceedings of the IEEE International Conference on Neural Networks II, New York: pp. 363-370.

Kvanli, A. H.; Guynes, C. S.; & Pavur, R. J. (1989). Introduction to Business Statistics. St. Paul: West Publishing Company.

Lapedes, A., & Farber, R. (1987). Nonlinear Signal Processing Using Neural Networks: Predicting and System Modeling. Preprint LA-UR-87-2662: Los Alamos National Laboratory.

Laudon, K. C., & Laudon, J. P. (1991). Management Information Systems, A Contemporary Perspective, Second Edition. New York: Macmillan Publishing Company.

Lawrence, J. (1991a). Introduction to Neural Networks, 3rd edition. Grass Valley, CA; California Scientific Software.

Lawrence, J. (1991b, November). Data Preparation for a Neural Network. AI Expert, 34-41.

Lippmann, R. (1987, April). An introduction to computing with neural nets. IEEE ASSP Magazine, 4-21.

Lucas, H. C. (1990). Information Systems Concepts for Management, 4th edition. New York: Mitchell McGraw-Hill.

Mantas, J. (1987). Methodologies in pattern recognition and image analysis. Pattern Recognition, 20, 1-6. As quoted in Silverman and Noetzel (1990).

Matyas, J. (1965). Random optimization. Automation and Remote Control, 26, 246-253. As quoted in Baba (1989).

McClelland, J. L., & Rumelhart, D. E. (eds). (1986). Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 2, Psychological and Biological Models. Cambridge, MA: Bradford Books/MIT Press. As quoted in Allman (1989).

Makridakis, S. (1986). "The Art and Science of Forecasting," International Journal of Forecasting, Vol. 2, p. 17.

McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. As quoted in Allman (1989)

Miller, M. D. (1985). Principles and a Philosophy for Vocational Education. Columbus, OH: The National Center for Research in Vocational Education.

Minsky, M. L., & Papert, S. (Expanded 1988 edition). Perceptron. Cambridge, MA: MIT Press.

Moss, F., & Wiesenfeld, K. (1995, August). The Benefits of Background Noise. Scientific American, 273(2) 66-69

Mougeot, M.; Azencott, R.; & Angeniol, B. (1991). Image Compression with Back Propagation: Improvement of the Visual Restoration using Different Cost Functions. Neural Networks, 4, 467-476.

Movellan, J. R. (1987). Self-regulating temperature in back propagation networks. Paper presented at the Ninth Annual Berkley-Stanford Conference. Berkeley, CA: As quoted in Kruschke and Movellan (1991).

Norris, D. (1990). How to build a Connectionist idiot (savant). Cognition, 35, 277-291.

O'Brian, J. A. (1990). Management Information Systems: A Managerial End User Perspective. Homewood, IL: Irwin.

Odom, M. D., & Sharda, R. (1989). A Neural Network Model for Bankruptcy Prediction. Proceedings of IEEE International Conference on Neural Networks. II 163. As quoted in Cnang (1992, p 3).

Oja, E. (1982). A simplified neuron model as a principal component analyzer. Journal of Mathematical Biology, 15, 267-273.

Parker, D., & Werbos. (1985). Learning-Logic, Report TR-47, Massachusetts Institute of technology, Center for Computational Research in Economics and Management Science. Cambridge, MA: As quoted in Neural Computing, Neural Ware Inc., 1993.

Parker, C. S. (1989). Management Information Systems: Strategy and Action. New York: McGraw Hill.

Patuwo, E.; Hu, M. Y.; & Hung, M. S. (1993, July/August). Two-Group Classification Using Neural Networks. Decision Sciences, 24(4), 825-845.

Peterson, C.; & Hartman, E. (1989). Explorations of the mean field theory learning algorithm. Neural Networks, 2(6) 475-494.

Plaut, D. C., & Hinton, G. E. (1987). Learning sets of filters using back-propagation. Computer Speech and Language, 2, 35-61. As quoted in Ye, et al.

Plunkett, K., & Marchman, V. (1991). U-Shaped learning and frequency effects in a multilayered perceptron: Implications for Child language acquisition. Cognition, 38, 43-102.

- Prechelt, L. (1995). comp.ai.neural-nets FAQ, parts 1 through 7. URL: <http://www.ipd.ira.uka.de/~prechelt/FAQ/neural-net-faq.html>. Maintained by: prechelt@ira.uka.de (Lutz Prechelt, Institut fuer Programmstrukturen und Datenorganisation, Universitaet Karlsruhe, Germany. Last modified, 1995/04/28.
- Pugh, G. A. (1989). Synthetic Neural Networks for Process Control. Computers & Industrial Engineering, 17(1-4) 24-26.
- Pugh, G. A. (1991). A comparison of Neural Networks to Statistical Process Control Computers & Industrial Engineering, 21(1-4) 253-255.
- Ratcliff, R. (1990). Connectionist Models of Recognition Memory: Constraints imposed by Learning and Forgetting Functions. Psychological Review, 97(2), 285-308.
- Renolds, S. B.; Mellichamp, J. M.; & Smith, R. E. (1995, June). Box-Jenkins Forecast Model Identification. AI Expert, 10(6) 15-28.
- Reyneri, L.; & Filippi, E. (1991, December). An analysis on the performance of silicon implementations of back propagation algorithms for artificial neural networks. IEEE Transactions on Computers, 40, 1380-1389.
- Robins, D. (1990). Helping Teenagers Make Wise Decisions. In "Option Plays" Becchetti, Noel, ed.; Youth Specialties, El Cajon, CA; p. 25.
- Rolston, D. W. (1988). Artificial Intelligence and Expert Systems Development. New York: McGraw-Hill Book Company.
- Rosenblatt, F. (1958). The Perceptron: A probabilistic model for information storage and organization in the brain. Psychological Review, 65, 386-408.
- Rosenfield, A.; & Kak, A. (1982). Digital pictures processing. Computer Science and Applied Mathematics, 1, 116-204. As quoted in Mougeot, et al., 1991)
- Roy, A., Govil, S, & Miranda, R. (1995). An algorithm to Generate Radial Basis Function (RBF)- Like Nets for Classification Problems. Neural Networks. 8(2) 179-201.
- Rucker, R., Sirus, R. U., & Mu, Q., eds. (1992). *Mono 2000: A Users Guide to the Edge*. New York: Harper Collins Publishers, Inc.
- Rumelhart, D.; Hinton, G.; & Williams, R. (1986a). Learning internal representations by error propagation. In Parallel Distributed Processing, vol. 1: Foundations, Cambridge: MIT Press.
- Rumelhart, D.; Hinton, G.; & Williams, R. (1986b, October). Learning representations by back-propagation errors. Nature. 323(6088), 533-536.
- Rumelhart, D.; Smolensky, P.; McClelland, J.; & Hinton, G. (1986). Schemata and Sequential Thought Processes in PDP models. In Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 2. D. Rumelhart, J. McClelland and the PDP Research Group, Eds. Cambridge, MA: MIT Press.
- Rumelhart, D. E. (1988). Learning and Generalization. Plenary Address, IEEE International Conference on Neural Networks, San Diego.

Samuel, A. (1967). Some studies in machine learning using the game of checkers II -Recent Progress. IBM Journal of Research and Development, 11(6) 601-617. Quoted in Kolen and Goel (1991).

Santini, S., Del Bimbo, A., & Jain, R. (1995). Block-Structured Recurrent Neural Networks. Neural Networks, 8(1), 135-147.

Sarel, W. S. (1994a). TNN Macor—Alpha release—Experimental, Version 2. SAS Institute Inc. Document downloaded from: <ftp.sas.com/pub/sugi19/neural/tnn2.doc>

Sarel, W. S. (1994b). Neural Networks and Statistical Models. Proceedings of the Nineteenth annual SAS Users Group International Conference, downloaded from: <ftp.sas.com/pub/sugi19/neural/neural1.ps>

Sarel, W. S. (1994c). Neural Network Implementation is SAS® Software. Proceedings of the Nineteenth annual SAS Users Group International Conference, downloaded from: <ftp.sas.com/pub/sugi19/neural/neural2.ps>

Sarel, W. S. (1995). Stopped Training and other Remedies for Overfitting. To appear in the Proceedings of the 27th Symposium on the Interface, 1995, downloaded from: <ftp.sas.com/pub/sugi19/neural/interface.ps>

Schulethes, R.; Sumner, M.; & Bock, D. (1992). Management Information Systems: The Manager's View. Homewood, IL: Irwin.

Senn, J. A. (1990). Information Systems in Management, 4th Edition. Belmont, CA: Wadsworth Publishing Co.

Servan-Schreiber, H., Printz, H., & Cohen, J. D. (1990). A network of neuromodulatory effect: Gain, signal-to-noise ratio, and behavior, *Science*, 249, 892-895.

Shoemaker, P. A.; Carlin, M. J.; & Shimabukuro, R. L. (1991). Neural Networks, 4, 231-241.

Sietsma, J., & Dow, R. J. F. (1991). Creating Artificial Neural Networks that Generalize. Neural Networks, 4(1), 67-79.

Silverman, R. H., & Noetzel, A. S. (1990). Image Processing and Pattern Recognition on Ultrasonograms by Back propagation. Neural Networks, 3, 593-603.

Sima, J. (1995). Neural Expert Systems. Neural Networks, 8(2) 261-271.

Sharda, R., & Patil, R. (1990). Neural Networks as Forecasting Experts: An Empirical Test. Proceedings of the IJCNN Meeting, Washington DC: vol. 2, 491-494.

Smith, M. (1993). Neural Networks for Statistical Modeling. New York: Van Nostrand Reinhold.

Sontag, E. D., & Sussmann, H. J. (1991). Back propagation separates where perceptrons do. Neural Networks, 4(2), 243-249.

Specht, D. F. (1990). Probabilistic Neural Networks. Neural Networks, 3, 109-119.

Springer, P. L., & Gulati, S. (1995). Parallelizing the Cascade-Correlation Algorithm using Time Warp. Neural Networks, 8(4) 571-577.

- Stair, R. M. (1992). Principles of Information Systems, A Management Approach. Boston: Boyd & Fraser Publishing Company, a division of South-Western Publishing.
- Stein, J. (1991). Neural Networks: From the Chalkboard to the Trading Room. Futures: The Magazine of Commodities & Options, 20(6), 26-30.
- Stein, R. (1993, February). Selecting Data for Neural Networks. AI Expert, 42-47
- Stone, M. (1974). Cross-validators Choice and Assessment of Statistical Predictors. Journal of the Royal Statistical Society B36: 111-147.
- Stone, G. O. (1986). An analysis of the delta rule and the learning of statistical associations. In D. E. Rumelhart and J. L. McClelland (Eds.) Parallel Distributed Processing (444-459). Cambridge, MA: MIT Press. As quoted in Silverman & Noetzel (1990).
- Sutton R. S., & Barto, A. G. (1981). Toward a modern theory of adaptive networks: Expectation and prediction. Psychology Review, 88, 135-170.
- Tang, Z.; de Almeida, C.; & Fishwick, P. A. (1990). Time series forecasting using Neural Networks vs. Box-Jenkins Methodology. Proceedings of the First Workshop on Neural Networks: Academic/Industrial/NASA/ Defense. February 5-6, Auburn, Alabama, Auburn University Hotel and Conference Center, 95-100.
- Tavel, R. (1989). Does the neuron 'learn' like the synapse? In Advances in Neural Information Processing Systems, D. S. Touretzky, ed., San Mateo, CA: Morgan Kaufman, pp. 169-176. As quoted in Kruschke and Movellan (1991).
- Thacker, N. A., & Mayhew, E. W. (1990). Designing a layered network for context sensitive pattern classification. Neural Networks 3, 291-299.
- Tollenaere, T. (1990). SuperSAB: Fast Adaptive Back Propagation with Good Scaling Properties. Neural Networks, 3(5), 561-573.
- Turing, A. M. (1936). On computable numbers with an Application to the Entscheidungs problem, Proceedings of the London Mathematical Society. As quoted in the Encyclopedia Britannica Macropaedia (1979), vol. 2, p 497.
- Tveter, D. (1991, July). Getting a fast break with Backprop. AI Expert, 36-43.
- Van Dalen, D. B. (1979). Understanding Educational Research. 4ed. New York: McGraw-Hill.
- Varfis, & Versino, A. V. (1990). Univariate Economic Time Series Forecasting by Connectionist Methods. Proceedings of INNS, 1990, Paris. 1, 342-345.
- Versaggi, M. R. (1995, April). Understanding Conflicting Data. AI Expert, 10(4) 21-25.
- Vogl, T. P.; Mangis, J. K.; Rigler, A. K.; Zink, W. T.; & Alkin, D. L. (1988). Accelerating the Convergence of the Back-Propagation Method. Biological Cybernetics, 59, 257-263.
- Webb, A. R., & Lowe, D. (1990). The Optimized Internal Representation of Multilayer Classifier Networks Performs Nonlinear Discriminant Analysis. Neural Networks, 3, 367-375.
- Weigend, A. S.; Huberman, B. A.; & Rumelhart, D. E. (1990). Predicting the future: A Connectionist approach. International Journal of Neural Systems, 1 (3).

Weiss, S. M., & Kulikowski, C. A. (1991). Computer Systems that Learn. San Mateo, CA: Morgan Kaufmann Publishers, Inc.

White, H. (1990). Connectionist nonparametric Regression: Multilayer Feedforward Networks can learn Arbitrary Mappings. Neural Networks, 3, 535-549.

Wittner, B. S., & Denker, J. S. (1987). Strategies for teaching layered networks classification tasks. In D. Anderson (ed.), Proceedings of the Conference on Neural Information Processing Systems (pp. 55-58). New York: American Institute of Physics. As quoted in Sontag and Sussmann (1991).

Wray, R.; Phillip, J.; Rogers, S.; & Walsh, B. (1994). Survey of Cognitive and Agent Architectures. Ann Arbor, MI: University of Michigan, <<http://ai.eecs.umich.edu/cogarch0/common/prop/nonomono.html>> Downloaded January 3, 1996.

Wray, J., & Green, G. G. R. (1995). Neural Networks, Approximation Theory, and Finite Precision Computation. Neural Networks, 8(1) 31-37.

Yaconelli, M. (1990). Understanding and using option making and consequencing. In "Option Plays" Becchetti, Noel, ed.; Youth Specialties, El Cajon, California; p. 16.

Yao, Y.; Freeman, W. J.; Burke, B.; & Yang, Q. (1991). Pattern Recognition by a Distributed Neural Network: An Industrial Application. Neural Networks, 4(1), 103-121.

Yoon, Y., & Swales, G. S. (1991). The Application of an Artificial Neural Network to Predict Stock Prices. Proceedings of the National Decision Sciences Institute Conference, 1713-1715. As quoted in Cnang (1992, p 3).

VITA

Gene Barton Binning

Candidate for the Degree of

Doctor of Education

Thesis: A REVIEW OF BACK PROPAGATION NEURAL NETWORKS AND
TRADITIONAL STATISTICAL METHODS

Major Field: Occupational and Adult Education

Personal Data: Born in Denver, Colorado, December 7, 1953, the son of Gene H. and Bette
Binning.

Education: Graduated from John Marshall High School in Oklahoma City, Oklahoma, in May
1971; received Bachelor of Arts Degree in Economics and Business Administration
from Vanderbilt University in May, 1975; received Master of Business Administration
from the University of Oklahoma in May, 1977; completed requirements for the Doctor
of Education degree at Oklahoma State University, December, 1996.

Professional Experience: Salesman, Department Manager, and Office Controller for Oklahoma
City Office of Trane Company, 1976 to 1988; Instructor, Department of Decision
Sciences, University of Central Oklahoma, Edmond, Oklahoma, September 1988.