INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

University Microfilms International
A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
313/761-4700 800/521-0600

Order Number 9325095

Economic rationality and neural networks

Garavaglia, Susan Berger, Ph.D.

City University of New York, 1993

Copyright ©1993 by Garavaglia, Susan Berger. All rights reserved.

U·M·I
300 N. Zeeb Rd.
Ann Arbor, MI 48106

A

Economic Rationality and Neural Networks

by

Susan Garavaglia

A dissertation submitted to the Graduate Faculty in Economics in partial fulfillment of the requirements for the degree of Doctor of Philosophy, The City University of New York

1993

© 1993

Susan Garavaglia

All Rights Reserved

This manuscript has been read and accepted for the Graduate Faculty in Economics in satisfaction of the dissertation requirement for the degree of Doctor of Philosophy.

4/20/93
Date

Chair of Examining Committee

4/20/93
Date

Michael Grossman
Executive Officer

Peter S. Albin

Michael Grossman

Salih N. Neftci

Supervisory Committee

THE CITY UNIVERSITY OF NEW YORK

Abstract

Economic Rationality and Neural Networks

by

Susan Garavaglia

Advisor: Professor Peter S. Albin

Artificial Neural Networks are constructed by linking primitive computing structures with weighted connections. The information content of a neural network develops through the adaptation of the connection weights by learning algorithms to meet some measurable criterion. This adaptability, combined with parallelism and distributed information processing, produce a highly effective paradigm of a decision making organization. Higher degrees of organizational complexity are represented through a larger number of primitive units and connections which can be decomposed into any meaningful subdivision. Because of the efficiency and modularity of this representation of a decision making entity, a neural network exemplifies Simon's *procedural rationality*.

To establish foundations, an overview essay includes a survey of economic applications, including predictive econometric models, classification, and behavioral models. Specific examples of neural networks developed to perform estimation of non-linear functions, discriminant analysis and clustering, and pattern recognition, are discussed. This essay also discusses some of the key statistical and mathematical

properties of neural networks, reviewing the work of Halbert White and others.

The principal justification of the application of neural networks to economic problems is delivered in the context of procedural rationality (Herbert A. Simon). The use of neural networks to reduce the complexity of a model is demonstrated and compared with Peter Albin's work on complexity measurement and organizational committee structures. In addition, the theme of biological analogy is examined in the work of Kornai and others. An economic system is often compared to a biological organism to explain its structure and behavior. One essay demonstrates the application of a neural network to modeling Akerlof's reciprocal gift exchange theory.

A new neural network model is proposed for application to rational decision making in organizations, based on R. K. Sah's and J. E. Stiglitz' work on optimal consensus size in committees and hierarchies, and Nils Nilsson's committee machine. This neural network learns the optimal size of a consensus in a committee and the characteristics of good decisions by observing the odds ratio of bad to good options. The behavior of the network is instructive in understanding decision-making behavior in large organizations.

PREFACE

This dissertation comprises a collection of essays relating to the economics of employing artificial neural networks as problem solving, decision support, statistical modeling, and behavioral modeling tools. Each essay is self-contained with regard to content and purpose, and the order in which the essays appear is meant to lead the reader from the well-established research and econometric applications to the author's research in applying neural networks to classification, clustering, and organizational behavior.

The predominating theme of the dissertation is the inherent economic rationality of a neural network approach to economic modeling, whether one is modeling a demand function, a financial market, or a collection of agents in some organizational structure. Simplification of a complex structure by redesigning it as a combination standard easily constructable components has precedents in many domains, from industrial manufacturing to the structure of a corporation as a collection of job descriptions and hierarchical relationships. Gains in economic utility may arise from increased profits due to reduced costs or time savings as well as the non-monetary benefits of confusion avoidance, in that people's roles are simplified to the extent that no one person in a organization is overwhelmed by information or assigned tasks.

Several essays, included as chapters, have been presented at conferences. The author gratefully acknowledges the permissions granted by the IEEE Computer

Society Press, the Software Engineering Press, and High-Tech Communications, Inc. The original idea for this disseration arose out of the research and writing of "An Application of a Counterpropagation Neural Network: Simulating the Standard & Poor's Corporate Bond Rating System" (Chapter 5) As bond ratings are actually produced by the work of a committee, the next logical step was to investigate the properties of neural networks that could model committee behavior. The paper, "Bi-directional Associative Memory Applied to Non-Neoclassical Economic Behavior" (Chapter 4), written for the International Joint Conference on Neural Networks - 1992, was the watershed because it was a liberation from the constrained view of neural networks solely as a replacement for more established statistical methods.

This author considers herself extremely fortunate to have Professor Peter S. Albin as a mentor and dissertation supervisor. Professor Albin has provided valuable guidance on all aspects of academic research, including background reading, other contacts in the field, writing style, and thinking beyond the conventional engineering approach to neural networks. In addition, Professors Michael Grossman and Salih Neftci, as the other dissertation committee members, have contributed ideas, advice, and encouragement.

The completion of a dissertation yields satisfaction for its accomplishment, recognition, opportunities, and an excuse for celebration. It also sets the stage for for a level of research that is not bound by the constraints of establishing the foundations, as they have been established as part of this work. As such, this dissertation is as much of a springboard as a finish line.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF TABLES (continued)

LIST OF FIGURES

LIST OF FIGURES (continued)

CHAPTER 1

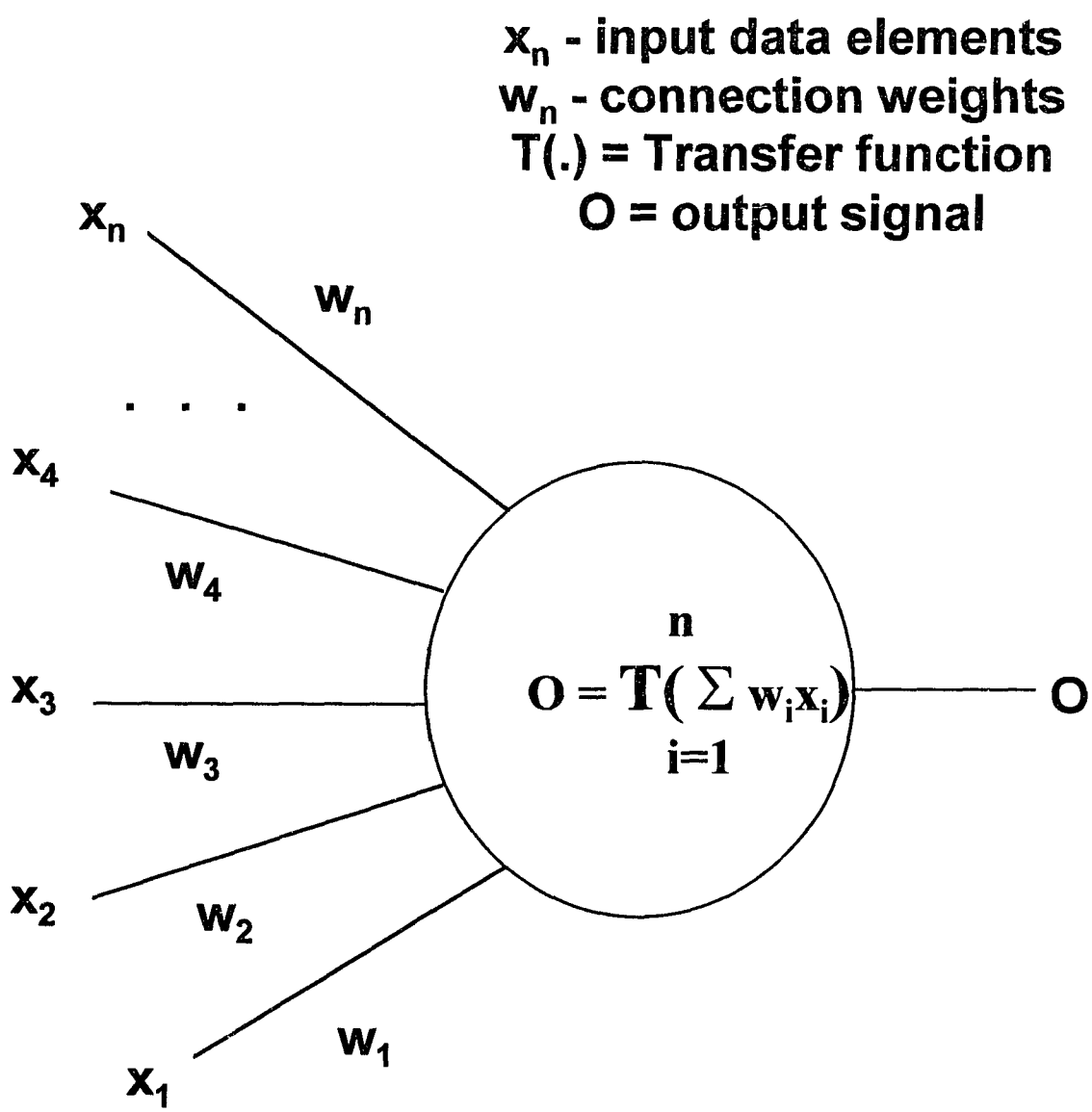
AN OVERVIEW OF NEURAL NETWORKS FOR THE ECONOMIST

1.0 Introduction

Artificial neural networks (neural nets) are *adaptive* multiprocessing structures that are conceptually parallel and distributed in their function. The neural net structure comprises a set of independent processing elements which are arranged in subsets called *layers* or *slabs*[1] and connected to processing elements in other subsets by weighted connections. Generally, the number and internal characteristics of individual processing elements and their connectivity scheme stay the same through the adaptation process, and we will restrict our discussion to this type of network. Each processing element contains some local memory in which a summation function, a transfer function which maps to a bounded interval, and the required data vectors are stored. A processing element accepts one or more inputs from either external sources or another processing element, performs a weighted summation on the inputs, submits the weighted sum to a transfer function which produces a single bounded output value (See Figure 1-1a). The adaptation is contained entirely in the

[1] *Slab* is used interchangeably with *layer* by Hecht-Nielsen (1990).

Figure 1-1a -- Diagram of a Neural Network Processing Element



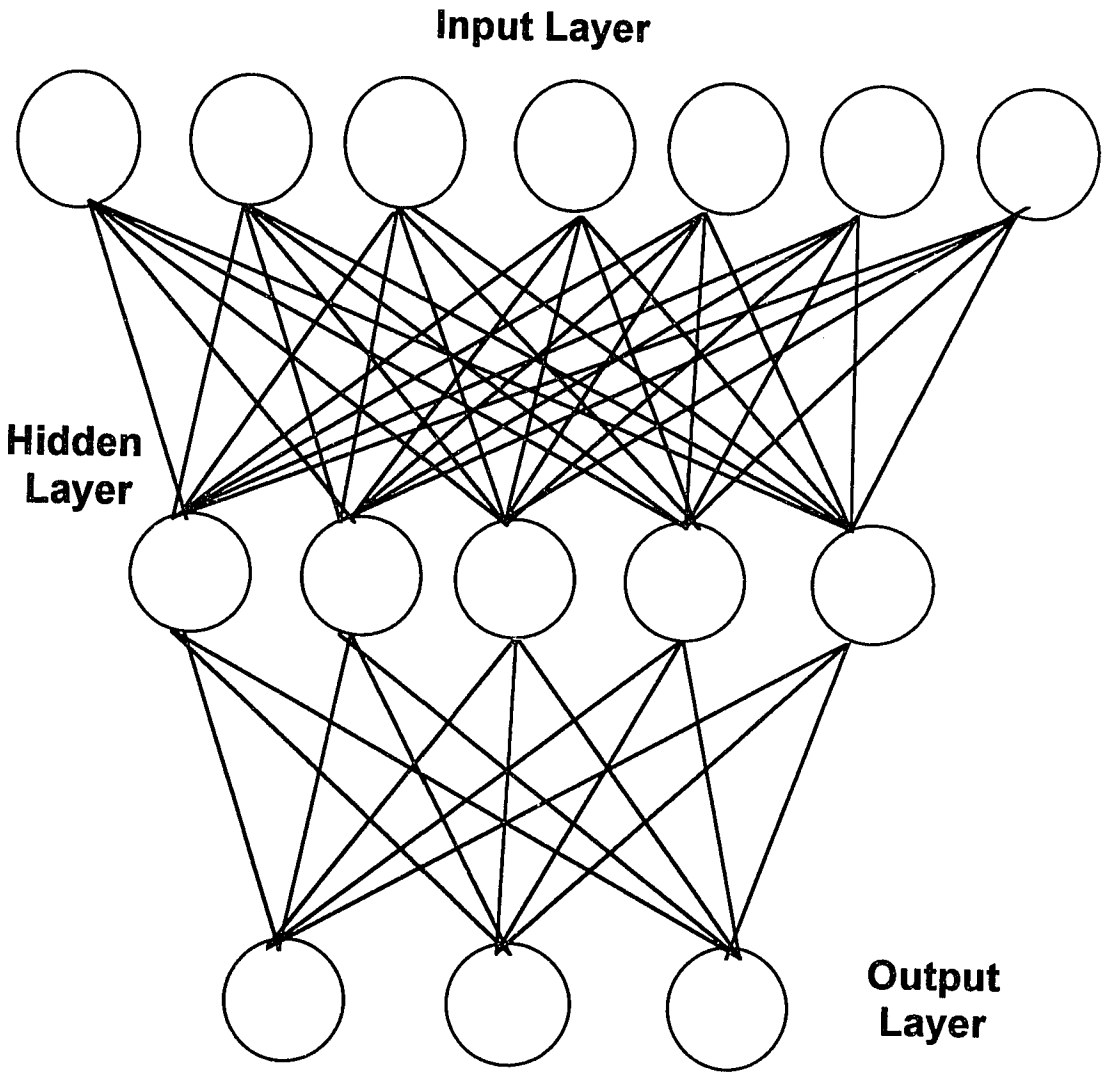
changing values of weights which are associated with each connection. Therefore, the "information content" or "knowledge" of a neural net is in the set of final values of the weights.

The final values of the weights are derived through an iterative training process requiring learning or updating algorithms which are typically a function of the weight's previous value, the output signal, and other parameters. If error correction algorithms are to be applied, output values have to be compared to some norms or "right answers." This is called supervised learning. When a steady state is the objective of the processing, algorithms update weight values until the "delta" is zero, or very close to zero. This is called unsupervised learning. Thus, the difference between supervised and unsupervised learning is whether or not a set of desired output vectors is presented to the network during training.

The number of layers in a neural network structure, and especially the number of "hidden" layers is a key consideration in network design and implementation. Layers or processing elements in layers that are not input or output layers are usually called "hidden" because their inputs and output are not observed external to the network activity. Most neural network architectures have at least one hidden layer, and it will be shown that the hidden layers play a major role in the network's effectiveness (See Figure 1-1b).

There are a number of variations on the basic structural concept. What remains constant is the single output value from any neural processing element, weighted summation of inputs, and adaptatation through changes in the values of the weight

Figure 1-1b -- A Three-Layer Neural Network



vectors. In terms of a taxonomy of structure, such as that of Albin (1983), neural networks belong to the category of "Loose Structures," which are of the highest level of complexity, having the ability to incorporate and combine more basic structural elements, such as cellular automata in the form of neural processing elements and graphs in the form of network connections.

1.1 From Biological to Mathematical Modeling

Artificial neural networks are loosely modeled on the architecture and functioning of a biological nervous system. The rationale is that, since nervous systems of simple animals can process complex sensory input far beyond the capabilities of the most sophisticated computer systems, imitating the functioning of a nervous system with computers will enhance their capabilities. Cotman and McGaugh (1980) express well the universal reverence for the biological neural network: "Through the actions of chemicals, squirted onto other cells organized in specific circuits, emerges a capacity to process signals that is beyond that of any computer yet constructed and probably in fact that ever will be constructed" (p. 153). A neural net processing element represents a single neuron. Like a biological neuron, it may generate an excitatory signal, an inhibitory signal, or a modulating signal, i. e., a signal which causes the receiving neuron's behavior to change. The weighted connections which direct the input signals to the summation function are analogous to synapses, which allow stimuli to pass from one neuron to another. The single output signal would be sent to other biological neurons via the axon.

However, in computer models of neural networks, the dendrite, which receives input signals, the synapse, and the axon, which transmits an output signal to the synapse, are all represented with a weighted link that transports the signal from one neural processing element to another.

Neural networks as information processors is an idea with beginnings traditionally traced back to a 1943 paper by Warren S. McCulloch and Walter Pitts (1943). McCulloch and Pitts proposed, in essence, that because a neuron's output signal is "all or nothing," i. e., the axon either "fires" its signal or not, it can represent logical propositions, which are also two-valued, namely, true or false. Another significant contribution that is indirectly related to neural networks is that of Claude Shannon (1948), who provided a theoretical framework for determining how to organize and design a system to communicate information based on the complexity and distribution of the information and the accompanying "noise."[2] As information is transmitted through an artificial neural network through digital signals, Shannon's work is relevant to selecting the number of processing elements to represent the input and output data.

In addition to representing propositional logic, the McCulloch and Pitts paper made brief but significant statements about the neural network architecture and causality. Given the possibility that more than one combination of weighted inputs can produce the same output signal; a neural network model is one in which causality

---

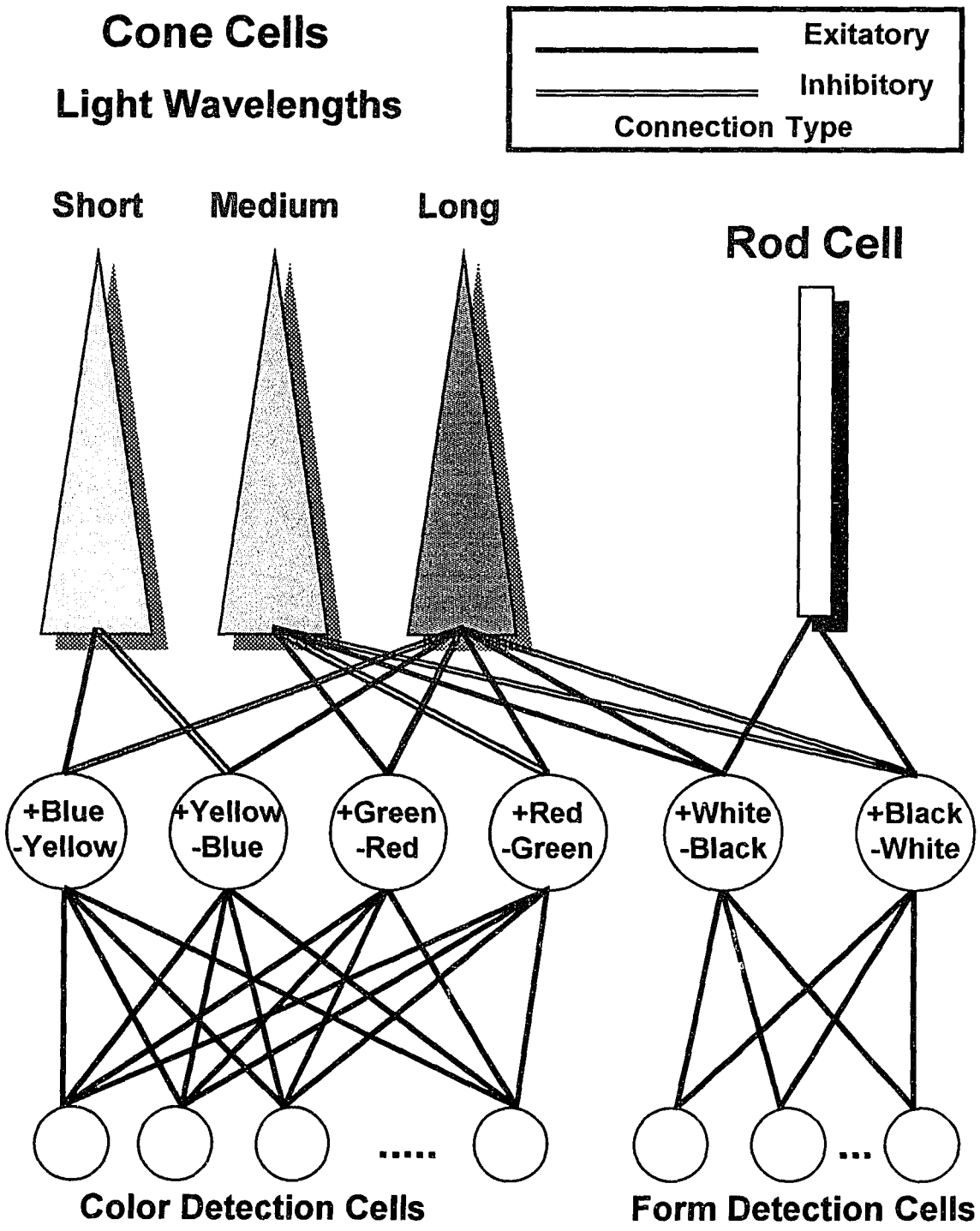
[2] White (1989) cited Shannon in reference to the the connection between information theory and an optimal set of weights.

is clearly defined as an intrinsic part of the model. Granger (1969) discusses modeling and testing for different types of causality by methods of spectral analysis. Neural networks can more directly represent simple causality, feedback, and instantaneous causality by the architecture of their layers and connections. Each layer of a neural network can represent a step in time and connections can be either feed-forward or feedback. Granger causality used a comparison of variances as a test, e. g., X causes Y if including X in the model produces a lower variance than excluding X. In neural network modeling, the value of a global error function can indicate if the network has accurately modeled the causal relationships, and experiments can be conducted to set connection weights to zero with the effect of excluding some data. However, without the present capabilities of computer systems, in terms of processor speeds, memory capacity, and visualization through the use of graphics, neural networks would probably not be a feasible methodology.

1.2 Color Vision as an Example of Biological Neural Network Efficiency

Economists seek both the ability to isolate and model complex dynamic processes and parsimony of representation. A brief discussion of human color vision, summarized from DeValois and DeValois (1975), illustrates how a network of relatively few types of processing elements allows discrimination of subtle distinctions in a rich environment, such as the color spectrum. Humans can easily perceive differences of 5-nm in wavelength a color spectrum range of just under 400 nm to about 700 nm, their retinae are equipped with only 4 types of photoreceptor cells.

Figure 1-2 -- A Biological Network for Color Vision



Named for their respective shapes, rod cells are sensitive only to forms in dim light and only detect shading, not color, and three types of cone cells are sensitive to overlapping ranges of wavelengths, not precisely corresponding to Blue, Green, and Yellow-Red. How do three types of photoreceptor distinguish 60 or more color differences, and what are the implications for economic modeling? Although somewhat simplified, Figure 1-2 shows the neural network subset for color vision. The rod and cone cells are stimulated by photons and send excitatory or inhibitory signals to the next layer, which comprises two general cell types, 4 subtypes of cells (called "opponent cells") for color perception, and 2 subtypes of cells (called "non-opponent cells") for form discrimination. Signals are then sent to two groups of neural cells in the cortex, one for specific color detection, and one for form information. There are two obvious benefits to this type of complex system; one is the economy of form, a small and finite number of units and unit types; the other is the processing of a continuous range of values with no discrete boundaries which translate into fuzzy and overlapping sets. For example the set of all variations on the color orange may overlap with the set of yellows and the set of reds. In addition, if we consider the cost elements of this system, we would consider it cost efficient as well. The rod and cone cells may be relatively "expensive," in that, by their location, they are the most exposed to physical damage and they do not transmit the simple "all-or-nothing" signals of most other neuron types. Instead, their output signals are graded responses to light. Thus, this neural network design has minimized "costs" by lowering the need for the most "expensive" neurons and keeping the "inventory" of

different types of neural cells low as well.

1.3 Neural Network Development: Training, Testing and Recall

Neural network design mainly includes selection of the numbers of processing elements and layers, the number and direction of the connections (i. e., feed-forward and/or feedback), the learning algorithm for updating the connection weights during training, the values of some learning coefficients, and the transfer function. The number of layers, transfer functions, scheme of connections, and the learning algorithms employed comprise a neural network architecture. Other factors include representation of the input and interpretation of the output, which determine the number of processing elements. All symbolic and analog data must be mapped to some form of bounded numerical representation, under the assumption that the implementation is through the programming of software on general purpose computer systems.

Training a neural network requires exposing it to corresponding input and output data vectors and activating the learning algorithms. The input-output vectors pairs can either represent the entire universe of possibilities, such as an entire alphabet, or be a statistically valid sample of the data population. Part of the value of a neural network is its ability to successfully process out-of-sample data. Testing a trained neural network requires feeding it only the input data and observing the output. If the output it produces matches the output used in training, learning is considered successful although the network's ability to generalize has not yet been

tested. In recall, the term commonly used to refer to the practical use of the trained neural network, the network may be given any input data. Output may be measured against expectations or any other success criteria.

The next section is a detailed walk-through of a feed-forward neural network architecture, the backpropagation network. Backpropagation causes the connection weights to change in direct proportion to each weight's contribution to the overall output error.

1.4 Supervised Learning: A Backpropagation Example

Neural networks can be used to obtain some of the information provided by regression analysis, in that the post-learning weight vectors represent the relationships between independent and dependent variables. In conventional linear regression, the relationship is modeled by the formula $y = X\beta + u$, where X is the dependent variable matrix, β is a column vector of coefficients, and u is a disturbance vector. Usually, there will be more elements in the weight vectors than in the β vector for the same sample data, and the weights will not have the same values as the β's derived from statistical regression techniques, although they can be used for prediction. The obvious disadvantage using neural networks is that the individual elements in the β vector reflect the role of any one independent variable in the overall result as a multiplier, while no such set of individual multipliers can be calculated from a neural network's system of weights and connections. However, conventional statistical regression analysis forces one to presume that the model is

in a specific mathematical form before any parameter estimation can take place. A neural network requires no prior modeling of the relationships between variables.

This section discusses the backpropagation learning algorithm. A learning algorithm determines how the connection weights are updated. Backpropagation is one specific method belonging to a class of gradient-descent algorithms. Generally, backpropagation causes each connection weight to change in the opposite direction of and in proportion to its contribution to the global error of the network. Rumelhart, Hinton, and Williams (1986) are usually credited with the introduction of the backpropagation learning algorithm; however Anderson and Rosenfeld (1990) also cite papers by Y. Le Cun, D. Parker, and the 1974 Ph.D. thesis of P. J. Werbos as describing backpropagation as well. Because of its efficiency in learning and its applicability to non-linear and linear function approximation, backpropagation is one of the most widely applied learning algorithms.

In order to illustrate function estimation with a neural network, a student's textbook problem from Johnston (1984) will serve as the example. The vector for the dependent variable y and the X matrix for the independent variables are:

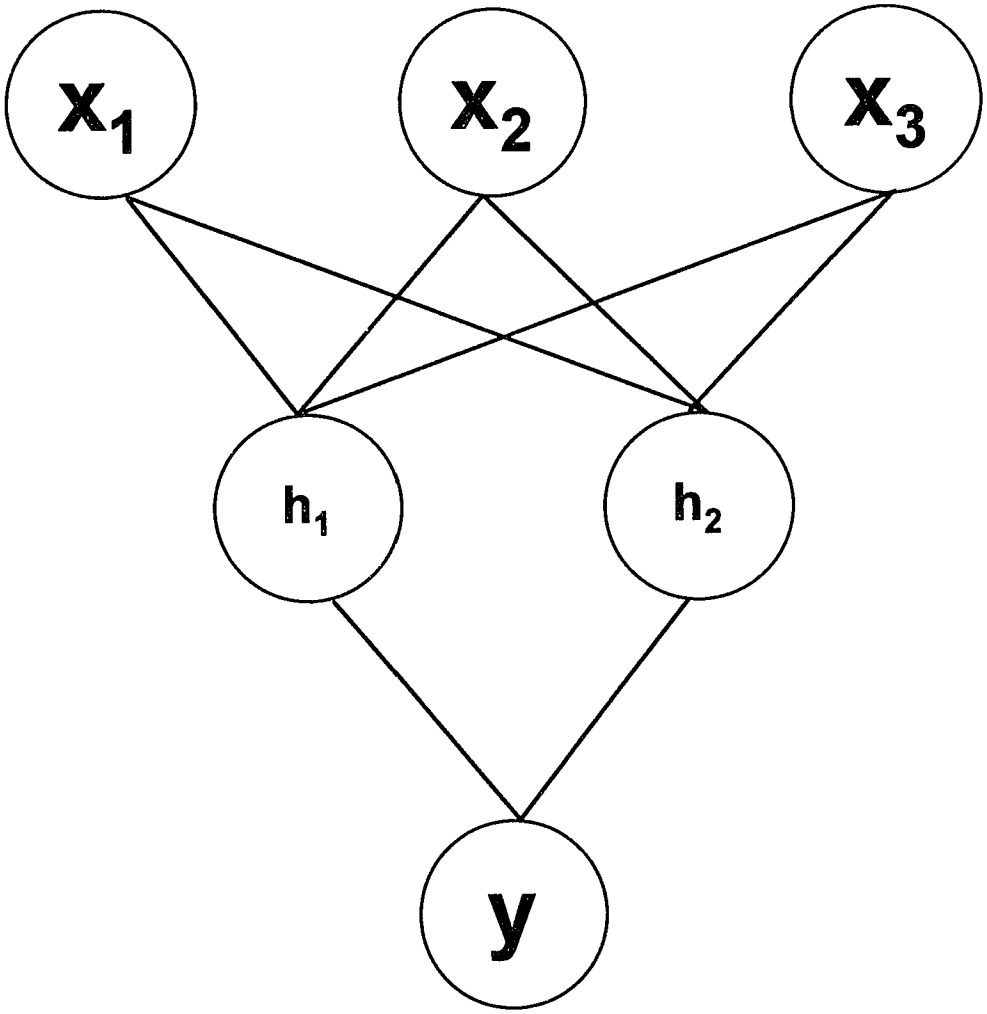
$$y' = \begin{matrix} 3 \\ 1 \\ 8 \\ 3 \\ 5 \end{matrix} \qquad X = \begin{matrix} 1\ 3\ 5 \\ 1\ 1\ 4 \\ 1\ 5\ 6 \\ 1\ 2\ 4 \\ 1\ 4\ 6 \end{matrix}$$

Using the Ordinary Least Squares (OLS) regression technique, the function derived is $\hat{Y} = 4 + 2.5X_2 - 1.5X_3$. Although a relatively high R^2 is obtained in this

regression (0.9464), some standard hypothesis testing generally reveals the weakness of these estimators (see Johnston (1984), pp. 190-193).

Consider how a simple neural network with one hidden layer would be used to determine the relationship between the independent and dependent variables given the same data. Generally, the inputs enter the structure by values assigned to the dedicated input layer; each element of an input vector is given to a processing element in this layer. A single output value from each input element "fans out" to function as input to the processing elements in the next layer, which is called the "hidden" or "feature detection" layer. Each connection from the input to the hidden layer has a weight associated with it, and each hidden layer element receives a weighted sum of the input layer's output as its input. The "hidden" layer gets its name from the fact that its results are not observable as either input or output. Choices can be made by the network designer as to the number of hidden layers to include and as to how to connect the various layers together. Our example will be a fully-connected strictly feed-forward network with 3 processing elements (PEs) in the input layer, 2 PEs in the hidden layer and 1 PE in the output layer. Every output from a previous layer is connected to every processing element in the next layer with no feedback connections (see Fig. 1-3). Some details about the architecture of the network need to be discussed before proceeding. If a function is to be estimated from data points, it is important to know that the extreme values of the sample data represent the extreme values of the population. In addition, truncations may have to be made if outliers exist, or the network may have to be trained to recognize

Figure 1-3 -- The Backpropagation Neural Network from the Student's Example from Johnston (1984)



outliers and treat them appropriately. In order to preserve this information, some scaling must be performed on the input and output to allow the neural network to "fit" the sample[3]. In addition, the hidden and output layers' output is controlled by a "squashing" function, here specifically the hyperbolic tangent function, *tanh(x)*[4] which produces values in the interval [-1, 1]. The purpose of a "squashing" function

---

[3] The scaling algorithms, which perform simple proportional scaling, are from NeuralWare, Inc. (1990):

a. for scaling the input:

$$i_j = \frac{(R_I - r_I) \times f_j + (M_j \times r_I - m_j \times R_I)}{(M_j - m_j)}$$

b. for scaling the desired output:

$$d_k = \frac{(R_D - r_D) \times f_k + (M_j \times r_D - m_k \times R_D)}{(M_k - m_k)}$$

c. for mapping the actual output back to the original data:

$$g_k = \frac{(M_k - m_k) \times o_k + (R_D \times m_k - r_D \times M_k)}{(R_D - r_D)}$$

where: (r_I, R_I) are the lower and upper boundaries of the actual data range; (r_D, R_D) are the lower and upper boundaries of the desired output ranges, f_j and f_k are the actual input values for processing elements j and k, o_k is the output value from processing element k,
(m_j, M_j) and (m_k, M_k) are the lower and upper boundaries of the range of values for processing elements j and k.

[4] The hyperbolic tangent function:

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

is to transmit a meaningful signal and to work within measurable boundaries.

In the training phase, each independent variable matrix row is input to the network, the network produces an output value from its output layer, using the weighted summation and *tanh* transfer functions at both the hidden and output layers to calculate the output value. The network's calculated output and the corresponding dependent variable's value, i. e., the desired value, are used in the learning algorithm to update the weights. The process of presenting the input and desired output to the network and the updating of the connection weights continues until the global error value is reduced to some predetermined acceptable value. Then, learning is complete. If the global error value does not get to the chosen value, but seems to be "stuck" within a range of higher values, then a number of remedies may be applied to change the network's architecture or learning algorithm. Changes may include increasing the number of hidden units or choosing different learning coefficients, but a revision in the representation of the data may also be needed.

The backpropagation learning algorithm minimizes the network global error by the following gradient-descent formula:

$$\Delta w_{ji} = -C \frac{\partial E}{\partial w_{ji}} \tag{1.1}$$

where w_{ij} is the connection weight from processing element i to j, C is a constant

known as the learning coefficient, and E is the global error defined as a function of the sum of the local errors:

$$E = 0.5 * \sum_{k} ((d_k - o_k)^2) \tag{1.2}$$

where $d_k - o_k = e_k$, the local error at processing element k. By using the chain rule, (1.1) can be expanded to

$$\Delta w_{ji} = -C \frac{\partial E}{\partial I} \frac{\partial I}{\partial w_{ji}} \tag{1.3}$$

where I is the weighted sum of the inputs feeding the current processing element

$$I = \sum_{j=1}^{n} w_{ji} x_i \tag{1.4}$$

Because

$$\frac{\partial E}{\partial I} = e_j \text{ and } \frac{\partial I}{\partial w_{ji}} = x_i, \tag{1.5}$$

the formula for updating each connection weight of any PE at any layer thus becomes:

$$\Delta w_{ji}^{[s]} = -C * e_j^{[s]} * x_i^{[s-1]5} \tag{1.6}$$

In (1.6), e_j is the local error at the current processing element in layer s and x_i is the

---

[5] The derivation of this formula is from Klimasauskas (1990); however, Kosko (1992), Rumelhart, Hinton, and Williams (1986) are also recommended sources for a detailed working out of the learning algorithm.

input received from processing element i in layer s. Each connection weight change is a therefore a function of the local error at the PE and the value passed to the PE on that specific connection. Because errors are corrected proportionally at their source, the corrections are optimal. At each training iteration, (1.1) is applied to the all the weights at all layers, starting with the output layer and proceding back to the weights connecting the input layer with the first (or only) hidden layer.

In the testing phase, the sample input data is fed to the network to see if learning was successful by comparing the desired output with the network output. For the sample data, the network performed better than the least squares regression in producing the desired output values (see Table 1-1). In fact the network was able to achieve an extremely close mapping between the sample inputs and outputs. Table 1-1 also contains other values produced by different sets of input vectors that will be used in Figure 1-4 to show how the neural network fits the data.

Table 1-1 -- OLS Results Compared with Backpropagation

X1	X2	Neural Network Result	Y	OLS Result	Neural Network Error	OLS Error
1	1	5.545215		5		
1	2	5.384619		3.5		
1	3	4.089423		2		
1	4	1.006888	1	0.5	-0.006888	0.5
1	5	0.361951		-1		
1	6	0.309524		-2.5		
2	1	5.959208		7.5		
2	2	5.976874		6		
2	3	5.598617		4.5		
2	4	2.99637	3	3	0.00363	0
2	5	0.655771		1.5		
2	6	0.415129		0		
3	1	7.163494		10		
3	2	7.310686		8.5		
3	3	7.371517		7		
3	4	6.734244		5.5		
3	5	3.00249	3	4	-0.00249	-1
3	6	1.169289		2.5		
4	1	8.459324		12.5		
4	2	8.522523		11		
4	3	8.56806		9.5		
4	4	8.550723		8		
4	5	7.980544		6.5		
4	6	4.998018	5	5	0.001982	0
5	1	8.773665		15		
5	2	8.78237		13.5		
5	3	8.788829		12		
5	4	8.788395		10.5		
5	5	8.733432		9		
5	6	8.001577	8	7.5	-0.001577	0.5
		Sum of Squared Err.			0.0000285	0
		Variance			0.0000135	0.3

The true test of learning, however, is to present the network with input data that was not used in training. Success with new data means the network has truly learned the relationships and is able to generalize. A number of new input vectors were presented to the network and also to the regression formula. Figure 1-4 shows the result in a 3-dimensional graph. A quick visual inspection shows that the network has more closely approximated the function suggested by the sample data.

The learned functional relationship can be "flattened" to equation (1.7), where *O* is the "raw" output value of the output layer's single processing element representing the dependent variable *y*.

$$(1.7)\quad O = \tanh(0.3051 + 1.1104 * \tanh(-0.2056 + 1.0840 * x_1 - 1.0604 * x_2) - 1.1959 * \tanh(0.3828 - 1.5015 * x_1 - 0.0744 * x_2))^{6}$$

Each constant is a connection weight, and the inner parentheses are the weighted

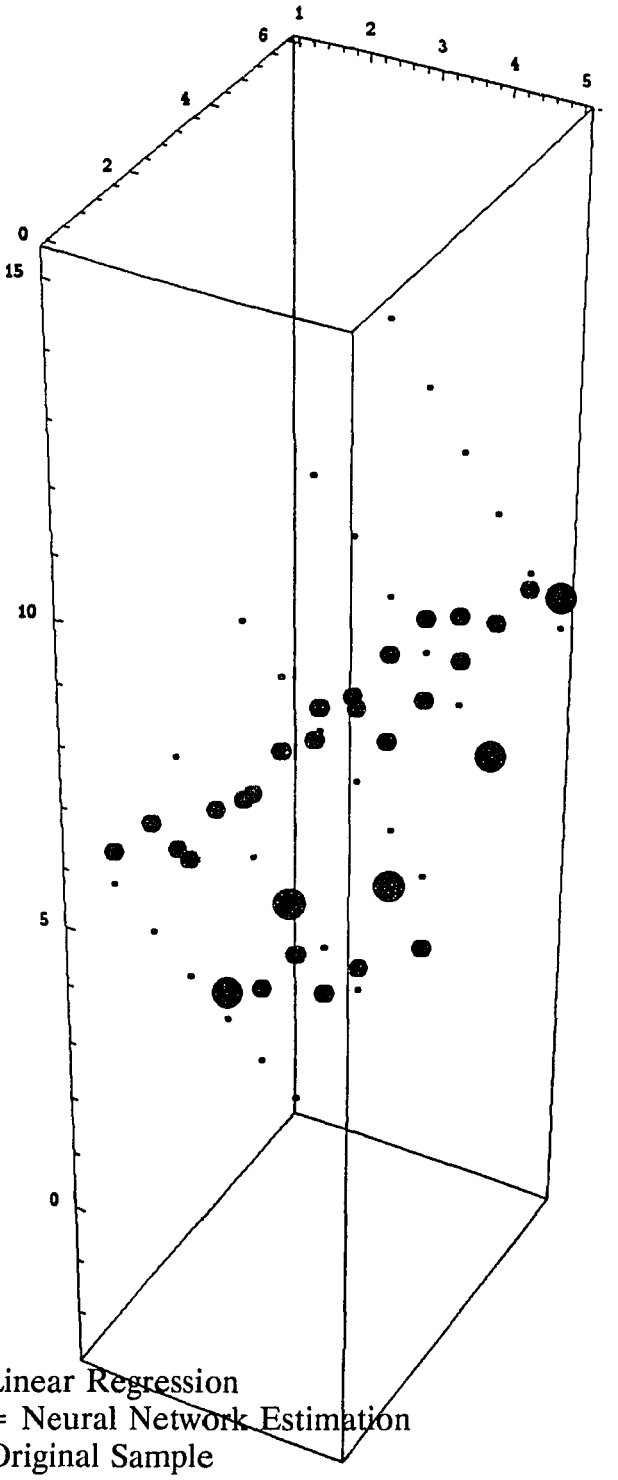
---

[6] Each constant is the weight w_{ij}, of the connection between processing elements i and j.

w_{16}	=	0.3051
w_{46}	=	1.1104
w_{56}	=	-1.1959
w_{14}	=	-0.2056
w_{24}	=	1.0840
w_{34}	=	-1.0604
w_{15}	=	0.3828
w_{25}	=	-1.5015
w_{35}	=	-0.0744

summations at each hidden processing element. The significance of this equation is in the combined use of the *tanh* function and the introduction of a hidden layer. Because the hyperbolic tangent function can be approximated as a power series (see Spanier and Oldham (1987)) and the hidden layer generates more coefficients than a regression using only the sample data points through the connection weights, the result is that the network becomes a polynomial equation that approximates a non-linear regression function. The theoretical basis for this result is discussed in section 1.13. The intuitive result is that the neural network creates a more refined approximation because the additional weight/coefficients allow more opportunities for iterative "fine adjustments." Just as a garment with more seams and darts allows a tailor to achieve a more custom fit, the hidden layer provides an analogous

Figure 1-4 -- A Three Dimensional Representation of the Results of OLS Compared with Backpropagation



Small Points = Linear Regression
Medium Points = Neural Network Estimation
Large Points = Original Sample

capability to fit a function to a set of sample data.

1.5 Comparison to Traditional Methods

White (1991) generalizes the output of a 3-layer (input, hidden, and output layers) feedforward neural network as the function

$$(1.8) \qquad f(x, \Theta) = F(x'a + \beta_0 + \sum_{j=1}^{q} G(x'\tilde{k}_j)\beta_j),$$

where x is the input vector, Θ is the set of weight vectors $(a, \beta_0..\beta_q k_1...k_q)$ with q being the number of hidden units, a being the z x 1 input-to-output weight vector for z output units, β_0 the bias weight, $\beta_1...\beta_q$ the hidden-to-output layer weights, $k_1...k_q$ the input-to-hidden layer weights, G the hidden layer transfer function, and F is the output transfer function. In many neural net structures the a vector would be a zero vector, as the inputs are not directly connected to the output layer. Typically, F and G are non-linear and bounded to the interval [0,1] or [-1,1]. White characterizes this representation as a "linear model augmented by nonlinear terms," and proposes that the hidden unit activations be considered as *latent* variables. Maddala (1983) discusses latent variables in terms of a representation of unobserved variables or errors in variables. Maddala also demonstrates how latent variables assist the modeling of interrelations between the variables. Because in a fully connected network, each hidden unit receives its own weighted sum of the inputs with a unique weight vector, hidden units can contain information about the interrelationships

among the inputs. If errors in variables exist, the hidden units effective combination of variables in different "strengths" using the weight vectors may provide a filtering process that corrects for both errors and outliers.

After training, the weight parameters in (1.7) can be subjected to the same type of hypothesis testing and inference process as is customary for non-linear estimation. Kuan and White (1991) provide examples of the Lagrange multiplier and the Wald test for neural network weight parameters.

Algorithms for non-linear parameter estimation are recursively defined and work iteratively until some condition is reached. Neural network learning algorithms have counterparts in non-linear regression, which are discussed in the next two sections.

1.6 The Kalman Filter

It may be useful for the econometrician to compare and contrast the neural network paradigm to the Kalman filter.[7] A neural network approach to estimation can overcome some of the limitations of the Kalman filter. In general, the Kalman filter requires a state space representation and uses a combination of the most recent observation with an estimator derived from all of the previous estimators to calculate a new estimator. The Kalman filter thus allows the incorporation of new information without re-doing any calculations with past observations, through a set of updating and prediction equations. This *ad hoc* type of updating is not available with neural

---

[7] Refer to Harvey (1981), Chapter 4, for a detailed discussion of the Kalman filter.

networks, and remains an advantage of the Kalman filter. For normally distributed observations or for linear estimation of the observations, the Kalman filter will produce the best available current estimator and optimal prediction.

The Kalman filter updating equations are strictly linear, using the difference between the most current obseration and estimator multiplied by a linear function of estimator and disturbance convariance matrices, called the Kalman gain, to calculate the estimator. Unlike the backpropagation learning algorithm, no partial derivatives are used to pro-ration the error. To begin generating the Kalman filter updating equations, initial values for the estimator and its covariance matrix are required, and the accuracy of these values will affect the time required for the Kalman filter to reach a steady state. In addition, the variances of the observation vector and the state vector process are assumed to be known. A neural network is initialized by randomizing the weights and requires no other input than the actual observations.

Although smoothing is available with the Kalman filter, a second set of recursive calculations must be performed after all the observations have been incorporated into the estimation. These recursive calculations must work backwards from the most recent information. Because a neural network produces a non-linear estimation of the functional relationship, it performs some smoothing during the same training process.

1.7 Non-linear Regression Iterative Methods

Two of the most widely used algorithms for parameter estimation are the Gauss-Newton and the Newton-Raphson iteration methods. In Judge, et. al. (1988), it is noted that Gauss-Newton and Newton-Raphson belong to a class of algorithms taking the following general form:

$$\beta_{n+1} = \beta_n - t_n P_n^{-1} k_n \tag{1.9}$$

where S is the objective function to be maximized or minimized, e. g., the sum of squared errors,

$$k_n = \frac{\partial S}{\partial \beta_n}\Bigg|_{\beta_n} \tag{1.10}$$

t_n is a positive constant and P_n^{-1} is some function of the reciprocal of the Hessian matrix,

$$\frac{\partial^2 S}{\partial \beta \partial \beta'}\Bigg|_{\beta n} \tag{1.11}$$

Using this general form, iteration proceeds until $\beta_{n+1} = \beta_n$.

Using this method, a non-linear functional form such as a Cobb-Douglas production function

(1.12) $f(K,L) = aL^bK^c$

must be specified as the proposed functional relationship. A neural network solution requires no prior commitment to a functional form as it produces a polynomial equation such as (1.7) based on the number of processing elements and connection weights.

1.8 General Applicability

Neural nets are used in a decision making process where incomplete or inconclusive knowledge about causality or linearity would make regression, discriminant analysis, maximum likelihood estimation, and other mathematical model-driven techniques ineffective. In this sense neural nets allow some form of imperfect or limited information processing that yields some benefit over no processing being performed. As such, we can consider neural nets a device for operating under *bounded* or limited rationality. Chapter 2 contains a detailed discussion of neural networks in this context.

1.9 Brief Summary of the History of Neural Network Research

McCulloch and Pitts (1943) are usually cited as the first researchers examining a neural network structure as a potential computing architecture, proposing it to be capable of any aritmetic or logical function. Wiener (1948) discussed neurons as

relays, and compared them to the computer hardware of that period in terms of information processing, long-term memory, and analogous malfunctions. For example, Wiener compared "out of memory" conditions to "anxiety neuroses," probably because the mind was so filled with problems that it had no room for solutions. These early explorations were highly theoretical, and there were no serious experiments until the late 1950's, when Frank Rosenblatt introduced the Mark I Perceptron, a neurocomputer that was capable of recognizing letters of the alphabet from a physical image.

Although Marvin Minsky's doctoral thesis was based on a model neurocomputer, he later collaborated with Seymour Papert (Minsky and Papert 1988) to show some severe limitations of perceptrons. These limitations included certain pattern recognition tasks and logical operations, namely, the "exclusive or", which a single perceptron could not learn. This work is credited with creating a "neural network winter," a period in which Hecht-Nielsen (1990) said, "neurocomputing had to go underground."

The Spring thaw which occured in the mid 1980's is traditionally credited to the physicist John Hopfield, who authored two papers on neural networks and personally convinced the academic and scientific communities to seriously consider the potential of the technology. The field quickly became re-legitimized by DARPA research, publications, conferences, journals, and business investment. One of the most popular research successes of the period was NETtalk, an experimental speech synthesis system developed by Terrence Sejnowski and Charles Rosenberg in 1985.

NETTalk uses a backpropagation neural network to learn to pronounce English words. Sejnowski and Rosenberg demonstrated NETTalk using the synthesized voice of a child learning to pronounce "tricky" words, such as "doubt."[8]

The research and application of artificial neural networks is currently supported by a substantial volume of theoretical and applied research, a number of neural network societies, a small number of journals, neural network computer hardware and software, and a growing number of fielded applications. However, a neural network approach to economics and other social sciences is still considered unconventional. What is needed is a bridge of theory and applications to connect traditional economic and econometric with neural network research methodology.

1.10 Economic Applications

In this section, neural networks are discussed as general function estimators, data classifiers, process simulators, and behavioral models. There are other applications beyond the scope of this work, mainly in the analysis of time series. Baum (1988) has presented neural networks as tools for economists, focusing on the backpropagation learning algorithm in the context of prediction of economic time series. Baum proposes that backpropagation works well because it can vary a large number of parameters (i. e., the weights), in relation to the "true number of

---

[8] The author heard a tape of the NETTalk system at a neural network tutorial led by T. Sejnowski and G. Hinton in 1987. The effect was extremely convincing in demonstrating the potential power of neural nets and drew spontaneous and enthusiastic applause.

independent variables." Chapter 7 contains a discussion on the topic of the degrees of freedom in a neural network, and section 1.13 in this chapter is intended to lead to the conclusion that it is the ability to easily create a relatively higher order polynomial expression that is the key to the predictive powers of a neural network rather than the ability to bypass the statistical requirement of sufficient degrees of freedom. Intuitively, if there are more connection weights than observations or data vectors, more than one solution may exist, and there is no constraint to find a better solution.

1.11 Case Study: Econometric Demand Analysis

Kuan and White (1989) compare a neural net approach to predicting appliance ownership with logistic regression and regression tree models. The purpose of their study was to come up with methods of forecasting energy demand by determining the demographic factors that influence ownership of specific household appliances. If appliance ownership can be predicted by commonly available demographic data reflecting household types, and the number of households in each category can be estimated, then demand for electricity and natural gas can be estimated as a function of the distribution of these households.

Their neural network architecture was relatively simple, having only 4 hidden units in a single layer. The weight vectors were derived by non-linear least squares (NLS) regression instead of back-propagation because of relatively slow convergence

of backpropagation relative to NLS.

The results of the study included the neural network out-performing the logit and regression tree models in many categories, especially in electric heating systems and electric appliances. What is interesting and not discussed by Kuan and White is the "story" told by the weight vectors that is consistent with expectations. The three highest values for each set of weights connected from each input variable to each hidden unit (not counting the constant) are as follows in Table 1-2:

Table 1-2 -- Neural Network Model for Electric Main Heating Systems[9]

From Input:	To Hidden Unit> #1	#2	#3	#4
Home Ownership		2nd (0.444)		
Year Built				
No. Bedrooms	3rd (0.59)		3rd (1.79)	1st (1.738)
Education		1st (0.891)		
No. Sq. Ft.		3rd (0.139)		
No. in Household	2nd (2.524)		2nd (1.992)	3rd (1.139)
Income	1st (2.60)			2nd (1.353)
Dwelling Unit Type			1st (2.302)	

Note: Entries in the table are the ranks of the weight values from the input units (rows) to the hidden units (columns) and the actual values of the weights (in parentheses).

[9] From Table 3B, Kuan and White (1989).

Number of bedrooms and number in household are among the strongest weights for 3 out of 4 hidden units. Income is in the strongest three weights for two out of 4 hidden units. Hidden units #1 and #4 both have Number of Bedrooms, Income, and Number in Household as the strongest weights. It would not be unreasonable to suggest that these hidden units detect a "relatively wealthy and large family." Hidden Unit #2, with Education, Number of Square Feet, and Home Ownership as the strongest weights, might suggest a preference for an investment in modernizing a home. A high income provides the means for a relatively expensive electric heating system, and household size provides the rationale for the flexibility and responsiveness of an electric heating system. For Hidden Unit #3, the type of dwelling is the strongest weight, and indicates another significant factor. In some literature, the hidden units are called "feature detectors" because certain hidden units will emphasize different variables. The results of Kuan and White could be interpreted to identify different economic rationales for choosing an electric heating system: (1) because household size makes it the best choice and income makes it feasible (Hidden Unit #1); (2) because the number of bedrooms makes it optimal and income makes it feasible (Hidden Unit #4); (3) because education level and home ownership suggest a preference for investing in modern technology (Hidden Unit #2); and, (4) because the type of dwelling unit usually calls for an electric heating system (Hidden Unit #3).

In standard econometric analysis, one might be concerned with collinearity between income and number of bedrooms, for example, because they might move up

or down in a close functional relationship. When a neural network is used, assumptions about the structure of the model and the expectations of knowledge to be gained are relaxed in exchange for a *satisficing* result. If the exact relationships were known, collinearity could be addressed a number of ways, and a conventional regression technique could be used successfully.

In three of the samples for electric appliances the logit model outperformed the neural network for predicting refrigerator ownership. It turns out that in the sample proportions in the paper, refrigerator ownership had the highest proportion in all three samples. Refrigerators are appliances that are highly scalable to family size and income; they range in price from about $100 for a 3 cu. ft. model to $6,000 or more for a restaurant-style model. The neural network was weak in its ability to separate the relatively few who don't own refrigerators from the majority that do own them.

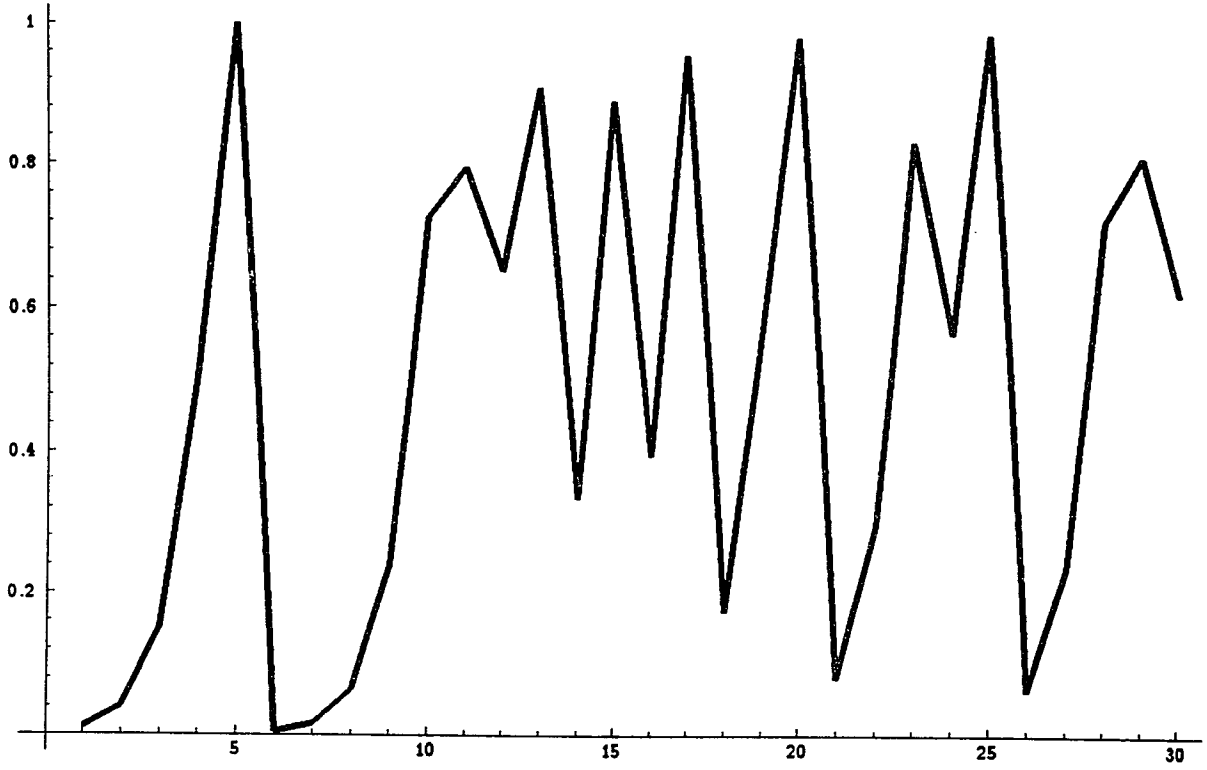
1.12 General Applications

Much focus in neural network research is devoted to the architecture of the hidden units, their number, how they are organized in one or more layers, and how these hidden units reorganize and interpret the information from the input layer. The terminology "hidden layer" intended to express the idea that, unlike the input or output layers, this layer does not have counterpart that was observable in a physical or sensory way. In a computer model of an artificial neural network, the term "hidden layer" acquires some misleading overtones of mystery and magic. It is

important to remember that a computer program is essentially deterministic in a properly functioning computer system, including one representing a neural network, and all processing elements are equally visible and obserable through the use of software utilities, such as displays, memory dumps, and so forth.

Lapedes and Farber (1987) proposed that the backpropagation algorithm could perform predictions for complex time series with chaotic characteristics with results superior to those gained using more conventional regression methods. Their experiments involved a number of 3-layer neural networks with a relatively small number of units in the hidden layer (five units), and their performance measurement statistic was the *root mean squared prediction error*, which is the variance in prediction divided by the standard deviation of the data. Two of the functions successfully estimated were the simple $y = x^2$ and the Feigenbaum map, $x_{t+1} = 4bx_t(1 - x_t)$. Figure 1-5 shows the first 30 terms of the Feigenbaum map, with $b=1$ and $x = .01$ to give an idea of the complexity. These accomplishments led Lapedes and Farber to state,"..non-linear neural nets can very accurately model polynomials, in addition to more general nonlinearities." Their work is frequently cited in the literature, and is an enthusiastic complement to the proof of Hornik, Stinchcombe, and White, which is discussed the next section.

Figure 1-5 -- The First 30 Terms of the Feigenbaum Map



1.13 Function Approximization by Polynomial Equations

Mathematical functions and automatons are related in that one of their results is a set of input and output mappings. MacLane (1986) makes the analogy of a function as a machine, and Narendra and Thathachar (1989) define the automaton as a set including two functions, and suggest that "simulated" neural networks are a form of learning automaton. The workings of the automaton include a set of states in addition to the sets of inputs and outputs. A static mathematical function can be thought to have one "state," which is the specification of the terms and coefficients of the function, and a neural network's states are the values of the weight vectors during training. A trained neural network, like a mathematical function, has a single state.

Earlier, an example of using a neural network to perform the same role as linear regression was discussed. The result was a polynomial function that produced a curve which was a better fit than the linear regression, facilitated by the use of a hidden layer. The next question is thus: how strong a statement can be made regarding the use of neural networks as approximators of functions?

Hornik, Stinchcombe, and White (1989), hereafter HSW, have published a rigorous proof that multilayer feedforward neural networks with at least one hidden layer can approximate "any Borel measurable function from one finite dimensional space to another to any desired degree of accuracy, provided sufficiently many hidden units are available." The requirements of Borel measurability and finite dimensional spaces make this concept more "rational" to the economist and econometrician

because economics "deals in" constraints and resource limits as a justification for its own existence.

Their proof is derived from the Stone-Weierstrass theorem. Among the many published versions which all say the same thing, Rudin's (1964) is the one of the most concise:

> If f is a continuous complex function on $[a,b]$, there exists a sequence of polynomials P_n, such that
>
> $$\lim_{n \to \infty} P_n(x) = f(x)$$
>
> uniformly on $[a,b]$. If f is real, the P_n may be taken real. (p. 146)

The Stone-Weierstrass theorem can be paraphrased to say, "any functional relationship that can be expressed in real numbers can be defined as a polynomial as long as there can be enough polynomial terms to approach the required degree of accuracy or match to the original function." The Taylor series approximation, familiar to economists, is a example of using the Stone-Weierstrass theorem.

The hidden units of the neural network create the polynomial terms through the use of a "squashing function," i. e., a function $Z: \mathbb{R} \to [0,1]$ which is non-decreasing and $\lim_{x \to \infty} Z(x) = 1$ and $\lim_{x \to -\infty} Z(x) = 0$ (see HSW). Squashing functions, such as the *sine* and *tanh* functions which are commonly used as transfer functions for neural networks, all have power series approximations (see Spanier and Oldham (1987)). In addition, the use of a hidden layer increases the number of coefficients

and polynomial terms, by increasing the total number of connections. For example a network with 5 input units, a hidden layer of 4 hidden units and 3 output units has 5 x 4 + 4 x 3 = 32 available connections. Without the hidden layer, only 5 connections would be created, the same as for the number of coefficients in a standard linear regression.

Although it can be said that, in general, the hidden units create a polynomial approximation of a function, specific neural network learning architectures use hidden units differently. In backpropagation, it was seen in the example from Johnston (1984) that the hidden units allowed the creation of a non-linear functional relationship. A counterpropagation network uses the hidden units to learn the mean values of the data elements for each class of data. Other architectures, such as the Bidirectional Associative Memory network and the Self-Organizing Map, do not have hidden units in the exact same sense as the backpropagation network, but the presence of processing elements in addition to those for input and output creates additional weighted connections. Chapter 8 presents a comparison of the activity of the hidden layer in various neural network architectures.

1.14 Classification

A counterpropagation network, introduced by Hecht-Nielsen in 1986, functions as a data classifier. As the network is trained, the hidden units become "prototypes" for the different classes represented by the data. The learning algorithm forces each hidden unit to "specialize" in recognizing a member of a particular class. If enough

hidden units are available, more than one hidden unit may recognize the same class. Some hidden units may fail to learn to recognize any class. The result after training is that the weight vectors to each hidden unit assume the mean values for their respective input elements. This is accomplished by a special transfer function

(1.13) h_i = 1 if $D(w_i^{old}, x) <= D(w_j old, x)$ for all j

0 otherwise.

In other words, the hidden unit h_i with a weight vector w_i that is closest to x by having the smallest Euclidian distance D from x will output a value of 1, while all other hidden units will output a 0. In addition, during learning, this "winning" hidden unit will adjust its weight vector with the Kohonen learning algorithm

(1.14) $w_j^{new} = (1 - a(t))w_i^{old} + a(t)x.$

The time-dependent constant $a(t)$ assumes a value between 0 and 1 which decreases in time.

In Chapter 6, a counterpropagation network to simulate the classification process of corporate bond ratings is presented. Using the Standard & Poor's ratings classes and corporate industrial financial data, our networks were trained and tested,

differing only in the number of hidden units. All four networks succeed well in discriminating investment grade companies and speculative grade companies, but failed to consistently recognize poor quality or be able to distinguish between rating classes such as A and A-. The following chapter, Chapter 7, addresses the problem of neural network stability and explains the counterpropagation network's performance for this application.

In this type of application, the goal is to have the network learn to simulate the process based on the data. Its performance can never be a more accurate rating system than the Standard & Poor's examples fed to it during training. In other words, it would be a mistake to have this neural network compete on the validity of the ratings with the human committee at Standard & Poor's. It might be appropriate to compete on consistency, however. A direction for future research would be to incorporate some of the qualitative factors used by S & P, such as confidence in management, and assess the results.

1.15 Summary

The theory and application of artificial neural networks provides a useful extension to existing and accepted practices in economic analysis and econometrics. Neural networks can be especially effective when non-linear or fuzzy set relationships are suggested by the data or problem to be considered. Neural networks can model non-linearity through the use of non-linear transfer functions. For fuzzy set modeling, the output values produced in the counterpropagation network can be

interpreted as a form of fuzzy set membership. In econometrics, the same statistical issues exist; a neural network cannot compensate for a poor sample, extreme heteroscedasticity, or any qualities in the variance which make estimation difficult in general. However, neural network methods can reduce the dichotomy between linear and non-linear methodologies.

Future research should emphasize neural networks to study the behavior and interactions among economic agents. Interactions between social and economic factors in human behavior can be modeled in the activity of the hidden units. In addition, where there is competitive learning, such as that in counterpropagation networks, there is a potential for game theory modeling as well. From the perspective of neural network efficiency, some work needs to be done in better estimation of the required number of hidden layers and units. At present, this is done by trial and error.

What is most important is to recognize that artificial neural networks are an extension of existing theory in mathematical analysis, optimization and logic, as well as an extension of statistical methods. Powerful computer software and hardware make the massive parallelism feasible as well minimizing the frustrations of the trial and error method of neural network design.

PART ONE

RATIONALITY AND BEHAVIOR

CHAPTER 2

NEURAL NETWORKS: AN IMPLEMENTATION OF HERBERT A. SIMON'S PROCEDURAL RATIONALITY

2.0 Overview

Herbert A. Simon has proposed approaches to problem solving and decision making that address real world conditions, including the computational, data storage capacity and temporal limitations of the human mind. *Bounded rationality* (Simon 1955) is the *portmanteau* for his proposed set of replacements to "classical" assumptions about rational economic behavior. *Procedural rationality*, as a subcategory of bounded rationality, is the theory of "...efficient computational procedures to find *good* solutions,"[1] as opposed to exhaustive computational procedures to find *optimal* solutions. Decisions or selections are almost always made under time and information constraints and therefore, optimality is sacrificed in exchange for a solution that is good enough to meet the requirements at hand. This is Simon's well-known idea of *satisficing,* which is a procedurally rational strategy. The economic effect of procedural rationality is the cost-benefit result from the process used to reach a goal. The cost of goal attainment or decision making could

---

[1] p. 133.

be significant, and should be factored into the ultimate payoff. The intent of procedural rationality is to prevent scenarios such as that of the stereotypical "bargain hunter," who, in pursuit of the lowest possible price, often spends enough time and money to use up the additional savings. More formally, Walliser (1989) discussed the search procedure employed by an agent as an "instrumental rationality" problem; the agent visits a number of producers until the desired product could be purchased at a price lower than a "reservation price," as opposed to some "optimal" price which may be costly to determine. Instrumental rationality and procedural rationality are synonomous in this context.

It is proposed here that neural networks provide a technology for decision support applications that is consistent with procedural rationality, and, hence, bounded rationality as well. In (Simon 1982), a number of ways of applying bounded rationality to practical problems are introduced. Of particular relevance to neural network applications are the bounded rationality assumptions that "the actor has only incomplete information about alternatives," and that there is "complexity in the cost function or other environmental constraints so great as to prevent the actor from calculating the best course of action."[2] Neural networks accommodate the incomplete information and computational complexity by their ability to generalize, for example by the use of "nearest neighbor" algorithms (see (Kohonen 1989)) and to approximate non-linear functions (see Hornik, Stinchcombe, and White 1989). Neural networks are constructed from a set of easily understood primitives, usually

---

[2]p. 163-164.

called artificial neurons or neural processing elements (PEs), which contain weighted summation and non-linear transfer functions and use simple weighted connections to pass information from one PE to another. As such, neural networks allow the magnitude of complexity to be driven by the numbers of these primitives. Thus, the effect of small increments in complexity can be measured, and the overall cost of the model is a simple function of the costs of the primitive elements. If relevant, neural networks' capabilities extend to modeling the effects of time; delays are represented by the arrangement of the PEs. A neural network architecture contains a highly generalized definition of the functional relationship between the inputs and outputs, and an adaptation (or learning) mechanism for deriving a solution or arriving at a goal state. Thus, there is relief from the task of choosing a specific functional form and going through expensive parameter estimation procedures only to discover that the chosen function is incorrect. We examine some essential characteristics of neural networks to understand the relationship to *procedural rationality*: adaptation, the basic structures common to all neural networks, and communication through bounded transfer functions. As decision support or information processing devices, neural networks provide a cost effective method by being relatively simple to design and modify. An analogy is the building of a complex machine using interchangeable parts from a standard inventory. Radner's (1992) references to Simon's "parable of the two watchmakers," is apropos here, in that the point is made that to be useful, complex systems must be designed in such a way as to be both composable from and decomposable into standard "subassemblies." As models for economic functional

relationships or decision support systems, neural networks meet these criteria well.

2.1 Background

There is a certain "attraction" to neural network concepts, and to modeling decision and control systems after the nervous system. Anderson (1986) cites a long history of humans aspiring to build machines to emulate all or some aspect of themselves. Even the early computers of the 1940's and 1950's, which were limited to certain classes of calculations, were called "electronic brains," as if they functioned identically to human intelligence. Wiener (1948) also aroused considerable interest with Cybernetics:or Control and Communication in the Animal and the Machine, which introduced the study of feedback and adaptation, drawing parallels between the human brain and electronic computer systems. There appears to be a consensus that emulating brain function, specifically the cortex, is a way to make machines behave more intelligently. As Anderson (1986) points out, the cerebral cortex "...evolved recently and is anatomically rather homogeneous, with only a few cell types and standard connection patterns to deal with," and "...may be more easily understandable than the older, more optimized parts of the nervous system."[3] As the cortex contains the functions of language and perception, it is of greater interest than other parts of the nervous system.

Human intelligence, with emphasis on the ability to adapt, generalize, reason with missing information and/or "fuzzy" logic, and Simon's *satisficing* concept are

---

[3] See *General Introduction*, page xiv.

closely related. Biological networks have evolved based on requirements for survival, which means making adequate decisions as necessary under non-negotiable time and resource constraints. Decisions to quickly hide or flee from a larger animal without being certain whether it will attack, and to avoid consuming bitter-tasting substances without knowing if they are poison have probably done more good than harm to the continuation of the species.

Nervous system development is related to the size and complexity of the animal. It takes more neurons and a larger network to control the actions of a larger animal, versus, for example, an insect. From a biological standpoint, there is an efficiency tradeoff - longer gestation periods and longer dependency on parents, which are necessitated by a complex nervous system's longer developmental and training time. There is also a biological tradeoff in the number of species versus the complexity of the nervous system. For example, there are over 700,000 known species of insects, which is more than three times as many species as all other animals together. The primary difference is that insect species adapted their physical characteristics to their environment, requiring many species for the varieties of climate and terrain, while higher animals adapted their behavior. Economic systems and social organizations also encounter the need for complex information networks and typically show slower adaptation as their size grows.

The seeking of inspiration from the biological nervous system is a common theme among economists. Both the limitations and the sophistication of the nervous system are used to describe economic institutions and economic activity. Knight

(1933) discusses economic organizations in terms of the human body with specialized organs and cells, and compares the complexity of the nervous systems of higher animals with the hierarchies of large organizations. Arrow (1974) indirectly compares the human body to organizations by discussing the physiological limits of the brain and sensory organs that constrain the acquisition and use of information to the organization concept of "span of control." (p. 39). Simon (1988) compares a government to a human being, citing both as examples of serial processing systems that can only focus on one thing at a time. Kornai (1971) has taken the view that an economic system has two sets of functions analogous to the "autonomous and higher functions" of the nervous system. The autonomous part is the basic production and trade activities common to all economic systems, analogous to breathing and digestion. The higher functions are related to macro-level decisions that are concerned with investments, technology, and broad aspects of production and consumption. Thus, the higher functions are analogous to the part of the nervous system performing "intelligent" activities, such as reasoning and language.

In a very general sense, there exists many parallels between neural networks and organizational structures created by societies to manage social and economic behavior. These parallels are in the very basic structures themselves, and the rationality behind these structures is to assure survival under time and resource constraints. Designing and modeling complex relationships and actually implementing real working systems with interconnected simpler structures appears to be an intuitive tactic to gain control over the complexity.

2.2 The Evolution of Structure: Firms, Social Organizations and Neural Networks

Much of traditional economic theory is concerned with the evolution of the firm, markets, and organizational hierarchies. As outcomes of allegedly rational behavior, these entities are credited with the economic benefit of lower costs for any given level of economic utility. Layard and Walters (1978) offer the idea that the firm lowers transactions costs by offering long term employment contracts, thus eliminating the time consuming negotiations that would take place if labor and the firm bargained over every task. The firm also generates relatively fewer transactions by purchasing and selling in volume, and creating a finite set of product lines and prices. In contrast, an individual craftsperson may not have a finite product set and must engage in price negotiations for each purchase of raw materials and sale of finished goods. Many organized markets also lower transactions costs by disseminating information, and creating and enforcing rules for trade. A major part of the lowering of costs comes from the structures that characterize firms, markets, and organizations. These structures standardize the exchange of goods and information essentially by putting constraints on the channels and the data that flow through these channels. These constraints go hand in hand with the specialization of function that characterizes economic entities. Firms, governments, and other organizations all seem to have the same general pattern as they grow larger: tasks become more specialized and compartmentalized, so that one person does not have to know everything to be productive, presumably because it is impossible to know everything when an organization reaches a certain size and complexity.

Communication also becomes more structured and rigid in the sense that "hard wired" channels are established between departments. Neural networks also employ structure, specialization, and "hard wired" connections. In organizations and neural networks, adaptation is usually achieved by small changes at local levels. Neural networks adapt by very small changes in the weights associated with the connections. Organizations may adapt by changes of procedure in individual departments. Radical changes in organizations, such as top-down restructuring, do occur, but they occur less frequently than the smaller changes, and they are much more disruptive, often temporarily impairing the organization from performing its primary functions. Biological organisms cannot afford this level of disruption; they will usually die.

Consider price negotiations as an economic activity which has evolved to be a highly structured and standardized process. Stock markets, automobile dealers, and labor markets, as examples, all have highly specialized protocols. What these markets have in common is that buyer and seller must adapt somewhat to each other's needs before an exchange can take place. Agreement on price is thus usually an adaptive process, similar to the learning process of neural networks. In bargaining, buyer and seller begin at some level that is "optimal" for each party individually, in that the buyer's first offer is as low as it can be without cutting off communications from the seller, and the seller's price is as high as it can be without discouraging the buyer. Then, each successive offer and counteroffer is based on the most recent price "on the table," with the buyer's price moving up and the seller's price moving down. The final price is less than optimal for each party separately, but

allows the transaction to proceed. In a neural network, the connection weight is adjusted by a learning algorithm until a value is reached that allows the network to function at some acceptable error level. The difference between the ask and bid price can be thought of as an equilibrium error that is corrected by the adaptive bargaining process. Albin and Foley (1992), discussed in Section 2.6, use an adaptive process to reach a state of equilibrium in a decentralized market model.

Standardization of the communications between structures allow these forms of adaptation to take place. Another form of adaptation in economic systems is how firms and industries grow to a size that benefits from the economies of scale. For example, in any one service industry, one expects to find a greater variance in the size of firms than in durable goods manufacturing. A software developer can be a one employee firm or a firm with thousands of employees, and this wide variation in size benefits this industry, because the consumer arguably has the choice between the innovation and flexibility of a one-person firm and the production capacity of a large firm. In contrast, an automobile manufacturer is dependent on the massive amount of capital and the degree of labor specialization that can only be made available in a very large firm. Of course, a one-person automobile firm could exist for the purpose of making highly customized automobiles for a limited market. More than the type of industry, organizational complexity is driven by goals such as gaining and retaining significant market share, providing support and continuity for customers, and increasing the wealth of the owners or shareholders. Arbib (1987) proposes that a combination of structural complexity and interconnectivity

serves as a facilitator of the kind of adaptation that is recognized as human intelligence. Albin (1975, 1978) suggests a parallel in societies of a necessary "critical mass" of complexity in order to generate new technologies and culture, which could be considered intelligence on a cooperative or societal level. Hopfield (1982), in a similar vein, reintroduced neural networks after a period of neglect by the research community:

> Much of the architecture of regions of the brains of higher animals must be made from a proliferation of simple local circuits with well-defined functions. The bridge between simple circuits and the complex computational properties of higher nervous systems may be the spontaneous emergence of new computational capabilities from the collective behavior of large numbers of simple processing elements.[4]

All of these statements carry the theme of increased complexity accompanying positive advances, and of gaining control over complexity through the use of simple structural elements and communications protocols.

2.3 Neural Network Parallels to Social Organization Structures

Learning, adaptation (or, behavior modification), and connectivity are important characteristics of neural networks, and, as such, a comparison to social structures is intuitive and appropriate. In a society, communication is the binding factor that

---

[4] p. 464, reprinted in Anderson and Rosenfeld.

allows the perception of a collection of humans or animals as a group. In his chapter on "Information, Language, and Society," in Cybernetics, Norbert Wiener describes various forms of communication in both human and animal societies to draw a relationship between information exchange and adaptation. He also views society-level information content and communication to be linked. If members in a group possess information, but no communication between them takes place, the information content of the group is nil. This is precisely the same state as an untrained neural network because no information has traversed the connections. Another way of viewing this is that the characteristics of the group are contained entirely in the communications between its members - it is impossible to learn the nature of the group from observing one member without observing communication. Conversely, it may be possible to learn the significant characteristics of a group by observing only its communications. In fact, much of archeological research is conducted by studying ancient communications. Thus, a neural network, in which all of the network-defining information is contained in the weighted connections, is an appropriate model for organizations. The remainder of Section 2 discusses organizational models that have similarities to neural networks.

2.4 Neural Networks and Complexity Measurement

Albin (1980) has proposed that representation of social structures as non-directed graphs provides a means of quantifying organizational complexity. He also demonstrates the measurement of complexity through the use of complexity ratios.

In his analysis, the nodes of a graph can represent committee members or committees (which can include a committee of 1 person) and the edges (or connections) to each node represent communication channels. The complexity of the total structure is a function of the order (number of nodes) and the average degree (i. e., connections) of each node. The example discussed involves comparing a group of individuals each communicating directly with each other with the same number of individuals organized in committees. The communication path is one bit wide, which simplifies the example. In addition, Albin's 'complexity measures' are "cardinal numbers or indices that will preserve rankings under linear transformations."[5] In this section, complexity measure will be called a score. Scores conforming to the aforementioned can be used to develop a cost function by assigning a price per complexity unit. Although neural networks are usually a form of directed graph, Albin's principles can be applied with useful results.

The committee/sub-committee example from Albin (1980) uses the following formula for complexity measurement:

$$C = N^3 d 2^d + N^2(2d2^d + 1) + N(d2^d + 1) \tag{2.1}$$

where N is the number of nodes and d is the average degree (number of connections) of a node. C is related to the longest path without retracing an edge or connection, and is a computational approximation for large values of N and d. The discussion

[5] p. 103.

proceeds to a group of 24 people each communicating directly with each other; the graph is 24 nodes of degree 23, and C = 2.8940697606E+12. Reducing the average degree to 4 by reorganizing the group to 6 subcommittees of 4 people each results in C = 960600. The ratio of the first measure to the second is 2.8940697606E+12 / 960,600 or 3,012,773 to 1. Thus, it is shown that committees reduce organizational complexity. Carrying the example further to include cost considerations, the cost of a unit of complexity could be a fixed price, incorporate "volume discounts," or conversely increase with volume, reflecting the increased costs of managing a complex organization. Therefore, the relative costs of varying degrees of complexity can be measured and factored into the decision making procedures.

In (2.1) the complexity increases at a faster rate with the growth of d than with the growth of N. For example, at $N=d=2$ and above, first partial derivatives show that d always has the greater effect because it appears as both a coefficient and an exponent.[6]

Neural networks tend to resemble directed graphs, but the same principles apply. A complexity measure for neural networks could also be developed as a function of processing elements (analogous to graph nodes) and connections (analogous to edges). Since the connections contain all the network information (i.

---

[6] $\frac{\partial}{\partial d} = 2^d N^3 + 2^d d N^3 \text{ Log}(2) + N(2^d + 2^d \text{ d Log}(2)) + N^2(2 \text{ x } 2^d + 2 \text{ x } 2^d \text{ d Log}(2))$

$\frac{\partial}{\partial N} = 1 + 2^d \text{ d} + 2 (1 + 2\ 2^d d)N + 3\ 2^d \text{ d } N^2$

e., the values of the weights), and the processing elements contain mainly summation and transfer functions, and do not retain any information unique to the data, the number of connections should dominate as a complexity measure. For a fully connected three-layer network, in which each processing element a layer is connected to each processing element in the next layer, the total number of connections is

(2.2) $$X = (i * h) + (h * o), \text{ or } (i + o) * h$$

where i is the number of input PEs, o is the number of output PEs, and h is the number of hidden PEs. Clearly, in a fully connected network, each type of PE has an equal role in changing the overall complexity. In practice, much effort and experimentation has been devoted to the hidden layer, and to learning algorithms for improving network performance. Formula (2.2) shows that a change in the representation of the input or output that would reduce the number of PE's needed in these layers would have an equally beneficial effect on reducing complexity. Any change in this neural network would result in a *linear change* in the number of connections, which is a useful measurement attribute.

Neural network complexity could be measured and assessed with regard to economic benefit. In Chapter 6, it was shown that three neural networks of varying size performed similarly for S&P bond rating estimation and general bond quality classification (i. e., investment grade, speculative, and poor quality). For the quality classification, the size of the network and overall accuracies were:

Table 2-1 -- Relative Accuracy for Three Classification Networks

Input	Hidden	Output	Total Connections	Accuracy
88	170	3	15,470	83.78%
88	264	3	24,024	84.24%
88	425	3	38,675	84.40%

The ratios of total connections and accuracies at each level show that going from 170 to 264 hidden PEs increased the complexity by 55.3% and only increased the accuracy by 0.5%; from 264 to 425 hidden PEs, increased the complexity by 61% and the accuracy by 0.2%; and going from 170 to 425 hidden PEs increased the complexity by 150% and increased accuracy by only 0.7%. Clearly, in this context, if a cost per connection were set, a procedurally rational question could be asked: "How much is one willing to pay for each 0.1% increase in accuracy." The answer is not obvious without a context because the application determines the benefit of each increment in accuracy. Matching colors of house paint versus preventing meltdowns in nuclear reactors are two extreme examples of possible contexts, which

suggest that a wide range of choices could exist regarding very small changes in accuracy.

2.5 Committees

There are a number of qualitative parallels between neural networks and committees (which may also apply to departments, agencies, or any other subgroup in organizations):

(1) *Standardized structures:* Committees have meetings, chairpersons, agendas, reports, and so forth. A particular organization may establish several committees to perform different functions, but the structure of each committee is practically identical and is broad enough to accommodate all of that organization's needs. The committee protocols allow for both communication within the committee and for interaction with other committees. Neural processing elements are also standardized structures, having weighted summation and transfer functions. In addition, the network of weighted connections is standardized for communication of information to other neurons and network layers.

(2) *Constrained Inputs and Outputs*: Committees select information (input) based on its relevance to their assigned function. Committee deliverables (output) are also strictly defined by their function. It is very rare in an organization to have a committee report on interesting by-products of its research and effort, especially when these by-products are relevant to the function of another committee. For example in an organization, there may be one committee to decorate the offices for

the Christmas holidays and another committee to take care of the refreshments for the Christmas party. If someone in the decorating committee independently decides to bake cakes for the party, it will cause problems in the organization, because the act itself can be interpreted as either a "good faith" gesture to help or an attempt to undermine the other committee's power. Even if the decorating committee member "asks permission" to supply the cakes to the refreshments committee, the information may not be well received or considered because it is not expected output from the decorating committee.

People frequently cite this type of situation as an example of the maladaptive qualities of organizations. One often hears the complaint that creativity and initiative are discouraged in large organizations and that the constraints cause inefficiencies. The counter-argument is that the purpose of committees is to preserve the larger organization, and that too many independent actions are threats to the organization's survival. This is a form of bounded rationality, in that there are insufficient resources to continually review ideas and proposals that are not part of the existing systems. Organizations can achieve a balance by establishing departments which have the sole purpose of evaluating new ideas.

Neurons can also be thought of as committees that select their inputs by adapting so that irrelevant inputs have their respective connection weights set to zero, so that they have no influence on the neuron's output. Neural network functioning does break down when the input varies greatly from that used in training, or when the training input has not been properly scaled. An analogous breakdown occurs in

organizations when information is not recognized or pre-conditioned.

(3) *Weighted Summation and Transfer Functions:* Individuals and organizations apply weighted summation and transfer functions to decision making. For example, a manager may consult several subordinates and give different weights to each subordinate's comments, based on the manager's assessment of the respective subordinate's competence. For a transfer function, the manager may modulate and interpret the weighted sum of the subordinates' opinions to produce a decision that is compatible with the goals and expectations of the manager's superiors.

(4) *Parallel Operation*: One reason that organizations use committees is to have dedication to various tasks in parallel. The benefit of parallelism is time savings. Returning to the Christmas party analogy, having the entire organization work on one thing at a time would result in a longer elapsed time to plan the party. Neural networks are organized in layers of neurons, so that the neurons in each layer can act in parallel on the same set of inputs.

(5) *Limited Global Information*: The processing elements in most neural network models share virtually no information. All adaptation is based on a PE's immediate inputs and the learning algorithm is a function of local information only. In large organizations, each department or committee also has primarily local information.

2.6 Applications to Communication and Decision Making in Organizations

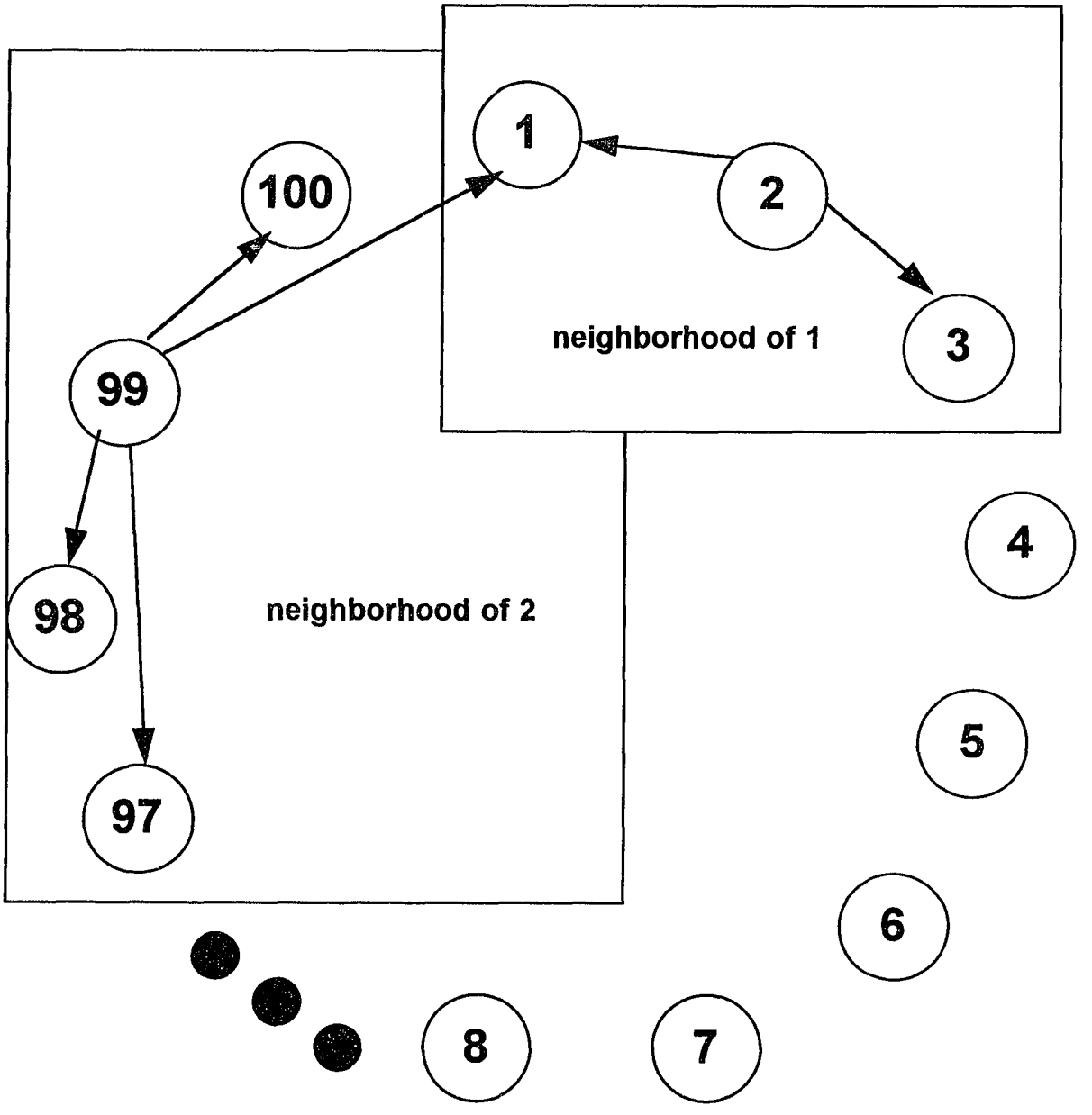
A remarkable result of neural network processing is the arrival at some desirable global information state using almost exclusively local information.

Processing elements contain only local memory, and, when organized in layers, receive information from a previous layer. No PE has information about the desired global state, so each PE is acting in a strictly bounded rationality context. This concept has been used in a number of network-like models of organizational behavior. The following examples show how adaptation compensates for lack of globally available information.

Radner (1992) and Radner and Van Zandt (1992) have applied a neural network-like model to study the effects of information processing capacity on returns to scale. Their model uses the parallelism and decentralized processing with the intention of minimizing a time delay. The application is decision making in large firms, where tasks may be too large for an individual to complete in a reasonable amount of time. Although the task is decomposed into subtasks that can be assigned to individuals working in parallel to reduce the time delay, there is a point at which adding more individuals ceases to improve the response time. This is around the point at which there are more individuals than subtasks. The tradeoff is the cost of the processors (personnel) versus the cost of the delay in information processing. Radner (1992) also discusses decentralization of information in large firms in which the decision-making is shared by many managers.

Albin and Foley (1992) incorporate the neural network concepts of adaptation and localized information in their model of decentralized exchange between agents. Their model has agents arranged in a circle (Figure 2-1) bidding and trading only with their 1 or 2 nearest "neighbors" on either side, aiming to maximize their own

Figure 2-1 -- Decentralized Agents from Albin and Foley (1992)



utility. There is an advertising, or communication cost that is proportional to the number of neighbors receiving the messages. Agents have no global knowledge of the aggregate wealth or of the Walrasian equilibrium price. In spite of each agent's limited influence and loss of wealth through advertising expenses, the result is that there is an overall improvement in the allocation of resources, as measured by the Debreu *coefficient of resource utilization*, which determines how to distribute the available wealth to equalize the utility of each agent. In the end, all agents have increased their utility and hence, everyone is better off. Just as neural network learning causes modification of connection weights with each training example, the agents use prior advertising and trading experiences to improve their chances of increasing their utility in the next round of trading. This is an example of how neural network like structure provides a framework for globally beneficial economic activity with limited computational resources. A classical optimizing approach would be all agents having knowledge of every other agent's situation in addition to some global parameters, but the result would be computationally intractable for a realistically sized economy.

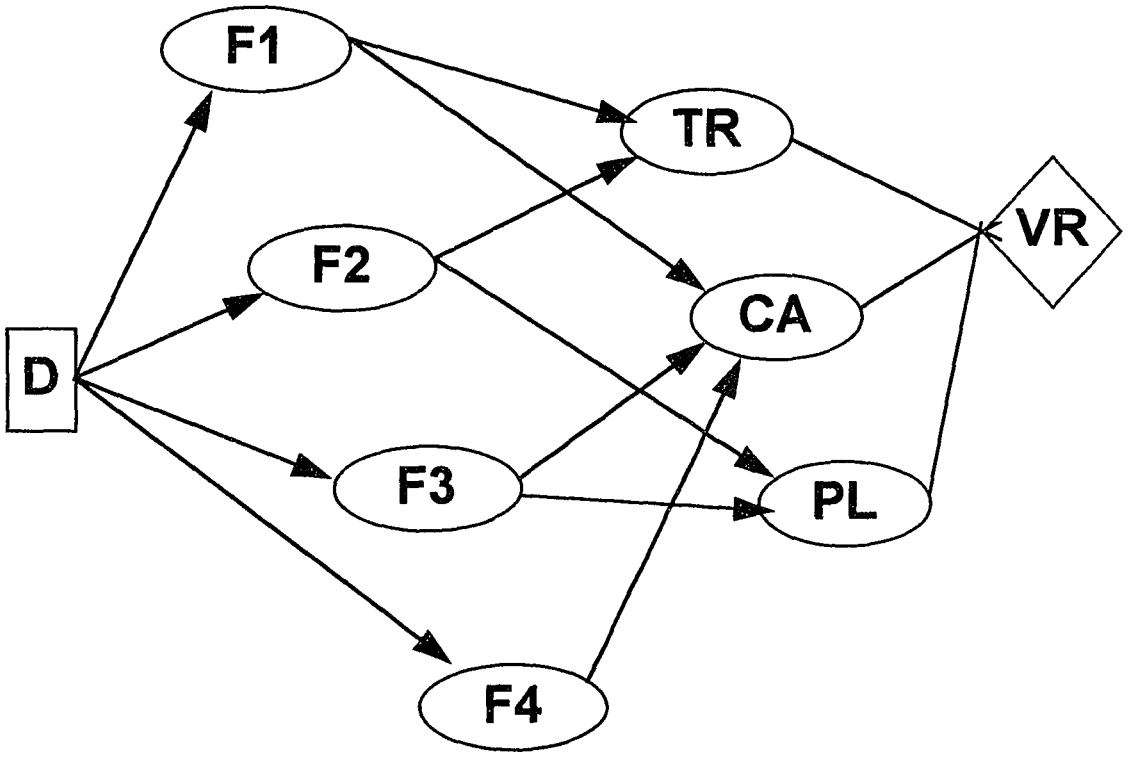
Lucas (1973) used a model with distributed information and decentralized agents to show conditions under which the inflation-unemployment tradeoff, or Phillips curve, breaks down. Lucas' model involves supplier agents who must independently set their supply levels by estimating the market price. The only global knowledge is the relative distribution of the global price, and the mean of the general price level. The result is that even though agents have mostly local information about their past

supply and prices, they are able to learn to distinguish the difference between price increases caused by inflation versus price increases caused by real growth in demand, mainly by measuring the volatility of past prices. Thus, agents with imperfect information, acting under bounded rationality conditions, could not be easily "fooled" because they adapted to responding to the local information that most accurately described the global economic conditions.

Although general uncertainty was more of a factor than the unavailability of global knowledge in Munier's (1989) exploration of using *influence graphs* as a decision making model under procedural rationality constraints, this form of network model is relevant to the present discussion. The *influence graph* is organized in layers; the "procedures physically available which would give different levels of VR (Valued Result)"[7] correspond to the hidden layer of a neural network; the factors that influence the procedures correspond to the input layer; and, the Valued Result corresponds to the output layer (see Figure 2-2). In Munier's model, the layers are

[7] (Munier 1989) p. 96.

Figure 2-2 -- Munier's Influence Graph



D - decision
F1-F4 - factors of Influence
TR,CA,PL - possible processes
VR - valued result

interconnected in exactly the same fashion as neural network layers, each node from each layer is connected to every other node in the next layer. This model has no weighting of the connections or adaptation, but could be combined with neural network concepts to simplify the structure of each node, especially the procedure nodes, which might not have to be initially specialized, but would become specialized by the learning process.

2.7 Committee-Like Neural Networks

Nilsson (1990) introduced the general concept of a "committee machine,"[8] a multilayered neural network which modeled a layer of neurons as a committee that would vote to determine how to respond to an input pattern. The simple majority vote determines the output value. The committee machine is a method to identify patterns which are not linearly separable, by having a majority of the neurons select the correct response. Thus, the committee is capable of solving a problem more complex than that solvable by a single processing element.

The counterpropagation model of Hecht-Nielsen (1986) uses a slightly different form of "agreement" among the neurons in the hidden layer. Instead of voting, there is a "winner take all" strategy, whereby the neuron that has connection weights that most closely fit the input vector communicates a positive signal of 1, while the rest of the neurons communicate a zero signal. Counterpropagation networks are used for classification of data; the benefit of the committee approach, in which each

---

[8] Actually appeared in the first edition of the book, published in 1965.

hidden unit acts as a committee, is that committee members learn the characteristics of different classes, so, as a whole, they are capable of differentiating among the classes. Each hidden unit committee member becomes a "specialist" in identifying a specific class.

The common thread between Nilsson's committee machine and Hecht-Nielsen's counterpropagation network is a primitive form of cooperation among neurons, which is enforced by a higher-level process. In this case, the entire layer acts as a committee, which each PE serving as a member of the committee. Although there is a global process in effect, there is still no global information, as each PE independently reacts to the input. Chapter 3 introduces a variation of the committee network that combines Nilsson's paradigm with Hecht-Nielsen's counterpropagation and incorporates the committee model of Sah and Stiglitz (1988).

2.8 Simon's "Simplified Payoff" and the Transfer Function

Neural network transfer functions play a special role in reducing complexity by converting all signals to finite open intervals. One of Simon's (1955) suggestions for applying bounded rationality was to simplify the payoff function. His example of a seller of a house is very similar to the neural network concept of a threshold value and a transfer function. In his example, the price of a house is set at $15,000, and although the seller prefers the highest offered price, the seller is in a position where a limited number of offers arrives at a point in time, and the seller must accept or reject each offer without knowledge of any future offers. Therefore, it is suggested

that an "acceptable" price is set, which is at the boundary of indifference between selling and not selling. If the offered price is at or above this acceptable price, the seller sells, and his payoff (utility) is +1. Otherwise the payoff is 0. This concept is functionally identical to a neural network step, or signum, transfer function: if the weighted sum exceeds a value, conventionally 0, the neuron's output is +1; otherwise it is zero.

In both Simon's example and in neural network architecture the transfer function serves two roles: to create a bounded interval for standardizing communications and to give a rational shape to the output function. Three typical neural network transfer functions are the signum function (Figure 2-3), the logistic function, $e^x / (1 + e^x)$, similarly, the sigmoid function $1 / 1 + e^x$, (Figure 2-4) and the hyperbolic tangent function (*tanh*) (Figure 2-5). These three functions have key common characteristics:

1. *Their boundaries are related to the unit interval*, either exactly [0,1], or [-1,+1] in the case of tanh, which is a bipolar transformation of the sigmoid function.

2. *Their first derivatives reach the maximum value at a mid-point value for each transfer function*, which is zero for the *tanh* and 0.5 for the logistic function. Thus, the slopes are at their maximum at the point that is farthest away from the function limits. If the extreme values of these functions represent "true" and "false," the slope peaks at the point where uncertainty is greatest. Although there is no smooth derivative function defined for the *signum* function, at zero, the derivative is infinite and positive. In the *tanh* and *signum* functions, the values {-1,0,+1} can represent

Figure 2-3 -- The Signum Function

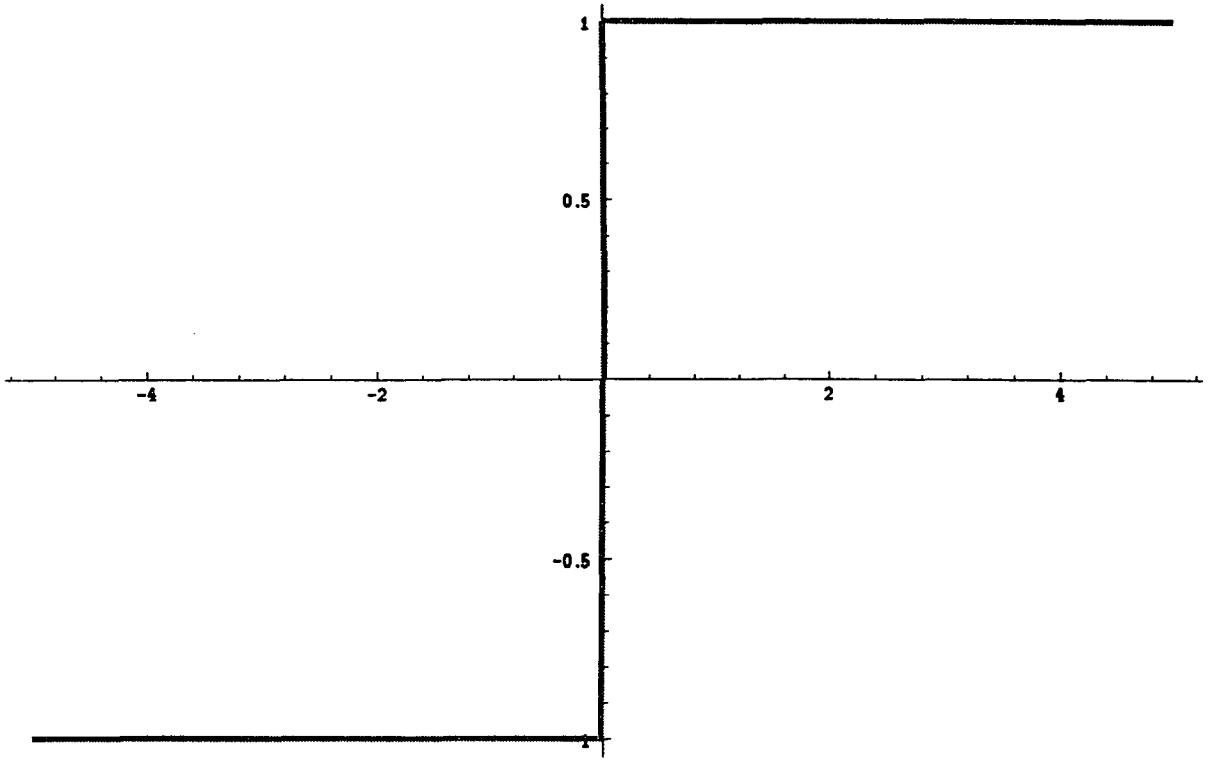
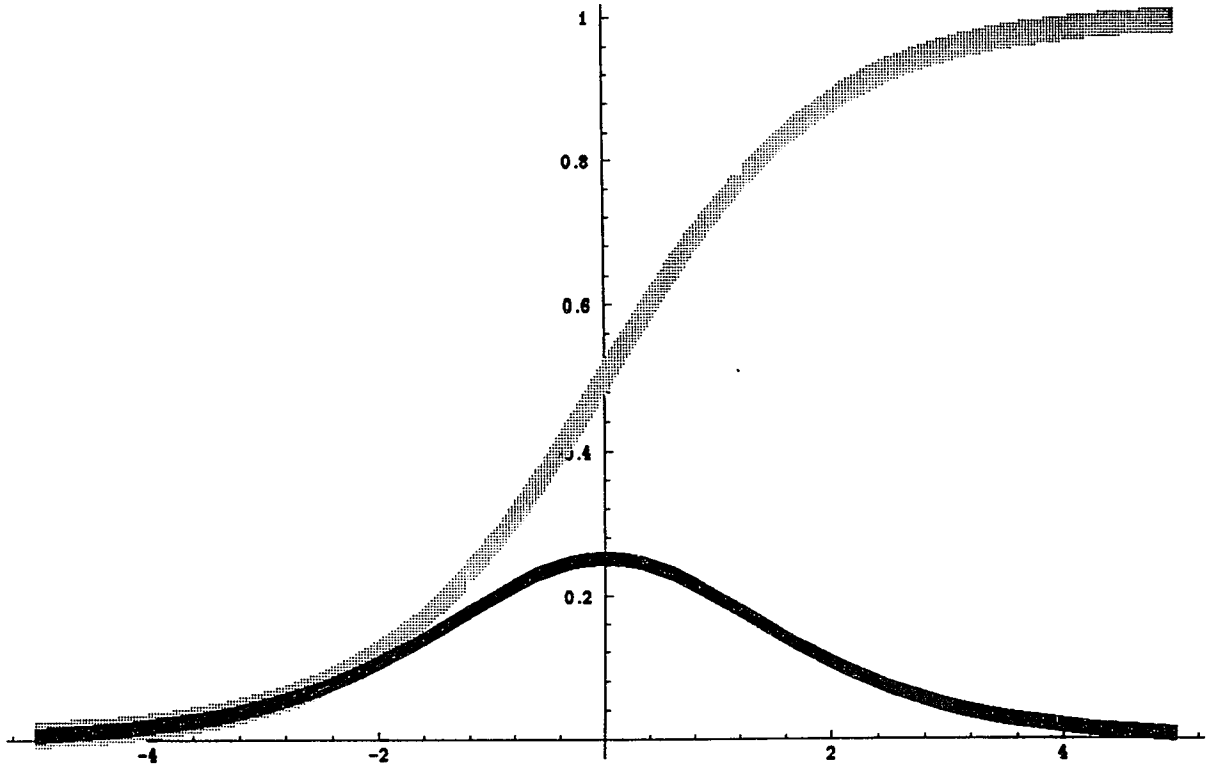
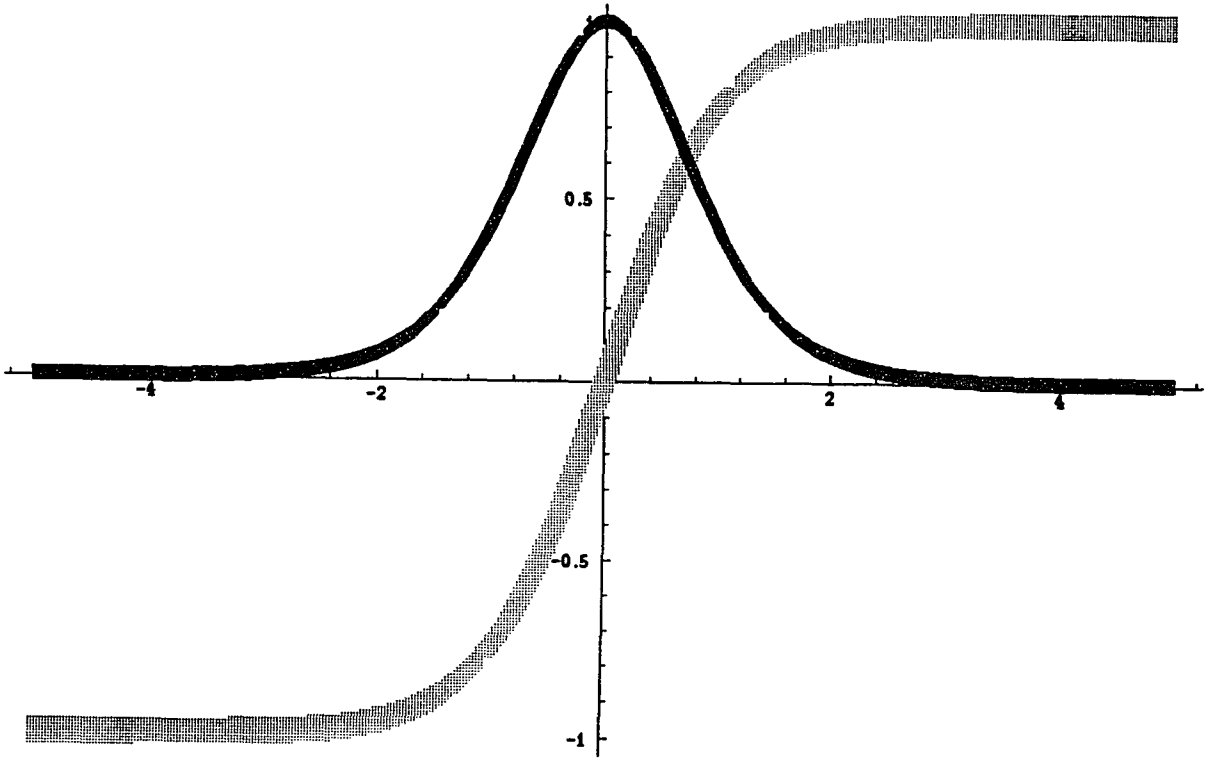


Figure 2-4 -- The Logistic Function



Gray Line = function Black Line = first derivative

Figure 2-5 -- The Hyperbolic Tangent Function



Gray Line = function **Black Line = first derivative**

{False,Don't Know, True} or {Reject,Abstain,Accept} or any other set of values in which zero means "no information" or "no decision." The steepness of the slope at these points helps to "repel" the value towards one of the boundary values, and the relative flatness of the slopes at the boundary values tends to keep, or "attract" the value at that level. Kohonen (1989) discusses the flatness at the extrema as a saturation effect represented by a loss term in the transfer function. The biological basis is that the physical characteristics of any neuron limit its reaction to stimuli, regardless of the strength of the stimuli.

Applied to economics, this shape can represent any relationship in which the rate of change is not easily stimulated at the lower bound and slows down as upper limits are reached. Intuitively, this might apply to the concept of scarce resources. Minimal investments or endowments will have a very slight effect on a complex economy; due to other constraints, it is possible for an economy to become nearly saturated as well.

In statistical measurement, this curve could represent the treatment of outliers - at either extreme, outliers are treated like values that are expected at the limits.

3. *First derivatives of the continuous functions can be expressed in terms of themselves:*

(2.3) $$F'(x) = c[(1 + F(x)) * (1 - F(x))] = c[1 - F(x)^2]$$

In addition, the continuous functions can also be expressed in terms of e^x [9], which is to say the slope at any point of each function depends on its value. c is any constant.

2.9 Practical Implications

Neural networks have been proposed as alternatives to statistical methods in estimation and analysis. Practical application has been relatively scarce, and the association of neural networks with "modeling the brain," have burdened this technology with too many metaphysical associations that are inappropriate. Neural networks also compete with a group of analytical methods that are relatively sophisticated and are well accepted.

Neural network research has been taking place over the last 40 to 50 years, and has been highly constrained until recently, by the limitations of computer systems. It is only in the last 10 years or so that computers with the requisite memory capacity, processing speed, and interfaces have been available to a large number of researchers. As with other statistical techniques, the necessary data transformations still take up a large proportion of time allocated for problem solving. There is no

---

[9] $\tanh(x) = \dfrac{e^x - e^{-x}}{e^x + e^{-x}}$

$\sin(x) = \dfrac{e^{ix} - e^{-ix}}{2i}$

"cookbook" set of agreed-on guidelines for building and implementing neural networks, although a basic architecture can be chosen from among an established set.[10] What is lacking is a broad collection of experience on representation of input and output values.

Another factor is the interdisciplinary nature of neural network technology. The concept of expertise has evolved away from the "Renaissance person," who had a variety of competencies, to very narrow fields of research. Modern experts have tended to be specialists. Neural network research and applications have been inspired by the areas of neurobiology, computer science, physics, mathematics, economics, finance and other fields. In addition, the field is also divided between pure researchers and practitioners, the former emphasizing theory, especially learning algorithms, and the latter confining themselves mostly to small-scale problems using the backpropagation algorithm (see (Hecht-Nielsen 1990)).

2.10 Procedural Rationality and Statistical Modeling

One practice in statistical modeling which is in the spirit of bounded rationality is the linearization of whatever relationship is being modeled. There are a number of techniques to accomplish this, and the end result is that the linear relationships present are given stronger coefficients and the non-linear influences end up combined

---

[10] Hecht-Nielsen (1990), Kohonen (1988), and Kosko (1992) describe a number of different types of neural networks and provide all of the necessary instructions for implementation. A number of software tools for the development of neural networks are also available. See, for example, NeuralWare (1991).

in the residuals. The argument is that estimating non-linear, and especially non-monotonic, functions is too computationally intensive, especially if there are many independent variables. Once linearity is abandoned, the choice has to be made on a different type of functional form, and to intuitively justify the decision.

For many applications, non-linear models are not as usable as linear models. It is easier to computerize and to make policies based on linear relationships. For example, if a financial institution offers a credit line based on income, it would be difficult to enforce a policy that, for example, gave a $1000 credit line up to $20000 of income, reduced the credit line to $500 at an income level between $20000 and $25000, and then increased it to $5000 at a $25000 income level, and so forth. Even if this non-linear relationship were the "true" relationship between income and risk, the cost of maintaining such a complex policy might exceed the benefit of the loss avoidance.

Another consideration in general statistical or other economic modeling is that practical models often contain many independent variables and intermediate relationships, making parameter estimation difficult. Leontief (1947) introduced a theory for the identification of subsidiary or intermediate functions that employed mathematical analysis and did not require any information outside of the data and proposed functional relationship. Leontief's theory provided a methodology for decomposing a function containing a large number of variables into smaller and simpler functions that could be estimated more easily and with greater accuracy. In a neural network architecture, each hidden unit performs a decomposition of the

overall relationship between the input and output data vectors. A hidden unit receives the set of weighted inputs, sums them, and submits the sum to a transfer function to produce an output value that is the result of only its local processing. Thus, each hidden unit produces a subsidiary or intermediate function, although not by dividing the independent variables into subsets. In Leontief's theory, simplification was achieved through determination of functional separability, which involved complicated analysis of the variables and the actual search for the subsidiary functions. These subsidiary functions allowed certain variables to be eliminated from the final estimation of the original target function, because their role was completely subsumed in the subsidiary function. A neural network does not produce the type of intermediate functions that easily identify subsets of variables, but it does eliminate the need to pre-specify these intermediate functions. In addition, in the case of non-linear relationships, the kind of functional separability discussed by Leontief, is rarely easily obtainable.

Neural networks can model non-linear relationships with relatively fewer and simpler decisions made "up front" (i. e., prior to parameter estimation) to build and train the network. This architecture includes the number of layers, the learning algorithm, the transfer function(s), and may include some additional data transformations. Adjustments to the network, which may include eliminating some of the interconnections, changing the constant values of coefficients in the learning algorithm, can be made after observing some training results. The cost tradeoffs in neural networks are mainly computational memory versus computational time and

overall computational resources versus accuracy. Neurons are "cheap" if memory is "cheap," and training time usually decreases as a function of the available memory. From a practical standpoint, running a neural network training procedure overnight may be more rational than running a computationally intensive regression if the neural network can produce more accurate results without pre-specifying the functional form.

5.0 Summary

It has been proposed that the concept of artificial neural networks evolved as part of the same rationality that causes humans to organize themselves in various forms of subgroups and to organize their economic activities in firms and industries. When a group of individuals reaches some critical number, it becomes counterproductive to continue to act as pure individuals, and subgroups are formed. A negative side-effect is that some valuable non-standardized communication between individuals and some innovation may be lost. The neural network analogy is that groups of neurons evolve to be able to quickly generalize and recognize key patterns. As a result, human intelligence is not optimized to have a memory of encyclopedic detail or rapidly perform many successive arithmetic calculations. Therefore, when a visual image is committed to memory, some detail of the specific image may be lost while some generalizations are formed. This is to say a person will not remember every dog that he or she has ever seen, but will have the capability to recognize that a new breed of dog seen for the first time is still a kind

of dog.

The understanding of biological neural networks reinforced the use of artificial neural networks as decision support and analytical systems. The underlying idea is to invent a small number of highly generalized processors or agents, communication channels and signals, and combine these in sufficient number to accommodate the complexity of the task. Thus, it is the structure that becomes complex, not its components. The components provide the means for expanding or contracting the structure as needed.

Simon's introduction of bounded rationality was contemporaneous with the onset of widespread use of computers. It would seem that Simon foresaw the potential for a situation called "information overload"[11], which is the state of having so many pieces of data that it is impossible to go through the process of deciding which pieces are relevant, and that he saw that computerization could either contribute to pathological information saturation or to the management of complexity. One way of procedurally rational management is the use of adaptive systems, such as neural networks, in situations where there are large quantities of data and it is felt that the functional relationships are very complex. The attributes of artificial neural networks, namely adaptation and distributed processing, compensate for the limitations of the human mind and complement its strengths.

---

[11] Radner (1992) attributed "information overload" to the "..relative cheapness of communication compared to Processing (digestion) of information."

CHAPTER 3

A CONSENSUS-LEARNING COMMITTEE NETWORK

3.0 Introduction

Neural Networks can model many aspects of organizational behavior. Decentralized computation and parallelism are especially well-suited to modeling committees, because members of organizations are often characterized as independent decision makers acting on limited, or local information. The layered architecture of neural networks is also applicable to hierarchical organizations, in which information is filtered as it passes through each organizational layer. In addition, the adaptive properties of neural networks can simulate the process of information gathering or learning by individual agents, or the dynamics of a changing environment.

At the highest level, the economics of decision making is often no more complicated than: good choices yield profits, poor choices result in losses, and some cost is involved in information gathering and processing. In any organization, the economic climate may influence how decisions are made. When the climate is good,

and good choices greatly outnumber the bad, the organization may operate as a polyarchy, requiring the approval of only one person, who is not challenged by peers or management, to proceed with a choice or action. Under adverse economic conditions, organizations may rely on committees or hierarchies to make decisions. The intuition of "two (or more) heads being better than one," prevails. Another approach is to centralize and assign the decision making to a single designated expert, or senior and experienced member of the organization. The benefits of centralization in this manner are that the experience and expertise compensates for a lack of other opinions and the focusing of accountability on one person is an incentive for that person to exercise extra care and effort in decision making.

Economic climates may change quickly without an obvious signal. For example, the decision to extend credit to a firm may be based *a priori* on an estimated likelihood of the individual firm going bankrupt. Certain characteristics of a firm, such as its size, industry classification, or payment performance history, may be significant predictors of severe financial stress when certain combinations of indicators are present. Moreover, when the overall bankruptcy rate increases, the general environment may become even more risky. To set the context, from 1982 to 1983, there was a 25% increase in the rate of business failures and from 1990 to 1991 the estimated increase in failure rate is 43%[1] These significant changes in the environment took place within a year but there was no one point during the year

---

[1] Source: The Business Failure Record, published by The Dun & Bradstreet Corporation, New York, 1992

when the jump in failures could be observed. Therefore, a dynamic re-evaluation of the environment is desirable.

In this essay, we briefly consider the conclusions of R. K. Sah and J. E. Stiglitz regarding the optimal, or profit maximizing consensus size. Consensus size is defined as the minimum number of members of a committee needed to agree on a decision or action. A neural network architecture is introduced which implements a portion of the Sah and Stiglitz model and is related in design to the committee machine of Nilsson (1990). The results obtained with the neural network do not strictly conform to the all conclusions of Sah and Stiglitz but are compatible in spirit, especially with regard to the types of errors made by committees for different levels of consensus.

3.1 The Sah and Stiglitz Model

Sah and Stiglitz (1985, 1988) have proposed a model and methodology for determining the optimal or profit maximizing size of a consensus for any given size of committee. In their model, the committee's activity is described as accepting or rejecting projects, which may be good or poor quality (hereafter called "goods" and "bads"). This type of all-or-nothing decision can easily be generalized to most economic decisions, especially those made by senior management, such as an investment, a hiring decision, introduction of a new product, and so forth. The optimal consensus is a function of portfolio quality (the odds ratio, or proportion of "bads"/proportion of "goods"), the amount of profit or loss resulting from each

decision, the probability of Type I errors (rejecting a "good"), and the probability of Type II errors (accepting a "bad"). Sah and Stiglitz (1985) gave a formal proof that profits are single-peaked in consensus size k and that an optimal k exists for a committee of any size.

The key formulae (Sah and Stiglitz 1988) to compute the optimal consensus k are:

(3.1) $k < n/2 + 1$ if $\beta < 1$ and $p_2 <= 1 - p_1$

(3.2) $k > n/2$ *if* $\beta > 1$ *and* $p_2 >= 1 - p_1$

(3.3) $k = n/2 + 1$ if $\beta = 1$ and $p_2 = 1 - p_1$

where k is the consensus size, n is the total number of committee members, β the odds ratio or proportion of "bads"/proportion of "goods," p_2 is the probability of a Type II error and $1 - p_1$ is the probability of a Type I error. If n is an even number, then the majority rule is greater than $n/2$ but a k strictly less than the majority rule is less than $n/2 + 1$. For an odd n, the majority rule is $(n+1)/2$. Formulae (3.1-3.3) hold for both even and odd n because $n/2 < (n+1)/2 < n/2 + 1$. Thus, formula (3.1) says that the optimal consensus size is less than a majority if the portfolio quality is relatively good and Type II errors are less likely than Type I errors. The

interpretation is that if the members of the committee are adept at avoiding "bads" and the economic environment is friendly, then a small consensus is optimal. Formula (3.2) is the opposite situation, in which the portfolio quality is relatively poor, and the committee is likely to select "bads". Here, Sah and Stiglitz explain that a larger consensus is needed to provide additional "scrutiny" to all decisions. Formula (3.3), in which the portfolio quality is neutral and the committee is equally likely to make Type I or Type II errors, is the case when the simple majority is the optimal consensus size.

Sah and Stiglitz make an important assumption -- that an increase in the consensus that results in increased vigilance will reduce the number of project approvals. When Type II errors are prevalent, this increases profits, by reducing "bads." However, when Type I errors are prevalent, more scrutiny increases the rejection of "goods," and profits are sacrificed to lost opportunity. The underlying rationale is that the committee members are inherently risk adverse; a greater number of rejections with an increased consensus could result from an augmented capacity to examine a project and discover negative aspects or from some collective "inertia." It is easier to reject a project, because, in the short term, rejecting a project usually requires no investment or change in behavior.

The distinction between Type I and Type II errors is an important one, even if the economic outcomes of rejecting a "good" or accepting a "bad" are identical. Organizations tend to punish Type II errors to a greater extent than Type I errors.

This is because the outcome of a Type II error is felt earlier and more demonstrably than a Type I error. Consider an investment decision, for example, adding an equity security to a portfolio. If it is a bad investment and the consensus is to approve, the committee and the organization observe the investment lose value. If it is a good investment, and the committee does not approve, it is less likely that the investment's rising value will be observed, because the organization is occupied watching the investments it approved. There is also the "sour grapes" posture; humans tend to compensate for their Type I errors by finding a reason that the decision would have been a bad one anyway.

3.2 The Committee Machine and the Counterprogation Network

The committee machine described by Niisson (1990) uses a simple majority rule for determining the output value (also discussed in Chapter 2). The purpose of the majority rule is to render a decision or answer when the decision space or solution space is not linearly separable. This means that unanimity is impossible to achieve. Nilsson's committee machine was specified always to have an odd number of units in the committee layer, to prevent deadlocks or the necessity of a tie-breaker unit outside of that layer.

The counterpropagation network of Hecht-Nielsen (1990) uses a "winner take all" strategy in the hidden layer, so named because, for each training iteration, one processing unit in the hidden layer is the "winner" and its "prize" is to be allowed to

learn through the adjustment of its input weights and send a non-zero output signal to the next layer. All the other hidden units, which are "losers," have their weights remain the same. The rule for winning the competition is to have input weights that most closely match the input vectors. The winning unit is then allowed to signal the next layer, and its weights are adjusted to further reduce the distance between the their values and those of the input vectors. As this is analogous to a committee in which only one member's approval is required for the committee to approve a project, it is closely related to the polyarchy organizational model described by Sah and Stiglitz.

Both the committee machine and the counterpropagation network use Kohonen learning to adjust the weight vectors, which causes adaptation in the direction of minimizing the Euclidean distance between the input vectors and the weight vectors.

There is a close relationship between Kohonen learning and *k-means* analysis as described by MacQueen (1967). Hecht-Nielsen (1990) has discussed this in some detail. In both k-means analysis and Kohonen learning, the vector representing the means for a class is dynamically adjusted with each new data vector. MacQueen describes k-means as appearing "..to give partitions which are reasonably efficient in the sense of within-class variance,"[2] and later states that "In general, the k-means procedure will not converge to an optimal partition."[3] This approach is consistent

---

[2] p. 281.

[3] p. 282.

with Simon's procedural rationality discussed in Chapter 2, in that the costs of implementing and maintaining a classification scheme based on k-means analysis is probably less than a model based on regression analysis. This is because k-means is simpler and less memory-consuming from a computational requirements standpoint.

3.3 The Consensus-Learning Committee Network

The neural network architecture proposed herein learns two important characteristics of its decision making environment:

(1) It learns the optimal consensus for the size of its committee by observing the odds ratio of bad choices to good choices, and dynamically recalculating the consensus size.

(2) Its individual agents, or neural processing elements in the hidden layer, adapt to recognize good choices and avoid bad ones. In other words, there is no outright rejection of a project; in essence, the committee either selects or "ignores" a project. This was felt to be more consistent with the actual behavior of people in organization than an explicit vote to reject.

The network is a four-layer network. The layers are called *input*, *committee*, *vote counter*, and *consensus maker*. The committee layer corresponds to the hidden layer usually referenced in the literature, and the consensus maker also functions as the

conventional the output layer. The committee layer contains an odd number of neural units, and the vote counter and consensus makers each contain one neural unit. See figure 3-1.

3.3.1 Processing Rules

Inputs are proportionately scaled and normalized so that the input vectors are equinorm, i. e., for each input vector $v = \{x_1, x_2, ...x_n\}$, $|| v || = L$, where L is the maximum norm of all the inputs.

(1) The input vectors are sent to the input layer one at a time.

(2) The committee layer units each receive the complete input vector over their weighted connections. Each hidden unit performs the following:

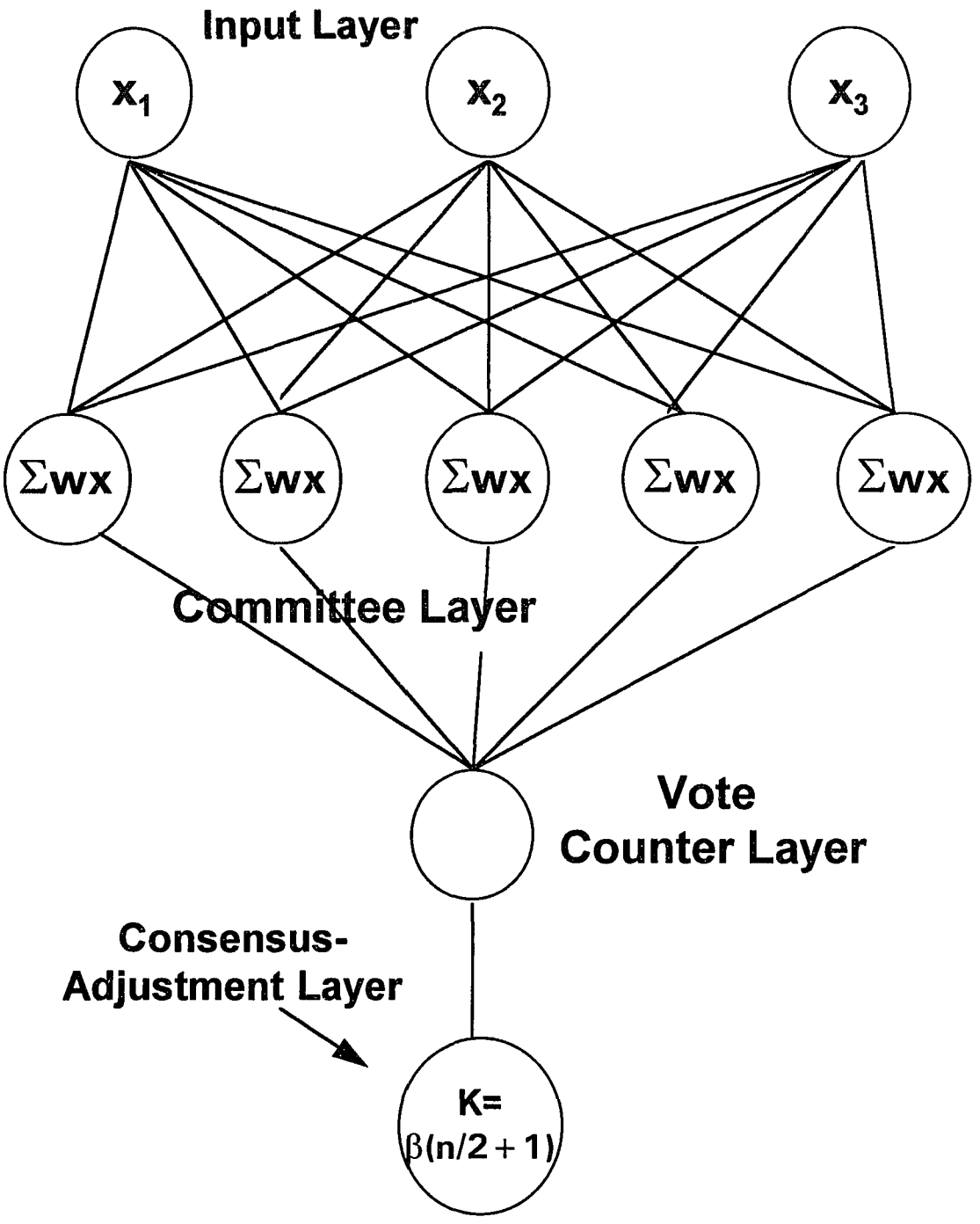
(i) Calculates the weighted sum of the inputs.

(ii) Calculates the Euclidean distance between
the weighted sum of the inputs and the squared inputs.

(iii) Tests if the Euclidean distance is less than 1. If it is,
the hidden unit outputs a 1; otherwise it outputs a 0.
The values are output to the vote counter unit over
connections with a constant weight of 1.

(3) The vote counter layer counts the votes and transmits the count,
or consensus to the next layer over a connection with a constant weight of 1.

Figure 3-1 --Consensus-Learning Committee Network



(4) The consesus maker compares the consensus to the minimum consesnsus for the size of committee. If the consensus meets or exceeds the minimum, then the decision is a "yes" represented by a value of one. Otherwise the decision is a "no," or zero. In both training and recall, the consensus maker dynamically updates the odds ratio with each iteration, by incrementing the total count and either the *good* count or the *bad* count. The actual minimum consensus is calculated according to the rule from Sah and Stiglitz:

$$\text{Minimum Consensus} = R * K$$

where R is the odds ratio, K is the majority rule $n/2$, or K can be set to any integer from 1 to n as a test of various levels of consensus. n is the committee size. If the odds ratio is greater (less) than 1, the bads (goods) outnumber the goods (bads) and the consensus requirement will be increased (reduced).

A key element of this process is that the minimum consensus rule stated above essentially allows the neural network to switch among the formulae a-c stated in section 3.1. If the odds ratio is less than one, signalling that the decision environment is favorable, $R * K < K$. The opposite situation, $R > 1$, signalling an adverse decision environment, results in $R * K > K$. For $R = 1$, $R * K = K$.

During learning, the following takes place:

(1) If the consensus meets or exceeds the minimum and the output value is a 1, representing a "good," or if the consensus is insufficient and the output is zero, representing a "bad," then no learning takes place, because the network has made the correct decision.

(2) If the consensus is insufficient and the output value is a 1, this is comparable to rejecting a good decision, or a Type I error. The number of additional units *j* required to make the consensus is calculated. Then the *j* units with the smallest Euclidean distance of all the units that output a zero have their weights adjusted as follows:

(3.4) $W_{new} = W_{old} + L1 * (X - W_{old})$

W_{new} is the updated weight vector, W_{old} is the previous weight vector, X is the input vector, and $L1$ is a constant learning coefficient $0 < L1 < 1$. This learning rule is the Kohonen learning law, and, with a sufficient number of iterations, it effectively adjusts the weight vectors to become very close to the input vectors, as measured by the Euclidean distance.

(3) If the consensus meets or exceeds the minimum and the output value is zero, this is effectively accepting a bad decision, or a Type II error. All the network weights between the inputs and the committee layers are adjusted as follows:

(3.5) $W_{new} = W_{old} - L2 * (X - W_{old})$

where W_{old} , W_{new}, and X are as described above and $L2$ is a learning coefficient $0 < L2 < 1$.[4] This effectively increases the Euclidean distance between the weights and the inputs.

Note: The dynamic updating of the odds ratio during recall is not typical of most neural network paradigms, because it is a form of learning and most network implementations do no learning at recall (production use) time. Also, in this model the Euclidean distance threshold has been set to 1.0, which is a very large value for a threshold. This amount of "leeway" is considered suitable in a model of this type, because the network is modeling decisions which made be made with some ambivalence and because committees may have to have a "production quota." This wide threshold guarantees that a consensus will be reached enough times to measure results.

3.3.2 The Data

---

[4] L1 may or may not be set equal to L2. The value of the learning coefficient is determined by trial and error. Generally, when learning begins, it is set to a relatively high value, say 0.8. After a certain number of learning iterations, it is reset to a lower value. The idea behind a gradually decreasing learning coefficient is that, when learning begins, the change in the weight values should be larger, and, as the weight vectors and the input vectors converge, the learning coefficient should become smaller as to promote convergence.

The data set for this work is from Kendall's (1980) discussion of an example of discrimination analysis. The example is taken from a 1947 paper by Cedric A. B. Smith, "Some examples of discrimination."[5] Results of tests given to 25 normal individuals and 25 psychotics were used as independent variables to discriminate between the psychotic and normal. A direct economic application of this application would be predicting successful employees, assuming that psychotic employees have a higher likelihood of causing losses to the firm. The set of independent variables could be hypothetical data for any two classes - good vs. bad investments, good vs.

[5] Annals of Eugenics. London. Vol. 13. p. 272.

Table 3-1 -- Original Data for CLCN[6]

X1	X2	Good=1; Bad=0	X1	X2	Good=1; Bad=0
22	6	1	24	38	0
20	14	1	19	36	0
23	9	1	11	43	0
23	1	1	6	60	0
17	8	1	9	32	0
24	9	1	10	17	0
23	13	1	3	17	0
18	18	1	15	56	0
22	16	1	14	43	0
19	18	1	20	8	0
20	17	1	8	46	0
20	31	1	20	62	0
21	9	1	14	36	0
13	13	1	3	12	0
20	14	1	10	51	0
19	15	1	22	22	0
20	11	1	11	30	0
18	17	1	6	30	0
20	7	1	20	61	0
23	6	1	20	43	0
23	23	1	15	48	0
25	9	1	5	53	0
23	5	1	10	43	0
21	12	1	13	19	0
23	7	1	12	4	0

[6] Source: Kendall (1980) p. 153.

bad credit risk, and so forth. Table 3-1 contains the original data.

In order to prepare the data for the neural network, a normalization was performed, so that the data vectors were equinorm, based on the maximum norm after proportional scaling. The norm $||\ X\ ||$ of the data vector $X = \{x1, x2, ... xn\}$ is defined as:

(3.6) $$\sqrt{\sum_{j=1}^{n} x_j^2}$$

First, the proportional scaling constrained all data values to the interval [-1,+1]. Then, one more element is added to each vector to complete the normalization. The value of the third data element is:

(3.7) $x_3 = (\text{Maximum Norm} - x_1^2 + x_2^2)^{1/2}$

Table 3-2 contains the normalized data.

Table 3-2 -- Normalized Data for CLCN

X1	X2	X3	Good=1;Bad=0
0.727273	-0.83607	0.664447	1
0.545455	-0.57377	1.021121	1
0.818182	-0.7377	0.675123	1
0.818182	-1	0	1
0.272727	-0.77049	1.000692	1
0.909091	-0.7377	0.546596	1
0.818182	-0.60656	0.79504	1
0.363636	-0.44262	1.158134	1
0.727273	-0.5082	0.939272	1
0.454545	-0.44262	1.125564	1
0.545455	-0.47541	1.070461	1
0.545455	-0.01639	1.171167	1
0.636364	-0.7377	0.848678	1
-0.09091	-0.60656	1.137209	1
0.545455	-0.57377	1.021121	1
0.454545	-0.54098	1.081733	1
0.545455	-0.67213	0.95924	1
0.363636	-0.47541	1.145066	1
0.545455	-0.80328	0.852434	1
0.818182	-0.83607	0.54863	1
0.818182	-0.27869	0.960382	1
1	-0.7377	0.353854	1
0.818182	-0.86885	0.495071	1
0.636364	-0.63934	0.925041	1
0.818182	-0.80328	0.595603	1
0.909091	0.213115	0.893061	0
0.454545	0.147541	1.200434	0
-0.27273	0.377049	1.205353	0
-0.72727	0.934426	0.517053	0
-0.45455	0.016393	1.209356	0
-0.36364	-0.47541	1.145066	0
-1	-0.47541	0.665888	0
0.090909	0.803279	1.007919	0
0	0.377049	1.235822	0
0.545455	-0.77049	0.882181	0
-0.54545	0.47541	1.070461	0
0.545455	1	0.609837	0
0	0.147541	1.283609	0
-1	-0.63934	0.510549	0
-0.36364	0.639344	1.062275	0
0.727273	-0.31148	1.021508	0
-0.27273	-0.04918	1.261992	0
-0.72727	-0.04918	1.066807	0

0.545455	0.967213	0.660605	0
0.545455	0.377049	1.108934	0
0.090909	0.540984	1.169826	0
-0.81818	0.704918	0.709289	0
-0.36364	0.377049	1.181111	0
-0.09091	-0.40984	1.221962	0
-0.18182	-0.90164	0.907419	0

Figures 3-2 and 3-3 show the normalized data and original data as scatter graphs. In both graphs, the *goods* are more concentrated in location than the *bads*.

3.4 Experimental Results

To test the Consensus-Learning Committee Network (CLCN), the simple data set described in 3.4.2 was used. The objective of the committee represented by the CLCN is to choose the right vectors, i. e., the normal individuals, and ignore the wrong vectors, i. e., the psychotic individuals. By failing to reach a consensus, the committee has effectively rejected a decision. The network was implemented in Mathematica,[7] and the computer program is listed in Appendix 1.

The initial connection weights were set by a pseudorandom function. Given an input vector size of three elements and a committee size of five, there are 15 adaptable weights. For this data set, a committee size of five was chosen to allow

[7] Mathematica Version 2.0 for MS-DOS 386/7 (Champaign, IL: Wolfram Research, Inc.).

Figure 3-2 -- Normalized Decision Data (XY Graph)

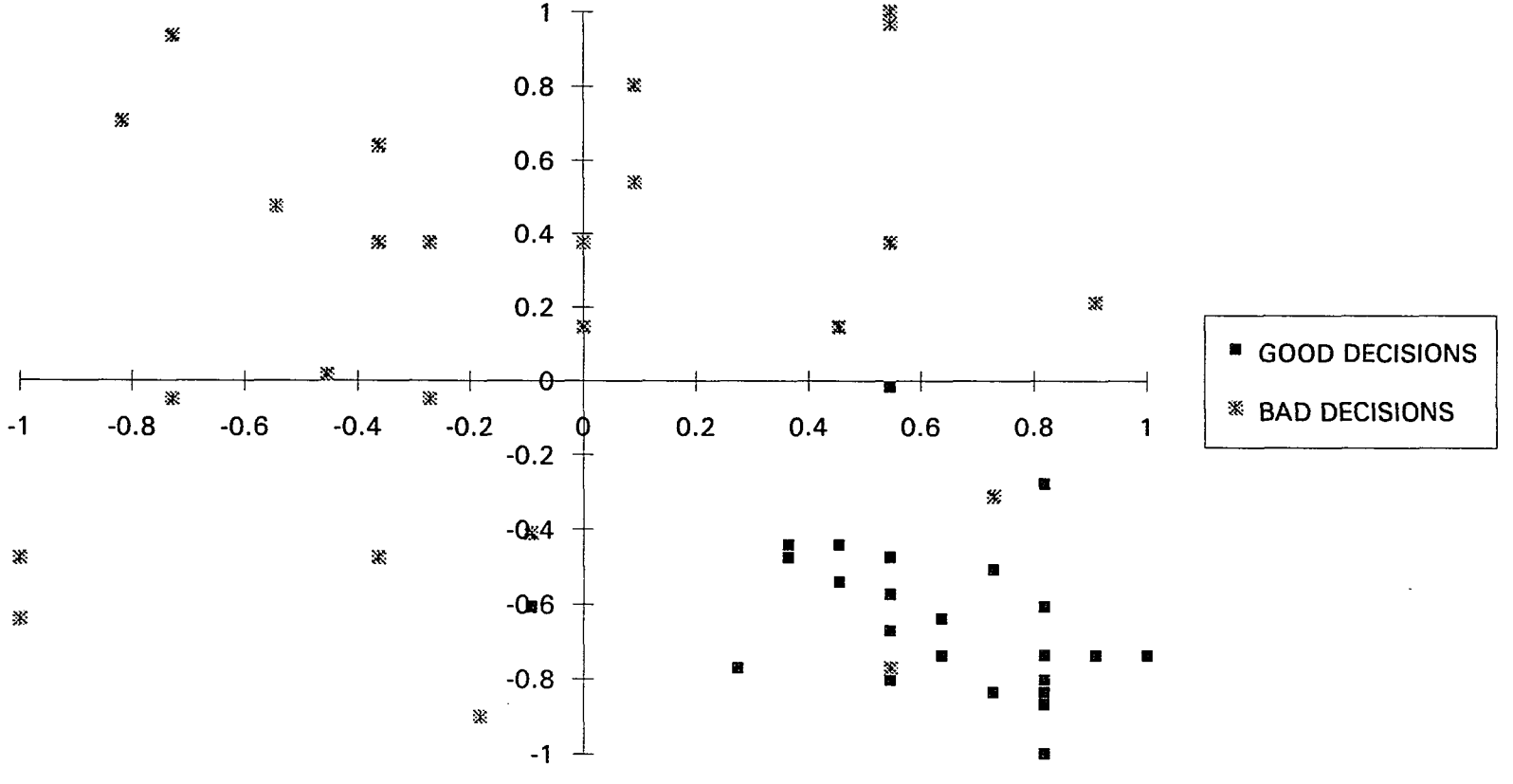
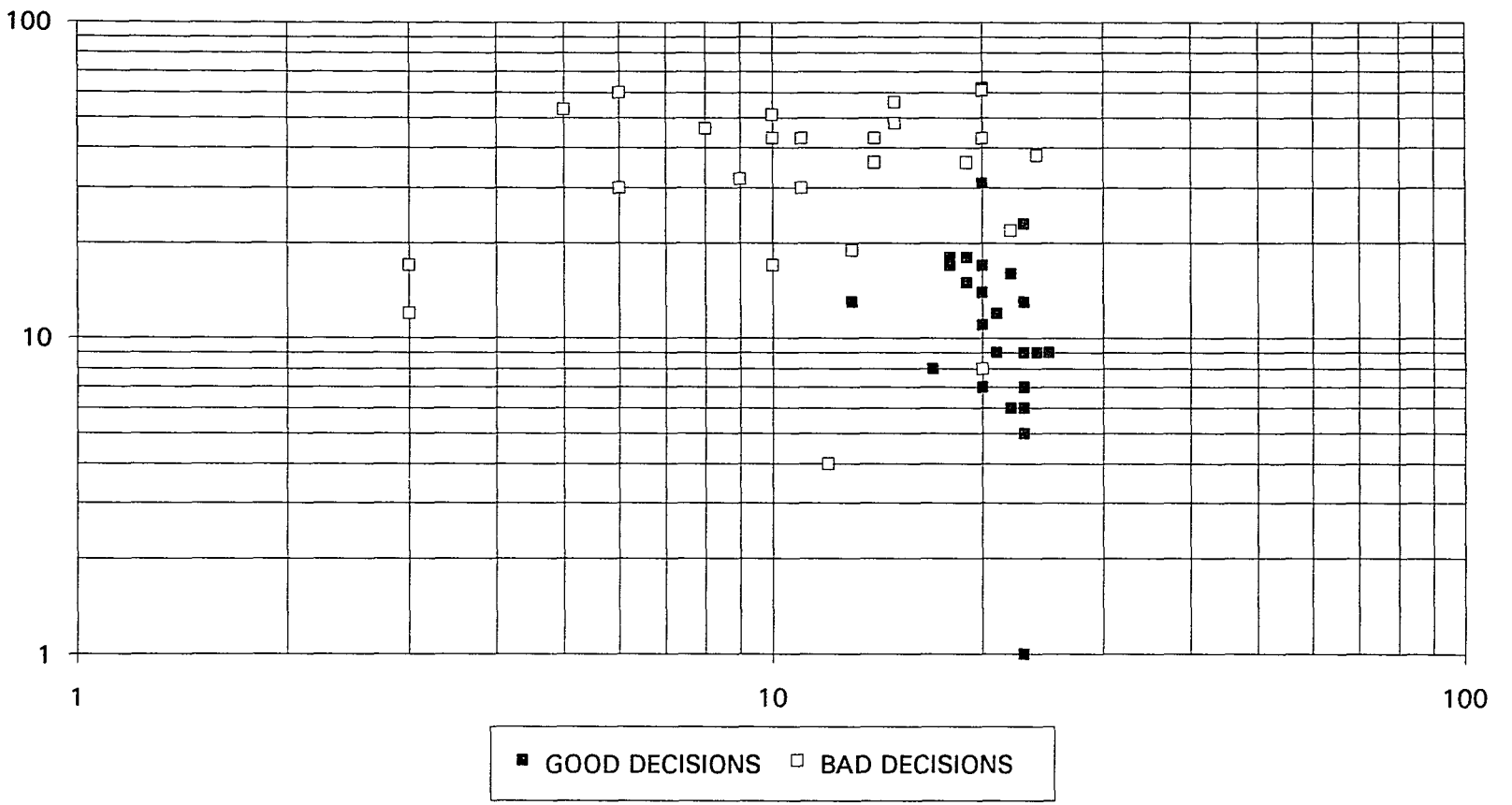


Figure 3-3 -- Decision Data (XY Log-Log Graph)



enough degrees of freedom (50 - 15 = 35), and at least 3 examples per adaptable weight. The order of presentation of the input vectors to the network was controlled by a randomization scheme.

Five different rules for the minimum consensus were tried, ranging from the most strict (unanimity) to the most lenient (approval by one):

(1) Odds Ratio * (5)

(2) Odds Ratio * (n/2 + 1)

(3) Odds Ratio * (n/2)

(4) Odds Ratio * (2)

(5) Odds Ratio * (1)

As there are an equal number of good and bad decisions, the Odds Ratio was (25/50) /(1 - 25/50) = 0.5/0.5 = 1. Because the odds ratio was calculated with every iteration and the input vector presentation was randomized, the odds ratio evolved over the training iterations, but, in each test, stablized to a value close to 1. In addition, as the rules above did not produce integer results, the minimum consesus was rounded up to the next integer. Thus, rules 2 and 3 produced minimum consensuses of 4 and 3, respectively.

Each rule was tested as follows: ten training sessions for each rule were run with a maximum of 1000 iterations (20 passes of 50 data vectors). It was decided to limit the training iterations and measure each result after the same number of training iterations rather than to train each network until it made no errors and compare

training times as a measure of performance. As the theme of procedural rationality has been discussed in depth in Chapter 2, performance under time and resource constraints is considered more apropos. A pass of all the data vectors is often called an epoch. After each 50 iterations, summary statistics were written to a file. Because the training vectors were input to the network in a random fashion, each set of 50 training iterations may have included duplicate vectors and consequently omitted some vectors. If 50 iterations took place without an error in classification before the 1000 iteration limit occured, then the system halted.

The summary statistics recorded were as follows:

Number of Iterations

Number of Learning Occurrences

Number of Type I errors

Number of Type II errors

The current Odds Ratio

The cumulative number of correct Goods

The cumulative number of correct Bads

The cumulative percentage of correct Goods

The cumulative percentage of correct Bads.

The most important statistics for this work are the Number of Type I errors, the

Number of Type II errors, and the overall percentage of correct decisions.

The results of the neural network model deviated from what was expected. As the odds ratio approached one, a majority rule was expected to outperform the other consensus options. However, the model with Rule 3 (the consensus stabilized at 2) performed the best, as determined by the average percentage of correct decisions, 90.02%. In terms of Type I and Type II errors, the results were consistent with the Sah and Stiglitz model. The larger the consensus, the more Type I errors and the fewer Type II errors, as a result of more overall rejections. Table 3-3 contains summary statistics for all training sessions run:

Table 3-3 --Selected Summary Statistics for All Network Training Sessions

Consensus Size is Odds Ratio * 1

Run Number	Type1 Err. Avg	Type1 Err. Var	Type2 Err. Avg	Type2 Err. Var	Correct % Avg	Correct % Var
1	2.80%	0.05%	8.90%	0.19%	88.30%	0.31%
2	2.00%	0.05%	8.00%	0.23%	90.00%	0.33%
3	3.30%	0.07%	6.50%	0.10%	90.20%	0.18%
4	1.60%	0.02%	7.60%	0.50%	90.40%	0.63%
5	2.00%	0.02%	7.30%	0.15%	90.70%	0.18%
6	3.80%	0.02%	6.40%	0.12%	89.30%	0.19%
7	3.00%	0.04%	7.00%	0.13%	90.00%	0.21%
8	2.60%	0.04%	8.20%	0.09%	89.20%	0.17%
9	3.30%	0.03%	9.30%	0.23%	87.10%	0.35%
10	2.60%	0.02%	7.50%	0.16%	89.90%	0.18%
For Group:						
Average	2.7000%	0.0346%	7.6700%	0.1910%	89.5100%	0.2730%
Varianed	0.0042%	0.0000%	0.0082%	0.0001%	0.0107%	0.0002%

Consensus Size is Odds Ratio * 2

Run Number	Type1 Err. Avg	Type1 Err. Var	Type2 Err. Avg	Type2 Err. Var	Correct % Avg	Correct % Var
1	4.00%	0.06%	5.50%	0.10%	90.50%	0.24%
2	5.50%	0.06%	6.90%	0.10%	87.50%	0.22%
3	4.90%	0.07%	6.00%	0.10%	89.10%	0.22%
4	4.70%	0.05%	4.90%	0.09%	90.40%	0.18%
5	6.00%	0.21%	5.38%	0.12%	88.63%	0.53%
6	4.30%	0.05%	5.80%	0.09%	89.90%	0.16%
7	5.10%	0.09%	6.00%	0.12%	88.80%	0.27%
8	4.90%	0.07%	6.50%	0.05%	88.60%	0.16%
9	2.50%	0.03%	5.25%	0.32%	91.75%	0.39%
10	3.00%	0.09%	1.00%	0.01%	95.00%	0.25%
For Group:						
Average	4.4900%	0.0771%	5.3225%	0.1085%	90.0175%	0.2633%
Variance	0.0105%	0.0000%	0.0239%	0.0001%	0.0408%	0.0001%

Consensus Size is Odds Ratio * 5/2 (Majority rule)

Run Number	Type1 Err. Avg	Type1 Err. Var	Type2 Err. Avg	Type2 Err. Var	Correct % Avg	Correct % Var
1	3.60%	0.04%	3.90%	0.07%	92.50%	0.12%
2	5.40%	0.10%	5.00%	0.04%	89.60%	0.22%
3	5.10%	0.09%	4.90%	0.09%	90.00%	0.26%
4	5.50%	0.12%	4.90%	0.06%	89.40%	0.28%
5	6.20%	0.13%	5.90%	0.09%	87.70%	0.31%
6	3.64%	0.08%	4.00%	0.12%	92.36%	0.30%
7	4.20%	0.09%	4.40%	0.10%	91.40%	0.30%
8	7.20%	0.15%	6.70%	0.07%	85.80%	0.35%
9	5.30%	0.05%	4.90%	0.13%	89.60%	0.28%
10	5.50%	0.11%	4.00%	0.10%	89.50%	0.39%
For Group:						
Average	5.1636%	0.0948%	4.8600%	0.0867%	89.7864%	0.2809%
Variance	0.0112%	0.0000%	0.0071%	0.0000%	0.0372%	0.0000%

Consensus Size is Odds Ratio * 5/2+1 (4 out of 5 committee members)

Run Number	Type1 Err. Avg	Type1 Err. Var	Type2 Err. Avg	Type2 Err. Var	Correct % Avg	Correct % Var
1	4.40%	0.14%	3.20%	0.07%	92.40%	0.29%
2	6.20%	0.14%	5.40%	0.12%	88.40%	0.37%
3	5.70%	0.17%	4.60%	0.05%	89.60%	0.19%
4	5.60%	0.09%	5.00%	0.07%	89.30%	0.23%
5	8.62%	0.31%	5.85%	0.08%	85.23%	0.41%
6	7.20%	0.13%	5.60%	0.09%	87.10%	0.29%
7	6.40%	0.31%	4.40%	0.07%	89.20%	0.38%
8	6.10%	0.10%	4.50%	0.08%	89.40%	0.25%
9	6.90%	0.28%	4.70%	0.07%	88.10%	0.50%
10	6.10%	0.12%	5.90%	0.06%	88.00%	0.26%
For Group:						
Average	6.3215%	0.1801%	4.9146%	0.0755%	88.6731%	0.3148%
Variance	0.0111%	0.0001%	0.0061%	0.0000%	0.0311%	0.0001%

Consensus Size is Odds Ratio * 5 (Unanimity)

Run Number	Type1 Err. Avg	Type1 Err. Var	Type2 Err. Avg	Type2 Err. Var	Correct % Avg	Correct % Var
1	5.60%	0.17%	4.40%	0.08%	90.00%	0.31%
2	33.30%	3.52%	1.40%	0.06%	65.00%	2.83%
3	21.53%	3.85%	2.59%	0.07%	75.88%	3.21%
4	5.60%	0.15%	3.80%	0.07%	90.60%	0.32%
5	7.07%	0.32%	5.07%	0.09%	87.73%	0.59%
6	8.40%	0.27%	5.20%	0.10%	86.40%	0.58%
7	6.20%	0.14%	4.00%	0.11%	88.80%	0.47%
8	7.07%	0.17%	4.27%	0.10%	88.53%	0.39%
9	7.00%	0.11%	4.90%	0.07%	88.10%	0.30%
10	5.70%	0.16%	3.90%	0.07%	90.40%	0.31%
For Group:						
Average	10.7463%	0.8856%	3.9522%	0.0819%	85.1449%	0.9310%
Variance	0.7707%	0.0197%	0.0124%	0.0000%	0.6144%	0.0111%

In order to see if the consensus size matters, it is necessary to see if differences in percentages of Type I or Type II errors or the total percentage correct is statistically significant, The appropriate F-test ratio, given the null hypothesis that the consensus size has no influence on the Type I or Type II errors, or the total percentage correct, is:

$$F = \frac{\text{The variance of the averages (between group variance)}}{\text{The average of the variances (within group variance)}}$$

In all cases the number of degrees of freedom in the numerator is 5 - 1 = 4 and the degrees of freedom in the denominator is the number of test statistics for each group minus 1, or (10 - 1 = 9) times the number of groups (4), or 45. Table 3-4 contains the F-test calculation and result for Type I errors, Type II errors, and the Total Percentage correct. As each F-test statistic exceeds the critical value of 3.78, the null hypothesis is rejected.

Table 3-4 -- F-Test Calculations for Between Group and Within Group Variations

F-Test Data	Type I Errors	Type II Errors	Total % Correct
Numerator	7.294664	1.555336	3.236536
Denominator	0.25444	0.10872	0.4126
Ratio	28.66949	14.30589	7.844246

d. f. of numerator. 4
d. f. of denominator 45

Critical Value at 1% significance level = 3.78

The neural network also adds the dynamics of improvement in managerial quality because learning takes place over the iterations. The average percent improvement in learning was calculated by taking, for each run: the percentage correct after one epoch (50 iterations), the percentage correct after 20 epochs (1000 iterations) or at the epoch after learning was terminated, dividing the latter by the former,subtracting one, and then taking the average over all these individual performance improvement statistics. All five consensus levels showed an improvement in performance over time through learning, as shown in Table 3-5. The best improvement was in the unanimous, or consensus size=n, network. The network learning rule dictates that the larger the consensus, the more learning takes place in the hidden layer. This could justify larger consensus values in committees; as more committee members get involved in reviewing projects, more of them are required

to develop expertise.

Table 3-5 -- Percentage of Improvement in Performance

Consensus Size	Average Percent Improvement
1	7.1%
2	8.1%
3	10.7%
4	7.8%
5	16.6%

The intuition from these results is that committees are more useful in avoiding Type II errors than in avoiding Type I errors, especially when the approval of a majority or more committee members is required. This is consistent with the Sah and Stiglitz model. It is easier for organizations to turn down bad decisions than to approve good decisions, all other things being equal. This is because turning down decisions usually maintains the status quo and one is less likely to "get caught" turning down a good decision than approving a bad one.

3.5 Flexibility of the Model

The model can be converted from a committee-type network to a hierarchy-type network by using variable weights to connect the committee layer to the vote counter layer. These weights could represent the relative degree of confidence that the vote counter has in the committee members. Therefore, the vote counter's "opinion"

would enter into the final decision. The weights would be adapted by increasing the weights for committee members that made correct decisions and decreasing the weights of members that made incorrect decisions. This eventually might result in some hidden units having no influence on a decision because their connection weight to the Vote Counter has adapted to a zero or near zero value.

3.6 Applications to Voting Behavior

The Consensus-Learning Committee Network can be applied to determining the level of consensus that will perform the best, in a Simonian "satisficing" sense for a given type of decision. Typically, informal or ad hoc committees choose a majority rule, because, intuitively, it seems "democratic." Elster (1989) points out "that majority voting is flawed because it neglects the intensity of preferences."[8] Although neural network design practice is to normalize data vectors to prevent the "intensity" of extreme values from distorting the information content of the network, useful results might be obtained by relaxing this requirement for a voting model.

Tullock (1967) discusses the conditions under which voting rules will be established, namely, "...in situations where the number of choosers greatly exceeds the number of issue dimensions."[9] Thus, we must consider that a committee layer in a neural network should have more units than the output layer to function in a rational

---

[8] p. 156.

[9] p. 27.

manner. For the intuition of this relationship between issues and voters, it appears to make sense to vote for a leader among two or more candidates; however, four people voting on the menu for a five course dinner strikes one as futile.

In a situation where several issues exist among a group of voters, a self-organizing map (Kohonen 1992) can be applied to reducing the dimensionality of the issue space to a two-dimensional topology. This topology allows the perception of the areas of relative agreement, allowing the conflict resolution effort to focus on the areas of discord.

3.7 Observations About Voting Behavior

Observations about voting behavior in organizations that are consistent with the behavior of the Consensus-Learning Committee Network:

(1) As remarked in section 3.1, organizations tend to punish Type II errors to a greater extent than Type I errors. To avoid this punishment, which may manifest itself in the form of dismissal or demotion, agents form larger consensuses that, for any given level of risk in the environment, will produce a relatively smaller number of Type II errors.

(2) At an extreme, certain types of organizations exhibit a profound reluctance to go forward with a decision on any consensus less than unanimity. This is a form of "herd instinct" or self-preservation - if everyone on a committee approves a decision, and it turns out to be a bad one, the organization will probably not punish

the entire committee. However, if a subset of the committee approves the same bad decision, the organization may punish those who were in the consensus. Albin[10] has suggested that committee members may offer their agreement as a form of gift exchange, similar to the employer-employee gift exchange described by Akerlof (1983) and modeled by the author as a bi-directional associative memory network (see Chapter 4). Sah and Stiglitz (1985) introduced their model by stating:

> This paper also provides an insight on why there is such a widespread sense of powerlessness in modern societies, even among individuals who occupy seemingly inportant decision making positions. One interpretation of this phenomenon is that an individual feels powerless if the collective decision is contrary to his judgment; for example if a project is accepted (rejected) when this individual disapproves (approves) of the project. The theory which we present suggests that when decision making is well-organized, the nature of human fallibility is recognized and, thus, this form of powerlessness is an essential counterpart of economic decision making.[11]

This is not inconsistent with Albin or Akerlof, in that committee members exchange the gift of helping to build a consensus for the gift of the feeling of empowerment in their unanimity. If every committee member walks away from a meeting being part of the consensus for every decision, then no committee member will need to "keep score," in order to try to "get even" at the next meeting. As there would be a time and effort cost in this scorekeeping, there is an economic benefit in the quid pro

---

[10] discussion with Peter S. Albin, March 23, 1993.

[11] p. 3

quo behavior in cost avoidance.

(3) In many organizations, the cost of the committee is invariant, even if the consensus size varies. Although, in practice, it may be costlier to make decisions with a larger consensus requirement, it is more difficult for management to attribute specific costs to the increased time spent trying to form a larger consensus than to attribute costs to the total committee. In the neural network model, the costs are equal for all consensus levels, because the hidden units do not communicate with one another. However, it will probably take longer to train a committee network with a larger consensus size.

3.8 Summary

The Consensus-Learning Committee Network can be used to model organizational behavior, when the environment is dynamic and organization members have the ability to modify their future behavior when observing after the fact outcomes. As demonstrated by Sah and Stiglitz and confirmed by the results herein, a larger consensus will increase the proportion of good decisions that are rejected and reduce the proportion of bad decisions that are accepted. The results from the neural network model also confirm the intuition that larger consensuses, over the long run, do not necessarily make an organization more profitable, as the overall percentage of correct decisions did not increase with consensus size.

PART TWO

APPLICATIONS

CHAPTER 4

BIDIRECTIONAL ASSOCIATIVE MEMORY NETWORKS APPLIED TO MODELING NON-NEOCLASSICAL ECONOMIC BEHAVIOR[1]

4.0 Introduction

Economic behavior is often studied by creating simplified and abstracted models of real world conditions and events. Underlying (and often conflicting) human traits, such as preferences for consumption, leisure, and altruism are maintained as assumptions and boundaries in these models. Initially, Artificial Neural Networks (ANN's, or neural networks) were applied to character recognition and other processes related to visual stimuli. However, neural networks have characteristics that make them suitable for modeling economic behavior. The property of convergence in the adaptation, or learning, is a form of optimization. The global stability achieved by neural networks is an equilibrium condition.

This paper proposes a neural network modeling reciprocal, or gift exchange, between both workers and their employer and workers among themselves. The gift

---

[1] Copyright 1992 IEEE. Reprinted, with permission, from Proceedings of the International Joint Conference on Neural Networks, Baltimore, MD, June 7-11, 1992, Vol. II, pp. 490-497.

exchange of workers' productivity for employer leniency produces an optimal wage rate for profitability and creates a non-market-clearing equilibrium between labor supply and demand. This neural network model simulates two intuitive results: 1) overall decline in workers' performance when the employer imposes stricter rules, and, 2) increased effort on the part of good workers to maintain overall productivity in exchange for the friendship of the poorer workers. Labor slowdowns and covering up for incompetent or shirking co-workers are frequently observed and can have a severe impact on society. Thus, we consider it useful to increase the understanding of the gift exchange process.

In sections 4.1 - 4-3,[2] there are references to other research on neural networks and economics, a discussion on the selection of the Bidirectional Associative Memory network model for this research, and some background on the application. Sections 4.5 and4.6 discuss the details of the experiments and results. Section 4.7 is a summary and conclusions section. The Chapter Appendix follows Section 4.7.

4.1 Some Prior Economic Applications of Artificial Neural Networks

ANN's have been applied as techniques of econometrics to demand forecasting, stock market predictions and risk classification. These same applications are usually approached by econometricians using linear and nonlinear regression, time series analysis, and maximum likelihood estimation. Kuan and White (1989) use

---

[2] This paragraph has been revised to conform to the section numbering conventions of the rest of the dissertation. In addition, references may be found in the Reference List.

feedforward ANN's to predict appliance ownership with the objective of forecasting demand for electricity. They input demographic data into each network to derive a likelihood of ownership of a specific appliance. Their results compare favorably with logistic regression. White (1988) also has studied prediction of stock returns with ANN's, focusing on methods to refute the efficient markets hypothesis. Dutta and Shekar (1988), Surkan and Singleton (1990), and Garavaglia[3] have each developed different bond rating classification models using various neural network models. Grossberg (1988) has proposed applying ANN's to modeling firms' production strategies and a resulting competitive equilibrium; each firm "adapts" its own production schedule based on public knowledge of market prices.

4.2 The Bidirectional Associative Memory Network as Behavioral Model

There are several approaches to applying a neural network to economic behavior. One is to build a network in which each processing element represents an economic agent, receiving inputs from other agents or from the environment, using the weighted summation and transfer functions as decision making devices, and producing some form of output which can be communicated further down the network or used as a final result. This output could represent an economic decision or action, or represent some equilibrium state.

A second approach is to have a network represent a more general functional relationship between two data sets. Just as in the former example, the result could

---

[3] Reprinted as Chapter 6

represent a decision, action, or equilibrium state. The benefits of the first method include the ability to examine and measure each agent as a separate entity. In the second method, the benefit is a working model independent of the number of agents.

The approach which actually we have here entails using a Bidirectional Associative Memory (BAM) network. The BAM network, which learns and represents associations between pairs of vectors, was introduced by Kosko (1992). The learning process involves the development of a matrix of weights that minimizes the energy function

$$E(x,y) = -x' * W * y \tag{4.1}$$

where x and y are a pair of associated vectors, x' is x transposed, and W is the weight matrix after training. Because the network is bidirectional, it is not appropriate to consider the vectors as strictly "inputs" or "outputs," i. e., each vector acts as both input and output in any one recall of an association. The BAM network receives a set of associations, an (x,y) pair, and responds with an (x,y) pair which represents the closest match. Unlearned pairs of vectors presented to a trained BAM network will be "fit" to the closest learned association.

This network was chosen for the following reasons:

(1) A feedback network seemed appropriate to model human behavior involving social interaction and behavioral norms.

(2) The Bivalent BAM Theorem, i. e., that every matrix is bidirectionally stable, indicates that a BAM weight matrix can represent economic equilibrium (see Kosko [8]).

(3) The BAM network can accommodate noisy data, suitable for the complexity of worker's characteristics, and also allows for some degree of fuzziness. When presented with an association that it has not explicitly learned, the BAM network can respond with the closest match that it has learned.

4.3 Background on the Original Case Study on Efficiency Wage Theory

The existence of involuntarily unemployed labor violates the neoclassical market-clearing condition of a supply-demand equilibrium. Various theories have been presented to explain why firms do not lower wages and hire workers until the market clears, or raise the minimum required output for each worker until the worker is on the verge of being indifferent between his or her current job and a less demanding position at lower pay. The *efficiency wage* is the wage paid by the firm that maximizes its profit but does not necessarily clear the labor market.

There are many plausible explanations for the existence of the efficiency wage. At the micro-level, the transactions cost of screening, hiring, firing, and training labor may counter the benefit of replacing or downgrading the current workforce when the supply increases. Barro (1987) discusses long-term labor contracts in the context of

the sticky price component of the Keynesian model. Another theory is the presence of unemployment acts as a threat, reminding employed workers that substandard performance may cost them their jobs.

Sociologist George Homans (1954) studied a group of ten female clerical workers for a five-month period (December, 1949 to April, 1950). All ten workers exceeded an employer-determined minimum output quota in an environment that both clearly lacked opportunities for advancement and held no real penalties for substandard performance. The workers all had knowledge of each other's job performance and socializing habits. Homans' conclusions were: (1) that the sub-organization of the ten workers into various groups for socializing purposes influenced their behavior; those who socialized within the group tended toward higher performance levels than those who socialized outside of the group, and (2) the lack of pressure to meet quotas and the "friendliness" and "niceness" of the group made each worker desire to contribute to the best of her ability.

Akerlof (1986) used the same case study to develop his economic theory of the standard-exceeding phenomenon as the employer's wage and employee's output having a reciprocal, or gift exchange, component. The employer offers a wage above the market-clearing wage and gives the employees significant control over their effort, by making the work rules fairly lenient. The employee reciprocates with consistent high levels of output and the quality of a low error rate.

Table 4-1 contains the data used for the neural network experiments. It is excerpted from the original Table 1 in Homans [7].

Table 4-1 -- Original Cash Posters Data

Name of Clerk	Cards Per Hour	Total Social Interactions	Total Number of Different People
Asnault	363	46	13
Burke	306	64	24
Coughlin	342	58	17
Donovan	308	32	14
Granara	438	40	16
LoPresti	317	56	14
Murphy	439	92	22
Rourke	323	60	16
Shaugnessy	333	16	9
Urquhart	361	32	13

Cards per hour represents the number of payment transactions posted to customers' ledger cards per hour. Total Social Interactions is the number of times the clerk spoke to another worker during the measurement period. Total Number of Different People is the total number of workers spoken to. For example, the clerk Asnault's hourly average was 363 postings per hour, and, during the measurement period, she spoke to 13 different workers in 46 instances.

The next two sections will describe how an artificial neural network can represent the behavior described by Homans and modeled by Akerlof.

4.4 The Cash Posters' Study As Utility Maximization

Akerlof's important conclusion was that workers' norms of effort were their own determinant of output as well as the work rules and the workers' innate ability. His economic model of the effort norm includes terms reflecting "the decline in the norm of effort as the work rules become increasingly binding on the worker's choice of effort," and "the effect on the norm of an inequality in the treatment of the two [good and poor] types of workers." Each worker's utility defines the overall benefit she receives, and is defined as a function of the norms, her effort, wages, and her "type" (i.e., "good", "poor", etc.). We assume rational utility maximizing behavior on the worker's part.

4.5 The BAM Network Model for the Cash Posters' Behavior

The purpose of this experiment was to determine how a BAM network could represent changes in worker behavior when the firm raised output standards. The network needed to represent the firm's minimum requirements, each worker's ability and effort as inputs, each worker's social behavior, and the worker's actual performance as output. The BAM network is restricted to bi-polar values in its data vectors; Figure 4-1 shows the encoding scheme used.

Figure 4-1 -- Encoding Scheme for Cash Posters BAM Network

'X' Vector for Each Worker's Characteristics:
Element 1: Firm's Minimum Requirements 1 if met by worker, else -1
Elements 2 - 6: Worker's Ability/Effort scaled as follows:

Sub-Vector	Cards Per Hour Range
[1, -1, -1, -1, -1]	300 - 314
[1, 1, -1, -1, -1]	315 - 329
[1, 1, 1, -1, -1]	330 - 349
[1, 1, 1, 1, -1]	350 - 364
[1, 1, 1, 1, 1]	365 or greater

Elements 7 - 10: Worker's Social Behavior, based on Interactions

Sub-Vector	Social Category[4]
[1, -1, -1, -1]	Socially isolated
[-1, 1, -1, -1]	Low social activity
[-1, -1, 1, -1]	Moderate social activity
[-1, -1, -1, 1]	High social activity

'Y' Vector for Each Worker's Results:
Element 1: 1 if below group norm, -1 otherwise
Element 2: 1 if at or above group norm, -1 otherwise
Element 3: 1 if below firm's minimum requirements, -1 otherwise
Element 4: 1 if at or above firm's minimum requirements, -1 otherwise

A 10-by-4 processing element BAM network was trained with the 10 worker's traits encoded as above, using the median value of 337 cards per hour as the group

[4] These four social behavior categories were based on a combination of the Total Social Interactions and the Total Number of Different People spoken to by each cash poster.

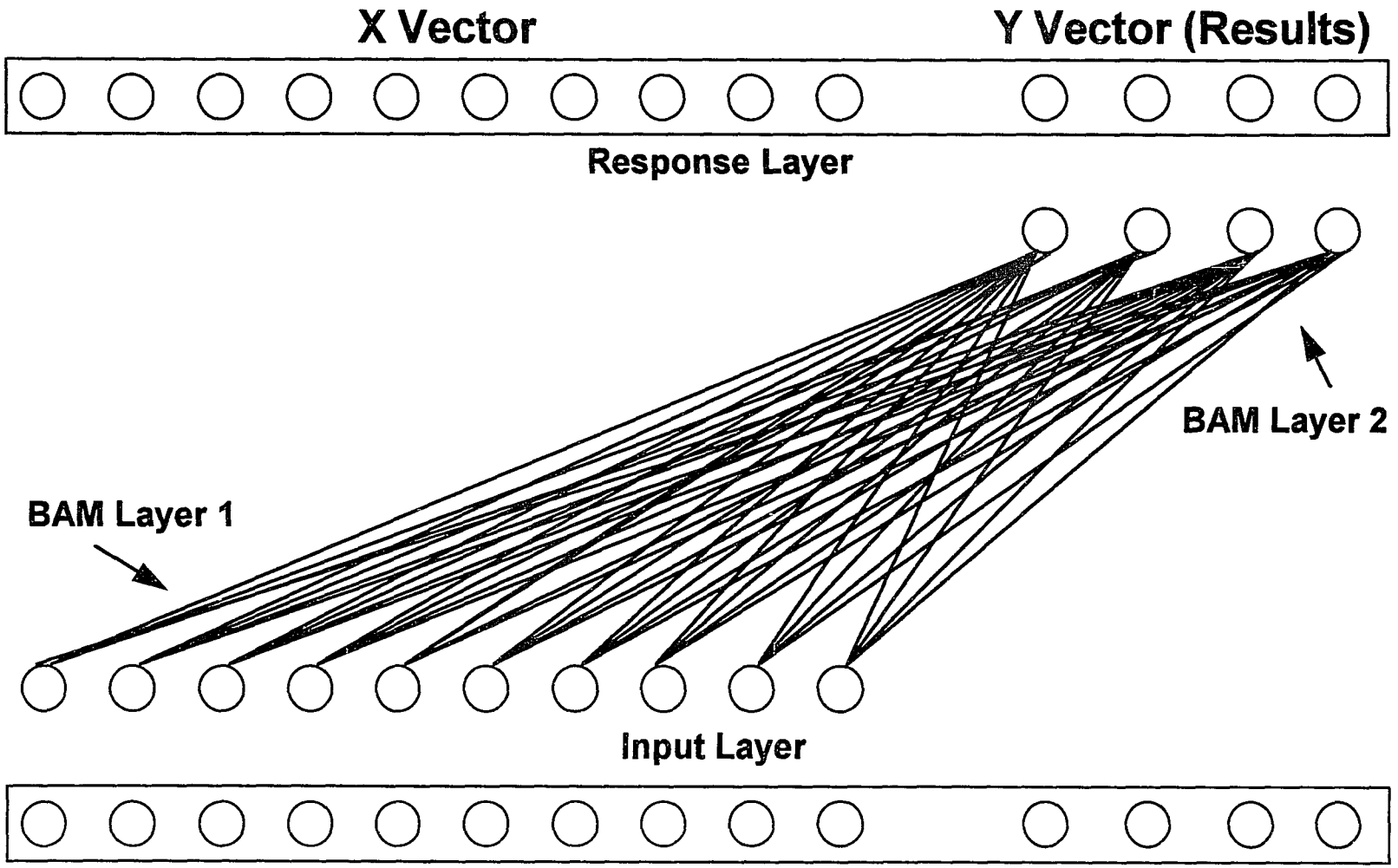
norm. Recall was performed with all possible trait combinations, i. e, 5 ability classes times 4 social interaction classes, or 20 "hypothetical workers." Training and testing was performed with the actual firm minimum of 300 per hour, 325 per hour, and 350 per hour, the last two being hypothetical. Figure 4-2 diagrams the BAM network used for all three experiments. The differences in the results are reflected by the resulting weight matrices (See Chapter Appendix for the actual training vectors, the result vectors and the weight matrices).

4.6 Interpretation of the Network's Responses

A reasonable assumption is that the network results obtained during recall could be interpreted consistently with the training sets, i. e., the elements representing effort/ability would continue to represent effort/ability in the same scale.

The results of the network recall indicate that when the firm minimum was 300 cards per hour, workers fell into two ability/effort groups, very high (above the norm) and medium (at the norm). One interpretation is that when workers can determine their own output norm, their utility is maximized when they strive to meet or exceed that norm. When the firm's requirements were raised to 325 per hour, there were still two performance groups, but these were polarized into very high and very low based on whether the firm's minimum was met. Thus, when the firm raised requirements, the worker's effort was determined by the firm's requirements, not the group norm. The network results can be interpreted to show that poorer workers reduced their effort, good workers increased their effort (supposedly to compensate

Figure 4-1 -- The Cash Posters' BAM Network



for their co-workers as a gift to them), but even in a group with more good workers, the end result was still less overall effort and productivity.[5] Under the actual 300-per-hour work rule, workers performed more homogeneously toward the higher effort scale. At a firm-set minimum of 350 cards per hour, the results were about the same as for the 325 card minimum. Refer to the Chapter Appendix for the actual results.

Some observations regarding the workings of the BAM network are appropriate here. The characteristics of the workers overlap and have fuzzy boundaries; therefore, they cannot be represented as orthogonal vectors. Without orthogonality, we expect the BAM network to be trained to recall perfectly each association exactly as it is presented.[6] However, in this application, the "inaccuracy" of the neural network results in a generalization of putting all workers into a limited number of performance groups. Thus, if we considered the BAM network as a classifying system, we could say that the network was capable of taking fuzzy data and finding the best grouping. The network produced two groups for each training set because the Y vectors indicated that the association of the worker's characteristics was an "above or below" condition.

When the firm minimum was set to 325 or 350 cards per hour, the network's

---

[5] This conclusion is based on the 350 card per hour network. The total number of -1's in the Ability/Effort subvector increases from 20 to 28 when the firm's minimum is raised to 350 cards per hour. See the Chapter Appendix.

[6] If the X vectors are orthogonal, the BAM network learns and recalls each association exactly as presented. With X vectors that are not orthogonal, the network's responses to associations become less accurate. This is a well-known limitation of Hebb's learning law. Hecht-Nielsen[6] discusses orthogonality, error estimation and error minimization with regard to Hebbian learning.

responses were two orthogonal vectors, one representing "good workers" and the other, "poor workers." As a result of the generalizing, new vectors were created that the BAM network did receive as part of the training. The three sets of weight matrices show that, as the firm minimum increased, the extreme values, i. e., +10 and -10, shifted from the lower end of the Ability/Effort subvector to the higher end. In the 300 cards per hour network, the +10 and -10 appear in the connection from the processing element (PE) representing 300-314 cards per hour; in the 325 cards per hour network, they are in the connection from the 330-349 cards per hour PE; in the 350 cards per hour network, they are in the connection from the 350-364 cards per hour PE. The extreme weight values shift according to the firm's minimum requirements.

Thus a BAM network was created, showing the same behavior described in equation-based models by Akerlof, i. e., that, for any given level of utility, workers produce more if they have more control over the work norms. The disutility of effort is compensated by the positive utility of having control over one's work environment. In addition, if the employer retracts the gift of leniency, the workers intentionally retract their gift of above-minimum productivity.

4.7 Summary and Conclusions

Artificial Neural Networks can be applied to analyzing and understanding economic behavior. Feedback networks, such as the BAM network are appropriate when economic agents make decisions based on other agents' behavior. In the case

presented, the BAM weight matrix represented the influence of the group of workers on any one worker. Each 'X' vector represented a specific worker's characteristics and each 'Y' vector represented the results given the firm's work rules. It was shown that imposing more constraints on the workers polarized them into two extreme performance groups with an overall result of reducing the effort offered by poorer workers. It also appeared that the presence of poor workers caused good workers to work harder, which might be attributed to their sentiments for them, so it is not conclusive that replacing the poor workers with better workers would have increased the productivity of the group.

Future research should include additional studies with other feedback network architectures and representing individual economic agents, such as firms in a single industry, as processing elements in a neural network.

- APPENDIX TO CHAPTER 4 -

A. Training Vectors

1. 300 Cards per hour

	Min	Ability/Effort						Soc. Behavior				Results		
Asnault	1	1	1	1	1	-1	-1	1	-1	-1	-1	1	-1	1
Burke	1	1	-1	-1	-1	-1	-1	-1	-1	1	1	-1	-1	1
Coughlin	1	1	1	1	-1	-1	-1	-1	1	-1	-1	1	-1	1
Donovan	1	1	-1	-1	-1	-1	-1	1	-1	-1	1	-1	-1	1
Granara	1	1	1	1	1	1	-1	1	-1	-1	-1	1	-1	1
LoPresti	1	1	1	-1	-1	-1	1	-1	-1	1	1	-1	-1	1
Murphy	1	1	1	1	1	1	-1	-1	-1	1	-1	1	-1	1
Rourke	1	1	1	-1	-1	-1	-1	-1	1	-1	1	-1	-1	1
Shaugnessy	1	1	1	1	-1	-1	1	-1	-1	-1	1	-1	-1	1
Urquhart	1	1	1	1	1	-1	-1	1	-1	-1	-1	1	-1	1

2. 325 Cards per hour

	Min	Ability/Effort						Soc. Behavior				Results		
Asnault	1	1	1	1	1	-1	-1	1	-1	-1	-1	1	-1	1
Burke	-1	1	-1	-1	-1	-1	-1	-1	-1	1	1	-1	1	-1
Coughlin	1	1	1	1	-1	-1	-1	-1	1	-1	-1	1	-1	1
Donovan	-1	1	-1	-1	-1	-1	-1	1	-1	-1	1	-1	1	-1
Granara	1	1	1	1	1	1	-1	1	-1	-1	-1	1	-1	1
LoPresti	-1	1	1	-1	-1	-1	1	-1	-1	1	1	-1	1	-1
Murphy	1	1	1	1	1	1	-1	-1	-1	1	-1	1	-1	1
Rourke	-1	1	1	-1	-1	-1	-1	-1	1	-1	1	-1	1	-1
Shaugnessy	1	1	1	1	-1	-1	1	-1	-1	-1	1	-1	-1	1
Urquhart	1	1	1	1	1	-1	-1	1	-1	-1	-1	1	-1	1

3. 350 Cards per hour

	Min	Ability/Effort						Soc. Behavior				Results		
Asnault	1	1	1	1	1	-1	-1	1	-1	-1	-1	1	-1	1
Burke	-1	1	-1	-1	-1	-1	-1	-1	-1	1	1	-1	1	-1
Coughlin	-1	1	1	1	-1	-1	-1	-1	1	-1	-1	1	1	-1
Donovan	-1	1	-1	-1	-1	-1	-1	1	-1	-1	1	-1	1	-1
Granara	1	1	1	1	1	1	-1	1	-1	-1	-1	1	-1	1
LoPresti	-1	1	1	-1	-1	-1	1	-1	-1	1	1	-1	1	-1
Murphy	1	1	1	1	1	1	-1	-1	-1	1	-1	1	-1	1
Rourke	-1	1	1	-1	-1	-1	-1	-1	1	-1	1	-1	1	-1
Shaugnessy	-1	1	1	1	-1	-1	1	-1	-1	-1	1	-1	1	-1
Urquhart	1	1	1	1	1	-1	-1	1	-1	-1	-1	1	-1	1

B. Result Vectors

1. 300 Cards per hour

	Min		Ability/Effort					Soc. Behavior				Results		
Good Worker:	1	1	1	1	1	-1	-1	1	-1	-1	-1	1	-1	1
Poor Worker:	1	1	1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	1

2. 325 Cards per hour

	Min		Ability/Effort					Soc. Behavior				Results		
Good Worker:	1	1	1	1	1	1	-1	1	-1	-1	-1	1	-1	1
Poor Worker:	-1	-1	-1	-1	-1	-1	1	-1	1	1	1	-1	1	-1

3. 350 Cards per hour

	Min		Ability/Effort					Soc. Behavior				Results		
Good Worker:	1	-1	1	1	1	1	-1	1	-1	1	-1	1	-1	1
Poor Worker:	-1	1	-1	-1	-1	-1	1	-1	1	-1	1	-1	1	-1

C. Weight Vectors

1. 300 Cards per hour

	Min		Ability/Effort					Social Behavior	
Below Norm	0	0	-4	-8	-8	-4	2	-2	0	0
Above Norm	0	0	4	8	8	4	-2	2	0	0
Below Min	-10	-10	-6	-2	2	6	8	0	6	6
Above Min	10	10	6	2	-2	-6	-8	0	-6	-6

2. 325 Cards per hour

	Min		Ability/Effort					Social Behavior	
Below Norm	-8	0	-4	-8	-8	-4	2	-2	0	0
Above Norm	8	0	4	8	8	4	-2	2	0	0
Below Min.	-10	-2	-6	-10	-6	-2	0	0	2	2
Above Min	10	2	6	10	6	2	0	0	-2	-2

3. 350 Cards per hour

	Min		Ability/Effort					Social Behavior	
Below Norm	-8	0	-4	-8	-8	-4	2	-2	0	0
Above Norm	8	0	4	8	8	4	-2	2	0	0
Below Min.	-10	2	-2	-6	-10	-6	0	-4	2	-2
Above Min	10	-2	2	6	10	6	0	4	-2	2

CHAPTER 5

A SELF-ORGANIZING MAP APPLIED TO MACRO AND MICRO ANALYSIS OF DATA WITH DUMMY VARIABLES

5.0 Overview

The two-dimensional self-organizing system, or map, introduced by Kohonen (1989) is well-suited to data clustering. Data clustering is often used for marketing applications and demographic research. One limitation of clusters is that one cannot easily control the sizes of clusters, although it is possible to combine related clusters to form groups of different sizes. The self-organizing system also can perform multidimensional scaling, that is, reduce the dimension of a set of vectors to that of a lower dimension without losing the ordering (See (Kendall 1980)).

The clustering process yields groups that possess similar characteristics, as measured by the distance of the data vectors from some defining central vector, but the question may arise as to how well individual clusters identify a subgroup that exhibits significantly different behavior from the general population. Especially in applications of human behavior, such as marketing response models, in which individual agents make decisions in a decentralized manner, it seems appropriate to search for a method of analyzing how a set of clusters can be compared to the population.

In order to produce a type of macro-analysis, namely, an understanding of the distribution of data and variability of the population, and a corresponding micro-analysis, a comparative analysis of data clusters, two self-organizing maps (SOM) were developed using a subset of the 1990 U. S. Census data. One SOM was developed using unscaled

data vectors, and produced the micro-topology. The other SOM was developed using the same data vectors expressed in standard scale format, which is essentially a measure of standard deviations from the mean, and produced the macro-topology.

A selection of data elements was chosen and converted to dummy variables, which assume the value of 0 or 1. This was done to take advantage of some of the special properties of dummy variables and to provide some results that could be applied to analyzing surveys or other data in which "yes-no" indications and qualitative elements play a significant role.

5.1 Applications of Dummy Variables

Dummy variables have two major applications in statistical and econometric analysis: (1) they are used for implementing "switches," such as those used for discrete shifts in a regression, i. e., seasonal adjustment, different "regimes," and so forth; and, (2) they used to represent qualitative factors, such as gender, race, geography, and other data. For further reference, see (Johnston 1984) and (Judge et. al., 1988).

A dummy variable also has the useful property and that the sample average for that variable represents the proportion of observations having a value of one. As an illustration, in a sample of 1000 observations, if a dummy variable is used to represent high school diploma status, with 1 meaning the subject has a high school diploma, and 0 otherwise, if 750 of the subjects have a high school diploma, the average is .75, and the standard deviation is about .43.

Another application of dummy variables which is apropos here is to allow simplification of data for specific analytical objectives. For example, the U. S. Census data provides median family income by census tract. An organization may be interested in consumer response rates only for families with income of at least $40,000 versus income of less than $40,000. In this case, the median family income may be represented by a

dummy variable that is set to 1 if median family income is at least $40,000, and 0, otherwise.

5.2 Creating the Self-Organizing Maps

The SOM experiment was designed with the objective of producing a set of results that could be analyzed relatively quickly with conventional computational tools.

5.2.1 The U. S. Census Data and Transformations

The data set is one Census Division from the 1990 U. S. Census, comprising data from the North East Division, including Connecticut, Maine, Massachusetts, New Hampshire, Rhode Island, and Vermont. Each data record, or vector, represents one census tract, and there are 3,202 records. This geographical division was chosen because it had the fewest number of records yet would still have enough data for a training data set and a testing data set. The odd numbered records were used for training; the even-numbered for testing; the data is sorted by census tract number. Because of the numbering system for census tracts, records from the same county are sorted together. Therefore, choosing every other record would still result in having every county represented in both the training and testing data sets.

The data element representing the state was extracted from the census tract number ; the first two digits of the tract number are the FIPS (Federal Information Processing Standards) code. These codes had to be converted into a dummy variable form, because the FIPS numbers are based on the alphabetic order of the state names, and their direct input into a a neural network or statistical analysis would imply the belief that the state name itself were predictive. A six element sub-vector was created to represent the state - {1,0,0,0,0,0} for Connecticut, {0,1,0,0,0,0} for Maine, and so forth.

Table 5-1 --
Input Data Elements for Self-Organizing Map
Each record represents a Census Tract

Data Element	Value	Sample Mean	Sample Standard Deviation
Median Age	1 if >=35, 0 otherwise	0.439100562	0.496354914
Race	1 if >= 90% white, 0 otherwise	0.780762024	0.413794829
Worker Type	1 if >= 70% White Collar, 0 otherwise	0.916614616	0.276507033
High School Graduate	1 if >= 75% are High School Grads, 0 otherwise	0.668332292	0.470885854
Connecticut	1 if Connecticut, 0 otherwise	0.260462211	0.438999806
Maine	1 if Maine, 0 otherwise	0.116177389	0.320531058
Massachusetts	1 if Massachusetts, 0 otherwise	0.411930044	0.492282481
New Hampshire	1 if New Hampshire, 0 otherwise	0.081511555	0.273700816
Rhode Island	1 if Rhode Is., 0 otherwise	0.07339163	0.260856509
Vermont	1 if Vermont, 0 otherwise	0.056527171	0.230406002
Median Family Income	1 if >= $40,000, 0 otherwise	0.522798251	0.499557983

Five other data elements were also converted to a dummy variable format and used for the SOM input Table 5-1 lists all the data elements and some basic statistics. .For the purposes of this research, the data is treated as a statistical sample of the entire U. S., even though it represents the entire population of the North East Division.

A second training and testing data set were created by transforming each data element in dummy variable form to the standard scale, or, as often referenced, *z-scale*. The standard scale formula is:

$$z = \frac{x - \mu(x)}{\sigma} \tag{5.1}$$

where $\mu(x)$ is the sample mean of x and σ is the sample standard deviation (See (Johnson & Bhattacharyya 1987) or (Phillips 1992)). The purpose of the standard score is to

Table 5-2
Input Data Elements for Self-Organizing Map
Transformed to Standard Scale

Data Element	Dummy=1	Dummy=0
Median Age	1.130037041	-0.884650378
Race	0.529822899	-1.886833688
Worker Type	0.301566955	-3.314977579
High School Graduate	0.70434842	-1.419308493
Connecticut	1.684597076	0
Maine	-0.36245283	0
Massachusetts	-0.836775752	0
New Hampshire	-0.297812613	0
Rhode Island	-0.281348663	0
Vermont	-0.24533723	0
Median Family Income	0.95524797	-1.046521664

Note: Once the data is transformed to standard scale, all means are zero and all standard deviations are 1.

facilitate comparisons among different variables. The standard score expresses in data in terms of the number of standard deviations from the mean.

Table 5-2 shows the transformed data elements. Note that the sub-vector for the state uses only the standard scale for the dummy variable equal to 1. This is because the six elements in the sub-vector represent only one variable.

5.2.2 The SOM Neural Networks

A separate SOM was created for each data format, i. e., unscaled dummy variable, and standard scaled dummy variable. As each data vector has 12 elements and each data set has 1,601 observations, a number of hidden units for the two-dimensional Kohonen layer had to be chosen to reflect the desired maximum number of clusters and allow for sufficient degrees of freedom. An 8 x 8 Kohonen layer was used for both SOM's.

Training efficiency was not an objective, that is, is was not considered important at this time to compare training times between the two forms of the data. Each network was trained for 160,100 iterations, or 100 passes of the data.

One unconventional aspect of this SOM development is that the data were not normalized to points on the unit sphere, but used directly in the formats described above. Generally, if the input vectors **x** are normalized, so that the norm of **x**, $||\mathbf{x}|| = 1$, the resulting weight vectors will also be normalized (See (Kosko 1992) , (Ritter et. al. 1992), Or (NeuralWare 1991)). This is more important for an analysis of the weight vectors than for the overall topology of the resulting SOM's. Meeting this requirement prevents any confusion between the magnitudes of the values of the data elements and their actual information content. Because the data elements were all two-valued, either unscaled or scaled, the ranges of the norms of the vectors are constrained to a bounded interval. The variation in individual norms created some noise, but, as a first experiment, the results were still considered usable.

5.3 Comparison of Unscaled and Standard-scaled Data

The two SOM's produced strikingly different topologies. The SOM created with the unscaled dummy variables produced a set of clusters that was more evenly distributed in the two-dimensional Kohonen layer than the standard-scaled dummy variables. Tables 4-1 and 4-2 illustrate the resulting topologies. The 64 circles represent the 8 x 8 Kohonen layer. The bold circles are the hidden units that were activated by at least 1 observation from the testing data set. In each circle, the top number is the hidden unit number and the bottom number is the number of observations from the testing data set that activated the hidden unit.

In Figure 5-1, there are 33 units representing active clusters. Figure 5-2 has only 18 clusters, and they are concentrated in one area (lower right) of the map.

The intuition from these contrasts is that the variability of the data was learned by the SOM with unscaled input, while the homogeneity of the data was learned by the SOM with the standard scaled data. This may appear to be a contradiction, but, on examining the standard scale values, only one value, worker type, represents a value that is more than 2 standard deviations from the mean. This is to say that individual clusters with defining characteristics exist within this sample, but the use of these clusters to predict significantly different behavior may be limited.

Another facet of these results is that the first SOM produces a larger number of clusters which are spread out in the topology, and this level of detail is a form of micro-analysis of the data. This type of clustering would satisfy a requirement for a large number of clusters . The second SOM, by producing a smaller set of clusters with a dense region in the topology, has provided a form of macro analysis of the data. Table 4-2 shows that there is a major subgroup defined by units 30, 32, 39, 40, 46, 48, 54, 55, 62, 63, and 64 that comprises almost 77% of the sample (1,229 out of 1,601 observations). The other active units represent clusters that are, for the most part, isolated from each other, and separated from the main subgroup by at least two units. The macro-view of the data leads to the suggestion that the population is primarily homogeneous with about 23% of the population in various outlying groups. This is supported by the actual values generated by the standard scale transformation, which reveal a low degree of variability.

Figure 5-1
SOM Topology Using Unscaled Dummy Variables

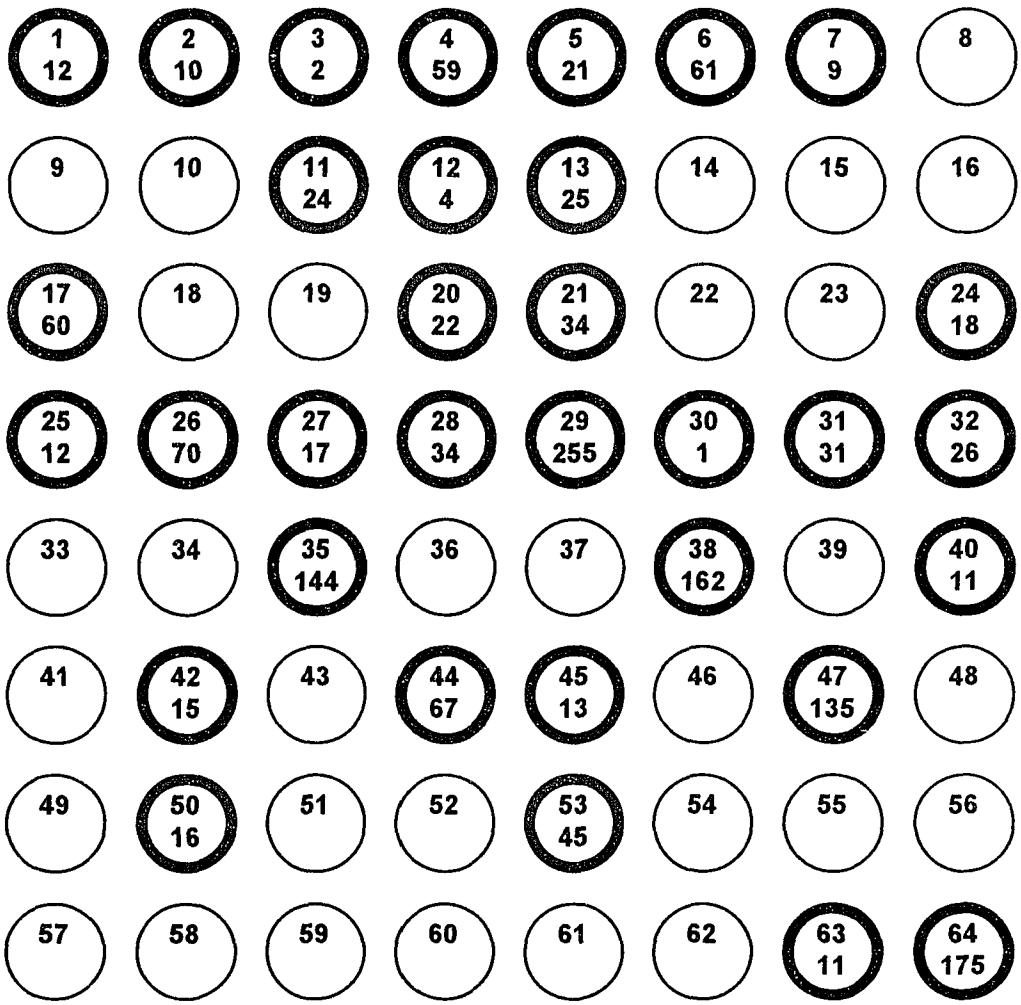
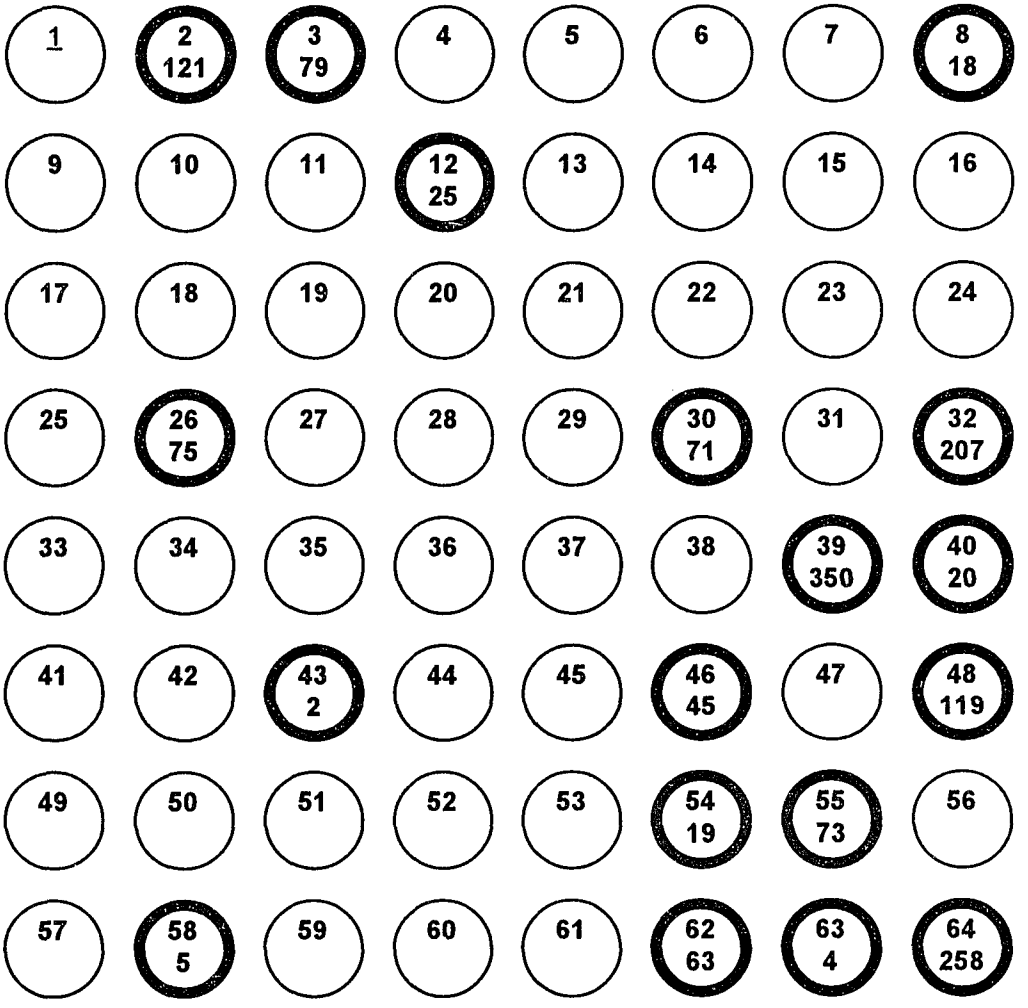


Figure 5-2
SOM Topology Using Scaled Dummy Variables



5.4 Summary

The Kohonen Self-Organizing System has the potential for providing multiple views of a single data sample. This is accomplished by rescaling the data vectors to emphasize different properties of the data. In addition, converting the data elements to dummy variables provides a form of simplification that allows identification of key characteristics of a population.

Future research will include incorporating all nine divisions of the U. S. Census data and a more conventional use of the SOM by normalizing all data vectors. This will permit integrating an analysis of the resulting weight vectors with the topology. In addition, using the entire U. S. Census data set will provide more degrees of freedom, and larger Kohonen layers will be feasible.

CHAPTER 6

AN APPLICATION OF A COUNTERPROPAGATION NEURAL NETWORK: SIMULATING THE STANDARD & POOR'S CORPORATE BOND RATING SYSTEM[1]

6.0 Introduction

6.0.1 Neural Networks as Classification Systems

Consider the process of classification as determining set membership. An object with one or more attributes is given membership in one of several predefined sets as a result of some transformation performed on the attributes. Examples of transformation processes may be a Bayesian calculation to assign the object to a class based on probability, a scoring algorithm, or reasoning with a set of logical clauses, all described in (Rich 1983). In addition, if-then rules combined with a forward or backward chaining inference engine have been applied successfully to classification problems. Winston (1984) summarizes two well known rule-based classification systems, MYCIN, which classifies blood diseases, and DENDRAL, which classifies general molecular structures, e. g., ketones. estrogens. etc. Fisher and Langley (1986) discuss conceptual clustering in the CLUSTER/2 system as a form of classification. More recently, fuzzy sets are being explored as a method of qualitative classification. The aforementioned classification techniques are commonly associated with artificial intelligence paradigms; however, discriminant analysis and various discrete regression models are also classification tools. See Maddala (1983) for

---

[1] Copyright 1991 IEEE. Reprinted, with permission, from Proceedings of the First International Conference on Artificial Intelligence Applications on Wall Street, New York, New York, October 9-11, 1991., pp. 278-287.

complete descriptions of the mathematical regression models which incorporate qualitative data.

In the above examples, exhaustive knowledge of the attributes of each class and construction of a specific algorithm or set of rules is required. The development of these types of classification systems is therefore expensive, time consuming, and difficult to maintain and expand if attributes change or a new class emerges. In addition, unknown but useful classifications may exist, but a lack of available expertise to articulate the attributes and code the rules or algorithms makes system development impossible.

Neural network models have the potential to overcome both the development and maintenance expense and unavailable expertise. Neural networks belong to the adaptive processing systems which are developed by a training process which includes inputting a subset of the data to be classified along with their correct outputs, then applying a learning algorithm for error correction, and a highly general transformation process to constrain the data to some measurable interval.

6.0.2 Comparison with Econometric Methods

Alternate econometric methods of classification (e. g. discriminant analysis), include regression techniques and maximum likelihood estimation. In regression analysis, the objective of accurate classification is achieved by pursuing mathematical goals of deriving unbiased estimators, minimizing the sum of squared errors. minimizing the covariance between the input data and the error terms, and other key statistical relationships. In maximum likelihood estimation. the mathematical goal is to derive estimation parameters that will result in the highest probability of having obtained the observed data, i. e., maximizing the probability that the observed sample reflects the "real world." Kennedy (1989) provides helpful insights for comparing

and contrasting econometric techniques and applications. Generally, neural network models work through analogous goals. The parameter sets sought after in an econometric model are replaced by vectors of weights that connect the: processing elements, which contain summing and transformation functions. Hence the term connectionism, which has become almost synonymous with neural networks, (see (Narendra and Thathachar 1989)). The sum of squared error minimization corresponds to convergence of the learning algorithm, which is to say achieving close as possible agreement between the neural network output values and the desired values. In neural network models, the error minimization process is the iterative application of a learning algorithm. In conventional regression analysis, calculus and matrix algebra are used to solve equations to obtain values of coefficients.

In both standard econometric and neural network methods, a set influential variables must be chosen in advance. The end result of parameter estimation may be to eliminate variables from the final model, due to near zero parameters associated with them. Misspecification is a serious issue in both methods.

Parsimonious representation is a common ideal. The fewest parameters necessary correspond to the simplest neural network model, expressed in number of layers and processing elements.

Neural networks have the advantage of not requiring prior information about the nature of the disturbances. In addition, neural networks' interconnectivity potentially can benefit from relationships between variables that normally impede regression analysis. In other words, collinearity is less of a problem.

The application which is the subject of this paper most closely represents discriminant analysis for a set of functional relationships that provide the key to grouping data into discrete classes based on similarities among the data, so that new data will be assigned to the most suitable group with the highest certainty possible.

6.1 Neural Networks as an Application of Bounded Rationality

When relationships are complex and nonlinearity exists, neural networks have the: potential to be more accurate than traditional methods. Rather than specify the model which represents the suspected relationships among parameters and data, a neural network approach defines a model of how to learn about those relationships. Thus, less information is required at the outset of the analysis.

With this in mind. neural networks can be considered a method of dealing with Herbcrt A. Simon's "bounded rationality" (Simon 1982). Given limited knowledge about the domains and bounds or the resources to perform calculations, neural networks are a strategy to cope with problems that must be solved in spite of many unknowns. Neural networks are adaptive models as opposed to pure optimization models and serve the goal of "satisficing" more than optimizing.

Neural network learning algorithms can be thought of as "rational adjustments." as described by Simon (1982) which function by "... adjusting behavior at a rate that is proportional to the difference between the actual and the criterion." Learning algorithms are a key component in distinguishing one neural network architecture from another.

6.2 Relevance to Finance Industry

The idea of using a neural network to simulate the S & P bond rating process arose in discussions with corporate finance professionals. Much of the recent corporate finance marketing activity has been directed at privately held companies and mid-sized public companies. Large highly rated companies. such as those in the Fortune 1000, are considered mature markets, where opportunities are scarce and competition with other corporate finance entities yields narrow margins. The ability to produce equivalent S & P corporate bond ratings for non-rated companies would allow better credit and risk analysis.

In addition to rating debt securities that are not rated by an agency, a bond rating simulator would have several other applications:

(1) A company could anticipate the rating it would receive prior to contacting the agency.

(2) The agency could run a "sanity check" against its own analysts' work.

(3) An organization could compare newly published ratings with the model's rating to determine if the rating process used by the agency has changed since the model's development.

6.3 Prior Research

Dutta and Shekhar (1988) also identified bond rating as a classification problem and designed their neural network experiments to be easily compared to statistical hypothesis testing. Their model used the neural network back-propagation model to determine if the hypothesis that a bond rating belonged to a specific rating class could be rejected or not. They chose 10 variables and used a training set of 30 companies, and a test set of 17 companies.

Surkan and Singleton (1990) also used back-propagation and modeled their research after earlier work using discriminant analysis. They chose the same variable and companies and obtained significantly better results than the more conventional approach. Their findings included the result that a neural network employing two hidden layers performed better than the more common one hidden layer network given the same total number of processing elements.

Counterpropagation was chosen as the neural network model for this research because it appeared to more naturally lend itself to classification than backpropagation because of

its two-valued (0 or 1) output.[2] Dutta and Shekhar (1988) and Surkan and Singleton (1990) have made specific comparisons to the results of conventional statistical methods, while this research emphasizes the performance of neural networks under the constraints of fuzzy and unevenly populated classes.

6.4 The Counterpropagation Model

Robert Hecht-Nielsen invented the counterpropagation network in 1986 to learn explicit functions and act as a statistically optimal lookup table. Alternative neural network models are described in both (Hecht-Nielsen 1990) and (Klimasauskas 1990). In general, if the counterpropagation network is to represent the function $y = f(x)$, counterpropagation approximates f by having the x and y vectors flow through the network in opposite (hence, counterpropagation) directions to yield approximation vectors x' and y'. The relationship between the vectors x' and y' infers f.

The actual result is a nearest neighbor classifier, with the weight vector for each hidden layer processing element, or PE, serving to represent the average value for each input for a specific class. The hidden layer PE whose weight vector most closely approximates the value of the inputs get the "privilege" of signaling its "win" by sending a value of one (1) to the output layer. "Losers" send a value of zero (0), which has no subsequent influence on the outcome. Most practical applications of counterpropagation need a feed-forward-only version, because the purpose of the effort is to perform classification, not to derive the function.

The counterpropagation model used for this research is described in detail in (Klimasauskas 1990). Although Klimasauskas lists three requirements for successful classification with counterpropagation, the results below will demonstrate that the strict

---

[2] Under perfect discrimination conditions. Chapter 9 discusses how to interpret counterpropagation output when the outputs are values between 0 and 1.

class separability requirements may be relaxed without sacrificing meaningful results. The three requirements are separable classes, sufficient PE's in the hidden layer, and training inputs equal distributed among the classes.

6.5 Objectives of This Research

This research was undertaken with the following objectives in mind:

(1) To determine if a connectionist paradigm could provide a good approximation of a classification process containing both quantitative and qualitative measurements. Even though no qualitative input was used, the idea occurred that the activity of the hidden layer and the complexity generated by the number of connections could provide a proxy for the qualitative judgments taking place. In the case of corporate bond ratings, Standard & Poor's position is "Ultimately, the judgment is qualitative in nature and the role of the quantitative analyses is to help make the best possible overall qualitative judgment because, ultimately, a rating is an opinion."[3]

(2) To contribute to a small but actively growing portfolio of application of neural networks to solving problems in business and finance. This research could lead to a system which could help predict changes in bond ratings or produce an "S & P equivalent" for a non-rated company, e. g., a privately held or foreign company.

(3) To determine how neural network models may fit in with the traditional set of econometric modeling tools.

---

[3] (Standard & Poor's Corporation 1979).

6.6 Input Data

6.6.1 The Standard & Poor's Bond Rating System

Standard & Poor's Corporation (S & P), a subsidiary of McGraw-Hill, is an independent entity which perform the services of rating various forms of securitized debt and publication of business and financial information. Corporate bond ratings, the subject of this research, have been offered by S & P since 1923. S& P does not make public their specific rating methodology; however,they publish a general description of the factors they consider and what their ratings mean (see (S & P 1979)). S & P was chosen over other agencies at this time because of the availability of data.

6.6.2 Input for this Research Project

Using the data element from an example of an S & P Corporate Credit Analysis (CCA) report and mapping them to the available data elements from the CompuStat public database, 87 input data elements were obtained for each public company that had S & P Corporate Bond Ratings available. A total of 797 out of a database of about 7,200 public companies had bond rating information. Although the CCA spread had 10 years. downloading and data formatting capacity limited this initial effort to 3 years (current rating with 2 lags). See (S & P 1979) for a sample of the CCA and the data elements. S & P actually uses a 3-year period in its comparative industry summary analysis. The rating tested was from 1987, so the two lags were 1986 and 1985. Older data was chosen so that any forecasting that was to be done as part of this research could be easily measured by looking at more current data. It turned out that forecasting did not become part of this initial research.

The bond ratings range from AAA (highest, most creditworthy) to D (lowest, in default). Generally, ratings AAA, AA, A, and BBB are considered "investment grade,"

i. e., the Comptroller of the Currency allows commercial banks to invest in those securities. The ratings BB, B. CCC. and CC are considered risky and speculative. A rating of C means the bond currently is not paying interest. A rating of D is for a bond in default of either principal or interest. Pluses and minuses added to those basic categories provide some more detailed information about an issue relative to the basic rating scale. See (S & P 1979).

The: data have characteristics that are not considered ideal for classification processing. First of all, the detailed classes and the three major grades, i. e., investment, speculative, and poor quality, are not equiprobable. In order to get a sufficient sample of poor quality, the three classes were grouped slightly differently than the standard S & P description: ratings AAA through BBB- are "investment," ratings BB through B- are "speculative." and ratings CCC through D are "poor quality." Normally, only ratings C and D are not considered good enough for the speculative class. The total sample is skewed heavily toward the investment and speculative grades. Out of a total of 797 in the sample, 581 are investment grade, 184 are speculative, and 44 are poor quality. See Table 6-1 for a distribution of ratings in the total data sample. This is not surprising; one would guess that companies who anticipate low ratings would not seek to be rated and vice versa.

Another challenge to classification is that the class boundaries are not believed to be crisp because of the qualitative and subjective factors used. Furthermore. the range of values for each input element varies widely, often by 3 or 4 powers of 10.

Table 6-1 --Data Distribution of Bond Ratings

Ratings	No. of Companies	Percent
AAA	36	4.5%
AA+	20	2.5%
AA	88	11.0%
AA-	51	6.4%
A+	94	11.9%
A	80	10.0%
A-	70	8.8%
BBB+	59	7.4%
BBB	51	6.4%
BBB-	32	4.0%
BB+	25	3.1%
BB	17	2.1%
BB-	20	2.5%
B+	27	3.4%
B	34	4.3%
B-	49	6.1%
CCC+	10	1.3%
CCC	9	1.1%
CCC-	9	1.1%
CC	3	0.4%
C	4	0.5%
D	9	1.1%

6.7 Methodology

In general, relatively inexpensive hardware and software were used to perform the research. In addition, the hardware and software (IBM PC/DOS compatible), are also commonly available in a financial services environment.

6.7.1 Preparing the Input Data for the Neural Network

A neural network system needs at least two sets of data for development: one for training which should be a subset of the actual data to be classified, and one for testing, which should be a larger sample of the universe. In the training phase, both the inputs and the desired outputs are given to the network. In testing, the network derives the output values, but they must be compared to the correct output in order to measure the results.

For each set of company data, the following had to be performed:

(1) The input data had to be constrained so that their values were between -1.0 and +1.0. This is a requirement of the counterpropagation neural network model and also many other models have similar requirements. Since the data would undergo a normalization step as part of the model, each data element was simply divided by some power of 10 as was necessary to make all the sample data elements lie in [-1.0,+1.0].

(2). Missing data had to be estimated. This was done by taking the average value for the available data for each class[4] and replacing the "N/A's." About 14% of the individual data elements were missing; a total of 9,926 out of 69,339 (87 input times 796 data sets) total data elements. Most of the time, these missing items were clustered around a group of inputs. i. e., large numbers of N/A's appeared in certain data fields. This would indicate data that companies are not required to report, or data that is not relevant for all types of businesses, e. g., inventory turnaround is not relevant for services. Sales, income, and debt were virtually complete, but pension data, working capital, and "turnarounds" (inventory and receivables) were highly populated with "N/As."

---

[4] This might create a problem when the network is asked to estimate the class membership of new data vectors in recall. One approach is to substitute the average values for every class in the missing elements, and see which output PE has the highest value, indicating the strongest indication of class membership. The usual approach is to substitute the sample mean for the data element in every observation.

(3) An output vector had to be created for each set of input data. Each output PE in the network represents a rating class. The output element representing the resulting output class should have a value of 1.0 and all others should have a value of 0.0.

(4) A representative training set had to be created. Given the total number of data sets for each class, 9 sample companies were chosen from each rating from AAA through B-, and 2 each were chosen from CCC through D to make up a single "class." The companies were in alphabetical order within each class and the first 9, and 2, respectively, were selected for the training set. This methodology was considered as good as selecting every nth data set for an initial experiment.

6.8 . Designing the Networks

The only decision factor in the network design other than the decision to accept the defaults of the software was to choose the number of PE's in the hidden, or Kohonen, layer. The number of PEs in the hidden layer(s) must be sufficiently large to separate the classes. Hecht-Nielsen (1990) implies that any non-trivial problem requires at least hundreds of hidden layer elements in a counter-propagation network.

Because of the complexity of mapping so many potential applications with some optimal number of hidden layer PEs, no universally objective standards exist. It is a topic for continued research.

Four experiments were performed. varying the number of PE's in the hidden layer. Although the output layer had one PE for each class of rating, except for the C and D ratings. which were grouped in one class, the nature of the S & P rating system allows this

architecture to be used to separate bond ratings in the three general classes: "investment." "speculative," and "poor quality."

All four models had the same number of input elements, 87 plus an activation element (with output constant at one), and the same number in the normalization layer, 88. Table 6-2 shows the number of PEs in the hidden and output layers. and the total number of PEs

Table 6-2 contains a summary of the four models:

Table 6-2 -- Number of PE's for Each Model

Networks	#1	#2	#3	#4
Input Layer	88	88	88	88
Normalization Layer	88	88	88	88
Hidden Layer	170	264	425	156
Output Layer	17	17	17	17
Total	363	457	618	339

Except for the model with 156 hidden layer PEs, the number of PE's for the hidden layer were chosen arbitrarily as functions of either the input or output PE numbers, i. e., 170 is 10 times the outputs, 264 is three times the inputs, 425 is 25 times the outputs. The number of training data sets is 156, so theoretically, each hidden layer PE could learn one to identify one of the is not believed that there is a linear relationship at this time, although future research plans

will contain additional experiments and results that may point the way to discovering this relationship.

6.9 Training

Training is accomplished by the continuous presentation of the training input to the network. The training data is 156 sets representing 9 companies from each rating group AAA through B- and 12 companies, 2 each with ratings CCC+ through D for the last output class. For Network #1, the: network converged (the Root Mean Square error met the criterion of being close enough to zero) after 7,409 training examples. Networks #2 and #3 did not meet the convergence criteria, but after 10,000 training examples, their weights had not changed for at least 1 pass of the entire training set. Network #4. converged after about 7,040 examples.

6.10 Measuring Results

6.10.1 Single Measure of Accuracy

An overall measure of accuracy is the percentage that the network classified correctly, which is simply the number of correct output vectors divided by the total number of output vectors (the sample). For each experiment. The results are presented in two ways; first, the detailed results by each separate class; then, the results grouped into the three major investment classes. The total data set used for the test was 641 examples. Tables 6-3 through 6-6 summarize the results of each experiment.

Table 6-3 --Network #l (170 hidden PE's)

Rating	Percent Correct
AAA	44.44%
AA+	50.00%
AA	20.25%
AA-	7.14%
A+	16.47%
A	19.72%
A-	31.15%
BBB+	20.00%
BBB	28.57%
BBB-	30.43%
BB+	18.75%
BB	0.00%t
BB-	18.18%
B+	11.11%
B	4.00%
B-	29.69%
Cs and Ds	34.38%

Investment Class

Class	Percent Correct
Investment	92.25%
Speculalive	62.50%
Poor Quality	32.26%

Misclassification Results

Error	Number
Speculative in Investment Class	30
Poor Quality in Investment Class	2
Investment in Speculative Class	38
Poor Quality in Speculative Class	19
Investment in Poor Quality Class	0
Speculative in Poor Quality Class	14

Accuracy for three investment classes 83.78%

Table 6-4 --Network #2 (264 hidden PE's)

Rating	Percent Correct
AAA	44.44%
AA+	50.00%
AA	18.99%
AA-	9.52%
A+	16.47%
A	19.72%
A-	29.51%
BBB+	28.00%
BBB	28.57%
BBB-	30.43%
BB+	18.75%
BB	0.00%t
BB-	18.18%
B+	11.11%
B	4.00%
B-	29.69%
Cs and Ds	34.38%

Investment Class

Class	Percent Correct
Investment	92.25%
Speculalive	65.50%
Poor Quality	32.26%

Misclassification Results

Error	Number
Speculative in Investment Class	30
Poor Quality in Investment Class	2
Investment in Speculative Class	38
Poor Quality in Speculative Class	19
Investment in Poor Quality Class	0
Speculative in Poor Quality Class	11

Accuracy for three investment classes 84.24%

Table 6-5 --Network #3 (425 hidden PEs)

Rating	Percent Correct
AAA	44.44%
AA+	50.00%
AA	18.99%
AA-	7.14%
A+	16.47%
A	19.72%
A-	29.51%
BBB+	28.00%
BBB	28.57%
BBB-	30.43%
BB+	18.75%
BB	0.00%t
BB-	18.18%
B+	11.11%
B	4.00%
B-	29.69%
Cs and Ds	34.38%

Investment Class

Class	Percent Correct
Investment	92.04%
Speculalive	66.67%
Poor Quality	32.26%

Misclassification Results

Error	Number
Speculative in Investment Class	28
Poor Quality in Investment Class	2
Investment in Speculative Class	39
Poor Quality in Speculative Class	19
Investment in Poor Quality Class	0
Speculative in Poor Quality Class	11

Accuracy for three investment classes 84.40%

Table 6-6 --Network #4 (156 hidden PEs)

Rating	Percent Correct
AAA	55.56%
AA+	20.00%
AA	20.25%
AA-	9.52%
A+	16.47%
A	22.54%
A-	31.15%
BBB+	36.00%
BBB	30.95%
BBB-	39.13%
BB+	18.75%
BB	0.00%t
BB-	18.18%
B+	22.22%
B	8.00%
B-	23.44%
Cs and Ds	34.38%

Investment Class

Class	Percent Correct
Investment	92.25%
Speculalive	68.33%
Poor Quality	32.26%

Misclassification Results

Error	Number
Speculative in Investment Class	30
Poor Quality in Investment Class	2
Investment in Speculative Class	38
Poor Quality in Speculative Class	19
Investment in Poor Quality Class	0
Speculative in Poor Quality Class	14

Accuracy for three investment classes 84.9%

6.11 Analysis of Results

The obvious result is that the counterpropagation network performed extremely well in separating three broad classes, but did rather poorly in separating 17 individual bond rating classes. Also, these results might challenge the belief that overall accuracy increases monotonically with the increase in hidden layer PEs. The experiment using only 156 hidden layer PEs (the number of examples in the training set) gave the best accuracy at 84.9%. In all cases, accuracy among the 17 single rating classes never seemed to improve.

There are several possible explanations for the performance discrepancy between three and 17 classes. Some of these explanations could serve as a basis for further research.[5]

(1) Not enough PEs in the hidden layer

The experiments to date are not conclusive, regarding the optimal hidden layer size for this application.

(2) Individual classes not separable enough

Given the amount of input, the complexity of the actual rating process and the degree of overlap in the data itself, it would be doubtful if there is any type of easily defined separation between the classes. Qualitative factors, like the maturity of the industry and the general state of the economy could be expresses as additional inputs, but S & P doesn't publish exactly how these aspects into their own equations.

---

[5] The following chapter, Chapter 7, contains more results from the bond rating problem and covers the topic of network stability. Appendix 3 contains results from a logistic regression on the bond rating data set.

(3) Further normalization of inputs needs to be done

Simply dividing a value by some power of 10 to assure that all the data for that specific input element fall in the interval [-1.0,+1.0] might not be sufficient, even though vector normalization takes place. Some other squashing function might be more appropriate for the range of values in the data.

(4) Distribution of classes too skewed

The self-selection of higher quality companies for rating services will probably continue. This will make it difficult to determine the effect of the skew on performance. In order to deal with skewed distributions, new neural network models or variants on existing models will have to be developed.

(5) Not enough information in the input layer

Both the among of "N/As" in the input and the fact that only 2 lags were used might have had an effect on the network's performance. Different approaches to missing data and the inclusion of more lags could be a topic of further research.

6.12 Summary and Conclusions

With some effort and experimentation, a fairly successful classification system was built using a feed-forward counterpropagation neural network. Success was based on the network's ability to produce the same general broadly defined class indicated by the corporate bond rating as published by S & P, without detailed knowledge of the mathematical and judgmental processes used by S & P.

If the true bond rating process is a many-to-one mapping, then, according to Hornik, Stinchcombe, and White (1989), then "...standard multilayer feed-forward network

architectures using arbitrary squashing functions can approximate virtually any function of interest to any desired degree of accuracy, provided sufficiently many hidden units are available. These result establish multilayer feed-forward networks as a class of universal approximators." Their research involves mathematical proofs of this concept. What is interesting is contemplation of the converse, except that the implied condition is that there is some correct network model and hidden layer size for a problem of a particular type that cannot fail to approximate the function, if, in fact the function exists.

The results of this research were not considered successful for producing a network to separate inputs into the complete spectrum of the 22 bond ratings defined by S & P. Possible causes for this failure were discussed. Hornik, Stinchcombe, and White (1989) assert "... failures in application can be attributed to inadequate learning, inadequate numbers of hidden units, or the presence of a stochastic rather than a deterministic relation between input and target." It might be true that the S & P bond rating classes overlap too greatly to be separable because of the intent by design, or because the rating system is unintentionally inconsistent.

Future experiments could be performed by varying the size of the hidden layer, the squashing function, and adding more input data. However, there should be some determination of some upper bound on the optimal complexity of the network, i. e., what is cost-justified and worth maintaining, prior to continuing.

The results of this experiment beg the questions: are 22 corporate bond ratings a real and meaningful guide for investors? Is there a consistent difference between, for example, an A and an A- rating? One application of neural networks might be to determine the true separability of existing classification systems.

CHAPTER 7

DESPERATELY SEEKING STABILITY: NEURAL NETWORKS WITH INSUFFICIENT DATA[1]

7.0 Background

A counterpropagation network for simulating the Standard & Poor's corporate bond rating systems was proposed by Garavaglia[2]. The initial results were that network performance measured in terms of overall accuracy ranged from 83.78% to 84.9% for classifying three general categories, AAA to BBB- as investment grade, BB+ to B- as speculative grade, and all lower ratings as poor quality, and ranged from 0% to 55.56% for classifying 17 specific rating grades. In the three category evaluation, about 98% of the misclassifications were "one off," i. e., investment grade companies were not misclassified as poor quality ("two off") and vice versa. These results raised a number of issues, including the feasibility of a usable model when the distribution of classes is uneven and the boundaries between the classes appears to be somewhat fuzzy.

This paper uses the same data and application but is concerned with the topic of neural network stability, i. e., the evaluation and maximization of network performance with out-of-sample data in terms of predictability, reliability, and usability. We consider the results of several new experiments that are evaluated in terms of the differences in performance between the training and testing data. The recommendations, given the non-optimal

---

[1] Copyright 1993. Software Engineering Press. Reprinted, with permission, from Proceedings of the Second International Conference on Artificial Intelligence Applications on Wall Street.New York, New York. April 19-22, 1993 (forthcoming)

[2] Reprinted as Chapter 6.

situation of an insufficient number of observations to provide the required degrees of freedom, are to: 1) take actions to reduce the degrees of freedom, and 2) analyze the rank ordering power of the network output values to determine if a subset of the network output is sufficiently stable and predictive. Dutta, Shekhar, and Wong (1993) have addressed the stability problem from the standpoint of preventing network overfitting and failure to generalize as well as further discussions of the bond rating application.

The problem of insufficient data is not trivial. One may observe that no matter how much data is available, the complexity of the application almost always seems to require more data. Categorization type applications in financial services in which only a limited amount of data may be available include:

(1) Corporate Bond Ratings: Standard and Poor's (1991) has outstanding ratings on about 2,000 corporations, 1,600 issuers of commercial paper, and about 8,000 public sector entities (municipals and other government-related). The coverage of Moody's Investors Service is similar in scope. Depending on the number of rating classes, these quantities may not be sufficient. Other rating agencies, such as Fitch and Duff & Phelps, have even fewer outstanding ratings.

(2) Senior Corporate Lending Risk Management: The top tier corporate lending function of a money center bank may have only between one and two thousand customers. Banks' internal rating systems usually have 9 or 10 classes. Depending on the number of data elements used, there may not be enough examples.

(3) Transaction Risk Identification: Any categorization task in which some categories are represented by a very small (e. g. less than 1 %) of the sample present problems. This is typical in the areas of fraud detection and predicting default. When a large sample, which has an artificially high proportion of these categories, can be provided, these problems can

be overcome.

7.1. Degrees of Freedom in Neural Networks

In conventional statistical techniques, the term degrees of freedom refers to "the number of free or linearly independent sample observations used in the calculation of a statistic."[3] In regression analysis, a requirement is that there are more observations than coefficients to be estimated, in order to guaranty a unique solution. This is because the solution process requires that the observation vectors be able to form a non-singular (invertable) square matrix when multiplied by themselves. With fewer observations than variables, the rows and columns of the square matrix are not linearly independent, and the rank of the matrix is less than the dimension of the matrix.

Another way of understanding this restriction is to consider the degrees of freedom as the number of observations minus the number of coefficients to be estimated. Zero or negative degrees of freedom means that a potentially infinite number of solutions exist to exactly fit the data, but these solutions will have no relationship to out-of-sample data. A simple example is a point in *n*-dimensional space. If this point were to represent a single observation, any number of lines of any slope can be drawn through that point. Therefore, if an arbitrary line is chosen, the predicted fit of this line to any population is indeterminate. If there are two points in *2*-dimensional space, a line, defined by a intercept and slope, will fit these two points exactly, but have no predictable relationship to any other point. Therefore, if two observations are used to estimate a 2 parameter model, there will be overfitting.

The strict definition of degrees of freedom requires that the number be greater than zero. Therefore, this discussion, which refers to negative degrees of freedom, is stretching the definition for the purposes of exploring the behavior of neural networks when there

---

[3] Kennedy (1985), p. 214.

are not enough data vectors, or observations, to produce a non-singular square matrix from the vectors. Kennedy (1985) and Phillips (1992) approach this subject from a viewpoint of practical implementations and Mac Lane's (1986) chapter on Linear Algebra is an excellent mathematical treatment.

The analogy in neural networks is the problem of overfitting the data in the training sample. This occurs when the number of training examples, or observations, does not exceed the total number of connection weights in the network. If the training sample does not represent the characteristics of the population, results may also look like overfitting.

In a fully-connected counterpropagation network, the total number of connections C is:

(7.1) $$C = (I + O) * H \qquad (1)$$

where I is the number of input layer units, O is the number of output layer units, and H is the number of hidden units. Therefore, the total number of connections in the bond-rating network #1 in Chapter 6 with 170 hidden PE's, is:

17,680 connections =
(87 input units + 17 output units)
* 170 hidden units

The number of degrees of freedom is therefore: 156 - 17,680 = -17,524.

7.2 The Revised Experiment

As a result of the initial research, a number of changes were made to the experiment:

(1) The training, or development, data set was enlarged from 156 to 415 examples, and the testing, or holdout, data set was reduced to 381 from 640 examples. The objective of this realignment of the data was to provide the maximum degrees of freedom in

development while still providing a full range of examples for testing. The fact that the sample did not contain equal proportions for each rating class made this more difficult.

(2) The data elements, with the exception of the SIC code, and the two prior ratings, were transformed to their natural logarithms. Formerly, the transformation was to simply divide these data elements by powers of 10. Using natural logarithms is more consistent with common practice.

(3) A separate test was performed using three output categories representing very general grades - investment, speculative, and poor quality, as described in the Introduction.

7.3 Evaluation Criteria

In the original research, the neural networks were evaluated based only on the results from the holdout data set. The main focus of this current research is to determine relative levels of model stability. Model stability will be evaluated on two criteria: 1) the overall accuracy results on the training (development) versus the testing (holdout) samples, and 2) the overall accuracy results on the highest output unit value. In counterpropagation, the output values are in the range [0,1]; the value 1 affirms membership in the category represented by the output unit, and only one output unit contains a 1 for any given presentation of input vectors. However, in actual implementation, output values may exhibit a full range of values in the [0,1] interval, which may correspond to the degree of fit to the category, although not in a fuzzy-set membership sense. In Chapter 9, using this range to produce a rank-ordering is discussed in detail.

7.4 Results

Table 7-1 contains the results from the original network #1 (From Table 7-2 of Chapter 6). Clearly, this is an example of overlearning or overfitting the training data. Overall accuracy falls to 22.46% from 97.44% and the accuracy of the highest output unit value falls to 11% from 100%. This network would be considered highly unstable as would be expected based on -17,510 degrees of freedom. Neither the overall accuracy nor the accuracy of any output unit value is stable.

Table 7-1
Stability Results of the Original 170-hidden Unit Network

Rating	Training Pct. Correct	Testing Pct. Correct
AAA	100.00%	44.44%
AA+	100.00	50.00
AA	100.00	20.25
AA-	100.00	7.14
A+	100.00	16.47
A	100.00	19.72
A-	100.00	31.15
BBB+	88.89	20.00
BBB	100.00	28.57
BBB-	88.89	30.43
BB+	100.00	18.75
BB	100.00	0.00
BB-	100.00	18.18
B+	88.89	11.11
B	88.89	4.00
B-	100.00	29.69
CD	100.00	34.38
Overall	97.44%	22.46%

Highest Output Unit Value :	0.997381
Accuracy in training:	100.00%
Accuracy in testing:	11.00%

Table 7-2 shows the results of a network that also contains 170 hidden units, but uses the new training data set of 415 examples. The number of degrees of freedom is -17,265. Overall accuracy falls from 60% to 32.02%. The highest output unit value is 100% accurate for both samples. As the difference in overall performance between the training and testing data sets is smaller than in the original 170-hidden unit network, and the accuracy of the highest output value does not degrade, this network is considered more stable.

Table 7-2
Stability Results of the New 170-hidden Unit Network

Rating	Training Pct. Correct	Testing Pct. Correct
AAA	70.00%	18.75%
AA+	60.00	0.00
AA	62.50	22.92
AA-	46.15	4.00
A+	90.00	70.37
A	62.50	47.50
A-	62.86	31.43
BBB+	43.33	33.33
BBB	60.00	16.67
BBB-	75.00	0.00
BB+	40.00	0.00
BB	30.00	0.00
BB-	60.00	20.00
B+	26.67	0.00
B	40.00	7.14
B-	93.33	73.68
CD	87.50	45.00
Overall	60.00%	32.02%

Highest Output Unit Value: 0.938848
Accuracy in training: 100.00%
Accuracy in testing: 100.00%

Table 7-3 shows the results when the number of hidden units is reduced to 51. The number of degrees of freedom is now "increased" to -4,889, (415 observations - (87 input units + 17 output units)*51 hidden units), based on formula (1). The difference in overall accuracy between the training and testing data sets is also reduced even further than the 170-hidden unit network, but the overall performance is poor. The highest output unit value is still reasonably accurate at 88%.

Table 7-3
Stability Results of the 51-hidden Unit Network

Rating	Training Pct. Correct	Testing Pct. Correct
AAA	35.00%	25.00%
AA+	20.00	44.44
AA	60.00	47.92
AA-	19.23	0.00
A+	17.50	5.56
A	57.50	47.50
A-	5.71	2.86
BBB+	20.00	10.34
BBB	6.67	0.00
BBB-	15.00	0.00
BB+	40.00	10.00
BB	40.00	0.00
BB-	20.00	0.00
B+	0.00	0.00
B	5.00	7.14
B-	73.33	73.68
CD	83.33	55.00
Overall	30.00%	22.10%

Highest Output Unit Value: 0.962427
Accuracy in training: 100.00%
Accuracy in testing: 88.00%

When the number of hidden units is again reduced to 17, the same number as the number of output units or classes, the overall difference in accuracy is minimal (see Table 7-4). The degrees of freedom has been increased to -1353. The stability of the output unit values is difficult to assess, however, because the five highest output unit values are assigned only one observation each in the training data set and the two highest output values are not assigned to any observations in the testing data set.

Table 7-4
Stability Results of the 17-hidden Unit Network

Rating	Training Pct. Correct	Testing Pct. Correct
AAA	70.00%	31.25%
AA+	10.00	0.00
AA	47.50	37.50
AA-	0.00	0.00
A+	22.50	27.78
A	17.50	17.50
A-	5.71	2.86
BBB+	0.00	0.00
BBB	3.33	0.00
BBB-	20.00	8.33
BB+	0.00	0.00
BB	0.00	0.00
BB-	0.00	0.00
B+	0.00	0.00
B	0.00	0.00
B-	73.33	68.42
CD	67.67	70.00
Overall	20.00%	19.42%

Highest Output Unit Value: 0.975326 (1 example)
Accuracy in training: 100.00%
Accuracy in testing: (N/A)

The last test involved creating a network that had a positive number of degrees of freedom. In order to create such a network with the data, the number of output classes was reduced from 17 to 3. The three classes are investment grade (ratings AAA to BBB-), speculative grade (BB+ to B-), and poor (below B-). As there are 415 training set examples, a positive number of degrees of freedom required that a maximum of 4 hidden units is allowed:

degrees of freedom =
415 observations
- (87 input units + 3 output units)
* 4 hidden units
= 415 - 360 = 55

Table 7-5 shows the results of this network. Since there were only four hidden units, there are only four output unit values. The network effectively used two hidden units for classifying investment grade bond ratings, two for speculative bond ratings, and did not classify poor grade ratings at all. The accuracy percentages of all four output unit values are shown at the bottom of the table.

Table 7-5
A General Investment Class Categorization Network

Class	Training Pct. Correct	Testing Pct. Correct
Investment	91.8%	94.8%
Speculative	63.0%	58.3%
Poor	0.0%	0.0%
Overall:	79.52%	82.9%
Output unit values:		
0.991468	99.16%	96.77%
0.718827	79.26%	84.53%
0.620933	62.86%	58.33%
0.513653	50.00%	51.85%

7.5. Discussion

In reviewing the five tests discussed above, it appears that the data are most suitable for a classification task that is limited to three general investment grades. It appeared that by increasing the degrees of freedom, network stability improved, even though the degrees of freedom remained negative in the first four tests. The overall accuracy for the testing

data set varied from 19.42% to 32.02%.

An attempt was made at a stable network that could classify the 17 ratings classes, by reducing the number of input data elements from 87 to 24, keeping only the SIC code, the past two period's bond ratings, and the data elements that represented financial ratios. The network could not discriminate at all, putting all of the examples in the same class in the hidden layer and having very low output values for every class. In addition, estimation of the model was attempted with logistic regression with no usable results.[4]

In the counterpropagation paradigm, each set of weight vectors connected from the inputs to a hidden unit adapts to become a form of linear discriminant. Because the hidden layer activation function treats each hidden unit as an independent competitor for "claiming" an input vector, the trained model conceptually has as many discriminant functions as hidden units. Therefore, the total number of coefficients may not be the best term to use to compute the degrees of freedom, in that, by having stability improve within a negative range of degrees of freedom, the model behaves as if there are more degrees of freedom available. The backpropagation paradigm, in contrast to counterpropagation, should require degrees of freedom to be calculated on the total number of connections, because backpropagation learning adjusts all of the connections for each learning iteration. This is topic for further research and may suggest the application of certain network architectures when only a small amount of data is available.

7.6 Conclusion

The availability of a sufficient number of observations is critical to neural network stability and reliability. To assure a positive number of degrees of freedom, more observations are needed for a neural network model than for a conventional statistical

---

[4] Using SAS software on an HP 750 processor, the problem required about 13 megabytes of memory and did not converge after 10,000 iterations.

regression model. The number of degrees of freedom is the number of observations minus the total number of connections, which, in the case of a fully connected three-layer network, can be calculated by using formula (1). Some improvement in stability may be gained, even if the number of degrees of freedom is negative, if the number of hidden units or input units can be reduced, but these results are not consistent with accepted practices in statistical modeling. In any case, the developer must be aware of the total number of connections generated by a neural network and anticipate stability problems if not enough observations, or training cases, are available.

PART THREE

PRACTICES

CHAPTER 8

EXPOSING THE HIDDEN LAYER[1]

8.0 Introduction

A hidden layer of an artificial neural network is a subset of the total number of neural processing elements. It may receive input signals from an input layer or another hidden layer and send an output signal to the output layer or another hidden layer. The hidden layer is named as such because it does not receive signals from or transmit them to any destination external to its network. Therefore, conceptually, its function is "hidden" from perception; however, with the ability to view any location in computer memory through software, the activity of the hidden layer can be examined. Every neural processing element in the hidden layer contains a weighted summation function and a transfer function, which is usually non-linear and bounded. The purpose of this transfer function is to translate the information provided by the weighted summation into a meaningful signal that the rest of the network can process.

In supervised learning, an artificial neural network is trained by repeatedly exposing it to input data vectors and corresponding desired output vectors, which are the training examples; in unsupervised learning, desired output values are not supplied. Through learning algorithms, the connection weights are generally updated after each training example is presented until some condition is reached. During recall, the network performs some input-output mapping function with the learned set of weight vectors. The set of weight vectors in a trained network do not change until re-training

---

[1] Copyright 1992. High-Tech Communications, Inc. Reprinted with permission, from Advanced Technology for Developers. Vol 1. (June), pp. 6 - 10.

is initiated; they represent the learned information about the relationships in the data.

Prior to training, it is customary to assign random values to the weight vectors. Once this randomization process has completed, training with a fixed and ordered set of data vectors is a deterministic process. There are a number of well-documented neural network architectures, each of which may be associated with a general type of statistical model for parameter estimation. Three neural network architectures are discussed in the sections that follow, and the activity of each hidden layer is examined.

8.1 Backpropagation as Continuous Function Approximation

Backpropagation is the probably the most widely used neural network architecture. The backpropagation architecture and its learning algorithm are described in detail in Kosko (1992). In general, the error, or difference between desired and network-derived output, is input to a learning algorithm to update the weights. Each weight is changed in proportion to and in the opposing direction of its contribution to the error. This leads to convergence, or zero difference, between desired and derived output because the transfer function of the hidden and output layers constrains the output signal to a closed interval $0 <= | 1 |$. Therefore, the range of the error is also a closed interval.

Backpropagation performs the same function parameter estimation as non-linear least squares. Both techniques seek to minimize the sum of squared errors. Non-linear models are very important because they have the capability to represent "real world" conditions that are impossible to show with linear models. Neftci (1986) discusses five characteristics of non-linearity: stationary cycles, asymmetric behavior, jumping, 3rd and higher moments availability, and time irreversibility. Business cycles and market fluctuations are examples that require non-linear forecasting models, and may display all five types of non-linearity.

In non-linear least squares (NLS), a functional form and an objective function and goal, such as minimizing the sum of squared errors (SSE) must be specified, and an iterative method, such as Gauss-Newton or Newton-Raphson is employed. In backpropagation, the goal is also to minimize the SSE, and in that process the network forms a polynomial approximation of a non-linear function; the number of polynomial terms depends on the number of neural processing elements in the hidden layer.

Consider the following NLS example from Judge, et. al. (1988), in which a 20 observation sample (Appendix A) is used to estimate the parameter β in the model:

$$y = \beta X_1 + \beta^2 X_2 + u \tag{8.1}$$

where y is the dependent variable, β is the parameter to be estimated, X_1 and X_2 are the independent variables, and u is the disturbance vector. Clearly, equation (1) is non-linear. A Gauss-Newton iteration method produces the value of $\beta = 1.1612$.

The same data was used to train a backpropagation network with 9 hidden units. As a result of training, a vector of 37 connection weights, 3 feeding into each of the 9 hidden units, and 10 feeding into the output unit, was derived. This 37 element vector functions as a set of estimators of the functional relationship between the X matrix and the y vector. If the trained network were to be expressed as a mathematical equation, the result would be:

$$O = T\left(\beta_{h0} + \sum_{j=1}^{q} T(X'\theta)\beta_{hj}\right) \tag{8.2}$$

where O is the output value, the β_{hj} are the connection weights from the hidden units to

the output unit, β_{h0} is the weight from the Bias unit to the output unit, X' is the transposed input matrix containing X_1 and X_2 as in (1), θ is the vector of weights connecting the input to the hidden layer, and T is some non-linear bounded transfer function, in this case the hyperbolic tangent. The inner expression is the hidden layer and there are q hidden units. Because T is non-linear, and has a power series approximation, (2) is a polynomial equation of potentially an infinite number of terms. This representation is based on that given in Kuan and White (1991).

The theoretical basis for a neural network as a universal function approximator is discussed, with proofs, in Hornik, Stinchcombe, and White (1989). Their proofs are based on the Stone-Weierstrass theorem, which states that any function on a closed interval can be approximated by a polynomial equation with the degree of accuracy increasing as the number of polynomial terms reaches infinity. Thus, in this example, the hidden layer allows the creation of a higher order polynomial equation without explicitly stating a functional form. The hyperbolic tangent function, for example, can be approximated by the power series

$$(8.3) \qquad tanh(x) = x - (1/3)x^3 + (2/15)x^5 - (17/315)x^7 + \ldots.$$

Thus, even a one hidden unit network can theoretically represent a polynomial equation with an infinite number of terms.

It is common practice to validate a neural network's predictive power by comparing its results with actual cases that do not belong to the training examples. Common statistics can be calculated relating to the size of the error or the percentage of incorrect results. However, sometimes not enough test cases are available at the time the model is built. Since both NLS and backpropagation minimize the sum of squared

errors, it is possible to calculate the SSE for both and compare; in this example, the SSE for the NLS regression is 16.30797 and the SSE for the neural network is 3.58862. This is one criterion to indicate that the neural network achieved a better fit than NLS with the given model specification in equation (1).

8.2 Counterpropagation as Discriminant Analysis

Counterpropagation is an appropriate architecture when the objective is to separate data into a pre-defined number of groups where the characteristics of each group are well-known. The same data as for the backpropagation example is used for comparison. Instead of a single continuous random variable for output, the output of counterpropagation indicates class membership, and the output layer contains one processing element per class. For training and testing, the original dependent variable was replaced by a 4 element vector separating the data into 4 classes - Class 1 for $y <= .99$, Class 2 for $1.0 < y <= 1.99$, Class 3 for $2.0 < y <= 2.99$, and Class 4 for $y > 3.0$. Because the data did not separate naturally into groups, counterpropagation was not nearly as effective as backpropagation. Several tests with different numbers of hidden units yield the following results in Table 8-1:

Table 8.1 -- Counterpropagation Network

Number of Hidden Units	Percent Correctly Classified
7	70%
9	75%
15	80%
31	95%

One obvious problem with this application is that with 31 hidden units, there are more units than classes, and the hidden units learn the characteristics of a single test case, and have may be deficient in generalizing to members of a class. The fact that, even with more than enough hidden units to cover every test case, the network only achieved a 95% accuracy rate is an indicator that this data did not lend itself to classification into groups.

In counterpropagation, the hidden units each learn to become a "prototype" for a class by having their input connection weights assume the values of the means of each of their respective inputs. In addition, the hidden layer elements interact with each other competitively. The hidden layer transfer function effectively selects the hidden unit which has weights that form the shortest Euclidean distance[2] between themselves the input vector. That hidden unit "wins" and sends an output signal of 1 to the output layer while all other hidden units send 0. The output units are trained to respond appropriately and communicate the correct class membership. A complete discussion of counterpropagation is in Hecht-Nielsen (1990), who invented this architecture.

In discriminant analysis, the objective is to derive linear functions that represent the characteristics of each class. Where more than two classes are required, the procedure is to assign the data to a class based on the probability of that data belonging to that class. The technique of this form of discriminant analysis involves logistic regression and can be found in Maddala (1983). The weighted summation function in each hidden unit in a counterpropagation network is a type of linear discriminant function and the transfer function represents the rational assumption that the data that is the minimum distance from that linear discriminant function has the highest probability

---

2 Euclidean distance d_{ij} from point i to point j in metric space:

$$d_{ij} = [\sum_{k=1}^{p} (x_{ik} - x_{jk})^2]^{1/2}$$

of being a member of that hidden unit's class. If the weight w is close or equal to the input x, the square root of $w * x$ will equal or be close to x, which is why the Euclidean distance is used in the transfer function.

8.3 Self-Organizing Map as Metric Space

The Self-Organizing Map (SOM) network architecture provides a method for gaining information about the possible grouping of data with unsupervised learning. In a self-organizing map, each input vector is mapped to one processing element in a two dimensional grid that represents a metric space. Input vectors that are nearly identical should map to the same processing element, and input vectors with very similar characteristics should map to adjacent processing elements.

In the counterpropagation example in Section 8.2, supervised learning was used, i. e., during training, the network was presented with the correct class membership. The primary difference between discriminant analysis and clustering is that, in discriminant analysis the classes are predefined and known while in clustering the primary objective is to develop a classification system including the number of classes and the criteria for assignment to a class. Gordon (1981) develops this comparison in more detail. When the same data was used in a self-organizing map with a 5 x 4 processing element arrangement, the data formed 7 classes (see Figure 8-1).

A straightforward method of assessment of this test is to compare variances of the entire data set with each group and combinations of groups that "clustered" together on the processing element "map." Table 8-2 shows variances by group.

Table 8-2 --Results of Self-Organizing Map Network

Data in Group	No. of Observations	Variances X_1	X_2	Y
Entire Dataset	20	0.096	0.067	1.042
Proc. Element 6	2	0.002	0.004	0.997
Proc. Element 7	2	0.002	0.028	0.041
Proc. Element 10	7	0.002	0.012	0.307
Proc. Element 13	1	n/a	n/a	n/a
Proc. Element 16	1	n/a	n/a	n/a
Proc. Element 19	1	n/a	n/a	n/a
Proc. Element 24	6	0.084	0.003	0.200
Proc. Elements 6 & 7	4	0.090	0.028	0.735
Proc. Elements 6 & 10	9	0.002	0.039	0.634
Proc. Elements 4 & 24	8	0.119	0.060	1.094

The variances for the entire dataset are greater than for any one subgroup, showing that the self-organizing map formed subgroups of the data that had similar characteristics. In addition, the groups comprising the data mapped to P. E.'s 6 & 7 and P. E's 6 & 10 had lower variances than the group comprising the data mapped to P. E.'s 6 & 24. In the case of the X_1 and Y variables, the variances in the "6 & 7" group were higher than the "6 & 10" group, but when looking at their proximity in Figure 4-1, it is observed that 10 is adjacent to 6 as well. Chapter Appendix B contains the actual data groups from this experiment.

SOM's are usually used in combination with supervised learning networks, such as backpropagation or counterpropagation, which were discussed earlier. This small example shows how a self-organizing map can make "discoveries," but it is necessary to supplement this activity with conventional statistical analysis of what the network accomplished.

Figure 8-1 --Arrangement of Processing Elements
in Two-Dimensional Hidden Layer of Self-Organizing Map

21	22	23	**24**
17	18	**19**	20
13	14	15	**16**
9	**10**	11	12
5	**6**	**7**	8

(The bold type indicates that these processing elements actively represent clusters)

8.4 Summary and Conclusions

Three different neural network architectures were discussed in terms of modeling functional relationships. In each architecture, the hidden layer's transfer function determined the characteristics of the activity. Backpropagation is highly suitable for estimating non-linear continuous functions, and the same test statistics are available for a direct comparison between any backpropagation network and a corresponding non-linear model. For discriminant analysis, counterpropagation is a rational choice, providing there exists a high degree of linear separability among the groups. For data clustering, a self-organizing map without a supervised learning layer will form data groups that are arranged in a two-dimensional metric space according to their similarity. More analysis, however, is needed to give some meaning to the SOM results.

Often backpropagation is applied to the discriminant analysis or the clustering with

satisfactory results. This has led to the opinion that backpropagation is the only architecture needed in order to benefit from the technology. However, when a problem lends itself well to counterpropagation or a SOM, more attention has to be given to the form of data representation to translate the problem into a backpropagation example (and vice versa). With a rich variety of network architectures available, it is not necessary to force an application into backpropagation. The effort saved on just using backpropagation may be lost on the analysis and data transformation required to avoid building other networks.

Chapter Appendix A

Sample Data for All Three Neural Network Examples (from Judge, et. al. (1988), p. 500))

X1	X2	Y
0.286	0.645	3.284
0.973	0.585	3.149
0.384	0.31	2.877
0.276	0.058	-0.467
0.973	0.455	1.211
0.543	0.779	1.389
0.957	0.259	1.145
0.948	0.202	2.321
0.543	0.028	0.998
0.797	0.099	0.379
0.936	0.142	1.106
0.889	0.296	0.428
0.006	0.175	0.011
0.828	0.18	1.179
0.399	0.842	1.858
0.617	0.039	0.388
0.939	0.103	0.651
0.784	0.62	0.593
0.072	0.158	0.046
0.889	0.704	1.152

Chapter Appendix B

Arrangement of the data from Appendix A in groups by the Self-Organizing Map

Class 1

X1	X2	Y	PE
0.286	0.645	3.284	7
0.384	0.310	2.877	7

Class 2

X1	X2	Y	PE
0.973	0.585	3.149	6
0.889	0.704	1.152	6

Class 3

X1	X2	Y	PE
0.276	0.058	-0.467	24
0.543	0.028	0.998	24
0.797	0.099	0.379	24
0.006	0.175	0.011	24
0.617	0.039	0.388	24
0.072	0.158	0.046	24

Class 4

X1	X2	Y	PE
0.973	0.455	1.211	10
0.957	0.259	1.145	10
0.948	0.202	2.321	10
0.936	0.142	1.106	10
0.889	0.296	0.428	10
0.828	0.180	1.179	10
0.939	0.103	0.651	10

Class 5

X1	X2	Y	PE
0.543	0.779	1.389	16

Class 6

X1	X2	Y	PE
0.399	0.842	1.858	19

Class 7

X1	X2	Y	PE
0.784	0.620	0.593	13

Class 8: Combining P. E.'s 6 and 7

X1	X2	Y	PE
0.286	0.645	3.284	7
0.973	0.585	3.149	6
0.384	0.310	2.877	7
0.889	0.704	1.152	6

Class 9: Combining P. E.'s 6 and 10

X1	X2	Y	PE
0.973	0.585	3.149	6
0.973	0.455	1.211	10
0.957	0.259	1.145	10
0.948	0.202	2.321	10
0.936	0.142	1.106	10
0.889	0.296	0.428	10
0.828	0.180	1.179	10
0.939	0.103	0.651	10
0.889	0.704	1.152	6

Class 10: Combining P. E.'s 6 and 24

X1	X2	Y	PE
0.973	0.585	3.149	6
0.276	0.058	-0.467	24
0.543	0.028	0.998	24
0.797	0.099	0.379	24
0.006	0.175	0.011	24
0.617	0.039	0.388	24
0.072	0.158	0.046	24
0.889	0.704	1.152	6

CHAPTER 9

INTERPRETING NEURAL NETWORK OUTPUT[1]

9.0 Introduction

A popular "podium joke" goes something like this:

Two campers were sitting by the fire in the woods when they spotted a huge aggressive looking bear. Camper #1 frantically grabbed his running shoes and started to put them on. "Are you crazy?" asked Camper #2, "You'll never outrun that bear." Camper #1 replied, "I don't have to outrun the *bear*. I only have to outrun *you*!"

The underlying concept was also expressed more formally by Herbert A. Simon as **satisficing** (Simon 1982), that is, finding solutions that are *good enough* to meet current requirements without an exhaustive search for an *optimal* solution. In this article, we explore the interpretation of the various neural network output signals, including the output unit values, the network error, and the distribution of the output unit values. Emphasis is placed on deriving utility from non-convergent neural networks. In some classification applications, very usable information is available even if the output values diverge substantially from the desired output.

---

[1] Copyright 1993. High-Tech Communications, Inc. Reprinted, with permission, from Advanced Technology for Developers, Vol. 2. (February) pp. 18-23.

9.1 Convergence Isn't Everything

It is common practice to run training iterations until a neural network *converges*, i. e., the root mean square (RMS) error reaches a very close approximation to zero. When convergence is achieved, it is assumed that the network has learned the relationship between the input and the desired output data vectors and will perform reasonably well on new input vectors.

More often than not, a neural network does not reach convergence after a large number of training iterations. At this point, if no other data is available, the developer's choices include alternative transformations of the data, increasing the number of hidden units, selecting a different neural network paradigm, working with various parameters within a network paradigm, and so forth. Given the permutations on the various combinations of these choices, the options quickly become too large to consider in any rational manner.

Neural networks produce several informational signals, all of which can be used to better understand the relationship being modeled. Convergence is only one of the many indications. It is proposed here that networks that do not converge may be usable if their information contents possess other characteristics. A case study using the counterpropagation network (Hecht-Nielsen 1986) will be used to illustrate a methodology for interpreting neural network output.

9.2 Application Background

The data for the case study are from a business-to-business credit approval application. The purpose of the application is to support a credit decision. A *bad* case is defined as one in which a firm has defaulted on its obligation and the funds are determined to be uncollectable. All other cases are classified as *good.* The solution originally proposed was a logistic regression model, which would produce a probability of a case

becoming bad. The logistic regression model did not sufficiently discriminate between goods and bads to meet the user's needs. Specifically, the logistic model produced too many *false positives,* namely, it identified too many good cases as bad, even though it was also very powerful in screening the bad cases.

9.3 The Counterpropagation Network

Since the basic nature of the model is to classify firms as "good" or "bad", the counterpropagation network was chosen because it is a form of nearest neighbor classifier and is well suited to applications in which the class boundaries are non-linear and complex.

The input data vectors had 8 elements, and the network was designed with 15 hidden units. For the training, or model development, data set, there were 167 cases each for good and bad for a total of 334. The network was tested with a data set of 544 cases, comprising 444 bad and 100 good cases. As the owner of the data kept more records on bad cases, there is a larger proportion of bad cases. For measuring results, the good cases are weighted to give the equivalence of a 1% bad rate. This helps assess how the network would perform in the actual working environment.

In order to obtain the most predictive power from the data, various transformations were used. For example, logarithmic transformations were used for data with a wide range such as sales and numbers of employees in order to linearize the relationship between the input (independent) and output (dependent) variables. In addition, ratios such as number of liens/sales and sales/number of employees were used to combine variables and reflect the relative effects of certain circumstances on companies of different sizes. For example, most companies experience some negative events such as a suit, lien, or judgment, but each event may have a relatively greater negative effect on a small company than on a larger company. Furthermore, each data element was converted to the *standard scale* (sometimes called the *z-scale*), which is

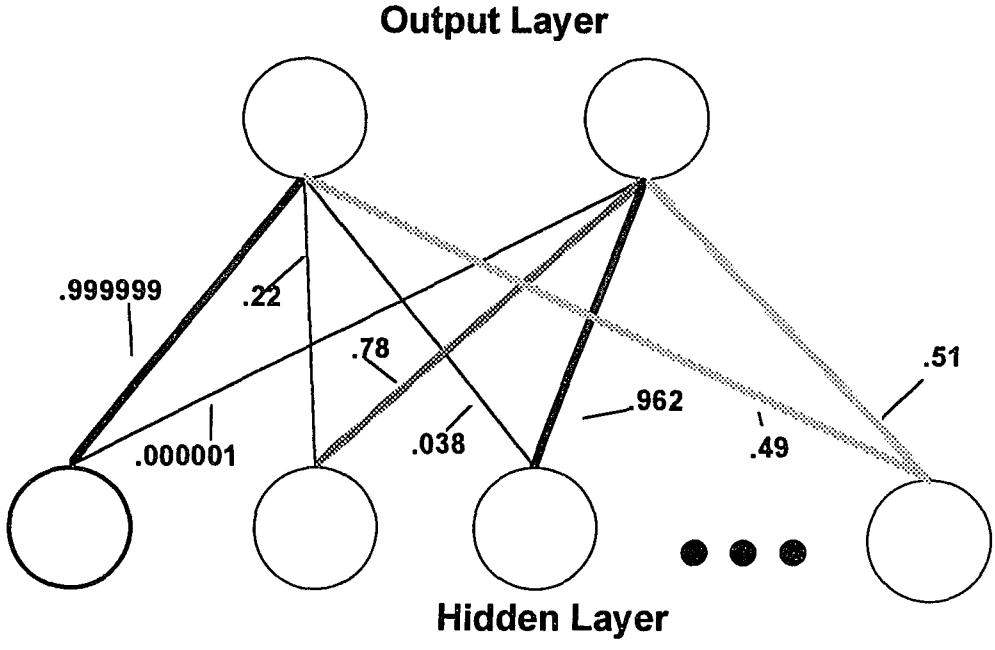
$$z = \frac{x - m_k(x)}{s_k} \tag{9.1}$$

where $m_k(x)$ is the mean of x and s_k is the standard deviation for class k (See (Johnson & Bhattacharyya 1987)). The purpose of the standard score was to enhance the predictive qualities of the data, and, in fact, a network that was identical in every other way except that the data were not expressed in the standard scale did not perform as well.

9.3 Ordered Ranking and Screening Effect

In the counterpropagation architecture, the output training vector contains one element for each class. The element that indicates which class corresponds to the inputs has a value of 1 and all other elements are zeroes. In a well-behaved model, after training, each hidden unit will have a fan-out of weights to the output units that sum to 1, or very close to 1. When the output units receive the one positive signal from the "winning" unit in the hidden layer, each output unit will output a value that equals the weight of its connection from the "winning" hidden unit. Another aspect of model "good behavior" is that the unit with the output value closest to 1 can be interpreted as the most likely correct unit, the value next closest to 1 as the 2nd most likely correct unit, and so forth. The output unit values can be used as a score to produce an ordered ranking. Figure 1 illustrates part of a counterpropagation network, showing examples of connection weights for a two class output layer.

Figure 9-1 -- Connection Weights in Counterpropagation
An Example



In credit models, the ordered ranking is used to assess the "screening effect." The general concept is: if, for example, the 10% of the total number cases receive the "worst" score and 30% of these are actual bad cases , the screening effect is 30% at the 10% level. A counterpropagation model may not produce neat deciles, but nevertheless, can produce a useful screening analysis for bad cases.

The tables below illustrate the "screening effect" concept. As an example, the first detail line from the Testing (Holdout) Sample is interpreted as:

Of the 654.5[2] cases that received the score .999999, there were 116 bads; therefore the score .999999 is separating the goods and bads with 17.72% accuracy. In the 1.21% of

[2] Cases may appear as fractions because the good cases were weighted to make the sample results look more like the actual bad rate of 1%.

the total number of cases that received the .999999 score, 26.13% of the bads were found. Therefore the screening effect is 26.13% at the 1.21% level.

In this example, the overall accuracy may not seem very impressive but the screening effect was over 2000% better than random, because, at a 1% bad rate, only 0.0121 of the cases are expected to be bad at the 1.21% level.

Table 9-1

COUNTERPROPAGATION NEURAL NET (NORMALIZED WITH Z-STATISTIC) (WEIGHTED BASED ON 1% BAD RATE)

TRAINING (DEVELOPMENT) SAMPLE
OVERALL CORRECT RESULTS -> 84.69%

Hidden Unit Value	"BAD" OUTPUT UNIT > 0.5					
	Cumulative				Cumulative Screening Effect	
	Total	Bads	Correct #	Correct %	Total %	Bad %
0.999999	50	50	50	100.00%	0.30%	29.94%
0.740659	152	53	53	34.87%	1.21%	31.74%
0.718556	355	58	58	16.34%	3.34%	34.73%
0.573603	2461	85	85	3.45%	14.74%	50.90%

Hidden Unit Value	"BAD" OUTPUT UNIT <= 0.5					
	Cumulative				Cumulative Screening Effect	
	Total	Bads	Correct #	Correct %	Total %	Bad %
0.434512	3359	92	976	29.06%	20.11%	55.09%
0.416427	13828	166	11272	81.52%	82.80%	99.40%
0.303646	14027	167	11470	81.77%	83.99%	100.00%
0.025295	14126	167	11569	81.90%	84.59%	100.00%
0.001238	14423	167	11866	82.27%	86.37%	100.00%
0.000202	14918	167	12361	82.86%	89.33%	100.00%
-0.000010	15809	167	13252	83.83%	94.66%	100.00%
-0.000103	16502	167	13945	84.50%	98.81%	100.00%
-0.004714	16700	167	14143	84.69%	100.00%	100.00%

TESTING (HOLDOUT) SAMPLE
OVERALL CORRECT RESULTS - 83.77%
>

Hidden Unit Value	"BAD" OUTPUT UNIT > 0.5					
	Cumulative				Cumulative Screening Effect	
	Total	Bads	Correct #	Correct %	Total %	Bad %
0.999999	654.5	116	116	17.72%	1.21%	26.13%
0.740659	2293	139	139	6.06%	4.22%	31.31%
0.718556	2321	167	167	7.20%	4.27%	37.61%
0.573603	8866	250	250	2.82%	16.33%	56.31%

Hidden Unit Value	"BAD" OUTPUT UNIT <= 0.5					
	Cumulative				Cumulative Screening Effect	
	Total	Bads	Correct #	Correct %	Total %	Bad %
0.434512	10492	260	1866	17.78%	19.32%	58.56%
0.416427	44595	438	35791	80.26%	82.14%	98.65%
0.303646	46216	443	37407	80.94%	85.12%	99.77%
0.025295	46754	443	37407	81.16%	85.12%	99.77%
0.001238	47293	443	37407	81.37%	85.12%	99.77%
0.000202	47831	443	37407	81.58%	85.12%	99.77%
-0.00001	50524	443	37407	82.56%	85.12%	99.77%
-0.0001	53755	443	37407	83.61%	85.12%	99.77%
-0.00471	54294	444	45484	83.77%	100.00%	100.00%

9.4 Indications of Non-Linearity or Higher Complexity

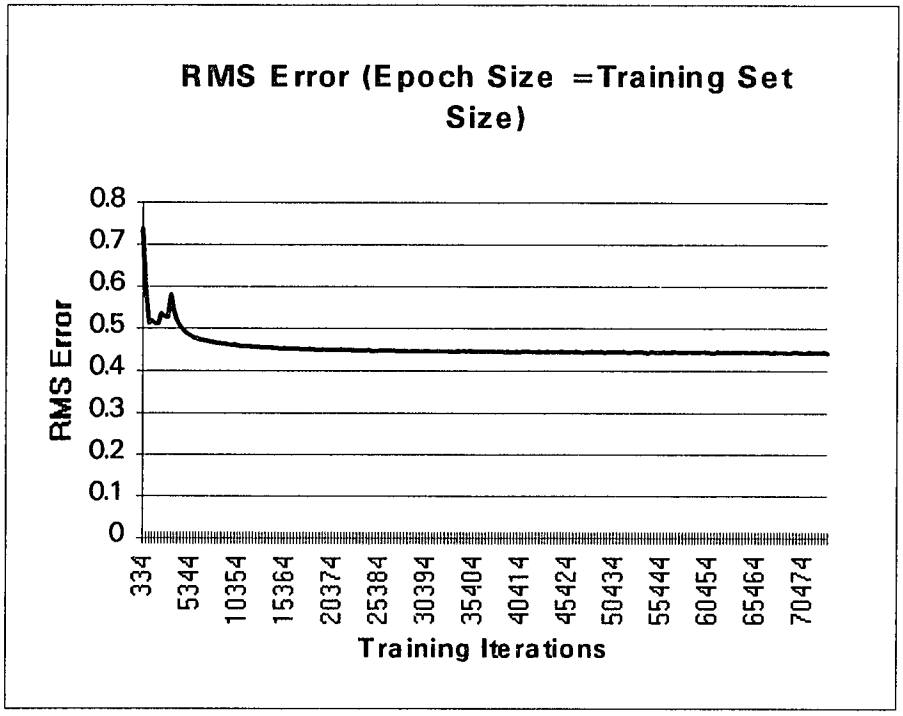
The results of the case study generally were organized to suggest an interpretation that, in a two unit output layer, the output unit with the value above .5 indicates the class membership for the data. In general, the output unit with the highest value (or score) can be interpreted to represent the class predicted by the network. Most of the time, the value of the output units serves as an indicator of the strength of the network's classification, although it cannot be literally interpreted as a probability or a fuzzy set membership.(See (Klimasauskas 1991) for a method of translating output unit values to probabilities). If the output vector of hidden units contains one element with a value very close to 1 and the other units all very close to zero, it can be concluded that the network achieved a high level of discrimination among the classes. If the output values are very close to one another, a low level of discrimination is indicated. If the values of the output units and the percentage of correct classifications do not decrease together monotonically, this indicates a higher degree of non-linearity.

9.5 Root Mean Square Error Graph

The root mean square (RMS) error is generally measured for the output vector error and is the average error over a training epoch. By changing the size of the training epoch, it is possible to obtain different measures of how well the network will discriminate among the classes. For example, if the epoch is set to the size of the training data set, the RMS error is a general indicator of whether or not the network is right more often than it is wrong. Figure 9-2 is a graph of the RMS error for this credit risk application in which the epoch size equals the training set size. In this two-class counterpropagation network, an RMS error of about .44 over the epoch indicates that the network is probably doing a good job of separating the classes. If the error for a particular case is exactly .44 the

output vector will be {.44,.56} instead of {0.1}, which is enough of a difference to indicate class membership. If the error were around .5, it would indicate that the network is not learning to discriminate between the two classes. How well a non-convergent network can discriminate given any RMS level depends on the application, but consider an error level significantly below ***1/n***, where ***n*** is the number of classes, or output units.

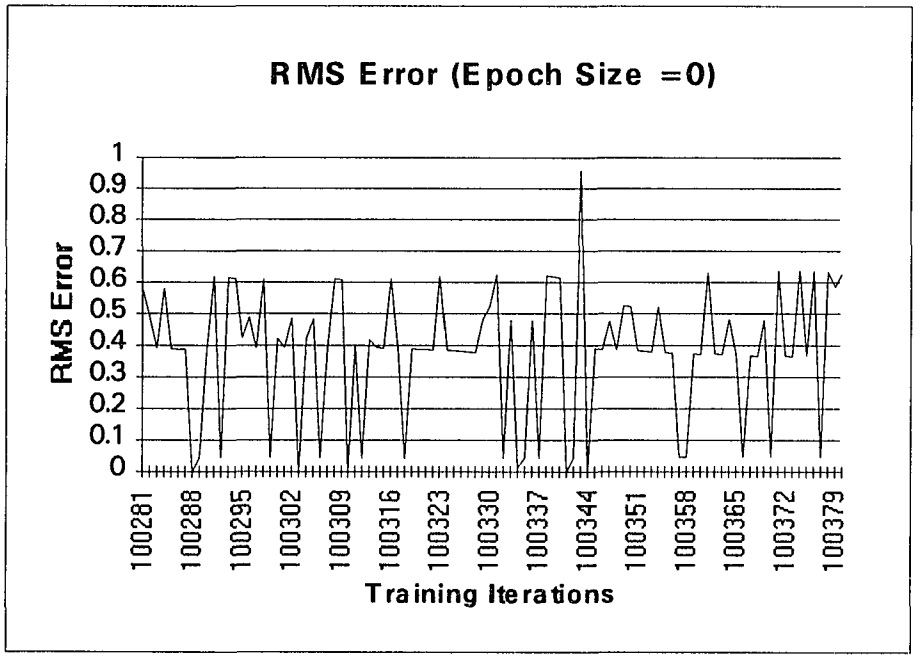
Figure 9-2



Unless the training set is very large, say, more than 10,000 cases, initially setting the epoch size to the training set size will provide some initial feedback of whether or not the network "gets it." Also, it is essential to assure that the RMS error is updated at the completion of each epoch. Once an overall ability to discriminate is determined, the next step is to look at the errors in more detail to see if enough of the cases are classified with a near zero error to provide one or more "high performance" output values that are very accurate and screens enough of a percentage of the cases to be useful. To do this, the

epoch size is set to 1 and the RMS error is also updated after every epoch. Now, the RMS error is really the individual error of the example. Figure 9-3 is a graph of the RMS error after every training example. A significant number of cases had a near zero error, and it is clear that more examples had errors of less than .5 than otherwise. The combination of these two views of the RMS error gives a complete picture of how network learning is progressing.

Figure 9-3



9.6 Summary

When a neural network does not converge, this does not mean that the network hasn't learned anything useful. There are several network signals that can be observed to evaluate the quality of learning. The output unit values are indicators of the strength of the network's discrimination power and certain output values may be extremely accurate for a percentage of the cases. The output values can also be used to produce a ranked

ordering of the results. The RMS error indicates both the networks overall learning and whether high performance output values exist. The example employed herein used the counterpropagation architecture; however these concepts can be applied to other network paradigms that do not constrain output values, such as backpropagation, as well.

APPENDICES

APPENDIX 1

MATHEMATICA[1] COMPUTER PROGRAM FOR THE CONSENSUS-LEARNING COMMITTEE NETWORK

```
(* Housekeeping and Setup Dialogue *)
Print["Consensus-Learning Committee Network"]
Print["Press Enter to Continue... "]
InputString[];

CommRule = Input["Type in the Consensus Size Expression "];
Coeff1 = Input["Type in the Learning Coefficient for Unlearning Bad Decisions "];
Coeff2= Input["Type in the Learning Coefficient for Learning Good Decisions "];
AValue = Input["Type in the Maximum Euclidean distance "];
IterMax = Input["Type in the Maximum Learning Trials "];
OutFile = Input["Type in the Output File Name in quotes "];

Print["Reading data...."];
OpenRead["psycho.dat"]

(* Read in Data into Arrays *)
NNI=ReadList["psycho.dat",Number,RecordLists -> True]
Close["psycho.dat"]

Do[NNInput[x]=NNI[[x,{1,2,3}]],{x,50}]

Do[NNOutput[x]=NNI[[x,{4}]],{x,50}]

(* Initialize Hidden Unit connections to pseudo-random numbers *)
hw[1] = {0.097205,-0.605215,0.854489}

hw[2] = {-0.394815,0.652089,-0.651598}

hw[3] = {0.413601,-0.395491,0.011375}
```

---

[1] Mathematica Version 2.0 for MS-DOS 386/7 (Champaign, IL: Wolfram Research, Inc.)

```
hw[4] = {-0.175763,0.045499,-0.700766}

hw[5] = {0.772962,-0.16631,0.166008}


(* Initialize Counters *)
Bads = 0
Total = 0
Epoch = 1
LearnCount = 0
Correct = 0
Iter = 0
T1 = 0;
T2 = 0;
GC=0;
BC=0;
TOTG=0;
TOTB=0;

OpenAppend[OutFile];

Print[SequenceForm["Consensus Size Rule is ",N[CommRule]]];
Print["Initialization Complete "];
Print["Type NNRun[] to start network training...."]

(* Main Routine to Train Network *)
NNRun[] :=

Block[{z},
While[Epoch == 1,

Epoch = 0;
For[q=1, q<51, q++,
Iter=Iter+1;

(* Choose a set of input and output vectors at random *)
z = Mod[Round[Random[]*50],50];
If[z==0,z=50];
If[z>25,TOTB=TOTB+1,TOTG=TOTG+1];

(* calculate Euclidean distance *)
Do[EuDist[x] = hw[x] - NNInput[z],{x,1,5}];

Do[EuDist[x] = EuDist[x]^2,{x,1,5}];
```

```
Do[Euclidean[x] = Sqrt[Apply[Plus,EuDist[x]]],{x,1,5}];

Do[If[Euclidean[x] > AValue,hwOut[x]=0,hwOut[x]=1],{x,1,5}];


(* Get consensus *)
Consensus = Sum[hwOut[x],{x,1,5}];

If[NNOutput[z]==={0},Bads = Bads + 1];

Total = Total + 1;

OddsRatio := (Bads/Total) / (1 - Bads/Total);

K = OddsRatio * CommRule;
If[K == 0,K = 1];If[K > 5,K = 5];

Do[SortEu[x]=Euclidean[x],{x,5}];

(* Sort Euclidean Distances *)
Do[Hold = SortEu[i];
  SortEu[0] = Hold;
  j = i - 1;
  While[Hold < SortEu[j],
  SortEu[j + 1] = SortEu[j];
  j = j - 1];
  SortEu[j + 1] = Hold,{i,2,5}];

(* Debugging code
Print[SequenceForm[Euclidean[1]," ",Euclidean[2]," ",Euclidean[3]," ",Euclidean[4],"
",Euclidean[5]]];
Print[SequenceForm[SortEu[1],"",SortEu[2],"",SortEu[3],"",SortEu[4],"",SortEu[5]]];
Print[SequenceForm[hwOut[1]," ",hwOut[2]," ",hwOut[3]," ",hwOut[4]," ",hwOut[5]]];
*)

(* Type I error *)
If[Consensus <K && NNOutput[z]=={1},LearnCount = LearnCount + 1;Epoch
= 1;T1=T1+1;
Print[SequenceForm[z," Con=",Consensus," K=",N[K]," Type I"]];
HowMany = Round[K+0.5] - Consensus;y=1;
Do[
    x=1;
    While[SortEu[y]=!=Euclidean[x] && x<=5,x=x+1];
    If[hwOut[x]==0,hw[x]=hw[x] + Coeff2*(NNInput[z] - hw[x]);hwOut[x]=2;
```

```
      HowMany=HowMany-1];
      y=y+1;If[HowMany==0,Break[]],
{v,5}]
];


(* Type II Error *)
If[Consensus >=K && NNOutput[z]=={0},LearnCount = LearnCount + 1;Epoch
= 1;
Print[SequenceForm[z," Con=",Consensus," K=",N[K]," Type II"]];
Do[If[hwOut[x]==1,hw[x]=hw[x] - Coeff1*(NNInput[z] - hw[x])],{x,1,5}];
Do[Print[SequenceForm[x," ",hw[x]]],{x,5}];
T2=T2+1];

(* Correct Response *)
If[Consensus >= K&& NNOutput[z]=={1}||Consensus <K&&
NNOutput[z]=={0},
Print[SequenceForm["Correct for ",z," Con=",Consensus,"
K=",N[K]]];Correct=Correct+1;
If[z>25,BC=BC+1,GC=GC+1]]

];

(* At end of each epoch *)
Print["Summary Statistics "];
Print[SequenceForm["Learn Count = ",LearnCount]];
Print[SequenceForm["Iterations = ",Iter]];
Do[Print[hw[x]],{x,1,5}];
Print[SequenceForm["Correct=",Correct]];
Print[SequenceForm["Type I Errors ",T1]];
Print[SequenceForm["Type II Errors ",T2]];
Print[SequenceForm["Odds Ratio ",N[OddsRatio]]];
Print[SequenceForm["Goods OK ",GC," Bads OK ",BC," Good % ",N[GC/TOTG],"
Bad % ",N[BC/TOTB]]];
Put[SequenceForm[Iter," ",LearnCount," ",Correct," ",T1," ",T2," ",N[OddsRatio],"
",GC," ",BC," ",N[GC/TOTG]," ",N[BC/TOTB]],OutFile];
T1=0;
T2=0;
Correct=0;
If[Iter > IterMax,Break[]]
];
Close[OutFile];
]
```

APPENDIX 2 - - AN EXAMPLE OF A COMPLETE RUN OF THE CONSENSUS-LEARNING COMMITTEE NETWORK (NETWORK TRAINING)

Consensus is 3 (majority)

Run # 1

Iter. Count	Learn Count	Number Correct	Type Errors	Type Errors	Odds Ratio	Goods Correct	Bads Correct	Goods % Correct	Bads % Correct	Type 1 Error %	Type 2 Error %	Total % Correct
50	3	47	2	1	0.785714	26	21	92.86%	95.45%	4.00%	2.00%	94.00%
100	6	47	2	1	0.724138	54	40	93.10%	95.24%	4.00%	2.00%	94.00%
150	7	49	1	0	0.898734	74	69	93.67%	97.18%	2.00%	0.00%	98.00%
200	14	43	2	5	0.923077	97	89	93.27%	92.71%	4.00%	10.00%	86.00%
250	20	44	4	2	0.908397	120	110	91.60%	92.44%	8.00%	4.00%	88.00%
300	24	46	2	2	0.948052	141	135	91.56%	92.47%	4.00%	4.00%	92.00%
350	31	43	2	5	1.023121	158	161	91.33%	90.96%	4.00%	10.00%	86.00%
400	35	46	1	3	1.072539	177	188	91.71%	90.82%	2.00%	6.00%	92.00%
450	39	46	2	2	1.083333	198	213	91.67%	91.03%	4.00%	4.00%	92.00%
500	40	49	0	1	1.127660	217	243	92.34%	91.70%	0.00%	2.00%	98.00%
550	44	46	3	1	1.115385	239	267	91.92%	92.07%	6.00%	2.00%	92.00%
600	47	47	1	2	1.054795	270	283	92.47%	91.88%	2.00%	4.00%	94.00%
650	50	47	2	1	1.044025	294	306	92.45%	92.17%	4.00%	2.00%	94.00%
700	51	49	0	1	1.005731	325	324	93.12%	92.31%	0.00%	2.00%	98.00%
750	53	48	1	1	1.043597	342	355	93.19%	92.69%	2.00%	2.00%	96.00%
800	57	46	2	2	1.067183	360	383	93.02%	92.74%	4.00%	4.00%	92.00%
850	62	45	2	3	1.058111	384	404	92.98%	92.45%	4.00%	6.00%	90.00%
900	65	47	2	1	1.050114	408	427	92.94%	92.62%	4.00%	2.00%	94.00%
950	70	45	4	1	1.038627	431	449	92.49%	92.77%	8.00%	2.00%	90.00%
1000	75	45	1	4	1.040816	454	471	92.65%	92.35%	2.00%	8.00%	90.00%

Run # 2

Iter. Count	Learn Count	Number Correct	Type Errors	Type Errors	Odds Ratio	Goods Correct	Bads Correct	Goods % Correct	Bads % Correct	Type 1 Error %	Type 2 Error %	Total % Correct
50	6	44	2	4	0.612903	29	15	93.55%	78.95%	4.00%	8.00%	88.00%
100	17	39	7	4	1.272727	35	48	79.55%	85.71%	14.00%	8.00%	78.00%
150	22	45	3	2	1.142857	58	70	82.86%	87.50%	6.00%	4.00%	90.00%
200	26	46	2	2	1.061856	83	91	85.57%	88.35%	4.00%	4.00%	92.00%
250	29	47	2	1	0.953125	112	109	87.50%	89.34%	4.00%	2.00%	94.00%
300	31	48	1	1	0.973684	135	134	88.82%	90.54%	2.00%	2.00%	96.00%
350	40	41	5	4	0.955307	157	153	87.71%	89.47%	10.00%	8.00%	82.00%
400	44	46	2	2	1.010050	175	181	87.94%	90.05%	4.00%	4.00%	92.00%
450	51	43	4	3	1.036199	193	206	87.33%	89.96%	8.00%	6.00%	86.00%
500	56	45	2	3	1.083333	210	234	87.50%	90.00%	4.00%	6.00%	90.00%
550	65	41	6	3	1.075472	229	256	86.42%	89.82%	12.00%	6.00%	82.00%
600	71	44	2	4	1.090592	249	280	86.76%	89.46%	4.00%	8.00%	88.00%
650	76	45	2	3	1.056962	276	298	87.34%	89.22%	4.00%	6.00%	90.00%
700	79	47	1	2	1.052786	300	321	87.98%	89.42%	2.00%	4.00%	94.00%
750	81	48	1	1	1.077562	319	350	88.37%	89.97%	2.00%	2.00%	96.00%
800	87	44	3	3	1.077922	340	373	88.31%	89.88%	6.00%	6.00%	88.00%
850	90	47	2	1	1.068127	364	396	88.56%	90.21%	4.00%	2.00%	94.00%
900	94	46	2	2	1.027027	395	411	88.96%	90.13%	4.00%	4.00%	92.00%
950	100	44	3	3	1.029915	416	434	88.89%	90.04%	6.00%	6.00%	88.00%
1000	104	46	2	2	0.996008	447	449	89.22%	89.98%	4.00%	4.00%	92.00%

Run # 3

Iter. Count	Learn Count	Number Correct	Type Errors	Type Errors	Odds Ratio	Goods Correct	Bads Correct	Goods % Correct	Bads % Correct	Type 1 Error %	Type 2 Error %	Total % Correct
50	5	45	3	2	0.785714	25	20	89.29%	90.91%	6.00%	4.00%	90.00%
100	7	48	1	1	0.694915	55	38	93.22%	92.68%	2.00%	2.00%	96.00%
150	14	43	3	4	0.807229	76	60	91.57%	89.55%	6.00%	8.00%	86.00%
200	23	41	6	3	0.886792	93	84	87.74%	89.36%	12.00%	6.00%	82.00%
250	24	49	0	1	0.923077	117	109	90.00%	90.83%	0.00%	2.00%	98.00%
300	29	45	3	2	0.875000	144	127	90.00%	90.71%	6.00%	4.00%	90.00%
350	31	48	1	1	0.851852	172	147	91.01%	91.30%	2.00%	2.00%	96.00%
400	38	43	2	5	0.886792	193	169	91.04%	89.89%	4.00%	10.00%	86.00%
450	43	45	2	3	0.931330	212	195	90.99%	89.86%	4.00%	6.00%	90.00%
500	47	46	3	1	0.945525	233	220	90.66%	90.53%	6.00%	2.00%	92.00%
550	50	47	2	1	0.957295	255	245	90.75%	91.08%	4.00%	2.00%	94.00%
600	57	43	3	4	1.013423	269	274	90.27%	90.73%	6.00%	8.00%	86.00%
650	62	45	2	3	1.044025	287	301	90.25%	90.66%	4.00%	6.00%	90.00%
700	66	46	3	1	1.034884	310	324	90.12%	91.01%	6.00%	2.00%	92.00%
750	76	40	6	4	1.010724	333	341	89.28%	90.45%	12.00%	8.00%	80.00%
800	83	43	3	4	0.995012	358	359	89.28%	89.97%	6.00%	8.00%	86.00%
850	85	48	1	1	0.985981	384	381	89.72%	90.28%	2.00%	2.00%	96.00%
900	92	43	3	4	0.991150	405	403	89.60%	89.96%	6.00%	8.00%	86.00%
950	99	43	3	4	0.991614	427	424	89.52%	89.64%	6.00%	8.00%	86.00%
1000	100	49	1	0	1.004008	448	452	89.78%	90.22%	2.00%	0.00%	98.00%

Run # 4

Iter. Count	Learn Count	Number Correct	Type Errors	Type Errors	Odds Ratio	Goods Correct	Bads Correct	Goods % Correct	Bads % Correct	Type 1 Error %	Type 2 Error %	Total % Correct
50	11	37	8	3	0.851852	19	18	70.37%	78.26%	16.00%	6.00%	74.00%
100	18	43	2	5	0.923077	42	38	80.77%	79.17%	4.00%	10.00%	86.00%
150	22	46	2	2	0.948052	65	61	84.42%	83.56%	4.00%	4.00%	92.00%
200	26	46	2	2	0.980198	87	85	86.14%	85.86%	4.00%	4.00%	92.00%
250	34	42	3	5	1.066116	104	110	85.95%	85.27%	6.00%	10.00%	84.00%
300	40	44	4	2	0.960784	132	126	86.27%	85.71%	8.00%	4.00%	88.00%
350	44	46	1	3	0.977401	155	149	87.57%	86.13%	2.00%	6.00%	92.00%
400	50	44	4	2	0.980198	176	172	87.13%	86.87%	8.00%	4.00%	88.00%
450	58	42	4	4	1.017937	193	197	86.55%	86.78%	8.00%	8.00%	84.00%
500	63	45	4	1	1.000000	216	219	86.40%	87.60%	8.00%	2.00%	90.00%
550	66	47	2	1	0.978417	242	240	87.05%	88.24%	4.00%	2.00%	94.00%
600	67	49	0	1	0.941748	273	258	88.35%	88.66%	0.00%	2.00%	98.00%
650	73	44	4	2	0.934524	296	279	88.10%	88.85%	8.00%	4.00%	88.00%
700	79	44	3	3	0.955307	315	304	87.99%	88.89%	6.00%	6.00%	88.00%
750	81	48	1	1	0.928021	345	322	88.69%	89.20%	2.00%	2.00%	96.00%
800	85	46	2	2	0.923077	370	343	88.94%	89.32%	4.00%	4.00%	92.00%
850	87	48	1	1	0.945080	390	371	89.24%	89.83%	2.00%	2.00%	96.00%
900	91	46	2	2	0.978022	406	401	89.23%	90.11%	4.00%	4.00%	92.00%
950	96	45	2	3	1.000000	424	428	89.26%	90.11%	4.00%	6.00%	90.00%
1000	104	42	4	4	1.004008	444	450	88.98%	89.82%	8.00%	8.00%	84.00%

Run # 5

Iter. Count	Learn Count	Number Correct	Type Errors	Type Errors	Odds Ratio	Goods Correct	Bads Correct	Goods % Correct	Bads % Correct	Type 1 Error %	Type 2 Error %	Total % Correct
50	5	43	4	1	0.666667	26	17	86.67%	85.00%	8.00%	2.00%	86.00%
100	11	44	1	5	0.818182	50	37	90.91%	82.22%	2.00%	10.00%	88.00%
150	19	42	5	3	0.898734	69	60	87.34%	84.51%	10.00%	6.00%	84.00%
200	28	41	5	4	0.960784	87	83	85.29%	84.69%	10.00%	8.00%	82.00%
250	37	41	4	5	0.923077	111	100	85.38%	83.33%	8.00%	10.00%	82.00%
300	42	45	4	1	0.875000	137	119	85.63%	85.00%	8.00%	2.00%	90.00%
350	44	48	0	2	0.861702	165	139	87.77%	85.80%	0.00%	4.00%	96.00%
400	52	42	4	4	0.869159	187	159	87.38%	85.48%	8.00%	8.00%	84.00%
450	57	45	3	2	0.829268	216	175	87.80%	85.78%	6.00%	4.00%	90.00%
500	61	46	1	3	0.858736	238	199	88.48%	86.15%	2.00%	6.00%	92.00%
550	73	38	7	5	0.890034	253	222	86.94%	85.71%	14.00%	10.00%	76.00%
600	77	46	2	2	0.910828	274	247	87.26%	86.36%	4.00%	4.00%	92.00%
650	85	42	4	4	0.928783	293	270	86.94%	86.26%	8.00%	8.00%	84.00%
700	90	45	2	3	0.939058	315	293	87.26%	86.43%	4.00%	6.00%	90.00%
750	94	46	3	1	0.923077	341	313	87.44%	86.94%	6.00%	2.00%	92.00%
800	102	42	4	4	0.960784	355	341	87.01%	86.99%	8.00%	8.00%	84.00%
850	107	45	2	3	0.954023	380	361	87.36%	86.99%	4.00%	6.00%	90.00%
900	110	47	2	1	0.943844	406	382	87.69%	87.41%	4.00%	2.00%	94.00%
950	120	40	5	5	0.950719	425	403	87.27%	87.04%	10.00%	10.00%	80.00%
1000	121	49	0	1	0.960784	448	429	87.84%	87.55%	0.00%	2.00%	98.00%

Run # 6

Iter. Count	Learn Count	Number Correct	Type Errors	Type Errors	Odds Ratio	Goods Correct	Bads Correct	Goods % Correct	Bads % Correct	Type 1 Error %	Type 2 Error %	Total % Correct
50	5	45	3	2	0.923077	23	22	88.46%	91.67%	6.00%	4.00%	90.00%
100	7	48	1	1	0.754386	53	40	92.98%	93.02%	2.00%	2.00%	96.00%
150	8	49	1	0	0.948052	72	70	93.51%	95.89%	2.00%	0.00%	98.00%
200	16	42	2	6	0.904762	98	86	93.33%	90.53%	4.00%	12.00%	84.00%
250	25	41	5	4	0.851852	123	102	91.11%	88.70%	10.00%	8.00%	82.00%
300	29	46	2	2	0.863354	147	124	91.30%	89.21%	4.00%	4.00%	92.00%
350	31	48	0	2	0.923077	168	151	92.31%	89.88%	0.00%	4.00%	96.00%
400	34	47	2	1	0.895735	195	171	92.42%	90.48%	4.00%	2.00%	94.00%
450	40	44	3	3	0.859504	223	187	92.15%	89.90%	6.00%	6.00%	88.00%
500	42	48	1	1	0.923077	240	218	92.31%	90.83%	2.00%	2.00%	96.00%
550	42	50	0	0	0.929825	265	243	92.98%	91.70%	0.00%	0.00%	100.00%

Run # 7

Iter. Count	Learn Count	Number Correct	Type Errors	Type Errors	Odds Ratio	Goods Correct	Bads Correct	Goods % Correct	Bads % Correct	Type 1 Error %	Type 2 Error %	Total % Correct
50	5	45	3	2	0.923077	23	22	88.46%	91.67%	6.00%	4.00%	90.00%
100	7	48	1	1	0.754386	53	40	92.98%	93.02%	2.00%	2.00%	96.00%
150	8	49	1	0	0.948052	72	70	93.51%	95.89%	2.00%	0.00%	98.00%
200	16	42	2	6	0.904762	98	86	93.33%	90.53%	4.00%	12.00%	84.00%
250	25	41	5	4	0.851852	123	102	91.11%	88.70%	10.00%	8.00%	82.00%
300	29	46	2	2	0.863354	147	124	91.30%	89.21%	4.00%	4.00%	92.00%
350	31	48	0	2	0.923077	168	151	92.31%	89.88%	0.00%	4.00%	96.00%
400	34	47	2	1	0.886792	196	170	92.45%	90.43%	4.00%	2.00%	94.00%
450	37	47	1	2	0.859504	225	188	92.98%	90.38%	2.00%	4.00%	94.00%
500	44	43	3	4	0.937984	238	218	92.25%	90.08%	6.00%	8.00%	86.00%
550	49	45	2	3	0.929825	263	238	92.28%	89.81%	4.00%	6.00%	90.00%
600	55	44	4	2	0.892744	291	254	91.80%	89.75%	8.00%	4.00%	88.00%
650	58	47	2	1	0.830986	327	265	92.11%	89.83%	4.00%	2.00%	94.00%
700	59	49	0	1	0.837270	353	288	92.65%	90.28%	0.00%	2.00%	98.00%
750	65	44	2	4	0.884422	368	317	92.46%	90.06%	4.00%	8.00%	88.00%
800	76	39	6	5	0.895735	386	338	91.47%	89.42%	12.00%	10.00%	78.00%
850	77	49	0	1	0.864035	420	353	92.11%	89.59%	0.00%	2.00%	98.00%
900	81	46	2	2	0.855670	447	372	92.16%	89.64%	4.00%	4.00%	92.00%
950	84	47	2	1	0.870079	468	398	92.13%	90.05%	4.00%	2.00%	94.00%
1000	86	48	2	0	0.890359	487	427	92.06%	90.66%	4.00%	0.00%	96.00%

Run # 8

Iter. Count	Learn Count	Number Correct	Type Errors	Type Errors	Odds Ratio	Goods Correct	Bads Correct	Goods % Correct	Bads % Correct	Type 1 Error %	Type 2 Error %	Total % Correct
50	10	37	8	2	1.173913	15	22	65.22%	81.48%	16.00%	4.00%	74.00%
100	16	44	2	4	1.173913	36	45	78.26%	83.33%	4.00%	8.00%	88.00%
150	19	47	1	2	1.083333	61	67	84.72%	85.90%	2.00%	4.00%	94.00%
200	22	47	1	2	0.941748	91	84	88.35%	86.60%	2.00%	4.00%	94.00%
250	28	44	3	3	0.937984	114	105	88.37%	86.78%	6.00%	6.00%	88.00%
300	35	43	3	4	0.986755	133	129	88.08%	86.58%	6.00%	8.00%	86.00%
350	41	44	4	2	1.011494	152	154	87.36%	87.50%	8.00%	4.00%	88.00%
400	46	45	2	3	1.000000	176	175	88.00%	87.50%	4.00%	6.00%	90.00%
450	57	39	6	5	1.036199	191	199	86.43%	86.90%	12.00%	10.00%	78.00%
500	66	41	4	5	1.092050	205	226	85.77%	86.59%	8.00%	10.00%	82.00%
550	73	43	4	3	1.052239	230	244	85.82%	86.52%	8.00%	6.00%	86.00%
600	77	46	2	2	1.083333	248	272	86.11%	87.18%	4.00%	4.00%	92.00%
650	85	42	4	4	1.076677	269	293	85.94%	86.94%	8.00%	8.00%	84.00%
700	94	41	3	6	1.108434	285	318	85.84%	86.41%	6.00%	12.00%	82.00%
750	108	36	8	6	1.094972	303	336	84.64%	85.71%	16.00%	12.00%	72.00%
800	112	46	2	2	1.110818	322	363	84.96%	86.22%	4.00%	4.00%	92.00%
850	118	44	3	3	1.083333	348	381	85.29%	86.20%	6.00%	6.00%	88.00%
900	127	41	5	4	1.088167	366	404	84.92%	86.14%	10.00%	8.00%	82.00%
950	134	43	4	3	1.106430	382	431	84.70%	86.37%	8.00%	6.00%	86.00%
1000	139	45	3	2	1.114165	401	457	84.78%	86.72%	6.00%	4.00%	90.00%

Run # 9

Iter. Count	Learn Count	Number Correct	Type Errors	Type Errors	Odds Ratio	Goods Correct	Bads Correct	Goods % Correct	Bads % Correct	Type 1 Error %	Type 2 Error %	Total % Correct
50	7	41	5	2	1.083333	19	22	79.17%	84.62%	10.00%	4.00%	82.00%
100	9	48	2	0	0.923077	45	44	86.54%	91.67%	4.00%	0.00%	96.00%
150	20	39	4	7	0.898734	68	60	86.08%	84.51%	8.00%	14.00%	78.00%
200	28	42	5	3	0.904762	89	81	84.76%	85.26%	10.00%	6.00%	84.00%
250	35	43	4	3	0.908397	111	102	84.73%	85.71%	8.00%	6.00%	86.00%
300	44	41	3	6	0.923077	133	121	85.26%	84.03%	6.00%	12.00%	82.00%
350	47	47	2	1	0.871658	162	139	86.63%	85.28%	4.00%	2.00%	94.00%
400	49	48	1	1	0.851852	190	159	87.96%	86.41%	2.00%	2.00%	96.00%
450	53	46	3	1	0.875000	211	184	87.92%	87.62%	6.00%	2.00%	92.00%
500	58	45	3	2	0.872659	235	205	88.01%	87.98%	6.00%	4.00%	90.00%
550	63	45	2	3	0.864407	261	224	88.47%	87.84%	4.00%	6.00%	90.00%
600	69	44	3	3	0.875000	283	246	88.44%	87.86%	6.00%	6.00%	88.00%
650	73	46	2	2	0.884058	306	269	88.70%	88.20%	4.00%	4.00%	92.00%
700	75	48	1	1	0.923077	324	299	89.01%	88.99%	2.00%	2.00%	96.00%
750	78	47	2	1	0.958225	341	329	89.03%	89.65%	4.00%	2.00%	94.00%
800	83	45	3	2	0.951220	365	350	89.02%	89.74%	6.00%	4.00%	90.00%
850	91	42	3	5	0.963048	385	372	88.91%	89.21%	6.00%	10.00%	84.00%
900	93	48	1	1	0.952278	412	393	89.37%	89.52%	2.00%	2.00%	96.00%
950	99	44	2	4	0.954733	435	414	89.51%	89.22%	4.00%	8.00%	88.00%
1000	102	47	2	1	0.930502	465	431	89.77%	89.42%	4.00%	2.00%	94.00%

Run # 10

Iter. Count	Learn Count	Number Correct	Type Errors	Type Errors	Odds Ratio	Goods Correct	Bads Correct	Goods % Correct	Bads % Correct	Type 1 Error %	Type 2 Error %	Total % Correct
50	5	43	4	1	0.785714	24	19	85.71%	86.36%	8.00%	2.00%	86.00%
100	13	42	4	4	0.960784	43	42	84.31%	85.71%	8.00%	8.00%	84.00%
150	19	44	3	3	1.027027	63	66	85.14%	86.84%	6.00%	6.00%	88.00%
200	19	50	0	0	0.941748	92	87	89.32%	89.69%	0.00%	0.00%	100.00%

APPENDIX 3
LOGISTIC REGRESSION COMPARISON FOR BOND RATING PROBLEM

In order to compare a neural network method to more traditional statistical analysis methods, the data from the bond rating problem (Chapters 6 and 7) was used for a logistic regression estimation. The LOGIST procedure from the SAS program was used with the SELECTION = S option for a stepwise regression. The stepwise regression option allows variables to be entered into the procedure one at a time, and only variables which meet the significance level criteria of .05 or less will stay in the model. Variables may be removed from the model during estimation if other variables entered later cause the earlier variable's significance levels to rise above the 0.05 level.

Of the 87 independent variables, only 9 were chosen by the LOGIST procedure for the final model. The completed model estimation begins on the bottom of page 224 and continues to page 225. The variables in the model are:

SP3Y - Previous year's S & P Rating

OPINC3Y - Previous year's operating income

DEFTAX2Y - Current year's deferred tax

WRKCAP2Y - Current year's working capital

COMEQU2Y - Current year's common equity

COMEQU3Y - Previous year's common equity

COMEQU4Y - Two years ago common equity

BEFTAX2Y - Current before tax income.

As there were 17 dependent variable categories, 17 intercepts were created. Therefore, in order to test the model, each observation has to be run though the model 17 times, once with each intercept value representing each category. The category with the highest probability is the best estimation.

The results from the logistic regression were unusable. The model assigned just about all observations multiple categories by having equal probability values. In addition, these categories did not sort out in three general classes of investment, speculative, and poor grade bonds. In examining the results of the estimation procedure, the failure of the logistic regression procedure is understandable. The intercept coefficients are magnitudes higher than the variable coefficients, indicating that the squared errors were very high. In addition, all of the measures of goodness of fit had very low values.

Further conditioning of the data would probably improve the results, but also there would be a cost of time and effort. Logistic regression, as implemented in SAS, is essentially a non-linear transformation of a multivariate linear regression, and the data must be linearized to the extent possible to get the best results.

APPENDIX 3 (continued) -- LOGISTIC REGRESSION RESULTS FOR A BOND RATING MODEL

The SAS System 07:45 Friday, April 9, 1993 1
The LOGISTIC Procedure

Data Set: BOND.BONDDEVL
Response Variable: SP2Y
Response Levels: 17
Number of Observations: 415
Link Function: Logit

Response Profile

Ordered Value	SP2Y	Count
1	2	20
2	4	10
3	5	40
4	6	26
5	7	40
6	8	40

The SAS System 07:45 Friday, April 9, 1993 2

The LOGISTIC Procedure

Response Profile

Ordered Value	SP2Y	Count
7	9	35
8	10	30
9	11	30
10	12	20
11	13	15
12	14	10
13	15	10
14	16	15
15	17	20
16	18	30
17	19	24

WARNING: 1 observation(s) were deleted due to missing values for the response or explanatory variables.

The LOGISTIC Procedure

Simple Statistics for Explanatory Variables

Variable	Mean	Standard Deviation	Minimum	Maximum
SIC	4.475325	1.533367	0.100	8.060
SP3Y	10.195181	4.996905	2.000	24.000
SP4Y	9.853012	4.697629	2.000	22.000
SALES2Y	3.075805	0.693273	0.000	5.008
SALES3Y	3.047315	0.669153	0.184	5.012
SALES4Y	3.028347	0.678545	0.195	4.984
OPINC2Y	2.025343	1.092715	-2.372	3.929
OPINC3Y	1.941089	1.147686	-2.688	3.918
OPINC4Y	1.934728	1.125909	-2.427	4.016
PTINC2Y	1.467417	1.538979	-3.052	3.893

The LOGISTIC Procedure

Simple Statistics for Explanatory Variables

Variable	Mean	Standard Deviation	Minimum	Maximum
PTINC3Y	1.404070	1.549325	-3.513	3.945
PTINC4Y	1.385393	1.501506	-2.932	3.991
TAXRTE2Y	1.250360	0.893870	-3.805	2.872
TAXRTE3Y	1.329971	0.814582	-3.334	3.434
TAXRTE4Y	1.277725	0.876583	-3.422	4.588
INCFOR2Y	1.273812	1.510286	-3.090	3.685
INCFOR3Y	1.192880	1.498121	-3.517	3.729
INCFOR4Y	1.197800	1.443889	-2.891	3.688
DEPR2Y	1.909189	0.702649	-0.226	3.786
DEPR3Y	1.875267	0.707493	-0.650	3.819
DEPR4Y	1.820051	0.726238	-0.554	3.793
DEFTAX2Y	0.553752	0.984229	-2.562	2.822
DEFTAX3Y	0.560727	1.067210	-2.616	3.040

The LOGISTIC Procedure

Simple Statistics for Explanatory Variables

Variable	Mean	Standard Deviation	Minimum	Maximum
DEFTAX4Y	0.656493	1.051592	-2.975	3.070
CAPEXP2Y	1.933390	0.848897	-0.790	3.849
CAPEXP3Y	1.953327	0.821033	-1.301	4.069
CAPEXP4Y	1.969442	0.840504	-1.137	3.963
CURRAT2Y	1.612495	0.853741	0.366	8.908
CURRAT3Y	1.668800	0.918887	0.283	12.214
CURRAT4Y	1.789464	1.062918	0.352	11.349
INVTRN2Y	0.937519	0.395569	-0.416	2.116
INVTRN3Y	0.923769	0.403312	-0.936	2.155
INVTRN4Y	0.920999	0.400300	-1.004	2.059
RECTRN2Y	0.868261	0.338248	-1.108	2.272
RECTRN3Y	0.871829	0.330118	-1.119	2.229
RECTRN4Y	0.879083	0.340043	-0.823	2.588

The LOGISTIC Procedure

Simple Statistics for Explanatory Variables

Variable	Mean	Standard Deviation	Minimum	Maximum
WRKCAP2Y	0.532024	1.054861	-2.486	4.089
WRKCAP3Y	0.323733	1.124512	-3.567	3.186
WRKCAP4Y	0.418193	1.072197	-3.185	3.413
CURDBT2Y	1.783267	1.056112	-1.194	4.707
CURDBT3Y	1.767308	1.051056	-1.149	4.707
CURDBT4Y	1.689590	1.054924	-1.337	4.630
LTDEBT2Y	2.597877	0.664805	0.000	4.490
LTDEBT3Y	2.553398	0.670773	-0.545	4.409
LTDEBT4Y	2.511512	0.627113	0.000	4.281
MININT2Y	0.606303	0.811065	-1.523	3.493
MININT3Y	0.568244	0.779961	-1.118	3.425
MININT4Y	0.505744	0.730197	-1.102	3.300
PRFSTK2Y	0.656493	1.054457	-3.000	3.212

The LOGISTIC Procedure

Simple Statistics for Explanatory Variables

Variable	Mean	Standard Deviation	Minimum	Maximum
PRFSTK3Y	0.642463	1.049883	-3.000	3.148
PRFSTK4Y	0.683732	1.056536	-3.000	3.164
COMEQU2Y	2.496915	1.235757	-3.312	4.527
COMEQU3Y	2.578320	1.006107	-3.410	4.505
COMEQU4Y	2.622770	0.812696	-2.761	4.467
WRKCPB2Y	1.845653	1.535540	-2.993	4.154
WRKCPB3Y	1.840962	1.487468	-2.884	3.917
WRKCPB4Y	1.908104	1.359770	-3.399	3.735
QUICK2Y	1.026518	0.646794	0.114	8.901
QUICK3Y	1.062602	0.742051	0.193	11.857
QUICK4Y	1.149412	0.864881	0.118	10.587
PENS2Y	0.117332	0.358835	-0.296	1.195
PENS3Y	0.203747	0.747972	-1.073	2.910

The LOGISTIC Procedure

Simple Statistics for Explanatory Variables

Variable	Mean	Standard Deviation	Minimum	Maximum
PENS4Y	0.116354	0.621918	-1.674	2.765
WRKCPL2Y	1.288303	4.249754	-20.680	64.155
WRKCPL3Y	0.152342	18.692998	-367.700	31.720
WRKCPL4Y	1.719439	7.404606	-1.118	146.810
TAXITC2Y	1.623727	1.071812	-0.848	4.074
TAXITC3Y	1.631011	1.037893	-0.854	4.035
TAXITC4Y	1.617880	1.013058	-0.602	4.043
NETINC2Y	1.304734	1.495057	-3.056	3.685
NETINC3Y	1.233153	1.481412	-3.512	3.729
NETINC4Y	1.236345	1.424815	-2.865	3.688
DIVPAY2Y	1.067573	1.067750	-2.428	4.439
DIVPAY3Y	1.151278	1.052226	-2.244	4.036
DIVPAY4Y	1.042327	1.106130	-3.287	3.273

The SAS System 07:45 Friday, April 9, 1993 9

The LOGISTIC Procedure

Simple Statistics for Explanatory Variables

Variable	Mean	Standard Deviation	Minimum	Maximum
LTDASS2Y	0.273888	0.217987	0.000	2.049
LTDASS3Y	0.267708	0.208607	0.000	2.144
LTDASS4Y	0.260602	0.199483	0.000	2.200
EXPGPP2Y	0.106493	0.071457	0.000	0.904
EXPGPP3Y	0.116459	0.071673	0.000	0.889
EXPGPP4Y	0.130298	0.080642	0.000	0.832
BEFTAX2Y	0.439969	0.447787	-1.308	2.504
BEFTAX3Y	0.393487	0.476102	-1.178	2.605
BEFTAX4Y	0.408308	0.511537	-1.831	2.778
LTDNPP2Y	-0.069086	0.604068	-2.029	2.771
LTDNPP3Y	-0.068458	0.659102	-2.074	4.555
LTDNPP4Y	-0.108176	0.560481	-2.096	2.632

The SAS System 07:45 Friday, April 9, 1993 10

The LOGISTIC Procedure

Stepwise Selection Procedure

Step 0. Intercepts entered:

Residual Chi-Square = 389.4497 with 87 DF (p=0.0001)

Step 1. Variable SP3Y entered:

Score Test for the Proportional Odds Assumption

Chi-Square = 205.4443 with 15 DF (p=0.0001)

The SAS System 07:45 Friday, April 9, 1993 11

The LOGISTIC Procedure

Criteria for Assessing Model Fit

Criterion	Intercept Only	Intercept and Covariates	Chi-Square for Covariates
AIC	2306.957	1144.387	.
SC	2371.409	1212.867	.
-2 LOG L	2274.957	1110.387	1164.570 with 1 DF (p=0.0001)
Score	.	.	365.080 with 1 DF (p=0.0001)

Residual Chi-Square = 116.6181 with 86 DF (p=0.0156)

The SAS System 07:45 Friday, April 9, 1993
12

The LOGISTIC Procedure

Step 2. Variable PTINC2Y entered:

Score Test for the Proportional Odds Assumption

Chi-Square = 285.0495 with 30 DF (p=0.0001)

The SAS System 07:45 Friday, April 9, 1993
13

The LOGISTIC Procedure

Criteria for Assessing Model Fit

Criterion	Intercept Only	Intercept and Covariates	Chi-Square for Covariates
AIC	2306.957	1122.347	.
SC	2371.409	1194.856	.
-2 LOG L	2274.957	1086.347	1188.610 with 2 DF (p=0.0001)
Score	.	.	366.504 with 2 DF (p=0.0001)

Residual Chi-Square = 106.6114 with 85 DF (p=0.0565)

The SAS System 07:45 Friday, April 9, 1993
14

The LOGISTIC Procedure

Step 3. Variable COMEQU2Y entered:

Score Test for the Proportional Odds Assumption

Chi-Square = 313.7073 with 45 DF (p=0.0001)

The SAS System 07:45 Friday, April 9, 1993
15

The LOGISTIC Procedure

Criteria for Assessing Model Fit

Criterion	Intercept Only	Intercept and Covariates	Chi-Square for Covariates
AIC	2306.957	1115.320	.
SC	2371.409	1191.858	.
-2 LOG L	2274.957	1077.320	1197.636 with 3 DF (p=0.0001)
Score	.	.	366.801 with 3 DF (p=0.0001)

Residual Chi-Square = 106.8179 with 84 DF (p=0.0472)

The SAS System 07:45 Friday, April 9, 1993
16

The LOGISTIC Procedure

Step 4. Variable COMEQU3Y entered:

Score Test for the Proportional Odds Assumption

Chi-Square = 340.4809 with 60 DF (p=0.0001)

The SAS System 07:45 Friday, April 9, 1993
17

The LOGISTIC Procedure

Criteria for Assessing Model Fit

Criterion	Intercept Only	Intercept and Covariates	Chi-Square for Covariates
AIC	2306.957	1089.238	.
SC	2371.409	1169.803	.
-2 LOG L	2274.957	1049.238	1225.719 with 4 DF (p=0.0001)
Score	.	.	370.330 with 4 DF (p=0.0001)

Residual Chi-Square = 90.9059 with 83 DF (p=0.2590)

The SAS System 07:45 Friday, April 9, 1993
18

The LOGISTIC Procedure

Step 5. Variable DEFTAX2Y entered:

Score Test for the Proportional Odds Assumption

Chi-Square = 389.6003 with 75 DF (p=0.0001)

The SAS System 07:45 Friday, April 9, 1993
19

The LOGISTIC Procedure

Criteria for Assessing Model Fit

Criterion	Intercept Only	Intercept and Covariates	Chi-Square for Covariates
AIC	2306.957	1082.995	.
SC	2371.409	1167.589	.
-2 LOG L	2274.957	1040.995	1233.961 with 5 DF (p=0.0001)
Score	.	.	370.641 with 5 DF (p=0.0001)

Residual Chi-Square = 85.4850 with 82 DF (p=0.3744)

The LOGISTIC Procedure

Step 6. Variable BEFTAX2Y entered:

Score Test for the Proportional Odds Assumption

Chi-Square = 453.4043 with 90 DF (p=0.0001)

The LOGISTIC Procedure

Criteria for Assessing Model Fit

Criterion	Intercept Only	Intercept and Covariates	Chi-Square for Covariates
AIC	2306.957	1075.853	.
SC	2371.409	1164.475	.
-2 LOG L	2274.957	1031.853	1243.104 with 6 DF (p=0.0001)
Score	.	.	372.559 with 6 DF (p=0.0001)

Step 7. Variable PTINC2Y is removed:

Score Test for the Proportional Odds Assumption

The LOGISTIC Procedure

Chi-Square = 410.5321 with 75 DF (p=0.0001)

Criteria for Assessing Model Fit

Criterion	Intercept Only	Intercept and Covariates	Chi-Square for Covariates
AIC	2306.957	1074.463	.
SC	2371.409	1159.057	.
-2 LOG L	2274.957	1032.463	1242.494 with 5 DF (p=0.0001)
Score	.	.	372.505 with 5 DF (p=0.0001)

Residual Chi-Square = 80.0398 with 82 DF (p=0.5407)

The LOGISTIC Procedure

Step 8. Variable WRKCAP2Y entered:

Score Test for the Proportional Odds Assumption

Chi-Square = 442.6367 with 90 DF (p=0.0001)

The LOGISTIC Procedure

Criteria for Assessing Model Fit

Criterion	Intercept Only	Intercept and Covariates	Chi-Square for Covariates
AIC	2306.957	1070.875	.
SC	2371.409	1159.498	.
-2 LOG L	2274.957	1026.875	1248.081 with 6 DF (p=0.0001)
Score	.	.	373.316 with 6 DF (p=0.0001)

Residual Chi-Square = 75.7570 with 81 DF (p=0.6437)

The LOGISTIC Procedure

Step 9. Variable COMEQU4Y entered:

Score Test for the Proportional Odds Assumption

Chi-Square = 465.7193 with 105 DF (p=0.0001)

The LOGISTIC Procedure

Criteria for Assessing Model Fit

Criterion	Intercept Only	Intercept and Covariates	Chi-Square for Covariates
AIC	2306.957	1067.730	.
SC	2371.409	1160.381	.
-2 LOG L	2274.957	1021.730	1253.226 with 7 DF (p=0.0001)
Score	.	.	373.967 with 7 DF (p=0.0001)

Residual Chi-Square = 71.5446 with 80 DF (p=0.7390)

The LOGISTIC Procedure

Step 10. Variable OPINC3Y entered:

Score Test for the Proportional Odds Assumption

Chi-Square = 497.3024 with 120 DF (p=0.0001)

The SAS System 07:45 Friday, April 9, 1993
28

The LOGISTIC Procedure

Criteria for Assessing Model Fit

Criterion	Intercept Only	Intercept and Covariates	Chi-Square for Covariates
AIC	2306.957	1062.593	.
SC	2371.409	1159.272	.
-2 LOG L	2274.957	1014.593	1260.363 with 8 DF (p=0.0001)
Score	.	.	374.477 with 8 DF (p=0.0001)

Residual Chi-Square = 64.7679 with 79 DF (p=0.8759)

NOTE: No (additional) variables met the 0.05 significance level for entry into the model.

The SAS System 07:45 Friday, April 9, 1993
29

The LOGISTIC Procedure

Summary of Stepwise Procedure

Step	Variable Entered	Variable Removed	Number In	Score Chi-Square	Wald Chi-Square	Pr > Chi-Square
1	SP3Y		1	365.1	.	0.0001
2	PTINC2Y		2	22.7571	.	0.0001
3	COMEQU2Y		3	7.7714	.	0.0053
4	COMEQU3Y		4	20.1036	.	0.0001
5	DEFTAX2Y		5	9.0982	.	0.0026
6	BEFTAX2Y		6	9.8938	.	0.0017
7		PTINC2Y	5	.	0.5632	0.4530
8	WRKCAP2Y		6	6.2456	.	0.0125
9	COMEQU4Y		7	5.5482	.	0.0185
10	OPINC3Y		8	6.7141	.	0.0096

The SAS System 07:45 Friday, April 9, 1993
30

The LOGISTIC Procedure

Analysis of Maximum Likelihood Estimates

Variable	Parameter Estimate	Standard Error	Wald Chi-Square	Pr > Chi-Square	Standardized Estimate
INTERC1	4.7888	0.8378	32.6690	0.0001	.
INTERC2	6.6735	0.8273	65.0656	0.0001	.
INTERC3	9.7576	0.8547	130.3416	0.0001	.
INTERC4	11.5194	0.9036	162.5207	0.0001	.
INTERC5	13.7792	0.9669	203.1068	0.0001	.
INTERC6	15.8524	1.0321	235.9014	0.0001	.
INTERC7	17.7729	1.1055	258.4585	0.0001	.
INTERC8	19.7846	1.1895	276.6667	0.0001	.
INTERC9	22.0942	1.2792	298.3218	0.0001	.
INTERC10	24.5165	1.4043	304.8048	0.0001	.
INTERC11	26.7341	1.5246	307.4936	0.0001	.
INTERC12	28.1502	1.5926	312.4300	0.0001	.
INTERC13	29.5888	1.6548	319.7155	0.0001	.

The LOGISTIC Procedure

Analysis of Maximum Likelihood Estimates

Variable	Parameter Estimate	Standard Error	Wald Chi-Square	Pr > Chi-Square	Standardized Estimate
INTERC14	31.6836	1.7311	334.9941	0.0001	.
INTERC15	33.7970	1.8013	352.0361	0.0001	.
INTERC16	37.5218	1.9396	374.2520	0.0001	.
SP3Y	-1.9418	0.0974	397.1949	0.0001	-5.349497
OPINC3Y	0.3111	0.1139	7.4607	0.0063	0.196851
DEFTAX2Y	0.3310	0.1076	9.4563	0.0021	0.179613
WRKCAP2Y	-0.2413	0.0937	6.6323	0.0100	-0.140318
COMEQU2Y	1.3320	0.1735	58.9144	0.0001	0.907535
COMEQU3Y	-1.0526	0.2482	17.9909	0.0001	-0.583847
COMEQU4Y	-0.5517	0.2240	6.0682	0.0138	-0.247218
BEFTAX2Y	1.0376	0.2476	17.5568	0.0001	0.256168

The LOGISTIC Procedure

Association of Predicted Probabilities and Observed Responses

Concordant	= 36.8%	Somers' D	= 0.365
Discordant	= 0.3%	Gamma	= 0.983
Tied	= 62.8%	Tau-a	= 0.341
(80149 pairs)		c	= 0.683

REFERENCE LIST

Akerlof, George A. 1986. Labor Contracts as Partial Gift Exchange. Efficiency Models of the Labor Market. George A. Akerlof and Janet Yellen, eds., 66-92. Cambridge: Cambridge U. Press.

Akerlof, George A. and Janet Yellen, eds. 1986. Introduction. Efficiency Models of the Labor Market. George A. Akerlof and Janet Yellen, eds., 1-21. Cambridge: Cambridge U. Press.

Albin, Peter S. 1975. The Analysis of Complex Socioeconomic Systems. Lexington Books. New York: D. C. Heath & Co.

Albin, Peter S. 1978. Measurable Complexity in Economic Systems. Sozialwissenschaftliche Annalen 2: 93-106. Vienna: Physica-Verlag.

Albin, Peter S. 1980. The Complexity of Social Groups and Social Systems Described by Graph Structures. Mathematical Social Sciences 1:101-129.

Albin, Peter S. 1983. Structural Theory and Structural Formations. Mathematical Social Sciences 6, no. 2 (November): 133-152.

Albin, Peter S. and Duncan K. Foley. 1992. Decentralized, dispersed exchange without an auctioneer. Journal of Economic Behavior and Organization 18: 27-51. Amsterdam: North-Holland.

Anderson, James A., and Edward Rosenfeld, eds. 1990. Neurocomputing: Foundations of Research. Cambridge, MA: The MIT Press.

Anderson, James A., 1990. General Introduction, and his introduction to William James' paper. Neurocomputing: Foundations of Research. James A. Anderson and Edward Rosenfeld, eds., xiii-xxi, 1-3. Cambridge, MA: The MIT Press.

Arbib, Michael A. 1987. Brains, Machines, and Mathematics. 2d ed. New York: Springer-Verlag.

Arrow, Kenneth J. 1974. The Limits of Organization. New York: W. W. Norton & Co.

Barro, Robert J. 1987. Macroeconomics. 2d ed. New York: John Wiley & Sons, Inc.

Carpenter, Gail A. 1989. Neural Network Models for Pattern Recognition and Associative Memory. Neural Networks 2: 243-257.

Cotman, Carl W., and James L. McGaugh. 1980. Behavioral Neuroscience: an Introduction. New York: Academic Press, Inc.

DeValois, Russell L. and Karen K. DeValois. 1975. Neural Coding of Color. Handbook of Perception: Volume V-Seeing. Edward C. Carterette and Morton P. Friedman, eds., 91-117. New York: Academic Press, Inc.

Dutta, Soumitra, and Shashi Shekhar. 1988. Bond Rating: A Non-conservative Application of Neural Networks. Proceedings of the IEEE International Cnference on Neural Networks. San Diego, 1988. 2 (July): 443-450. Los Alamitos, CA: IEEE Computer Society Press.

Dutta, Soumitra, Shashi Shekhar, and W. Y. Wong. 1993. Decision Support in Non-Conservative Domains: Generalization with Neural Networks. Decision Support Systems. forthcoming.

Elster, Jon. 1989. Nuts and Bolts for the Social Sciences. Cambridge: Cambridge University Press.

Fisher, Douglas, and Pat Langley. 1986. Conceptual Clustering and Its Relation to Numerical Taxonomy. Artificial Intelligence and Statistics. William A. Gale, ed., 77-116. Reading, MA: Addison-Wesley.

Fukushima, K., S. Miyake, and T. Ito. 1983. Neocognitron: a neural network model for a mechanism of visual pattern recognition. IEEE Transactions on Systems, Man, and Cybernetics. SMC 13:826-834, reprinted in Anderson and Rosenfeld.

Gordon, A. D. 1981. Classification. London: Chapman & Hall.

Granger, C. W. J. 1969. Investigating Causal Relations By Econometric Models and Cross-Spectral Methods. Econometrica 37, no. 3. (July): 424-38.

Grossberg, Stephen. 1988. Nonlinear Neural Networks: Principles, Mechanisms, and Architectures. Neural Networks 1, no. 1:17-61. Elmsford, NY: Pergamon Press.

Harvey, A. C. 1981. Time Series Models. London: Philip Allan.

Hecht-Nielsen, Robert. 1990. Neurocomputing. Reading, MA: Addison-Wesley.

Hornik, Kurt, Maxwell Stinchcombe and Halbert White. 1989. Multilayer Feedforward Networks are Universal Approximators. Neural Networks 2: 359-366. Elmsford, NY: Pergamon Press.

Johnson, Richard, and G. Bhattacharyya. 1987. Statistics Principles and Methods. New York: John Wiley & Sons, Inc.

Johnston, J. 1984. Econometric Methods. 3d ed. New York: McGraw-Hill.

Judge, George G., R. Carter Hill, William E. Griffiths, Helmut, Luetkepohl, and Tsoung-Chao Lee. 1988. Introduction to the Theory and Practice of Econometrics. 2d ed. New York: John Wiley & Sons, Inc.

Kendall, Sir Maurice. 1980. Multivariate Analysis. 2d ed. London and High Wyncombe: Charles Griffin & Company, Limited.

Kennedy, Peter. 1985. A Guide to Econmetrics. 2d ed. Cambridge, MA: The MIT Press.

Klimasauskas, C. 1990. Reference Guide to NeuralWorks Professional II/Plus and NeuralWorks Explorer. Pittsburgh, PA: NeuralWare, Inc., Technical Publications Group.

Klimasauskas, C. 1991. Applying Neural Networks: Part VI - Special Topics. PC-AI Magazine Nov.-Dec.: 46.

Knight, Frank H. 1933. Social Economic Organization. (reprinted from The Economic Organization by Frank H. Knight, Harper & Row, New York, 1933) in Readings in Microeconomics, ed. by W. Breit, H. M. Hochman, and E. Saueracker. 1-13. St. Louis: Times Mirror/Mosby College Publishing. 1986.

Kohonen, Teuvo. 1989. Self-Organization and Associative Memory. 3d ed. Berlin: Springer-Verlag.

Kornai, Janos. 1971. Anti-Equilibrium: on economic systems theory and the tasks of research. Amsterdam: North-Holland.

Kosko, Bart. 1992. Neural Networks and Fuzzy Systems. Englewood Cliffs, NJ: Prentice-Hall.

Kuan, Chung-Ming and Halbert White. 1991. Artificial Neural Networks: An Econometric Perspective. working paper. August, 1991.

Kuan, Chung-Ming and Halbert White. 1989. Predicting Appliance Ownership Using Logit, Neural Network, and Regression Tree Models. working paper, October, 1989.

Lapedes, A., and R. Farber. 1987. Non-linear Signal Processing Using Neural Networks: Prediction and System Modeling. Los Alamos National Laboratory report LA-UR-87-2662. Los Alamos, NM.

Layard, P. R. G., and A. A. Walters. 1978. Microeconomic Theory. New York: McGraw-Hill.

Leontief, Wassily. 1947. Introduction to a Theory of the Internal Structure of Functional Relationships. Econometrica 15, no. 4. (October):361-373.

Lucas, Robert E., Jr. 1973. Some International Evidence on Output-Inflation Tradeoffs. The American Economic Review 63, no. 3:326-334.

Mac Lane, Saunders. 1986. Mathematics: Form and Function. New York: Springer-Verlag.

MacQueen, J. 1967. Some methods for classification and analysis of multivariate observations. Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability 1:281-297. Berkeley and Los Angeles: U. of California Press.

Maddala, G. S. 1983. Limited-Dependent and Qualitative Variables in Econometrics. Cambridge: Cambridge U. Press.

McCulloch, Warren S., and Walter Pitts. 1943. A logical calculus of the ideas immanent in nervous activity. Bulletin of Mathematical Biophysics 5: 115-133. Reprinted in (Anderson and Rosenfeld 1990).

Minsky, Marvin L. and Seymour A. Papert. 1988. Perceptrons. expanded edition. Cambridge, MA: The Mit Press.

Munier, Bertrand. 1989. Cognition and Uncertainty. Theory and Decision 27: 93-106.

Narendra, Kumpati S., and Mandayam A. L. Thathachar. 1989. Learning Automata: an Introduction. Englewood Cliffs, NJ: Prentice-Hall.

Neftci, Salih N. 1985. Testing Non-Linearity in Business Cycles. Competition, Instability, and Nonlinear Cycles. Proceedings of an International Conference. 324-340. New School for Social Research. Willi Semmler, ed. Berlin: Springer-Verlag.

NeuralWare, Inc. 1990. Reference Guide to NeuralWorks Professional II/Plus and NeuralWorks Explorer. Pittsburgh, PA: NeuralWare, Inc., Technical Publications Group.

Nilsson, Nils J. 1990. The Mathematical Foundations of Learning Machines. (Previously published as: Learning Machines. 1965.) San Mateo, CA: Morgan Kaufman Publishers, Inc.

Philips, John L., Jr. 1992. How to Think About Statistics. rev. ed. New York: Wm. H.Freeman & Co.

Radner, Roy. 1992. Hierarchy: The Economics of Managing. Journal of Economic Literature. (Sept).

Radner, Roy and Timothy Van Zandt. 1992. Information Processing in Firms and Returns to Scale. Annales D'Economie et de Statistique, no. 25/26.

Rich, Elaine. 1983. Artificial Intelligence. New York: McGraw-Hill.

Ritter, Helge, T. Martinetz, and K. Schulten. 1992. Neural Computation and Self-Organizing Maps. Reading, MA: Addison-Wesley.

Rumelhart, David E., G. E. Hinton, and R. J. Williams. 1986. Learning internal representations by error propagation. Parallel Distributed Processing: Explorations in the Microstructures of Cognition. Vol. 1. D. E. Rumelhart and J. L. McClelland, eds. Cambridge, MA: The MIT Press.

Rudin, Walter. 1964. Principles of Mathematical Analysis. New York: McGraw-Hill.

Sah, Rah Kumar, and Joseph E. Stiglitz. 1986. The architecture of economic systems: hierarchies and polyarchies. The American Economic Review 76, no. 4 (September): 716-727.

_____ and _____. 1988. Committees, hierarchies, and polyarchies. The Economic Journal. 98 (June): 451-470.

_____ and _____. 1985. Economics of Committees. School of Organization and Management Working Paper No. D10. Yale University.

Shannon, Claude E. 1949. The Mathematical Theory of Communication. Urbana, IL: University of Illinois Press.

Simon, H. A. 1955. A Behavioral Model of Rational Choice. Quarterly Journal of Economics 69: 99-118.

Simon, H. A. 1982. From substantive to procedural rationality. Models of Bounded Rationality: Behavioral Economics and Business Organization. Vol. 2: 129-140. Cambridge, MA: The MIT Press.

Simon, H. A. 1988. Rationality as process and a product of thought. Decision Making: Descriptive, Normative, and Prescriptive Interactions. D. E. Bell, H. Raiffa, and A. Tversky, eds., 58-77. Cambridge: Cambridge University Press.

Simon, H. A. 1982. Theories of Bounded Rationality. Models of Bounded Rationality: Economics and Business Organization. Vol. 2: 161-176. Cambridge, MA: The MIT Press.

Spanier, Jerome, and Keith B. Oldham. 1987. An Atlas of Functions. New York: Hemisphere Pub. Corp.

Standard & Poor's Corp. 1979. S & P's Ratings Guide. New York: Mc-Graw-Hill.

Standard & Poor's Corp. 1991. S & P's Corporate Finance Criteria. Solomon Samson, ed. New York: Standard & Poor's Corp.

Surkan, Alvin J., and J. Clay Singleton. Neural Networks for Bond Rating Improved by Multiple Hidden Layers. Proceedings of the IEEE International Conference on Neural Networks. San Diego, 1990. Los Alamitos, CA: IEEE Computer Society Press.

Tullock, Gordon. 1967. Toward a Mathematics of Politics. Ann Arbor: The University of Michigan Press.

Walliser, Bernard. 1989. Instrumental Rationality and Cognitive Rationality. Theory and Decision 27: 7-36.

White, Halbert. 1988. Economic Prediction Using Neural Networks: The Case of IBM Daily Stock Returns. working paper #88-20. San Diego: UCSD.

White, Halbert. 1989. Learning in Artificial Neural Networks: A Statistical Perspective. Neural Computation 1, no. 4. (Winter): 425-464.

Winston, Patrick Henry. 1984. Artificial Intelligence. 2d ed. Reading, MA: Addison-Wesley.

Wiener, Norbert. 1948. Cybernetics: or Control and Communication in the Animal and the Machine. 2d ed. Cambridge, MA: The MIT Press.