2005

NEURAL NETWORK APPLICATIONS IN AGRICULTURAL ECONOMICS

Jianhua Chen
University of Kentucky, josh7chen@yahoo.com

Click here to let us know how access to this document benefits you.

Recommended Citation
https://uknowledge.uky.edu/gradschool_diss/228

This Dissertation is brought to you for free and open access by the Graduate School at UKnowledge. It has been accepted for inclusion in University of Kentucky Doctoral Dissertations by an authorized administrator of UKnowledge. For more information, please contact UKnowledge@lsv.uky.edu.
ABSTRACT OF DISSERTATION

Jianhua Chen

The Graduate School
University of Kentucky

2005
NEURAL NETWORK APPLICATIONS
IN AGRICULTURAL ECONOMICS

ABSTRACT OF DISSERTATION

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the College of Agriculture at the University of Kentucky

By
Jianhua Chen

Lexington, Kentucky

Director: David L. Debertin, Ph.D., Professor of Agricultural Economics

Lexington, Kentucky

2005

Copyright© Jianhua Chen 2005
Neural networks have become very important tools in many areas including economic researches. The objectives of this thesis are to examine the fundamental components, concepts and theory of neural network methods from econometric and statistic perspective, with particular focus on econometrically and statistically relevant models. In order to evaluate the relative effectiveness of econometric and neural network methods, two empirical studies are conducted by applying neural network methods in a methodological comparison fashion with traditional econometric models.

Both neural networks and econometrics have similar models, common problems of modeling and interference. Neural networks and econometrics/statistics, particularly their discriminant methods, are two sides of the same coin in terms of the nature of modeling statistic issues. On one side, econometric models are sampling paradigm oriented methods, which estimate the distribution of the predictor variable separately for each class and combine these with the prior probabilities of each class occurring; while neural networks are one of the techniques based on diagnostic paradigm, which use the
information from the samples to estimate the conditional probability of an observation belonging to each class, based on predictor variables. Hence, neural network and econometric/statistical methods (particularly, discriminant models) have the same properties, except that the natural parameterizations differ.

The empirical studies indicate that neural network methods outperform or are as good as traditional econometric models including Multiple Regression Analysis, Linear Probability Model (LPM), and Logit model, in terms of minimizing the errors of in-sample predictions and out-of-sample forecasts. Although neural networks have some advantages over econometric methods, they have some limitations too. Hence, neural networks are perhaps best viewed as supplements to econometric methods in studying economic issues, and not necessarily as substitutes.

KEYWORDS: neural networks, feedforward, backpropagation, economic growth, food manufacturing.

Jianhua Chen

December 08, 2005
NEURAL NETWORK APPLICATIONS
IN AGRICULTURAL ECONOMICS

By
Jianhua Chen

David L. Debertin, Ph. D.
Director of Dissertation

David Freshwater, Ph.D.
Director of Graduate Studies

December 08, 2005
RULES FOR THE USE OF DISSERTATIONS

Unpublished dissertations submitted for the Doctor’s degree and deposited in the University of Kentucky Library are as a rule open for inspection, but are to be used only with due regard to the rights of the authors. Bibliographical references may be noted. But quotations or summaries of parts may be published only with the permission of the author, and with the usual scholarly acknowledgements.

Extensive copying or publication of the dissertation in whole or in part also requires the consent of the Dean of the Graduate School of the University of Kentucky.

A library that borrows this dissertation for use by its patrons is expected to secure the signature of each user.
NEURAL NETWORK APPLICATIONS
IN AGRICULTURAL ECONOMICS

DISSERTATION

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the College of Agriculture at the University of Kentucky

By
Jianhua Chen

Lexington, Kentucky

Director: David L. Debertin, Ph.D., Professor of Agricultural Economics

Lexington, Kentucky

2005

Copyright © Jianhua Chen 2005
To Shi-Fu where I always remember with nostalgia;

To my teachers past and present who always encouraged me.
ACKNOWLEDGEMENT

I would like to express my gratitude to my advisor, Dr. David Debertin, for his support and encouragement throughout my graduate studies and dissertation preparation. His technical and editorial advice was essential to the completion of this dissertation.

My thanks also go to the members of my advisory committee, Dr. Angelos Pagoulatos, Dr. Jerry R. Skees and Dr. Mark Toma for reading previous drafts of this dissertation and providing many valuable comments that improved the contents of this dissertation. I would like to thank the outside examiner, Dr. Ram Pakath for his time input and support.

I will give a special thanks to Dr. Stephan J. Goetz for directing me to start my dissertation and researches at the University of Kentucky. I do not believe that I have been working in artificial neural networks and come out with this dissertation without his guide in the first phase of my graduate study.

The friendship of Dr. Maretno (Agus) Harjoto, Ms. Mei Chen is much appreciated and has led to many interesting and good-spirited discussions relating to this research. Especially, Mei Chen helped me with data collection for application two.

Special thanks go to my parents, Yunlun Chen and Shunxiu Tang, and Professor Peiming Huang, my former advisor in Beijing, for their encouragement.

Last but not least, I want to thank Ivy and Joanna for supporting me with their love and understanding during past several years.
TABLE OF CONTENTS

Acknowledgements ........................................................................................................ iii
List of Figures .................................................................................................................... vii
List of Tables .................................................................................................................... viii

Part One Neural Network Theory and Methods: A Review on Econometric and Statistic Perspective .......................................................... 1

Chapter One Neural Networks Overview and Heuristics ................................. 2
1. Motivation for the Studies ......................................................................................... 5
2. The Outline of This Thesis ....................................................................................... 5
3. Neural Networks Overview and Heuristics ............................................................... 8
3.1 Basic Concepts of Neural Networks ...................................................................... 8
3.2 Architecture and Organization of a Neural Network ............................................ 9
3.2.1 Structure of a Neuron ..................................................................................... 10
3.2.2 A Layer of Neuron ......................................................................................... 14
3.2.3 Structure of a Network ................................................................................... 14
4. Learning Schemes and Learning Rules in a Neural Network .............................. 16
4.1 Learning Schemes ............................................................................................... 17
4.2 Learning Rules ..................................................................................................... 18
5. Neural Network History as a Glance ................................................................. 20
5.1 Before 1940s: the Thoughts of Neural Networks ................................................. 21
5.2 The 1940s: the Beginning of Modern Era for Neural Networks ......................... 21
5.3 The 1950s and 1960s: the Golden Age of Neural Networks ................................ 22
5.4 The 1970s: the Stagnation Time of Neural Networks .......................................... 23
5.5 The 1980s: the Rejuvenation of Neural Networks .............................................. 23
5.6 The 1990s and Present: the Explosion of Interest in Neural Networks ............... 24
5.7 Neural Network Applications in Agricultural Economics ................................. 25

Chapter Two Linear Neural Networks: An Econometric Perspective ............... 26
1. Hebbian Linear Associator Network ...................................................................... 27
1.1 The Architecture of Linear Associator ................................................................. 27
1.2 Hebbian Learning Rule ....................................................................................... 28
1.3 Estimation Methods ............................................................................................ 29
2. Single Perceptron Network ................................................................................... 32
2.1 The Architecture of Perceptron ......................................................................... 33
2.2 The Learning Rule .............................................................................................. 34
3. Adaptive Linear Neural Network ........................................................................ 35
3.1 The Architecture of ADLINE ........................................................................... 36
3.2 The Learning Rule .............................................................................................. 37
4. Econometric Commentaries ............................................................................... 38
4.1 Neural networks provide an intuitive representational framework for econometric models .................................................................................. 38
4.2 Neural networks have the same concepts as econometrics but different terminologies ......................................................................................... 39
Chapter Three Non-Linear Neural Networks: An Advanced Statistical Perspective

1. Linear Separable versus Linear Non-separable .................................................. 44
2. Multilayer Perceptron Architecture ........................................................................... 48
3. Learning Algorithms .................................................................................................. 50
3.1 The Standard Backpropagation Algorithm ............................................................... 51
3.2 Levenberg-Marquardt Algorithm .............................................................................. 57
4. Statistical Commentaries .......................................................................................... 61
4.1 Similarities and Differences between Multilayer Perceptron and Nonlinear Statistic Models ................................................................................................................................. 61
4.2 The Advantages and Disadvantages of Multilayer Perceptron versus Nonlinear Statistic Models ................................................................................................................................. 63
4.2.1 The Advantages of Neural Networks .................................................................. 64
4.2.2 The Disadvantages of Neural Networks ............................................................. 65

Chapter Four Neural Network Design and Its Pragmatics ............................................ 68
1. Number of Hidden layers .......................................................................................... 68
2. Number of Neurons .................................................................................................... 70
3. Cross-Validation ......................................................................................................... 72
4. Data Re-scaling .......................................................................................................... 73
4.1 Along Norm[0,1] ....................................................................................................... 75
4.2 Along Norm [-1,1] .................................................................................................... 76
5. Evaluation of Forecast Accuracy ................................................................................ 75
5.1 Absolute error measures ........................................................................................ 76
5.2 The indices of squared errors .................................................................................. 77
5.3 Direction Errors ....................................................................................................... 80

Part Two Neural Network Applications in Agricultural Economics ............................ 81

Chapter Five Forecasting Economic Growth: A Comparison of Econometric and Neural Network Methods .............................................................. 82
1. Modeling Economic Growth ...................................................................................... 83
1.1 Literature Review .................................................................................................... 83
1.2 Specification of the Economic Growth Equation .................................................. 85
2. Methodology of Neural Networks ............................................................................ 86
2.1 Architecture of a Neural Network .......................................................................... 87
2.2 Econometric/statistic perspectives of neural networks .......................................... 91
2.3 Empirical design aspects of neural networks ....................................................... 95
3. Evaluation and Metrics for Prediction Accuracy ..................................................... 98
3.1 Absolute error measures ....................................................................................... 99
Chapter Six  Feedforward Neural Networks in Prediction of Food Manufacturing Establishment Growth ............................... 110

1. General Introduction ...............................................................110
2. Location Determinants and Hypotheses .......................................113
2.1 Market Access Variables ..........................................................114
2.2 Raw Material (Inputs) Variables ...............................................114
2.3 Labor Force Characteristics Variables .......................................115
2.4 Policy Variables .................................................................116
2.5 Agglomeration Economy Variables ..........................................116
3. Empirical Methodology and Data Description ..............................117
3.1 The Empirical Methodology ......................................................117
3.2 Data Description .................................................................121
4. Results and Discussion .............................................................122
4.1 The Results of the Linear Probability Models ..............................122
4.2 The Results for the Logit Model ...............................................125
4.3 Results of Neural Networks Results ........................................126
5. Summary and Conclusions .......................................................129

Chapter Seven Summary and Conclusions .................................133
1. Thesis Summary and conclusions .............................................133
2. Further Work .................................................................136

References .................................................................185
Vita .................................................................194
LIST OF FIGURES

Figure 1.1, The structure of a biological .......................................................... 138
Figure 1.2, The structure of a simulated ............................................................ 138
Figure 1.3, The structure of a neuron .............................................................. 139
Figure 1.4, The structure of a layer of neurons ................................................. 139
Figure 1.5, The structure of a neural network .................................................. 140
Figure 2.1, The architecture of linear network ................................................ 140
Figure 2.2, The architecture of perceptron network ........................................ 141
Figure 2.3, Decision boundary for two-input perceptron .................................. 141
Figure 2.4, Decision boundary for two-input ADALINE ................................ 142
Figure 2.5, General representation of a neural network and econometric model ... 142
Figure 3.1, Logic “OR” ..................................................................................... 143
Figure 3.2, Two input/single-output neuron .................................................... 144
Figure 3.3, Surface of transfer function .......................................................... 144
Figure 3.4, Surface of transfer function (scaled by 10) ..................................... 144
Figure 3.5, Neuron of logic “OR” ................................................................. 145
Figure 3.6, Decision boundary of logic “OR” ................................................. 145
Figure 3.7, Neuron of logic “AND” ............................................................... 146
Figure 3.8, Decision boundary of logic “AND” .............................................. 146
Figure 3.9, Neuron of logic “NAND” ............................................................ 147
Figure 3.10, Decision boundary of logic “NAND” ....................................... 147
Figure 3.11, The linear non-separability (XOR) ............................................. 148
Figure 3.12, The structure of a network for solving XOR ............................... 148
Figure 3.13, Network of “XOR” ................................................................. 149
Figure 3.14, Surface of XOR neural network ................................................. 150
Figure 3.15, Decision boundaries of logic “XOR” ......................................... 150
Figure 3.16, The model of classifier of neural ................................................. 151
Figure 3.17, The flow chart of the backpropagation algorithm ....................... 152
Figure 4.1, Discontinuous function .............................................................. 153
Figure 4.2, The relationship between error and training time or number of neurons/layers ................................................................. 153
Figure 5.1, Structure of a Biological Neuron .................................................. 154
Figure 5.2, Structure of a Artificial Neuron ................................................... 154
Figure 5.3, Bio-Neuron versus Artificial Neuron .......................................... 155
Figure 5.4, The structure of a layer of neurons .............................................. 156
Figure 5.5, The structure of a neural network .............................................. 156
Figure 5.6, The structure of a Linear Network .............................................. 157
Figure 5.7, The structure of a Perceptron ..................................................... 157
Figure 5.8, MPL with one hidden layer ......................................................... 158
Figure 5.9, Sum of absolute error of in-sample prediction ............................ 159
Figure 5.10, Sum of square error of in-sample prediction ............................ 159
Figure 5.11, Direction error of in-sample prediction .................................... 160
Figure 5.12, Sum absolute error of out-of-sample forecast ............................ 160
Figure 5.13, Sum of Square error of out-of-sample forecast .......................... 161
Figure 5.14, Mean absolute percentage error of in-sample prediction ......... 161
LIST OF TABLES

Table 1.1, The Common Transfer Functions ....................................................... 162
Table 1.2, The Comparison of econometric and Neural Network Terms .................. 163
Table 3.1, Output of Logic “OR” ..................................................................... 164
Table 3.2, Output of Logic “AND” ................................................................. 164
Table 3.3, Output of Logic “NOT AND” .......................................................... 164
Table 3.4, Output of Logic “XOR” .................................................................. 165
Table 5.1, Regression Model for Economic Growth ............................................ 166
Table 5.2, In-Sample Prediction Errors ............................................................. 167
Table 5.3, Out-of-Sample Forecast Errors ......................................................... 167
Table 6.1, Descriptions of the industries ............................................................ 169
Table 6.2, Variable description ....................................................................... 170
Table 6.3, Variable description (continued) ....................................................... 171
Table 6.4, The Summary Statistics for variables .................................................. 172
Table 6.5, The Results of the Linear Probability Models ....................................... 173
Table 6.6, The Results of the Linear Probability Models (continued) ..................... 174
Table 6.7, The Results of the Logit Models ......................................................... 175
Table 6.8, The Results of the Logit Model (continued) ......................................... 176
Table 6.9, The Prediction Accuracy and Errors of the LPMs (in-sample) ............... 177
Table 6.10, The Prediction Accuracy and Errors of the LPMs (out-of-sample) ...... 177
Table 6.11, The Prediction Accuracy and Errors of the Logit Models (in-sample) ... 178
Table 6.12, The Prediction Accuracy and Errors of the Logit Models (out-of-sample) .... 178
Table 6.13, The Weight Matrix of Hidden Layer of the Neural Network (SIC2000): Experiment One .............................................................. 179
Table 6.14, The Weight Vector of Output Layer of the Neural Network (SIC2000): Experiment One .............................................................. 179
Table 6.15, The Weight Matrix of Hidden Layer of the Neural Network (SIC2000): Experiment Two .............................................................. 180
Table 6.16, The Weight Vector of Output Layer of the Neural Network (SIC2000): Experiment Two .............................................................. 180
Table 6.17, The Prediction Accuracy and Errors of the Neural Networks (in-sample) .............................................................. 181
Table 6.18, The Prediction Accuracy and Errors of the Neural Networks (out-of-sample) .............................................................. 181
Table 6.19, Comparison of Correctness Counts among the Three Methods (in-sample) .............................................................. 181
Table 6.20, Comparison of Correctness Counts among the Three Methods (out-of-sample) .............................................................. 182
Table 6.21, Comparison of Mean Squared Errors (MSE) among the Three Methods (in-sample) .............................................................. 183
Table 6.22, Comparison of Mean Squared Errors (MSE) among the Three Methods (out-of-sample) .............................................................. 183
Table 6.23, Comparison of Mean Absolute Errors (MABSE) among the
Three Methods (in-sample) ........................................................................................................ 184
Table 6.24, Comparison of Mean Absolute Errors (MABSE) among the
Three Methods (out-of-sample) .................................................................................................. 184
**Notation:**

1) scalar, vector and matrix:

   Scalars: small italic letters, such $i, o$.

   Vectors: small bold italic letters, such $i, o$.

   Matrices: Capital bold italic letters, such as $W$.

2) The superscript and subscript: In the thesis, there are many notations such as $A_2$, except additional notation, the meanings of the subscript and superscript are as follow:

   - The subscript to the right of a letter is the layer number.
   - The subscript to the left of a letter is the number of the variable, such as $1^{st}, 2^{nd}$, … of observation or input.
   - The superscript to the left of the letter is exponential index. For example, $c^2$ is $c$ squared.
Part One

Neural Network Theory and Methods: A Review on Econometric and Statistic Perspective
Chapter One
Neural Networks Overview and Heuristics

1. Motivation for the Studies

Neural networks are a class of input-output models originally developed by cognitive scientists. These models are capable of learning and memorizing knowledge from in-sample data, and extrapolating patterns from out-sample data. In econometric/statistic terms, neural networks constitute a particular class of nonlinear parametric methods. “Learning” corresponds to statistical estimation of model coefficients (Kuan and White, 1992).

Neural networks have been applied as research tools in many fields such as speech and signal processing (Sejnowski, Yuhas, Goldstein, and Jenkins, 1990; Malkoff, 1990), handwritten character recognition (Le Cun, Boser, Denker, Henderson, Howard, Hubbard and Jackel, 1990), finger print recognition (Leung, Engeler, and Frank, 1990), prediction of bank failure (Coats and Fant, 1993; Altman, Marco and Varetto, 1994), prediction of financial stock market performance (Azoff, 1994; Refenes, Zapranis and Francis, 1994; Gately 1996; Trippi, 1996), modeling and forecasting economic issues (Joerding, Li, and Young, 1993; Maasoumi, Khotanzad and Abaye, 1994; Church and Curram, 1996; Kastens and Featherstone, 1996; Terasvirta and Dijk, 2004; Gavidia and Gupta 2004), and many more.

The reason why investigators from variety of areas have been attracted to apply the neural network models in their researches and applications is that the nature of neural network’s freedom from restrictive assumptions such as linearity that are often needed to
make the traditional mathematical models tractable (Moshiri, 1998). Although the usefulness of a neural network is popular, the meaning of neural networks differs among the applications. It means that not everyone treat them as the same thing. For example, neuro-scientists may consider them as a model of explaining biological data; Cognitive scientists may use them as formalism to capture cognitive processing; Computer scientists look them as a sub-domain of machine intelligence (Sweiss and Kulikowski, 1991); Statisticians see and use them as nonlinear regression and classification models, or non-parametric models as additional tools to their traditional toolbox (White, 1989; Bishop 1995; Sarle 1997); Engineers or other researchers employ them in applications requiring nonlinear processing of continuous data, or function simulation (Funahashi, 1989); and data-mining analysts use them to find pattern in massive data and see them as an information system, knowledge learning mechanism and decision making supporting system (Bigus, 1996). In short, people from both academic and industrial environments, consider neural networks as handy tools and apply them in their work, although the nature of the applications differs.

Successes in these or other areas suggest that neural network models may serve as a useful addition to the economic researcher tool kit (Kuan and White, 1994). From a methodological perspective in the regime of economics, two main issues have been extensively and intensively studied are regression and classification (Gujarati, 1995; Greene, 1997). For regression, the most common issue studied is linear regression, and the least squares method is a basic method to estimate the coefficients. For classification, the Logit, Probit and the Linear Probability Model (LPM) models are normally applied. More complex models, such as the simultaneous-equation model, are deployed to solve
the economic problems in the real world. The ultimate goal of these sorts of studies is to understand the objective laws of economic phenomena.

However, the validity of the results and conclusions of our researches strongly rely on many assumptions and hypotheses. First, for convenience or simplicity, we may prefer to assume the relationship of a dependent variable to an independent variable is linear (Gujarati, 1995; Greene, 1997). In fact, this assumption of linearity may not be true in the real world, because economic phenomena are factor-interactive, dynamic systems. A linear model may not accurately capture most economic phenomena, and these models often generate inaccurate results. Second, even if the non-linearity is introduced to the econometric models, some hypotheses are often imposed on the models. Since economic systems are complex and subject to human reactions and counter-reactions by different agents or players, it is difficult, if not impossible, to write down a complete model with all the potential reactions and counter-reactions determined. Given such complexity, it is nearly impossible to try to approximate in any details a model based on first principal of, for example, utility maximizations or profit maximization. In such complex systems, it is natural to turn to models, which emulate and simulate the economy or the society in question. That is exactly what the neural network methodology is capable of delivering (Shachmurovem, 2004). Finally, the validity of the property, Best Linear Unbiased Estimators (BLUE), of ordinary least square method, depends on many assumptions including zero mean values of disturbance, homoscedasticity of variance of disturbance, no autocorrelation between the disturbance and etc. (Gujarati, 1995; Greene, 1997). However, those assumptions may or may not be valid in practice.
This dissertation has two goals: first, to review the fundamental components, concepts and theory of neural network methods from an econometric and statistical perspective, with particular focus on econometrically and statistically relevant models; and second, to apply neural network methods to model agricultural issues in a methodological comparison fashion. Although the neural network method is not “native” in the regime of agricultural economics, it may be a potential alternative to model our economic issues. There is an old Chinese saying, “Stones from other hills may serve to polish the jade of this one”. It means that we can borrow tools with certain advantages from others to overcome our own shortcomings.

2. The Outline of This Thesis

This dissertation consists of two parts. In part one, we shall review neural network theory and methods from econometric and statistical perspective. The main theory and methods of neural networks are comparatively exploited with econometric and statistic models. Besides that the general framework of a neural network model is investigated, more emphasis is given to multilayer feedforward networks and backpropagation learning algorithms, which are the most common algorithm and training algorithm in the field of neural networks. Moreover, the two main methods of regression and classification, which are most common topics both in the field of neural networks and in the domain of econometrics, are addressed to set up a better foundation for the applications in part two. In part two, two applications of neural networks in agricultural economics are presented, one is titled as “Forecasting Economic Growth: A Comparison of Econometric and
Neural Network Methods”; the other application is “Feedforward Neural Networks in Prediction of Food Manufacturing Establishment Growth”.

In chapter one, a brief review is given of the literature employing neural networks. All the concepts, terminology, principles, frameworks and methods are illustrated with respect to the counterparts of econometric models. We shall investigate the basic “hardware” components of the neural network models such as a single neuron, transfer functions and a single layer, and then a general structure of a network. For better understanding neural networks, a glance is taken at the developing history of neural networks.

In chapter two, the focus is to study linear neural networks. We shall review three classic single-layer networks including Hebbian Linear Associator Network, a Single Perceptron Network and Adaptive Linear Neuron Network (ADALINE). We are going to introduce their architecture and learning rules at length. Although they are rarely being used to solve practical issues nowadays because of their limitations, these single-layer linear networks provide many useful insights into the properties of more complex multilayer networks. Especially, we will present these networks in comparison of econometric models such as the multiple linear regression, the linear probability model (LPM) and Logit model. We can observe the similarities between single-layer networks and econometric models from mathematical forms and the methods of parameter estimation.

In chapter three, non-linear neural networks (multilayer networks), specifically feedforward networks, will be reviewed from statistical perspective. Because one-layer networks can merely model data with simple pattern or solve linear separable issues, the
non-linear networks with multiple layers of successive transformations have been widely used in many areas. They become the main stream in this area. To understand the dynamic process of weights adaptation and mechanism of the network, we shall deploy networks to model the simple logic issues such as “OR”, “AND” and “NOT AND” (NAND). Finally, the most popular training algorithms for neural networks, the backpropagation and its variant the Levenberg-Marquadt algorithm, are presented at length since they will be employed to train the networks in the applications in chapter five and six.

In chapter four, several important issues for neural network design are discussed. First, the negative consequences of the networks with too many hidden layers and neurons are expounded, and the potential benefits of the cross-validation and data re-scaling elucidated. Second, since the various criteria used to evaluate forecast accuracy can profoundly alter the ranking of the two methods that generated the forecasts, effort to studying a variety of measures of forecast errors, including absolute, squared and directional errors, is devoted.

In chapter five, we evaluate the relative effectiveness of econometric and neural network methods in forecasting economic growth by comparing in-sample predictions and out-of-sample forecasts and associated errors from models estimated using cross-sectional, county-level income growth data. Since the criteria used to evaluate forecast accuracy can profoundly alter the ranking of the two methods that generated the forecasts, we devote considerable effort to testing a variety of measures of forecast errors, including absolute, squared and directional errors.
In chapter six, we shall use a feedforward network to predict food manufacturing establishment growth in this country. The purpose of this application is to determine the factors that affected food manufacturing establishment growth and compare the ability of econometric methods including the linear probability model and the Logit model and feedforward neural networks in forecasting food manufacturing establishment growth by comparing in-sample prediction and out-of-sample forecast. The aggregate industry of food manufacture and its 9 sub-industries were investigated. The data was cross-sectional data from 3049 continental U.S. counties. They were divided two sets. One set had 2444 observations for estimating the econometric models and training the neural network model. The other set had 655 observations for out-of-sample forecast so as to test the performance of the estimated models when they faced a new data set.

In chapter seven, we are going to make a summary for this thesis, and give out the directions for further researches.

3. Neural Networks Overview and Heuristics

3.1 Basic Concepts of Neural Networks

The “Neural Network” as employed here is obviously not a biological concept. It is artificial. In the regime of neural networks, “Neural Networks” and “Artificial Neural Networks” are interchangeable. Neural networks are inspired by the functioning of the biological network of neurons in human brain. A human brain contains approximately $10^{11}$ computing elements called neurons. Those neurons are fully connected and communicate with each other. The inputs are received by the sensory receptors and sent
to the neurons in the network. The neurons process the inputs and send information to the next neurons (Zurada, 1992).

What is an artificial neural network? In fact, there is not a standard definition in literature. One of definitions for neural networks, which is quoted by many researchers, is given by Haykin (1999). According to Haykin, a neural network is a massively parallel distributed processor made up of simple processing units, which has a natural propensity for storing experiential knowledge and making it available for use. It resembles the brain in two respects: First, knowledge is acquired by the network from its environment through a learning process; Second, interneuron connection strengths, known as synaptic weights, are used to store the acquired knowledge. It is certain that a neural network model is composed of many neurons, which are basic units that work like computer processors. The units are connected by communication paths (connections) with weights. The units operate only on the inputs that they receive via the connections, and then send the outputs to the next layer of units. As a neural network model, it should have some sort of "training" rule whereby the weights of connections are adjusted on the basis of training set data. Thus, neural networks “learn” from input examples and exhibit some capability for generalization beyond the example data (Sarle, 1999).

3.2 Architecture and Organization of a Neural Network

A biologic neural network is a net with many nodes, which are called neurons or units. Neurons are the basic units and building blocks of a net. The structure, function and working procedure of a single neuron of a neural network are based on a biologic brain neuron. The working process of a biological neuron is that: to receive signals from the
outside, collect and send them to the information processing area, when the combined signals of reach a certain threshold, the neuron is activated and pass a signal on to other neurons. Billions of those kinds of neurons construct a powerful brain. In terms of the rules of biologic neurons, a heuristic model of neural networks operates based on the following assumptions (Fausett, 1992):

First, information is processed at neurons. Neurons are parallel in each layer. It means that there is no information communication among neurons in the same layer. Second, signals are passed between neurons in contiguous layers over connection links (paths). Third, each connection link has a weight, which multiplies the signal transmitted. Fourth, there is a transfer function in each neuron to determine the value of the outgoing signal.

Hence, a neural network is characterized by: (1) a pattern of connections between the neurons that link specific inputs to outputs (the architecture), (2) a procedure on creating the “weights” (its training/learning algorithm), and (3) an activation function (or called transfer function). Although all neural networks share those characteristics, there are many different training algorithms employed for estimating the weights.

3.2.1 Structure of a neuron

First, let us take a look at a biological neuron. A neuron is the fundamental part of the nervous system. All the neurons have the same structure, independent of their size. The structure of a biological neuron is shown in Figure 1.1. There are four physical components for a neuron, including cell body, dendrites, axon and synaptic terminals. The dendrites are the signal receivers, which accept signal from outside or other neurons.
The cell body is the place for message processing, and generate impulses. The axon is the channel that transmits the message generated by the neuron to the next neurons, or to the outside such as muscle fibers.

According to the structure and function of a biological neuron, a simulated neuron can be constructed as illustrated in Figure 1.2. The neuron may receive the exciting signals and inhibitory signals at the same time, and then integrate them. The output is sent through an outlet to the other neurons.

It is not difficult to understand an artificial neuron in neural networks, which is a mathematical neuron. It is the basic block for building a neural network, and is fundamental to the operation of a neural network. A neuron acts as a tiny “processor”, receiving, processing and sending data to the next layer of the model. As Figure 1.3 shown, a neuron is composed of a summer (or an adder), which just likes the linear equation of the linear regression model, and a transfer function, which is linear or non-linear.

The adder is directly connected with the input vector \((x_1, x_2, x_3, \ldots, x_n)\) from outside of the neuron. There is a weight on each connection (path) between each input and the neuron. In addition, a bias which input value is 1 is also associated with the neuron. Therefore, input-output equation is

\[ a = b + w_1x_1 + w_2x_2 + \ldots + w_nx_n \]

where \(a\) is the output of the summer; \(b\) is the bias; \(w_1, w_2, \ldots, w_n\) are the weights. \(x_1, x_2, \ldots, x_n\) are the inputs.
Three major components make up an artificial neuron. All neurons in the artificial neural networks have these components without exception.

(1) Weighting Factors: A neuron receives external inputs. Each input has its own relative weight on the connection with the neuron, which provides the input’s impact on the summation function. These weights perform similar functions as do the synaptic strengths of a biological neuron. Obviously, inputs make different amount of contribution to the output of the neuron because of the magnitude of the weights. Furthermore, weights also represent the intensity and strength of the connections of the neurons. From mathematical perspective, the collection of weights is the matrix that can memorize the knowledge from the training data sets. The counterpart of weights in econometric models is coefficients.

(2) Summation Function: There are two operations inside a neuron. The first one calculates the weighted sum of all of the inputs. The second one converts the output of the summation in term of certain threshold. Mathematically, summation function is to operate the inner product of input vector and weight vector. The result is a single number. Geometrically, the inner product of two vectors can be considered a measure of their similarity. If the vectors point in the same direction, the inner product is maximum; if the vectors point in opposite direction (180 degrees out of phase), their inner product is minimum.

(3) Transfer Function: The transfer functions (or activation functions) are essential parts in a neural network model. It is analogous to the logistic function in a Logistic model, and the inverse of the link function in a Generalized Linear Model (GLM) (McCullagh and Nelder 1989). In literature, much emphasis has been given on the
transfer function. The transfer function is employed to transform the result of summation function, basing on the need of neuron output. For example, the neuron could use a “hard limiter” or step function to output zero and one, one and minus one, or other numeric combinations.

Three types of transfer functions, hard limiter (step function), linear and log-sigmoid, are often applied in an artificial neuron. Table 1.1 gives out all transfer functions. The most useful type of transfer function is log-sigmoid. There are two reasons. One is that linear (straight-line) functions simply increase or decrease the summation result proportionally. Linear functions cannot solve non-linear issues in practice. That was the problem in the earliest network models such as Perceptron and ADALINE models. We are going to discuss it at length in Chapter Two. The second reason why sigmoid functions are welcome is that these functions have the following properties:

a. They are continuous functions and monotonically increase or decrease.

b. They are non-linear. This is the reason why the neural networks are so powerful.

c. They are differentiable and it is efficient to take derivatives.

d. They saturate toward the minimum and maximum values.

Researchers often prefer to using logistic functions or hyperbolic tangent functions in hidden layers. Although they are different functions, the actual effects caused by them are negligible.
3.2.2 A layer of neurons

Figure 1.4 is of a layer of neurons. The left nodes are inputs. Each of the units corresponds to one input. The input layer is analogous to a series of explanatory (independent) variables in a regression model. The nodes in this layer are not considered to be neurons. They are merely inputs to the next layer (neurons). The next layer is a layer of “real” neurons. The nodes called neurons are different from those in the input layer. Neurons are vertically connected by channels. The channels are linked via a set of weights which are like the parameters of an econometric model.

Although there are only three neurons illustrated in the model, more than three neurons may be present in each layer. Neurons are independent and separate each other. Their computations are also parallel. Note that the weight \( W \) is a matrix, and the input \( a \) and output \( y \) are vectors, respectively. The computations in the adders and the transfer functions are:

\[
\begin{align*}
  a &= b + Wx \\
  y &= f(a)
\end{align*}
\]

where \( b \) is bias vector and \( f(.) \) is transfer function.

3.2.3 Structure of a Network

A network may consist of several layers. Figure 1.5 is of multilayer feedforward network. There is controversy about layer counting. Some researchers count the vector of raw inputs as a layer, but others do not.
One approach that avoids controversy is to count the hidden layers. Normally, the first (or left in our diagrams) layer is input layer, and the far right layer is the output layer. The layers falling between are hidden layers. If there is one hidden layer in a model, so the model is called as one hidden layer model. Hence, Figure 1.5 is a model with two hidden layers. The computations in each layer are as follows:

In the first hidden layer, the operations of the adder and the transfer function are

\[ \mathbf{a}_1 = \mathbf{b} + \mathbf{W} \mathbf{x} \]  \hspace{1cm} (1.3)

\[ \mathbf{y}_1 = f(\mathbf{a}_1) \]  \hspace{1cm} (1.4)

where, \( \mathbf{x} \) is inputs vector, \( \mathbf{W} \) is the weights matrix in current layer, \( \mathbf{b} \) is vector of biases. The output of summer (\( \mathbf{a} \)) is the input of the transfer function, respectively.

In the second hidden layer, the operations of the adder and the transfer function are as follows

\[ \mathbf{a}_2 = \mathbf{b} + \mathbf{W}_1 \mathbf{y}_1 \]  \hspace{1cm} (1.5)

\[ \mathbf{y}_2 = f(\mathbf{a}_2) \]  \hspace{1cm} (1.6)

where, the output of previous layer \( \mathbf{y}_1 \) becomes the input of current layers. The computation is similar to the first hidden layer.

In the output layer, the computations of the adder and the transfer function are

\[ \mathbf{a}_3 = \mathbf{b} + \mathbf{W}_2 \mathbf{y}_2 \]  \hspace{1cm} (1.7)

\[ \mathbf{y}_3 = f(\mathbf{a}_3) \]  \hspace{1cm} (1.8)
where, the output of the second hidden layer $2y$ is used as input. The output of this layer represents the final result of the whole network.

If all the operations are concatenated into one mathematical form that is

$$\hat{y} = 3f(3W2f(2W1f(Wx + b) + b) + b) + b$$  \hspace{1cm} (1.9)

A key neural network design problem is to determine the number of hidden layers and neurons for the empirical problem under study. If the data are linear, a hidden layer may not be required. For a non-linear problem, a hidden layer is crucial. A net with only one hidden layer can effectively “learn” the patterns contained within the data. The fewer neurons in a network, the fewer the number of algebraic operations needed, and the shorter the amount of computing time needed. Further, a complex network model may simulate the in-sample data very well, but not do well in out-of-sample projection. This is analogous to a regression model that contains enough variables such that nearly every observation is intersected by the regression hyperplane such a model would not be useful in making forecasts. Additional neurons and layers require more weights. Thus, an elaborate neural network, with many hidden layers, neurons and weights may simply “memorize” the information contained in the sample, and may not forecast well beyond the data set.

4. Learning schemes and Learning Rules in an Artificial Neural Network

Once the number of layers, and number of units in each layer, has been selected, the next step is to find the optimal coefficients (weights). This process is called
“estimation” in the context of econometrics, and is referred to as “learning” or “training” in the regime of neural networks. The network's weights and bias must be set so as to minimize the error made by the network. This is the role of the training algorithms. The input cases are used to automatically adjust the weights and thresholds in order to minimize this error. This process is equivalent to fitting the model represented by the network to the training data available. The error of a particular configuration of the network can be determined by running all the training samples through the network, comparing the actual output generated with the desired or target outputs. The differences are combined together by an error function to give the network error.

4.1 Learning schemes

Neural networks are trained towards specific outputs by imposing a learning scheme. Learning occurs where a network alters the weights of its component connections, so as to bring it closer to a desired output or problem solution. There are many learning rules in the context of neural networks, which fall into two main categories including supervised learning and unsupervised learning in terms of the role of target (Hagan, Demuth and Beale, 1996). Supervised learning rules use the error (the difference between the target and the output) to guide the adjustment of weights and bias during the training. Eventually, the output of the network is close to the target value at a desired level. In contrast, unsupervised learning rules do not use target values, and even the target values are not available. They just use the inputs information to categorize the input patterns.
4.2 Learning rules

Learning rules are mathematical algorithms that dictate how the connection weights of a neural network will be altered after learning. This is again a crude approximation of biological function as our knowledge of biological learning systems is incomplete. There are a number of learning rules for neural networks. Here we would like to take a look at a few of important ones (DACS, 2005).

(1) Hebb's Rule: it is the first learning rule in the history of neural networks that was introduced by Donald Hebb. Hebb described this rule in his book *The Organization of Behavior* in 1949. It is saying: “When an axon of cell A is near enough to excite a Cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A’s efficiency, as one of the cells firing B, is increased.” Basically, the theory goes like this: Two neurons in the brain may well have a connection between them. The neurons can be activate or inactive. If both the neurons are active (mathematically have the same sign) at the same time, then the strength of the synapse (connection) between the two will increase. If the neurons are not both active at the same time, then the strength of the synapse (connection) will decrease. In real implementations, the Hebb rule can be defined as an unsupervised learning rule, which dose not require any information of the target value, and supervised learning rule, which incorporates real target value in the equation. We are going to discuss more about the Hebb rule in detail in next chapter.

(2) Hopfield Law: It is similar to Hebb's rule with the exception that it specifies the magnitude of the strengthening or weakening. It states, “if the desired output and the input are both active or both inactive, increment the connection weight by the learning
rate, otherwise decrement the weight by the learning rate.” The learning rate usually is a positive number, which value is between 0 and 1. This law introduced the concept of a “learning rate” (Clark, 2005).

(3) The Delta Rule (Widrow-Hoff Learning Rule): the Delta rule is a variation of Hebb's Rule. This rule is also referred to as the Widrow-Hoff Learning Rule and the Least Mean Square (LMS) Learning Rule. It is one of the most commonly used in the regime of neural networks. In addition, it is the precursor to the backpropagation algorithm for multipler networks. This rule is based on the simple idea of continuously modifying the strengths of the input connections to reduce the difference (the delta) between the desired output value and the actual output. The training process is to change the network weights so as to minimize the mean squared error for the network.

The way that the Delta Rule works is that the delta error in the output layer is transformed by the derivative of the transfer function and is then used in the previous neural layer to adjust input connection weights. In other words, this error is back-propagated into previous layers one layer at a time. The process of back-propagating the network errors continues until the first layer is reached.

When using the delta rule, it is important to ensure that the input data set is well randomized. Good conditioned data set can lead the network to converge to the desired accuracy. Just as Meade (1995) states that neural networks are data-dependent, so the leaning algorithms are only as good as the data shown to them.

(4) The Gradient Descent Rule: This rule is similar to the Delta Rule because the derivative of the transfer function is still used to modify the delta error before it is applied to the connection weights. Here, however, an additional proportional constant tied to the
learning rate is appended to the final modifying factor acting upon the weight. This rule is commonly used, even though it converges to a point of stability very slowly.

It has been shown that different learning rates for different layers of a network help the learning process converge faster. In these tests, the learning rates for those layers close to the output were set lower than those layers near the input. This is especially important for applications where the input data is not derived from a strong underlying model.

(5) Kohonen's Learning Law: Teuvo Kohonen is one of the most famous and prolific researchers in neurocomputing, and he has invented a variety of networks (Sarle, 2002). Kohonen’s learning law was inspired by learning in biological systems. In the procedure, the processing elements compete for the opportunity to learn, or update their weights. The processing element with the largest output is declared the winner and has the capability of inhibiting its competitors as well as exciting its neighbors. Only the winner is permitted an output, and only the winner plus its neighbors are allowed to adjust their connection weights.

5. Artificial Neural Network History at a Glance

In this section, let us take a look at the developing history of neural networks so as to better understand the theory and methods of neural networks. The history of neural networks can be divided several stages as follows (Fausett, 1992).
5.1 Before 1940s: the Thoughts of Neural Networks

Even in late 19th and early 20th centuries, some thoughts underlying neural networks had been occurred in many disciplinaries such as physics, psychology and neuro-physiology (Haga, Demuth and Beale, 1995). The early researches focused on general theories of learning, vision and many more. But, no concrete neural network models were invented in that time.

5.2 The 1940s: the Beginning of Modern Era for Neural Networks

The concept of neural networks started from 19th, however these thoughts began to be used to computation models until the 1940s. There were some initial simulations using formal logic. McCulloch and Pitts (1943) developed models of neural networks based on their understanding of neurology. These models made several assumptions about how neurons worked. McCulloch was a psychiatrist and neuro- anatomist, and Pitts was a mathematical prodigy. Their networks were based on simple neurons which were considered to be binary devices with fixed thresholds. The results of their model were simple logic functions such as "a or b" and "a and b". It is a well known fact that the work of McCulloch and Pitts (1943) is the starting mile stone of modern era for neural networks. In the classical paper, “A logical calculus of the ideas immanent in nervous activity”, they introduced the first mathematical model for a neuron. Hence, their work is acknowledged as the beginning of the neural networks (Haga, Demuth and Beale, 1995).

The next major development happened in 1949 with the publication of Hebb’s book, the Organization of Behavior. His famous postulate was that if two neurons were exciting simultaneously and closed to each other, the strength of the connection between
them should be increased. He designed the first learning law for neural networks, the Hebb rule. It was Hebb’s work that other researchers pursued through the next two decades. It is still a basic learning rule in almost all the textbooks of neural networks today.

5.3 The 1950s and 1960s: the Golden Age of Neural Networks

During 1950s and 1960s, approaches for construction neural networks developed quickly. There were two key words should be mentioned in this period. One is Perceptron. The other is ADALINE.

In 1958, Rosenblatt invented the neuron-like element called a perceptron and its associated learning rule. A typical perceptron consisted of an input layer, which connected by paths to the neuron with weights on them. The weights could be adjustable during the training, and they would converge to the right values (Fausett, 1992). The thoughts of perceptron neuron and learning rule are the base-stone of modern multi-layer perceptrons.

In the early 1960s, a device called ADALINE (standing for ADAptive LINEar combiner) and its learning rule were introduced by Widrow and Hoff. The structure of ADALINE is very similar to perceptron. The learning rule minimized the summed square error during the training. The learning is still used today. It is also the precursor of the backpropagation learning rule for multi-layer networks (Fausett, 1992).
5.4 The 1970s: the Stagnation Time of Neural Networks

Despite the successes and enthusiasm of the early and mid-1960s, the existing neural network models (i.e., single-layer networks) and its associated learning rules could not solve more complex computing issues, such as the non-linear classification problem. Neural networkers could not find a way to overcome the inherent weakness of neural networks at that time. So, the 1970s were quiet years for neural network development. The financial and technological (powerful computers) supports were not sufficiently available for neural networks researches. Many researchers deserted studying in this fields (Haykin, 1999).

However, the researchers of neural networks never completely stopped in the 1970s. Many current leading scientists in this fields continued working on neural networks, and invented new networks in this period. Kohonen and Anderson independently developed associative memory networks in 1972. Grossberg intensively investigated self-organizing networks during this phase (Hagan, Demuth and Beale, 1995).

5.5 The 1980s: the Rejuvenation of Neural Networks

The most important two reasons for the stagnation of research and application of neural networks in 1970s were the inherent weakness of the neural networks (i.e. single-layer nets could not solve exclusive-or function) and the powerful computers. During the 1980s, both of these impediments were overcome so that the research in neural networks increased dramatically. Two new concepts including backpropagation and recurrent
networks were responsible for the rejuvenation of neural networks (Hagan, Demuth and Beale, 1995).

The idea of backpropagation was introduced by Werbose in 1974, but it had not drawn public attention. This method was also invented separately and independently by Parker in 1985 and by LeCun in 1986 (Fausett, 1992). The backpropagations can be applied to train multi-layer perceptrons, which can solve the problems of exclusive-or functions.

5.6 The 1990s and Present: the Explosion of Interest in Neural Networks

More and more researchers from different disciplines are becoming interested on neural networks, and many associations have developed. The number of conference, journals and books related to neural networks are also growing (Sales, 1999). Applications being made include pattern recognition, control, signal processing, medicine, speech production, business and even sports. Specifically, neural networks mainly used in the following four types of areas:

a. *Classification*: Medical diagnosis, target recognition, character recognition, fraud detection, speech recognition. In this type application, the target values are normally binary;

b. *Function approximation*: Process modeling, process control, data modeling, machine, diagnostics;

c. *Time Series prediction*: Financial forecasting, bankruptcy prediction, sales forecasting, dynamic system modeling;
d. *Data mining*: Clustering, data visualization, data extraction. This is often unsupervised learning, and find patterns from the data set.

5.7 Neural Network Applications in Agricultural Economics

Some neural network work has been done in agricultural economics. Kastens and his group at Kansas State University, used neural network methods to predict farmer risk preferences (1996), and to model used combine pricing in the Great Plains (1995). A group lead by Joerding from Washington State University, employed feedforward neural network to estimate a crop yield response function (1994). Erba and his colleagues from Cornell University used neural network approaches to compare the costs and efficiencies between cooperative, proprietary, and captive fluid milk processors (1996). Richard, Patterson and Ispelen modeled fresh tomato market margins by using econometric and neural network approaches (1998).
Chapter Two  

Linear Neural Networks: An Econometric Perspective

In this chapter, some classical linear neural networks are investigated. All the models are simple single neuron and single-layer models. Nowadays, they are rarely used in real-life applications as non-linear, more versatile, complex and powerful neural network models are available. Nevertheless, the models such as Perceptron and ADALINE are still the building blocks of the larger networks. To understand those simple linear models is the gateway to non-linear neural networks. Moreover, there are more similarities between those linear neural network models and econometric models (i.e., linear regression model and logistic model).

Three major classical linear neural networks are reviewed, including Hebbian linear associator network, Perceptron and ADALINE. These are compared with the counterparts in econometrics, focusing on model architectures and learning rules.

1. Hebbian Linear Associator Network

Perhaps the most influential work in neural network history is the contribution of Canadian neuropsychologist, Donald O. Hebb. In 1949, in his book, The Organization of Behaviour, Hebb presented a theory of behavior based as much as possible on the physiology of the nervous system. The most important concept to emerge from Hebb's work was his formal statement (known as Hebb's postulate) of how learning could occur. Learning was based on the modification of synaptic connections between neurons.
1.1 The Architecture of Linear Associator

Hebb’s learning rule can be used in combination with a variety of neural network architecture. The reason that we are going to use a linear associator network to present Hebb’s rule is that there are many similarities in the models and learning (estimation) rules between the network and econometric OLS model.

The linear associator is shown in Figure 2.1. The output $y$ can be presented as follows:

$$y = \text{purelin}(b + w^T x) = b + w^T x$$  \hspace{1cm} (2.1)

where $y$ is scalar, which is the output of the model. $b$ is scalar, and it is the bias. $w$ is the vector of weights (i.e., $w_1, w_2, \ldots, w_k$), and $x$ is vector of external inputs (i.e., $x_1, x_2, \ldots, x_k$).

If the samples are included into the model, the general form of the linear network can be written as:

$$y_i = b + w_1 x_{i1} + w_2 x_{i2} + \ldots + w_k x_{ik} \hspace{1cm} i = 1,2,3,\ldots,n$$ \hspace{1cm} (2.2)

Where $b =$ bias, $w_i$ to $w_k =$ weights, $i =$ ith example, $n$ being the size of the training size.

$$y = b + Xw$$ \hspace{1cm} (2.3)
where \( y \) is \( y_1, y_2 \ldots, y_n \) outputs, \( X \) is \( n \times k \) data matrix. \( w \) is \( (k \times I) \) vector.

This mathematic presentation is exactly the same as linear regression model in econometrics.

1.2 Hebbian Learning Rule

Hebb’s training rule was developed from his postulate. As mentioned in Chapter one, “When an axon of cell A is near enough to excite a Cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A’s efficiency, as one of the cells firing B, is increased.” The postulate implies that if a positive input produces a positive output, then the related weight should increase. This suggests that one mathematical interpretation of the postulate could be

\[
\mathbf{w}^{\text{current}} = \mathbf{w}^{\text{previous}} + \mathbf{y}\mathbf{x}^T \tag{2.4}
\]

For the supervised Hebb rule, the output value \( y \) can be substituted by target value \( t \). The equation of (2.4) can be written as

\[
\mathbf{w}^{\text{current}} = \mathbf{w}^{\text{previous}} + \mathbf{t}\mathbf{x}^T \tag{2.5}
\]

From equations (2.4) and (2.5), Hebbian learning adjusts the network's weights such that its output reflects its familiarity with an input. The more probable an input, the larger the output will become (on average). Unfortunately, plain Hebbian learning
continually strengthens its weights without bound (unless the input data is properly normalized). There are only a few applications for plain Hebbian learning, however, almost every unsupervised and competitive learning procedures can be considered Hebbian in nature.

1.3 Estimation Methods

If we assume that the initial weights are all equal to zero, equation (2.5) can be written

$$w = tx^T$$ (2.6)

Where $t = [t_1, t_2, \ldots t_k]$, $x = [x_1, x_2, \ldots x_k]$.

The output of network is

$$a = wx_k = \left( \sum_{q=1}^{Q} t_q x_q^T \right) x_k = \sum_{q=1}^{Q} t_q (x_q^T x_k).$$ (2.7)

Assume that the input $x_k$ are orthogonal and unit length (orthonormal),

$$(x_q^T x_k) = 1 \quad q = k$$ (2.8)

$$= 0 \quad q \neq k.$$

Therefore, Eq. (2.7) can be rewritten

$$a = wx_k = t_k$$ (2.9)

The network output is equal to the target value. This shows that the Hebb rule will produce the correct output for each input, given the input prototype vectors are orthonormal.
However, if the input vectors are not orthogonal, the Hebb rule produces some errors. In practice, the input patterns rarely meet the orthonormal condition. Hence, an alternative approach would to estimate the weights to minimize the following cost function:

\[
F(w) = \sum_{i=1}^{s} ||t_i - wx_i||^2 = e^T e.
\]  

(2.10)

This is least squares method, which is applied to estimate coefficients of linear models in econometrics as well.

Suppose that \( e \) is \((s \times 1)\) vector of the errors; \( t \) is \((s \times 1)\) vector of the dependent values; \( o \) is \((s \times 1)\) vector of the predicted value of the estimated model; \( w \) is the \((n \times 1)\) vector of the coefficients of the model; \( X \) is \((s \times n)\) matrix of the inputs and \( s \) is the number of the observations; \( n \) is the number of independent variables.

Minimize \( w \) \( f(w) = e^T e = (t - o)^T(t - o) \)  

\[
= (t - Xw)^T(t - Xw)
\]

\[
= t^T t - 2w^T X^T t + w^T X^T X w
\]

(2.12)

\[
\frac{\partial f(w)}{\partial w} = -2X^T t + 2X^T X w
\]

(2.13)

The necessary condition for minimizing the sum of squared errors is that the above equation is equal to zero. Setting the preceding equation to zero gives

\[
X^T X w = X^T t
\]

(2.14)

There are three methods solve eq. (2.14).
(1) The normal equation

The form of normal equation is

\[ w = (X^T X)^{-1} X^T t \]  \hspace{1cm} (2.15)

This method is the fastest method among the three but also the least accurate. However, it is adequate when the condition number of input matrix \(X\) is small.

The precondition for this method is that the inverse of multiplication of inputs \((X^T X)\) exists. However, in practice, difficulty may occur since \(X^T X\) could be singular or near singular. The reason for the singularity is that there are two or more inputs are nearly perfectly collinear (Masters, 1993, and Bishop, 1995). In this case, normal equation method would not be applied to solve the issue. Next two methods such as QR and SVD may be good alternatives.

(2) The QR decomposition

To avoid computing the inverse of \((X^T X)\), \(X\) could be decomposed into \(QR\), hence

\[ w = (X^T X)^{-1} X^T t \]

\[ = (R^T Q^T Q R)^{-1} R^T Q^T t \]

\[ = R^{-1} Q^T t \]  \hspace{1cm} (2.16)
where $Q$ is $s \times n$ orthogonal matrix, and $R$ is $n \times n$ upper triangular matrix.

QR decomposition is a standard method for solving the least squares problems, but it is slower.

(3) The Singular Value Decomposition (SVD)

$X$ is decomposed into $U \Sigma V^T$, where $U$ is $s \times n$ orthogonal matrix, $V$ is $n \times n$ orthogonal matrix, and $\Sigma$ is $n \times n$ diagonal matrix with singular values in the diagonal.

$$w = (X^TX)^{-1}X^Tt$$

$$= V^T \Sigma^{-1} U^T t$$  \hspace{1cm} (2.17)

SVD is the most widely used method, and especially, it is a good method for ill-conditioned problems.

2. Single Perceptron Network

In 1943, Warren McCulloch and Walter Pitts introduced one of the first artificial neuron, “Perceptron”. There were some initial simulations using formal logic. Specifically, Perceptron mimics the basic idea behind the mammalian visual system. They were mainly used in pattern recognition even though their capabilities extended a lot more. McCulloch and Pitts (1943) developed models of neural networks based on their understanding of neurology. These models made several assumptions about how neurons worked. Their networks were based on simple neurons which were considered to be binary devices with fixed thresholds. The results of their model were simple logic
functions such as "a or b" and "a and b". Another attempt was by using computer simulations. Perceptron is especially suited for simple problems in pattern classification. It is a fast and reliable network for the problems it can solve. In addition, an understanding of the operations of the Perceptron provides a good basis for understanding more complex networks.

2.1 The Architecture of Perceptron

In fact, it is a pattern classification system. The architecture of one-Perceptron Network is similar to that of ADALINE network, except that its transfer function is hardlim, instead of linear one. Perceptron network is shown in Figure 2.2. The mathematical presentation is as follows

\[ y = \text{hardlim}(b + w^T x) \]  \hspace{1cm} (2.8)

where \( y \) is the vector of output. \( b \) is bias. \( W \) is the matrix of weights and \( x \) is vector of external inputs. The transfer function is the hardlimiter function, which is defined as:

\[ y_i = \text{hardlim}(a_i) = 1 \text{ if } a_i \geq 0 \]  \hspace{1cm} (2.9)

\[ 0 \text{ otherwise} \]

The decision boundary is illustrated in Figure 2.3. The network output will be \( I \) for the region above and to the right of the decision boundary, and the output will be \( 0 \) for the region below and on the decision boundary line. Because the inner product of the
input vector with the weight vector is the same (i.e., 0) on the boundary, the boundary line must be orthogonal to the weight vector.

Perceptron may remind us of the Logit model and Probit model in econometrics. A typical Logit model can be presented as

\[
p_k = E(o=1|x_i) = \frac{1}{1 + e^{-(b + w_1 x_1 + w_2 x_2 + ... + w_n x_n)}} = 1/(1 + e^{-a}) \tag{2.10}
\]

where \(E(o=1|x_i)\) is the probability of event happening under the conditions of \(x\); \(e\) is the familiar base of the natural logarithm.

Equation (2.10) represents what is known as the (cumulative) logistic distribution function. As the output of the linear equation goes from \(-\infty\) to \(+\infty\), the probability of \(p_k\) goes to 0 to 1.

2.2 The Learning Rule

The training technique used is called the perceptron learning rule. The perceptron generated great interest due to its ability to generalize from its training vectors and learn from initially randomly distributed connections.

The weights \(w\) are adjusted by an adaptive learning rule. If the two classes \(C_1\) and \(C_2\) are linearly can be separated by a straight line (in general, a hyperplane), then there exists a weight vector \(w\), with the properties

\[
b + w^T x \geq 0 \quad \text{for inputs belongs to class } C_1 \tag{2.11}
\]

\[
b + w^T x < 0 \quad \text{for inputs belongs to class } C_2
\]

Assuming, to be general, that the perceptron has \(m\) inputs, then the equation \(b + w^T x = 0\), in an \(k\) dimensional space with coordinates \(x_1, x_2, \ldots, x_k\), defines a hyperplane as the switching surface between the two different classes of the inputs.
The core of the perceptron convergence algorithm for adapting the weights is as

If target value \( t = 1 \) and output \( y = 0 \), then \( w^{\text{current}} = w^{\text{previous}} + x \). \hspace{1cm} (2.12)

If target value \( t = 0 \) and output \( y = 1 \), then \( w^{\text{current}} = w^{\text{previous}} - x \).

If \( t = y \), then \( w^{\text{current}} = w^{\text{previous}} \).

From the third condition if the equation (2.12), the learning rule can be unified in to a single expression:

\[
 w^{\text{current}} = w^{\text{previous}} + (t - y)^{\text{previous}} x. \tag{2.13}
\]

This rule can be extended to train the bias by noting that a bias is simply a weight whose input is always 1. The expression for a bias is

\[
 b^{\text{current}} = b^{\text{previous}} + (t - y)^{\text{previous}}. \tag{2.14}
\]

3. Adaptive Linear Neuron Network

ADALINE (ADAptive LInear Element) is a simple single-neuron, which was developed in 1960 by Widrow and Hoff (of Stanford University). Both ADALINE and the Perceptron are considered to be landmark developments in the history of neural network. The ADALINE was an analogue electronic device made from simple components. It was developed to recognize binary patterns so that if it was reading streaming bits from a phone line, it could predict the next bit. It differs from the Perceptron in that its output is not binary (0 or 1) or bivalent (-1 or 1) but is instead continuous and its value can be anywhere on the real line. It was widely used in filtering
and signal processing. Being an early success in neural computation, it bears historical significance. Unlike the Perceptron, it cannot guarantee separation of two linearly separable classes, but it has the advantage that it converges fast and training in general is stable even in classification problems where the two classes are not linearly separable. ADALINE is trained using the LMS (least mean square) algorithm, otherwise known as the delta rule and also as the Widrow-Hoff algorithm. Also note that the widely used backpropagation algorithm is a generalization of the LMS algorithm.

3.1 The Architecture of ADALINE

The ADALINE unit functioned by taking a weighted sum of its inputs and producing an output of 0 or 1 depending on whether some threshold is exceeded. It could be considered to be an idealized version of a nerve cell, which "fires" if the total activity of connected neurons reaches a certain level. Like the Perceptron, the ADALINE is a single-neuron. It has the same architecture as the Perceptron. Single ADALINE units can be assembled in layers to build a larger architecture, a MADALINE network, which can produce complicated functions.

The ADALINE can be presented as follow

\[ y = \text{purelin}(b + w^T x) = b + w^T x \]  

(2.15)

where \( y \) is scalar, which is the output of the model. \( b \) is scalar, and it is the bias. \( w \) is the vector of weights (i.e., \( w_1, w_2, \ldots, w_k \)), and \( x \) is vector of external inputs (i.e., \( x_1, x_2, \ldots, x_k \)).
Equation (2.15) is identical to equation (2.1), except that the target values are continuous values in equation (2.1), whereas target values are binary for ADALINE so that it can be used to classify objects into two categories. As shown in Figure 2.4 Decision Boundary for Two-Input ADALINE, the boundary is always orthogonal to weight vector. For the points on the boundary, the inner product of the input vector with the weight is the same. This implies that these input vectors will all have the same projection onto the weight vector, so they must lie on a line orthogonal to the weight vector. Any points in the right side should be greater than 0 and these in the left side are less than 0.

ADALINE model could be thought as the Linear Probability Model (LPM) in the regime of econometrics, because both of ADALINE and LPM are used to model the dummy dependent (output) variable.

3.2 The Training Rule

ADALINE employed the Least Mean Squares (LMS) learning rule. The learning rule minimizes the mean square error and. That is, the total error E is

\[
F(x) = E[e_t^2] = E[(t_i - y_i)^2]
\]  

(2.16)

Where \( t \) is target values, and \( E[.] \) denotes expected value.

We are not going to derivate the LMS algorithm here. Please reference to Martin Hagan’s book. The LMS algorithm can written as
\[ w_{\text{current}} = w_{\text{previous}} + 2\alpha(t - y)_{\text{previous}} x^T \] (2.17)

and bias equation is

\[ b_{\text{current}} = b_{\text{previous}} + 2\alpha(t - y)_{\text{previous}} \] (2.18)

where \( \alpha \) is a small number, which is in the range of \((0, 1)\).

4. Econometric Commentaries

4.1 Neural networks provide an intuitive representational framework for econometric models

As discussed in previous sections, many econometric ideas and models can be expressed in neural network notation. Specifically, econometric models can be diagrammatically presented. For example, figure 2.1 can be used to present a multivariate linear regression model, and figure 2.2 is able to illustrate logistic models in econometrics if replacing the transfer function with a logistic function. In most cases, the economists may present familiar entities in econometric models by adopting representation, which is used in neural network community (Cheng and Titterington, 1994).

In general, figure 2.5 is a skeleton of a neural network or econometric model. From this diagram, the commonalities of the models from two regimes can be illustrated. Data layer: it is an independent variables (inputs) \((x_1, x_2, \ldots, x_n)\) layer at the far left, which is connected with the model body through the paths (connections) with coefficients (weights) on them.
Model kernel: it is the model’s main body. This part determines the nature/type of models (i.e., linear regress model or ADALINE, Logit model or Perceptron).

Error correction layer: the predicted value (output) is produced by the model body, and sent to the layer of error collected layer, in which the error term is computed through comparing the value of dependent variable (target) and the predicted value (output). The error term is used to optimize coefficients (weights) of the model.

4.2 Neural networks have the same concepts as econometrics but different terminologies

As mentioned in 4.1, the neural network and econometric literatures contain many of same concepts. However, the terminology may be different. For example, sample in econometrics is called as input; model coefficient, intercept, depend variable and independent variable are referred as to weight, bias, input, and output, respectively. Table 1 lists some terms in both econometrics and neural networks. In this thesis, I will use econometric terms to discuss the matters related to econometric models, in contrast, neural network terms are used in network oriented places. I may give out both sets of terms in some places.

4.3 Many common problems of modeling and interference have both neural networks and econometrics

In neural network practice, there are two main aspects to the treatment of modeling data (Cheng and Titterington, 1994):
a. specifying the architecture of a suitable network, including selecting input
variable, transfer function pick, number of hidden layers and number of
hidden neurons in each hidden layer; and
b. training the network to perform well with reference to a training set.

To economists, this is equivalent to

a. specifying a regression model, including variable and equation selection; and
b. estimating the parameters of the model given a set of data.

The differences between the two regimes lie in the ways to deal with the two
steps. Given perceptron and Logit model as examples, when neural network specialists
specify a single-layer perceptron model, they may only focus on the number of nodes and
connections from which a regression model can be got, whereas economists may start
from the mean of conditional probability of response variable on independent variables.

In econometrics, for estimating coefficients, the method that is used most extensively is
the method of ordinary least squares (OLS) (Gujarati, 1995; Greene, 1997). In contrast to
econometrics, the most popular rule is applied for training neural networks is the least
mean squares in which the weights (including the biases) are modified to minimize the
mean squared error. Both neural networks and econometrics use the Least-square
criterion. However, neural network specialists focus on adjusting the weights so as to
reach the minimum point of the cost function surface, whereas econometricians assume a
disturbance distribution and other characteristics (Greene, 1997. These characteristics
include a zero mean value of disturbance, homoscedasticity, a specific variance of the
error term, and others), and often use likelihood. In fact, neural networks involve exactly
the same sort of assumptions as econometric models (Sale, 2002). However, economists
study the consequences and importance of these assumptions while many neural network specialists may ignore them.

4.4 Neural networks and econometrics/statistics are two sides of the same coin

Theoretically, econometric/statistical discriminant methods can be categorized into two general types, the sampling paradigm and the diagnostic paradigm (Riley, 1992). The sampling paradigm estimates the distribution of the predictor variable separately for each class and combines these with the prior probabilities of each class occurring. Then using Bayes theorem, the posterior probabilities of belonging to each class are calculated. It can be represented as

$$p(t, x) = p(x/t)p(t)$$

(2.19)

where $t$ is class, $x$ is vector of predictors. On the other hand, diagnostic paradigm uses the information from the samples to estimate the conditional probability of an observation belonging to each class, based on predictor variables. The mathematical form is

$$p(x, t) = p(t/x)p(x)$$

(2.20)

Crudely put, the regression models (i.e. Logit model) has tended to focus on sampling paradigm, neural network is one of the techniques based on diagnostic paradigm (Jordan, 1995).

For further understanding, we may have binary classification as example to review these to paradigms. Two classes are, $t \in \{t_0, t_1\}$. Given an input vector $x$, we wish to assign it to $t_0$ of the two classes. We use Bayes’ rule and calculate the relevant the relevant posterior probability:

$$p(t_0|x) = \frac{p(t_0|x)}{p(x)}$$
\[
\begin{align*}
  &= \frac{p(x/t_0)p(t_0)}{p(x)} \\
  &= \frac{p(x/t_0)p(t_0)}{p(x/t_0)p(t_0) + p(x/t_1)p(t_1)} \\
  &= \frac{1}{1 + \exp\left\{-\log\left[\frac{p(x/t_0)}{p(x/t_1)}\right] - \log\left[\frac{p(t_0)}{p(t_1)}\right]\right\}} \\
  &\quad \quad \text{(2.21)}
\end{align*}
\]

Equation (2.21) is in the form of logistic function.

For sampling paradigm, we need to assume the class-conditional densities of \( p(x/t) \). In Logit model, we assume that they are multivariate Gaussians with identical covariance matrices. Substitute the Gaussians into the posterior formula, we can have:

\[
p(t_0/x) = \frac{1}{1 + e^{-\left(w^T x + b\right)}}, \quad \text{(2.22)}
\]

where \( w \) is the vector of parameters and it can be calculated:

\[
w = \Sigma^{-1}(\mu_0 - \mu_1), \quad \text{(2.23)}
\]

and intercept of \( b \) is

\[
b = 1/2(\mu_1 + \mu_0)^T \Sigma^{-1}(\mu_1 - \mu_0) + \log\left[\frac{p(t_0)}{p(t_1)}\right]. \quad \text{(2.24)}
\]

where \( \Sigma \) is multivariate Gaussian with identical covariance matrix. \( \mu_0 \) and \( \mu_1 \) are the mean of class \( t_0 \) and \( t_1 \).

For diagnostic paradigm, the conditional probability of \( p(t/x) \) need to be specified. As we already have the assumption of the class-conditional densities are Gaussian. The conditional should be a logistic-linear function of \( x \). Hence the logistic model is the common choice of the two paradigms. However, we can use different form functions including logistic function as transfer functions in neural networks. To complete the
calculation of diagnostic paradigm, we need to specify the probability \( p(x) \). However, it is always provided as evidence. On the other word, we ignore \( p(x) \) when using diagnostic paradigm.

As sampling paradigm (Eq. 2.19) and diagnostic paradigm (Eq. 2.20) are equivalent, the counterpart methods in two paradigms are the same (Reley, 1995). Just as Joran (1995) states, two paradigms are two sides of the same coin. The automatic corollary of which is that neural network and econometric/statistical discriminant methods have the same properties, except that the natural parameterization differ. As future inputs are merely classified from \( p(t/x) \), the neural network classifiers parameterizes directly the quantity of interest (Reley, 1995).
In chapter two, we discussed some linear neural networks. Such networks have just one layer so that they have a number of limitations in terms of the capability of classification and regression. In this chapter, we shall investigate the nonlinear neural networks with multiple layers, which are capable of modeling data with more complex patterns.

1. Linear Separable versus Linear Non-separable

Recall that a linear network such as perceptron, ADALINE model, is able to divide the input space into regions by the linear boundary (hyperplane). Any inputs, that can be separated by a linear hyperplane, are called linear separable. According to Fu (1994), linear separability refers to the case when a linear hyperplane exists to place the instances of one class on one side and those of the other class on the other side of the plane.

To understand linear separable issues and how a linear neural network solve these problems, we shall investigate the simple logic functions such as “OR”, “AND” and “NOT AND” (NAND). These will provide us with some insights for the Multilayer Perceptron (MLP) networks to solve non-linear problems.

Table 3.1 shows logic “OR” function. Note that target value is zero when $x_1$ and $x_2$ are both zero and output is one whenever $x_1$ “or” $x_2$ is a one. Figure 3.1 illustrates the problem graphically. It displays the input space with the output accordingly. The dark
circles (points) indicate the target value is 1, and the light one indicates the target value is 0.

The key is to find a decision boundary which separates the dark points and the light one. Certainly, there are an infinite number of solutions to this problem. However, it seems reasonable to choose the line that falls “halfway” between the two categories of the inputs.

As shown in Figure 3.2, a network with one single neuron, in which a logistic transfer function is employed, could be used to solve this problem. We used MatLab toolbox, neural network, to train this simple network. The training process quickly converged after 7 epochs. The weights are [16.5641, 14.6640], and the bias is -7.222. The network could be presented by the following function,

\[ y = \frac{1}{1 + (e^{7.222 - 16.5641x_1 - 14.6640x_2})} \]  

Recall that the activation level of a neuron is the weighted sum of the inputs, plus a bias value. This implies that the activation level is actually a simple linear function of the inputs. The activation is then passed through a sigmoid (logistic) curve as shown in Figure 3.3. Altering the weights and bias alter the response surface. In particular, both the orientation of the surface, and the steepness of the sloped section, can be altered. A steep slope corresponds to large weight values. Increasing all weight values proportionally gives the same orientation but a different slope. Figure 3.4 is illustrated the same transfer function but with 10 times of weights and bias as those in Figure 3.3. The sigmoid cliff is protected as a line in input space as shown in Figure 3.6. In short, the larger the slope is,
the clearer cut we can get in the input space, so that the more dots in the space could be categorized. In fact, during the process of neural network training, the weights and bias are initiated randomly with very small value. The slope of the surface of transfer function is small. No clear cut decision boundary is available in the input space. Many dots are in the surface with ambiguous states. As the epochs increasing, the weights become bigger and bigger, the projection line of cliff becomes clearer and clearing. Hence, all the dots can be classified correctly.

Finally, the trained network with real weights and bias for solving logic OR can be graphically presented in Figure 3.5.

The input-output relations of logic AND and logic NAND are shown in Table 3.2 and Table 3.3. Intuitively, there are also linear-separable problems like logic OR. We can use the same network shown in Figure 3.2 to map the relations.

The process of the training is very brief. The logic AND network is presented by

\[
y = \frac{1}{1 + (e^{22.2014 - 15.6236x_1 - 15.8425x_2})}, \quad (3.2)
\]

and it is graphically presented in Figure 3.7. The decision boundary is illustrated in Figure 3.8.

Similarly, the logic NAND network can be mathematically presented by

\[
y = \frac{1}{1 + (e^{-22.7121 + 15.3488x_1 + 14.9508x_2})}, \quad (3.3)
\]
and it is graphically presented in Figure 3.9. The decision boundary is illustrated in Figure 3.10.

Now it is time to examine the linear non-separable problems. There are two inputs \((x_1, x_2)\) and the correspondent response is \(t\). The relation between the inputs and the response is illustrated in Table 3.4, which is classical Exclusive-OR (XOR) problem. As Figure 3.11 shows, it is impossible to draw a single line to correctly separate the dots into two categories.

If a regular linear regression model and/or a Logit model are/is applied to model this, the parameter estimates will be zeros, and the measure of “Goodness of fit”, \(R^2\), for the linear model even is very low. The result strongly indicates that this type of issue is beyond the capacity of those conventional econometric models.

If we still use the linear network in Figure 3.2 to train the inputs in Table 3.4. The network performance will never reach the goal. A linear neural network cannot separate the patterns correctly. Historically, this issue was pessimistically addressed by Minsky and Papert in 1969. After that, neural network development stagnated until the 1980s. Then multilayer perceptron (MLP) architecture and its associated learning rules invented, and linear non-separable issues could then be solved by neural networks.

How does a MLP network work can solve nonlinear issues? In neural networks (especially classifiers), a neuron carries out a simple logic function of OR, AND and NAND (Bharath and Drosen, 1994). To solve the linear non-separable problems like XOR, we can decompose the problem into a collection of simple ones. In fact, XOR can be decomposed into following formula:
\[ \text{XOR} = \text{(OR)} \text{ AND (NAND)} \]  \hfill (3.4)

Three simple (linear) logic blocks are formed the XOR problem. One is logic OR, the other is logic NAND. Finally, there is a relation of logic AND between the OR and NAND.

A network for solving XOR problem is illustrated in Figure 3.12. In the hidden layer, one neuron implements the operation of logic OR; while another one implements NAND. In the output layer, a neuron implements logic AND on the outputs of the previous two neurons.

We trained the neural network in Matlab as well. The output is presented in Figure 3.13. The output is consistent to our previous analysis.

Figure 3.14 (StatSoft, 2003) illustrates a typical response surface for a network with only one hidden layer, of two units, and a single output unit, on the classic XOR problem. Two separate sigmoid surfaces have been combined into a single U-shaped surface. The projection of the U-shaped surface in the two dimension input space has two decision lines to separate the dots into right categories (Figure 3.15). Lines 1 and 2 correspond to neurons 1 and 2 in the hidden layer. In conclusion, MLP could overcome solve the linear non-separable problems like XOR problems.

2. **Multilayer Perceptron Architecture**

Multilayer Perceptron network is also referred as to feedforward network. Figure 3.17 shows the structure of a standard multi-layer perceptron network with one hidden layer. The external inputs form the input layer of the network; the outputs are taken from
the output nodes. The middle layer of nodes, visible to neither the inputs nor the outputs, is termed the hidden layer, and unlike the input and output layers, its size is not fixed.

We can use the following formula to present a multilayer perceptron model as illustrated in Figure 3.16.

\[ y = 2f(2w_2f(1Wx + 1b) + 2b) \]  \hspace{1cm} (3.5)

where \( y \) is the output; \( f \) is the transfer function of the output layer; \( w \) is the weight vector of the output layer; \( b \) is the bias of output layer; \( f \) is the transfer function of the hidden layer; \( W \) is the weight matrix of the hidden layer; \( b \) is the bias vector of the hidden layer.

Suppose the transfer function \( f \) is a pure linear function, Eq. (3.5) can then be rewritten as

\[ y = 2w_2f(1Wx + 1b) + 2b \]  \hspace{1cm} (3.6)

Eq. (2.6) is a class of additive/linear models equivalent to the statistic models, projection pursuit models (Friedman and Stuetzle, 1981).

Another statistic model for nonlinear regression, the generalized additive model, is analogous to Eq. (3.5). Hastie and Tibshirani (1990) proposed a series of generalized additive models. These models assume that the mean of the dependent variable depends on an additive predictor through a nonlinear link function. Generalized additive models permit the response probability distribution to be any member of the exponential family of distributions. Many widely used statistical models belong to this general class, including additive models for Gaussian data, nonparametric logistic models for binary
data, and nonparametric log-linear models for Poisson data. The class of general additive models takes the form

\begin{equation}
    y = z f(z) + b
\end{equation}

(3.7)

where the \(f\) are nonlinear function and \(z\) represents the logistic sigmoid function.

3. Learning Algorithms

In a linear model with sum squared error function, for example, this error surface is a parabola (a quadratic), which means that it is a smooth bowl-shape with a single minimum. The error is an explicit linear function of the network weights, and its derivatives with respect to the weights can be easily computed. It is therefore relative easy to locate the minimum point. In network with multiple layers, the error surfaces are much more complex, and are characterized by a number of unhelpful features, such as local minima, which are lower than the surrounding terrain, but above the global minimum, flat-spots and plateaus, saddle-points, and long narrow ravines.

It is not possible to analytically determine where the global minimum of the error surface is, and so neural network training is essentially an exploration of the error surface. From an initially random configuration of weights and thresholds (i.e., a random point on the error surface), the training algorithms incrementally seek for the global minimum. Typically, the gradient (slope) of the error surface is calculated at the current point, and used to make a downhill move. Eventually, the algorithm stops in a low point, which may be a local minimum, but hopefully may be the global minimum.
3.1 The Standard Backpropagation Algorithm

The best-known example of a neural network training algorithm is backpropagation (Haykin, 1994; Fausett, 1994). Backpropagation means “backpropagation of error”. Strictly, backpropagation refers to the method for computing the gradient of the case-wise sum of squares error function with respect to the weights (and biases) for a feedforward network, a straightforward but elegant application of the chain rule of elementary calculus (Warle, 1999). Backpropagation is a euphemism of the generalized delta rule, and is an extended LMS method, which observes the law of steepest descent gradient. The gradient vector of the error surface is calculated. This vector points along the line of steepest descent from the current point, so we know that if we move along it a short distance, we will decrease the error. A sequence of such moves will eventually find a minimum point. However, the difficult part is to decide how large the steps should be. Large steps may converge more quickly, but it may also overstep the solution or go off in the wrong direction if the error surface is very eccentric. A typical example of these cases, in neural network training, is where the progresses very slowly along a steep, narrow, valley, bouncing from one side across to the other. In contrast, very small steps may go in the correct direction, but they also require a large number of iterations. It takes long time to converge. In practice, the step size is proportional to the slope so that the algorithms settles down to a special constant, the learning rate. The correct setting for the learning rate is application-dependent, and is typically chosen by experiment.

The training progress of the standard algorithm of backpropagation is given as follows, and the procedures are illustrated in Figure3.14. Here, assume that there are \( n \) inputs, \( m \) neurons in the hidden layer, and \( l \) neuron in the output layer.
Step 0: *Weights initiation.*

Randomly initialize weights and biases, respectively, on all the connections (paths) in the network.

*Feedforward:*

Step 1: *Raw inputs feedforwarding.*

Each input \( (x: x_1, x_2, x_3, \ldots, x_n) \) is sent to all neurons in the hidden layer.

Step 2: *Summation at the summer in each neuron in the hidden layer.*

Each neuron \((j)\) computes the inner product of its weights and bias and the raw inputs

\[
ij = (w_j^T x + b_j), \quad (3.8)
\]

where \(ij\) is the inner product of the summer of the jth neuron in the hidden layer; \(w_j\) is weights vector of jth neuron, which is a \( n \times 1 \) vector; \(b_j\) is the bias for the jth neuron.

Step 3: *The inner product transferring in the hidden layer.*

The inner product from the summer is squeezed by the transfer function \((f(\cdot))\). In other words, each neuron \((j)\) will finish the computation of

\[
o_j = f(ij), \quad (3.9)
\]

Then its output \((o_j)\) is sent to the output layer as an input.

Step 4: *Summation at the adder in each neuron in the output layer.*
There is only one neuron in the output layer, and the neuron sums its weighted inputs as

\[ z_i = z\mathbf{w}^T \mathbf{o} + z b, \quad (3.10) \]

where \( z_i \) is the inner product of the summer of the output neuron; \( z\mathbf{w} \) is weights vector of the output neuron, which is a \( m \times 1 \) vector; \( \mathbf{o} \) is the output vector from the hidden layer, which is a \( m \times 1 \) vector; \( z b \) is the bias for the output neuron.

Step 5: The inner product transferring in the output layer.

The transfer function \((z f(\cdot))\) in the output neuron computes its output \((z\mathbf{o})\). It completes the computation of

\[ z\mathbf{o} = z f(z_i), \quad (3.11) \]

where \( z\mathbf{o} \) is the output of the neuron, and it is also the output of overall network.

Step 6: Squared error computation.

After the output neuron computed the output, then the error information term \( \{e_k = t_k - z\mathbf{o}_k\} \) can be calculated. For all input patterns (observations in econometric models) the performance index (cost function), \( v \), for the network is

\[ v = \frac{1}{2} \sum_{k=1}^{k} (t_k - z\mathbf{o}_k)(t_k - z\mathbf{o}_k) \quad (3.12) \]

Where, \( k \) is the \( k \)th example (a pair of input and target value);
$s$ is the number of examples (observations);

$t_k$ is the $k$th target value;

$z_o$ is the error term for the $k$th example.

**Step 7: Halt condition checking.**

If the error is less than a specific value, the training stops, and the weights and biases are exported. Otherwise, the training proceeds to the next step.

**Backpropagation:**

**Step 8: Sensitivity computation for the output layer.**

Sensitivity is the change of the index of performance responding to the input in the output layer. $\frac{\partial v}{\partial z_i}$ is defined as the sensitivity of the output layer. Because only one neuron is set in the output layer, the sensitivity term $z_\delta$ is scalar. The sensitivity is represented by

$$z_\delta = \frac{\partial v}{\partial z_i} = - z_i f'(z_i)(t - z_o),$$

where $f''$ is the derivative of output of the transfer function respective to input $z_i$. It can be represented by $d(f(z_i))/dz_i$.

**Step 9: The correction terms for the weights calculation in the output layer.**

According to steepest descent gradient rule, the correction terms of the weights and biases between the output layer and the hidden layer are calculated as
\[ \Delta_2 w_j = -\alpha \frac{\partial v}{\partial_2 w_j}, \quad (3.14) \]

\[ \Delta_2 b = -\alpha \frac{\partial v}{\partial_2 b}, \quad (3.15) \]

where, \( \alpha \) is the learning rate, Normally, the learning rate falls between (0, 1). Where \( j \) is the \( j^{th} \) weight between the hidden layer and the output layer.

Because \( \partial v/\partial_2 w_k \), \( \partial v/\partial_2 b \) can be written as,

\[ \frac{\partial v}{\partial_2 w_j} = \frac{\partial v}{\partial_2 i} \frac{\partial_2 i}{\partial_2 w_j} = \frac{\partial v}{\partial_2 i} i^{O_j}, \quad (3.16) \]

\[ \frac{\partial v}{\partial_2 b} = \frac{\partial v}{\partial_2 i} \frac{\partial_2 i}{\partial_2 b} = \frac{\partial v}{\partial_2 i}. \quad (3.17) \]

Hence, the correction terms are

\[ \Delta_2 w_k = -\alpha_2 \delta_1 o_k \quad (3.18) \]

\[ \Delta_2 b = -\alpha_2 \delta. \quad (3.19) \]

Step 10: The sensitivity \((2 \delta)\) propagation.

The sensitivity term of the output is sent back to the hidden layer. Each hidden unit gets its delta input \((2 \delta)\) from the output layer, and then the sensitivity \((1 \delta)\) is
computed for updating the weights in the hidden layer. Because there are \( m \) neurons in the hidden layer, the sensitivity term \( \delta_1 \) is a vector. Actually, the relationship between the two sensitivities is

\[
\delta_1 = F_1'(i_1)w_2\delta
\]  

(3.20)

note that, \( \delta_1 \) is a \((m \times 1)\) vector. \( w_2 \) is a \((m \times 1)\) vector. \( \delta \) is a scalar. \( F_1'(i_1) \) is \( m \times m \) matrix,

\[
F_1'(i_1) = \begin{bmatrix}
F_1'(i_1) & 0 & \cdots & 0 \\
0 & F_1'(i_2) & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & F_1'(i_m)
\end{bmatrix}
\]  

(3.21)

Step 11: The correction terms for the weights calculation in the output layer.

Now, the correction terms for the weight and bias are calculated. They are

\[
\Delta_1 w_k = -\alpha_1 \delta x_k
\]  

(3.22)

\[
\Delta_1 b_k = -\alpha_1 \delta
\]  

(3.23)

Update weights and biases:

Step 12: The output unit updates weights and biases.

At the end of each iteration, the weights and bias will be renewed. The renewed weights and bias are as follows:
where, “current” means the current iteration, and “previous” means the previous iteration.

Step 13: Each hidden unit updates its bias and weights as follows:

\[
2w_k^{\text{current}} = 2w_k^{\text{previous}} + \Delta 2w_k = 2w_k^{\text{previous}} - \alpha_2 \delta_j o_k \quad (3.24)
\]

\[
2b_k^{\text{current}} = 2b_k^{\text{previous}} + \Delta 2w_k = 2b_k^{\text{previous}} - \alpha_2 \delta, \quad (3.25)
\]

Step 14: Return to step 2 for next iteration.

3.2 Levenberg-Marquardt Algorithm

A standard backpropagation algorithm observes the law of the steepest descent gradient. It is a simple law for numerical computation in the nonlinear system. However, it is slower to converge than Newton method and conjugate method. Because of this inherent weakness, the standard backpropagation algorithm has following drawbacks (Hagan, Demuth and Beale, 1996):

a. **It is slow to converge.** The standard backpropagation method uses the steepest descent method, which is a slower learning law than others such Newton’s law and the Conjugated law. Especially, with the layers and neuron added in the network, more weights are need estimated. So, the training speed may be very slow if the standard backpropagation algorithm is applied.
b. *It may not converge at all.* The steepest descent method will converge under the condition that one layer has linear transfer functions, because the quadratic surface of the performance index function has a single unique minimum point, and the Hessian matrix is constant. However, the performance index function does not have the above properties when the multi-layer feedforward network has more than one layer with a nonlinear transfer functions. The function may not be quadratic, the curvature of the surface of the function may differ from region to region, and even it may have many local minimum points.

Because of those drawbacks, alternative backpropagation algorithms must be found. Another backpropagation algorithm, the Levenberg-Marquardt algorithm, based on an approximation of Newton’s method, is well suited for training multilayer feedforward networks. According to Sarle (1999), Levenberg-Marquardt algorithm is often better optima for a variety of problems than do the other usual methods. Levenberg-Marquardt algorithm is faster to converge than others. According to our experiments, this statement is true. Hence, let us lay out this algorithm.

In certain sense, the Levenberg-Marquadt method evolved from Newton’s method, which can be generally defined as

\[
    w_{\text{current}} = w_{\text{previous}} - (H_{\text{previous}})^{-1} \cdot g_{\text{previous}},
\]

(3.28)
where, $w_{\text{current}}$ is current iteration weights, $w_{\text{previous}}$ is the weights after previous iteration. $H_{\text{previous}}$ is the Hessian matrix of second partial derivatives at previous points in the surface of the performance index function. And $g_{\text{previous}}$ is the gradient vector.

Actually, Newton’s method can find the minimum of a quadratic function in one step. However, in practice, it needs to compute and store the Hessian matrix. Hence, it is a extremely inefficient method. Especially, the quality of the new point heavily depends on the Hessian matrix consistency (Masters, 1993). If the performance index is not pure quadratic function, and the Hessian matrix is not constant, so, the minimum point found by Newton’s method may not real one.

The Levenberg-Marquardt method can be noted as follows:

$$w_{\text{current}} = w_{\text{previous}} - (H_{\text{previous}} + \lambda I)^{-1} g_{\text{previous}}, \quad (3.29)$$

where, $\lambda$ is a number; and $I$ is the identity matrix.

This formula is similar to that of Newton’s method. However, the flexibility of the method can be realized by changing the value of $\lambda$. When the Hessian matrix is relative constant and error becomes small, $\lambda$ can take a number close to zero, and then $\lambda I$ can be ignored. The method becomes Newton’s method. At some point, the Hessian matrix changes substantially. The value of $\lambda$ can be large, and then the Hessian matrix will be swapped out. The method becomes steepest descend method. This is the beauty of the Levenberg-Marquardt method.

For avoiding computing Hessian matrix, Gauss-Newton method can be used. So, it leads to the Levenberg-Marquardt method.
\[ w_{\text{current}} = w_{\text{previous}} - (J^T J + \lambda I)^{-1} J^T (t-o), \] (3.30)

where \( J \) is the Jacobian matrix. The matrix could be very complicated as the number of targets (number of neurons in output layer) increases. For simplicity and ease of understanding, the following Jacobian matrix is based on one neuron. The form is

\[
J = \begin{pmatrix}
\frac{\partial v_1}{\partial w_1} & \frac{\partial v_1}{\partial w_2} & \cdots & \frac{\partial v_1}{\partial w_m} & \frac{\partial v_1}{\partial b}
\frac{\partial v_2}{\partial w_1} & \frac{\partial v_2}{\partial w_2} & \cdots & \frac{\partial v_2}{\partial w_m} & \frac{\partial v_2}{\partial b}
\vdots & \vdots & \ddots & \vdots & \vdots \\
\frac{\partial v_s}{\partial w_1} & \frac{\partial v_s}{\partial w_2} & \cdots & \frac{\partial v_s}{\partial w_m} & \frac{\partial v_s}{\partial b}
\end{pmatrix},
\] (3.31)

where, \( v \) is error \((t-o)\); \( s \) is the number of observations; and \( m \) is the number of inputs.

Both the standard backpropagation algorithm and Levenberg-Marquardt algorithm have similar procedures. The only differences are in the sensitivity and its associated correction term computations. Moreover, the key step in the Levenberg-Marquardt algorithm is the computation of the Jacobian matrix.

Here is the formula of computing sensitivities in the Marquardt-Levenberg algorithm. For a detailed discussion, see Seber and Wild, 1989, Hagan, Demuth and Beale, 1996 and others.

a. The sensitivity for the output layer

\[
\zeta \delta^{LM} = \frac{\partial e}{\partial \zeta i} = -\zeta f' (\zeta i),
\] (3.32)
where, $e = t - \mathbf{o}$ is the error vector.

b. The sensitivity for the hidden layer

$$i \delta_{LM} = i F'(i)_{2} w_{2} \delta_{LM}$$  \hspace{1cm} (3.33)

where, $i F'(i)$ is the same as the matrix in the standard propagation algorithm.

Although the Levenberg-Marquardt algorithm has better properties in convergence speed and stability than the standard propagation algorithm, it also has some limitations. First, the Levenberg-Marquardt method is also a gradient-driven method, so, it will have problems when the surface of performance index function is relatively flat (Masters, 1993). Second, the computation of the Jacobian matrix is still difficult and time consuming, so the performance of the Levenberg-Marquardt method will be poor if a network has hundreds of weights. However, if a network has few weights, the Levenberg-Marquardt algorithm is often a good choice. In practice, the number of weights for a network is generally not very big, hence, the Levenberg-Marquardt algorithm is widely accepted and applied in the field of neural networks.

4. Statistical Commentaries

4.1 Similarities and Difference Between Multilayer Perceptron and Nonlinear Statistical Models
As discussed before, the multilayer perceptron model with one hidden layer is remarkably similar to some nonlinear statistic models such as projection-pursuit regression and generalized additive models. The most obvious statistical interpretation of multilayer perceptron is that it provides nonlinear regression function that is estimated by optimizing some measure of fit to the training data. If data are noise-free, then the exercise is one of function approximation (Cheng and Titterington, 1994). However, there are some differences between the multilayer perceptron and those nonlinear statistic models (Bishop, 1995). We shall discuss the differences at length, between multilayer perceptron network and the most similar nonlinear regression statistic model, projection pursuit regression.

One important difference is that each hidden node in a projection-pursuit regression allows for a different transfer function, whereas the counterparts of multilayer network use the same activation function. The second difference is that the forms of transfer functions in projection-pursuit regression are not defined in advance, but are determined from the data in the process of training. This means that the pattern of data determine the forms of transfer functions. For the multiplayer-perceptron model, its transfer function is given in the architectural design. Furthermore, sigmoid functions are often used since they have some desirable properties as outlined in Chapter one.

A third difference is that all of the weights in multilayer perceptron networks are updated and optimized simultaneously, whereas those in projection-pursuit regression models are optimized cyclically and in groups. Overall, in the training process, the weights of the hidden layer are optimized first and one by one. Then the transfer
functions are determined. The last step is to optimize the weights in the first layer. This process is repeated until the error function reaches the minimum point.

4.2 The Advantages and Disadvantages of Multilayer Perceptron versus Nonlinear Statistical Models

Neural network models have some useful properties. The first of these is nonlinearity. Although the neural networks can be linear or nonlinear in terms of neuron properties, the form is not given priori, and it is very flexible. Because of its flexibility, a neural network model is a potentially promising tool for studying complicated economic phenomena, which may have higher latitude than conventional methods. The second property is Input-Output Mapping (Haykin, 1999). Although there are different types of supervised learning rules, the essence of those rules is input-output mapping. Each example consists of a unique signal (input) and a corresponding response. At the beginning of the training, a set of weights is selected randomly, and it is then adjusted to minimize the difference between the desired output and the actual response of the network produced by the input. This procedure is repeated until the weights are convergent. The weights do not convey much meaning by themselves. The only objective is to minimize the difference during the mapping. This property may bring some positive effects, and negative ones as well.

The third property is their non-parametric nature. Associated with the property of input-output mapping, neural network is a non-parametric method of statistical inference. It permits estimation of the decision boundary for classification problems and
approximate the functions in the regression by using the examples, without prior assumptions for the distribution of the inputs and the model itself.

4.2.1 The Advantages of Neural Networks

Therefore, neural networks have some advantages over statistic/econometric models as follows:

a. Flexibility. As opposed to a simple regression where linearity in parameters is imposed on the data, neural networks are nonlinear in nature. As for nonlinear econometric procedures, these in most cases, require specific assumptions regarding the non-linearity structure of the investigated relationship. For example, the popular Cobb-Douglas function form imposes zero production as zero input(s), unbounded output, constant unitary elasticity of substitution between inputs, and constant input elasticity. All of these assumptions are unrealistic (Joerding et al., 1993). A neural network system does not require such prior knowledge. This characteristic enables the neural network approach to capture relationships with a higher degree complexity and a higher degree of accuracy (Shachmurovem, 2004).

b. Broad Application. Because neural network models require no assumption and prior constraints, neural network is a more reliable and flexible model that is suitable to more modeling cases. In fact, a prior assumption or constraint in econometric models may not be valid at all in some cases. Specifically, the estimators in econometric models may be highly biased in that they assume an a priori model (e.g., linear relationship). However, neural networks are analogous to nonparametric regression models on that they make no a priori assumptions about the problem (i.e., let the data speak for itself), so the
model may fit the data very well. (Refenes, Zapranis and Francis, 1994; Shachmurovem, 2004).

c. Directly parameterize relationships of interest. As neural network methods are of diagnostic paradigm, they directly parameterize the relationship of interest, p(\(t/x\)). They are independent from input distribution and they are more robust to uncertainty data so that they have more chance of ‘generalization’ (Jordan, 1995; Riley, 1995). Many comparative empirical studies have shown that neural networks outperform conventional methods in terms of prediction accuracy (Altman, 1994; Erba and et al., 1996; Kastens and et al., 1996). They improve not only the accuracy of in-sample simulation but also that of out-of-sample prediction as well. This advantage is more evident in dealing with “dirty” data that do not distribute well.

4.2.2 The Disadvantages of Neural Networks

Neural network models have some weaknesses. A major and inherent problem of neural networks is that the internal structure cannot tell researchers how it processed the input information or reached a conclusion. That process is represented only in the matrix of connection weights, which cannot be reliably translated into meaning or interpretation (Hawley, Johnson and Raina, 1990). In other words, the weights in the model do not have economic meanings whereas the coefficients in econometric models can often be interpreted in an economic context. This is because econometric models generally have theoretical foundations whereas neural networks are based on emulation or simulation. The latter is much weaker from the standpoint of identification of structure and explanatory power.
However, some economists such as Joerding (1993) and others consider that this weakness of neural networks should not be overstated since neural networks can address many of the same questions for which other flexible nonlinear models are sometimes applied.

Secondly, related to the first weakness, although there are weights corresponding to each input, they do not indicate to the researcher anything about the inputs. Thus, that the quantitative relationship between the dependent and independent variables (inputs) is not as obvious as in an econometric model. Furthermore, the methods of significance testing cannot be applied to determine which independents (inputs) could be incorporated into the model. A sensitivity analysis can be carried out for each input to determine its marginal effect, but sometimes it is often misleading (Sarle, 2002). Hence, just as DeTienne and Lewis (2003) mentioned, “When results are sought without an extraordinary need to understand the parameters behind them, then neural networks are most beneficial. When researchers need both an accurate prediction and an understanding of the predictive parameters, both ANNs and Regression should be used in conjunction.”

Third, neural networks are data-dependent, so the algorithms are only as good as the data used to apply them (Meade, 1995). In this light, neural networks may be thought as weakly deterministic systems that converge to a predictable eigen-behavior (Shachmurovem, 2004). Furthermore, because of the complexity of error surface and randomness in the weights initiation, the learning results may not be stable. In other words, the weights from the model training cannot be identical for any two different training even with the same patterns (observations) for the same model.
The last weakness of a neural network should be mentioned is that training a network may be highly time-consuming. For a straightforward econometric model, estimation can be completed in seconds, but for training a neural network, it may take hours or even days to get the job done.

In summary, as discussed in chapter two, neural network and econometric and statistical methods are two sides of the same coin. Neither one is inherently better, rather they have complementary advantages and disadvantages (Jordan, 1995). There is considerable overlap between the fields. Neural networks can be seen as an addition to or as alternative to the existing statistical and econometric tool box. Because of their dynamic composition, neural networks are efficient at analyzing problems where that data are incomplete or fuzzy, and accurate predictions are sought more heavily than explanations.
Chapter Four

Neural Network Design and Its Pragmatics

1. Number of Hidden Layers

A breakthrough was to add hidden layers to networks. Because of added hidden layers, networks can solve the linear non-separable problems and non-linear regression problems. The question is how many hidden layers are sufficient, and more is not always better.

According to the Kolmogorov’s theorem (1957), one hidden layer is adequate for most practical issues. The Kolmogorov’s mapping neural network existence theorem states that any continuous function from the $n$ dimensional cube $[0, 1]^n$ to the real numbers $\mathbb{R}$ can be implemented by only one hidden layer network with $2n + 1$ neurons, and $m$ neurons in the output layer (Fu, 1994, Bishop, 1995, Reed and Marks, 1998). The transfer functions used in the hidden layer and the output layer accordingly are

$$h_j = \sum_{i=1}^{n} \lambda_j f(x_i + k\epsilon) + k$$
$$y_j = \sum_{i=1}^{2n+1} g_j(h_i)$$  \hspace{1cm} (4.1)

where, $j$ represents the $j^{th}$ neuron in the hidden layer and output layer; $0 < \lambda < 1$ are constant.
The problem is that the proof for this theorem is less constructive. It fails to tell researchers how to find the non-linear functions of $f$ and $g$, and the designed number of neurons is not suitable for many practical cases.

According to another neural network scientist, Masters (1993), for function approximation, if the function consists of a finite collection of points, is continuous and defined on a compact domain (it says the inputs are bounded), or is discontinuous but it can be met in real life, the function can be approximated at any desired accuracy level by a neural network with only one hidden layer.

Now, people in the field of neural networks have the common acknowledgement that one hidden layer is often adequate. More than one hidden layer is never beneficial. The reasons are as follows:

a. The training speed is lower (Sarle, 1999). More layers than necessary mean that more computation is needed. So the direct negative effect for more hidden layers is decrease the speed of learning.

b. More hidden layers make the gradient less stable (Master, 1993). Because the gradient-oriented algorithms can perform better if the gradients maintain relative stable when the weights change.

c. More hidden layers increase the number of local minima at the surface of the performance function (Master, 1993). Too many hidden layers make the surface more complex, and the curvature differs from region to region. Hence, it increases the probability of sticking at a local minimum.

d. More hidden layers may eventually be harmful to out-of-sample prediction. It is possible that too many layers may cause an over-fitting problem. Therefore,
a network with many layers can memorize the pattern of the inputs, but it is not good for generalization.

Although one hidden layer is sufficient for approximation of most functions, it is not always enough. It may be necessary to have more than one hidden layer to approximate some functions such as those shaped like a zig-zag as shown in Figure 4.1 (Master, 1993, Reed and Marks, 1998). Furthermore, when two hidden layers are set in a network, there are more neurons in the first hidden layer (close to raw input layer) than in the second hidden layer.

2. Number of Neurons

The number of input and output units is defined by the problem, but the number of hidden units to use is far from clear. Again, there is no convincing theory or formula to calculate number of neurons in the hidden layer(s). There are lots of “rules of thumb” in the literature, such as determining the number of neurons by the numeric relation between input size and output size, number of principle component needed and so on (Zurada, 1992, Sarle, 1999). But these rules can be misleading because they ignore the number of training cases, the amount of noise in the targets, and the complexity of the function.

Just like number of hidden layers, more is not necessarily better. For example, there are $s$ inputs in a classification case. If $s-1$ neurons are set in the hidden layer, those patterns are completely separable, because a line can be always found between any two patterns (Reed and Marks, 1998). However, this design is not efficient for learning but extremely bad for generalizing as well.
Although there are no existing rules to follow, the following factors should be concerned when number of neurons in the hidden layer is to be determined. These factors are: the numbers of input and output units, the number of training cases, the amount of noise in the targets, the complexity of the function or classification to be learned, the architecture, the type of hidden unit activation function and so on.

There is a rule of thumb to try a different number of neurons when a network is trained, and then select the appropriate number according to initial testing. “Over-fitting” means the network’s outputs fit the target values too closely. The weights have already memorized even the specialties (the noises in the target), so that the ability of generalization, which is the ability to recognize patterns learned from the training samples decreases dramatically.

Over-fitting occurs because the training time is too long (too many iterations) or because there are too many hidden layers and/or too many neurons in the hidden layer(s). The network that is too complex may fit the noise, not just the underlying pattern, leading to over-fitting. For example, in an extreme case, the number of weights is more than the number of inputs (observations). The net will memorize all the input vector’s target requirements. This kind of design will have no power of generalization (Azoff, 1994). As Figure 4.2 illustrates, the learning error (in-sample simulation) decreases as the learning time or the number of layers/neurons increases, however, the testing error curve (out-of-sample prediction error) takes the shape of “U” letter. It indicates that too much training time or too many layers and neurons in the networks are not appropriate for generalization. Therefore, to carefully select number of the hidden layers and their neurons is crucial for neural network design.
3. Cross-Validation

Cross-validation calculates generalization error for selecting an appropriate network design. Actually, the error minimized by the training process is not the error measure of ultimate interest (Smith, 1993). Bishop (1995) states, “Since our goal is to find the network having the best performance on new data, the simplest approach to the comparison of different networks is to evaluate the error function using data which is independent of that used for training. Various networks are trained by minimization of an appropriate error function defined with respect to a training data set. The performance of the networks is then compared by evaluating the error function using an independent validation set, and the network having the smallest error with respect to the validation set is selected.” Why the architecture making the training error smaller is not necessary to make the out-of-sample testing error smaller? The reason is that there is noise in the training data (Smith, 1993). Since the noise exists, the weights in the elected network based on smaller training error may not optimal. Therefore, cross-validation is required in network design.

There are two kinds of cross-validations (Reed and Marks, 1998, Sarle, 1999). Here we call them “two-set testing” and “Three-set validation”. For the two-set testing, the examples (observations) are divided into two sub-groups. One group data are used for training the different models (architectures with variant layers and number of neurons), and the other group of data (out-of-sample data) are applied to calculate the generation errors. According to the values of errors, the architecture producing the smallest generation error is selected as the optimal design.
Regarding the three-set validation, the samples are divided into three sub-sets. The purpose of the first two groups is identical to that of the two-set test. The third group of data is used to validate the selected architecture. One more validation can make the selected architecture more robust and more reliable.

Here the question is “what are the proportions of samples in each set?” There is no definite answer just like the number of neurons in the hidden layer. It depends on the size of the samples. If the sample size is small, and too many of them are allocated to the testing or validation set, it is very likely to make the learning over-fitting (Reed and Marks, 1998). On the opposite, if fewer samples are used to test or validate, the variation of the generalization error may be large, and the selection of optimal architecture is not reliable.

4. Data Re-scaling

Data rescaling is also referred to as “data standardization” or “data normalization”, in the context of neural networks. Rescaling data means to add a constant and/or then multiply by a constant. Rescaling is similar as changing the units of measurement of the data. In the neural network literature, “data normalizing” often refers to rescaling input data to make all the elements lie a specific range (e.g. between -1 and 1).

Although some computer scientists such as Smith (1998) state that it is not required to rescale the data, rescaling data may have some benefits for the network training. The reasons are:
a. Rescaling of an input vector can make the surface of the performance index smoother, make training faster and reduce the chances of getting stuck in local optima (Masters 1993, Sarle, 1999).

b. Rescaling of inputs into uniform range initially equalizes the importance of input variables. For example, if one input has an order of magnitude of 1,000,000, and the other is approximately 0.000001, so the weights associated with the first input variable are small but very large for the second input (Masters 1993). In some algorithms, the contribution of second input to the network may be swamped by the contribution of the first one (Sarle, 1999).

c. Although it is not necessary to rescaling the target values for the network with an unbounded transfer function in the output layer, and especially, it is nonsense to re-scale the target values in the case of prediction, in some cases such as network with bounded transfer function in the output layer, target variable also need be re-scaled.

In fact, it is algorithm dependent to rescale data. According to Sarle (1999), Steepest descent-like algorithms are very sensitive to scaling. Further, the learning process is slower to converge if the Hessian matrix is ill-conditioned. Hence, rescaling data is an important consideration for gradient-descent methods such as standard backpropagation. All the inputs data have been rescaled in the experiments presented here.

Two types (Azoff, 1994) of data rescaling methods are applied in most neural network applications.
4.1 Along Norm[0,1]

This type data rescaling makes all the data fall into the range of 0 to 1, inclusively. The formula is as follows:

\[
    t_{knorm}^k = \frac{(t_k - \text{min})}{(\text{max} - \text{min})},
\]

(4.2)

where \(t_k\) is the original data serial, and \(t_{knorm}^k\) is the re-scaled data serial. \textbf{max} and \textbf{min} refer to maximum and minimum values for the variable, respectively.

4.2 Along Norm [-1,1]

This type of data rescaling is similar to previous one, however, it make the data fall into the range of -1 to 1, inclusively. The formula is

\[
    t_{knorm}^k = \frac{2(t_k - \text{min})}{(\text{max} - \text{min})} - 1.
\]

(4.3)

In practice, the second data rescaling should be employed, since for nearly all transfer functions inputs could be positive or negative. If the data are rescaled into the range of [0, 1], this kind of rescaling could narrow the variation of the inputs and eventually, deteriorates the quality of learning.

5. Evaluation and Metrics for Prediction Accuracy

There are many ways to measure forecast accuracy. The evaluation results may not concur if different measures are used. The measures can be placed in three categories: (1) absolute errors, (2) squared errors and (3) directional errors. Both absolute error and squared error measures treat the positive and negative errors the same. The absolute error
method gives all errors the same weights no matter if the error is small or large. The squared error method gives most weight to the large errors, and the least weight to the small errors. Because of this, the measures obtained from the forecast techniques cannot always be the same. To measure the directional errors is to assess the forecast techniques from another angle. Directional errors occur when the sign of forecast values is not consistent with the corresponding sign of the values in the data set. A measure of directional error is the number of pairs in which the forecast values and actual values have different signs. Since each category of measure emphasizes a specific property of errors, different measures of forecast accuracy could lead different conclusions.

To better understand the properties of the forecast methods, we used series forecasting error measures.

5.1 Absolute error measures
The common absolute error measures are (1) the sum absolute error (SAE), (2) the mean absolute error (MAE), and (3) the mean absolute percentage error (MAPE).

Suppose that the forecast error of point $k$ is

$$e_k = t_k - o_k \quad (4.4)$$

where $o_k$ is the forecast value and $t_k$ is the actual value at point $i$.

Then the sum absolute error (SAE) is

$$SAE = \sum_{k=1}^{s} |e_k| \quad (4.5)$$

where $s$ is the number of observations.

The mean absolute error (MAE) is
\[ MAE = \frac{SAE}{s} \]  \hfill (4.6)

Both the \( SAE \) and the \( MAE \) measure the magnitude of the errors obtained to form the forecast. Forecast errors may sometimes appear large in an absolute index, but are acceptable relative to the magnitude of the observations contained in the data set. In this sense, relative index is more informative. Thus, mean absolute percentage error (\( MAPE \)), measures the magnitude of absolute errors in relative terms.

\[ MAPE = \frac{\sum_{k=1}^{s} \left( \frac{|e_k|}{t_k} \right)}{s} \]  \hfill (4.7)

5.2 The indices of squared errors

The most popular index for measuring squared errors is the sum of squared errors (\( SSE \)). The \( SSE \) is employed in many neural network algorithms, because its quadratic form is useful to finding the optimum solution. The sum of squared errors is

\[ SSE = \sum_{k=1}^{s} e_k^2 \]  \hfill (4.8)

The mean square error (\( MSE \)) refers to average square error for a set of prediction points.

\[ MSE = \frac{SSE}{s} \]  \hfill (4.9)
The root mean square error (RMSE) is the square root of MSE. It re-scales the errors in order to keep the errors’ dimension as the predicted value.

\[ RMSE = (MSE)^{\frac{1}{2}} \]  

(4.10)

Theil decomposed mean square error as

\[ \frac{1}{s} \sum_{k=1}^{s} (t_k - o_k)^2 = (\bar{t} - \bar{o})^2 + (d_t - d_o)^2 + 2(1 - r)d_t d_o \]  

(4.11)

Where, \( \bar{t} \) and \( \bar{o} \) are means of the predicted values and the actual values, respectively, and

\[ \bar{t} = \frac{\sum_{k=1}^{s} t_k}{s} \]  

(4.13)

\[ \bar{o} = \frac{\sum_{k=1}^{s} o_k}{s} \]  

(4.12)

d\(_o\) and d\(_t\) are the standard deviations of the predicted values and the actual values, which are defined as

\[ d_o^2 = \frac{\sum_{k=1}^{s} (o_k - \bar{o})^2}{s} \]  

(4.14)
\[ d_i^2 = \frac{\sum_{k=1}^{s} (t_k - \bar{t})^2}{s} \quad (4.15) \]

and \( r \) is the correlation coefficient of predicted and actual values:

\[ r = \frac{1}{s} \sum_{k=1}^{s} (o_k - \bar{o})(t_k - \bar{t}) \quad (4.16) \]

For convenience in this study, the error components are

\[ U^M = s(\bar{t} - \bar{o})^2 \quad (4.17) \]

\( U^M \) is called the bias proportion, which indicates errors in central tendency.

\[ U^D = s(d_t - d_o)^2 \quad (4.18) \]

\( U^D \) is called variance proportion, which reflects the difference between the variances.

\[ U^C = 2s(1 - r)d_td_o \quad (4.19) \]

\( U^C \) is called covariance proportion, which is due to incomplete covariance.
Normally, if $U^M$ is large, this means that the average of predicted values deviates from the mean of actual values. This is a serious error. The other two errors may be difficult to eliminate.

5.3 Direction Errors

Another index for measuring errors, is called “direction errors” ($DE$). Direction error is a “turning point” error in time series data. Direction errors occur when the predicted value and its corresponding value in the data set have different sign. For example, in our case of forecasting economic growth, suppose that the actual growth rate is negative at a specific point. If our prediction value is positive, a direction error occurs in this situation. Therefore, direction errors are often considered as serious prediction errors.
Part Two

Neural Network Applications in Agricultural Economics
Chapter Five

Forecasting Economic Growth:

A Comparison of Econometric and Neural Network Methods

Economists have long attempted to forecast economic growth or identify the conditions under which economies achieve faster rates of growth. Complex macro-econometric and sectoral models estimated from time series data, such as the Chase or Wharton Econometric models (e.g., Johnson, 1996), were widely used in the 1970s to forecast business conditions. More recently, work by Barro and others has stimulated renewed interest in empirical modeling of growth processes using single-equation econometric models. For the most part, these models involve cross-sectional data from different countries (Levine and Renelt; Pritchett, 1992), states (Evans and Karras, 1996; Goetz and Ready, 1995), or U.S. counties (Goetz and Hu, 1995). Recent efforts include extensions to panel data sets (Islam and Caselli, 1996; Esquivel and Lefort, 1999; Zipfel, 2005).

Despite the popularity of this research, the explanatory power of equations estimated from cross-sectional data often remains low. For example, in county-level regressions, the adjusted $R^2$ value may be as low as 20 percent, suggesting that the model does not fit the data well. A relatively new method, neural network analysis, has been applied in numerous disciplines to assist in identifying relationships between dependent and independent variables. This method is especially promising in cases where theory offers limited guidance in specifying the parameters of the estimating equation, its functional form and the distribution of the error term.
In this research, we compare the relative effectiveness of econometric and neural network methods in forecasting economic growth by comparing in-sample prediction and out-of-sample forecast errors from models estimated using cross-sectional county-level income growth data. With over 3,000 counties as cross-sectional observations, we have a large enough data set to be able to use neural network methods. Ordinary least squares and neural network methods are applied to the same data set to determine which method is superior in terms of minimizing forecast errors. Since the definition and type of forecast error used can profoundly affect the ranking of the two methods that generated the forecasts, we devote considerable effort to testing a variety of measures of forecast errors (and in-sample predictions). We conclude by reviewing advantages and disadvantages of each method.

1. Modeling Economic Growth

1.1 Literature Review

Since Adam Smith and others of the time, many economists have studied the determinants of economic growth. Their thoughts are embodied in the production function, \( Y = F[K, L, T] \), where \( Y = \) output, \( K = \) Capital, \( L = \) Labor, and \( T = \) Land. Smith argued in *The Wealth of Nations* that economic growth could be caused by any one of the three factors.

Economists since Adam Smith have discussed a plethora of various means to affect economic growth, however, the fundamental model has remained the same. (Zipfel, 2004). There are basically two categories of economic growth theories (David
and Ruffin, 1993): One is those based on exogenous growth theory, and the other is those based on the concept of endogenous growth.

Neo-classical economic growth model contains only two inputs, physical capital, $K(t)$, and labor, $L(t)$, and a function assuming the form: $Y(t) = F[K(t), L(t), t]$, where $Y(t)$ is output produced at time $t$. Function $F$ satisfies the following properties: constant returns to scale (CRS), or linear homogeneity, positive and diminishing marginal products. However, capital investment and labor failed to encapsulate all the information relevant to explaining the growth of a particular economy. Solow recognized that the inputs of capital and labor were not the only factors that contributed to the economic growth. He pointed out that a significant portion of economic growth is dependent on the technological progress (Zipfel, 2004). But, he considered that the factor of technological progress is the exogenous factor that contributes to economic growth through capital and labor so that it was kept out of the equation.

From the 1980’s, endogenous growth theory was popular. Economists recognized that economic growth itself had to be incorporated into the growth model. For example, Barro and his associates have carried out regressions based on initial per capita income levels and various macroeconomic variables to identify determinants of income growth across nations (Goetz and Hu, 1996). Endogenous growth theories emphasize that technological progress dose not just happen as a result of scientific operation outside the economy. From the perspective of long-run economic growth, it is determined by economic incentives. This indicates that productivity growth might be related to the preference, structure, policies and other variables inside the economy.
While exogenous and endogenous growth models have different implications to understand economic growth, nowadays economists use these models as guides to identify the determinants of economic growth. Economists attempted to add more variables into their equations. These factors include human knowledge (Barro 1991, Mankiw, Romer and Weil 1992, Levine and Renelt 1992), international trade (Ben-David, 1991), equipment investment (De Long and Summers, 1991), political and governmental factors (Barro 1989, 1990, 1991, 2001), religion (Barro and McCleary, 2003), and others.

1.2 Specification of the Economic Growth Equation

Since a consensus theoretical framework for specifying regional economic growth models does not exist, variable selection is essentially an empirical issue, conditioned by the units chosen for the analysis. In this study, specification of a functional form and selection of explanatory variables are guided by previous studies. Specifically, we are going to extend Goetz and Hu’s county-level economic growth model. The model is as follows

\[
DY = \alpha_0 + \alpha_1 IN + \alpha_2 HS + \alpha_3 IN \times ED + \alpha_4 UR + \alpha_5 LD + \alpha_7 HW + \alpha_8 CRP \\
+ \alpha_9 HWY + \alpha_{10} RTW + \alpha_{11} UN + \alpha_{12} WT + \alpha_{13} MW + \alpha_{14} SO + \varepsilon, \tag{5.1}
\]

where \(DY\) is the difference between the logarithm of real county income per capita in 1990 and 1995; \(IN\) is real income per capita in 1990; \(HS\) presents the proportion of adults who had received at least a high school education in 1990; \(IN \times ED\) is the interaction
between \( IN \) and \( ED \); \( UR \) denotes counties with a Beale code of less than 4 (urban counties); \( RU \) are counties with a Beale code greater than 7 (rural counties); \( LD \) is land area per capita; \( HW \) is highway spending in each county; \( CRP \) is the (state-wide) corporate tax rate; \( HWY \) is a dummy variable with value 1 if the county has interstate highway access and 0 otherwise; \( RTW \) is a Right-to-Work dummy variable; \( UN \) is the percentage of manufacturing workers who are unionized. Variables \( CRP \), \( RTW \) and \( UN \) are available only at the state level. \( WT \), \( MW \) and \( SO \) are dummy variables representing the West, Midwest and South regions, respectively (the Northeast in the excluded region); and \( \varepsilon \) is a random error term. In the process of estimating the above equation, we extend the results of Goetz and Hu from the U.S. South to all continental U.S. counties, and from the 1980s to the early 1990s.

2. **Methodology of Neural Networks**

Although the neural network method is relatively new and not often applied in agricultural economics research (Joerding, Li and Young, 1994; Kastens, Featherstone and Biere, 1995), it has been applied as research tools in many other fields such as speech and signal processing (Sejnowski, Yuhas, Goldstein, and Jenkins, 1990; Malkoff, 1990), handwritten character recognition (Le Cun, Boser, Denker, Henderson, Howard, Hubbard and Jackel, 1990), finger print recognition (Leung, Engeler, and Frank, 1990), prediction of bank failure (Coats and Fant, 1993; Altman, Marco and Varetto, 1994), prediction of financial stock market performance (Azoff, 1994; Refenes, Zapranis and Francis, 1994; Gately 1996; Trippi, 1996), modeling and forecasting economic issues (Joerding, Li, and Young, 1993; Maasoumi, Khotanzad and Abaye, 1994; Church and Curram, 1996;
Kastens and Featherstone, 1996; Terasvirta and Dijk, 2004; Gavidia and Gupta 2004) and many more. Just as Kuan and White states, successes in these and other areas suggest that artificial neural network may serve as a useful addition to the tool-kits of economists.

What is an artificial neural network? In fact, there is no standard definition in literature. One of definitions for neural networks, which is quoted by many researchers, is given by Haykin (1999). According to Haykin, a neural network is a massively parallel distributed processor made up of simple processing units, which has a natural propensity for storing experiential knowledge and making it available for use. It resembles the brain in two respects: First, knowledge is acquired by the network from its environment through a learning process; second, interneuron connection strengths, known as synaptic weights, are used to store the acquired knowledge. It is certain that a neural network model is composed of many neurons, which are basic unites that work like computer processors. The units are connected by communication paths (connections) with weights. The units operate only on the inputs that they receive via the connections, and then send the outputs to the next layer of units. As a neural network model, it should have some sort of “training” rule whereby the weights of connections are adjusted on the basis of training set data. Thus, neural networks “learn” from input examples and exhibit some capability for generalization beyond the example data (Sarle, 1999).

2.1 Architecture of a Neural Network

First, let us take a look a biological neuron. A neuron is the fundamental part of the nervous system. All the neurons have the same structure, independent of their size. The structure of a biological neuron is shown in Figure 5.1. There are four physical
components for a neuron, including cell body, dendrites, axon and synaptic terminals.

The dendrites are the signal receivers, which accept signal from outside or other neurons. The cell body is the place for message processing, and generate impulses. The axon is the channel that transmits the message generated by the neuron to the next neurons, or to the outside such as muscle fibers.

After observing a biological neuron, we may not have difficulty to understand an artificial neuron in neural networks, which is mathematical neuron. It is the basic block for building a neural network, and is fundamental to the operation of a neural network. As Figure 5.2 shown, three major components make up an artificial neuron. All neurons in the artificial neural networks have these components without exception.

a. **Weighting Factors**: A neuron usually receives external inputs. Each input has its own relative weight, which provides the input’s impact on the summation function. These weights perform similar functions as do the synaptic strengths of a biological neuron as shown in Figure 5.3. Obviously, some inputs make different amount of contribution to the neurons’ output. Furthermore, weights also represent the intensity and strength of the connections of the neurons. From mathematical perspective, the collection of weights is the matrix that can memorize the knowledge from the training data sets. The counterpart of weights in econometric models is coefficient.

b. **Summation Function**: There are two operations inside an artificial neuron, which is the counterpart of nucleus in bio-neuron (see Figure 5.3). The first one calculates the weighted sum of all of the inputs. The second one converts the output of the summation in terms of certain threshold. Mathematically, summation function is to operate the inner product of input vector and weight vector. The result is a single number.
Geometrically, the inner product of two vectors can be considered a measure of their similarity. If the vectors point in the same direction, the inner product is maximum; if the vectors point in opposite direction (180 degrees out of phase), their inner product is minimum.

c. Transfer Function: The transfer functions (or activation functions, squashed functions) are essential parts in a neural network model. In literature, much emphasis has been given on the transfer function. The transfer function is employed to transform the result of summation function, basing on the need of neuron output. For example, the neuron could use a “hard limiter” or step function to output zero and one, one and minus one, or other numeric combinations.

A number of neurons could be formed into a layer like Figure 5.4. Although there are only three neurons illustrated in the model, more neurons may be present in each layer. Neurons are independent and separate each other. Their computations are also parallel. Note that the weight \( W \) is a matrix, and the input \( a \) and output \( y \) are vectors, respectively. The computations in the adders and the transfer functions are:

\[
\begin{align*}
\mathbf{a} &= \mathbf{b} + \mathbf{Wx} \quad (5.2) \\
y &= f(a) \quad (5.3)
\end{align*}
\]

where \( \mathbf{b} \) is bias vector and \( f(\cdot) \) is transfer function.

A network may consist of several layers of neurons. Figure 5.5 is of multilayer feedforward network. Normally, the first (or left in our diagrams) layer is input layer, and
the far right layer is the output layer. The layers fallings between are hidden layers. If there is one hidden layer in a model, the model is called as one hidden layer model. In the first hidden layer, the operations of the adder and the transfer function are

$$1a = 1b + 1Wx$$  \hspace{1cm} (5.4)
$$1y = 1f(1a)$$ \hspace{1cm} (5.5)

where, $x$ is inputs vector, $W$ is the weights matrix in current layer, $b$ is vector of biases. The output of summer ($a$) is the input of the transfer function, respectively. In the second hidden layer, the operations of the adder and the transfer function are as follows

$$2a = 2b + 2W1y$$  \hspace{1cm} (5.6)
$$2y = 2f(2a)$$ \hspace{1cm} (5.7)

where, the output of previous layer ($y$) becomes the input of current layers. The computation is similar to the first hidden layer.

In the output layer, the computations of the adder and the transfer function are

$$3a = 3b + 3W2y$$  \hspace{1cm} (5.8)
$$3y = 3f(3a)$$ \hspace{1cm} (5.9)
where, the output of the second hidden layer $y_2$ is used as input. The output of this layer represents the final result of the whole network.

If all the operations are concatenated into one mathematical form that is

$$y = f_W (f_{W_1} (W_2 x + b_1) + b_2) + b_3$$  \hspace{1cm} (5.10)

2.2 Econometric/statistic perspectives of neural networks

In fact, a neural network is similar to an econometric equation in some ways. For example, in neural network modeling, the dependent variable (in our case, economic growth), the independent variables (exogenous determinants of growth), intercept and the predicted values are called target, inputs, bias and output, respectively. Both regression methods and neural networks minimize the respective sums of squared errors between actual and predicted values, or the target and the output. The process of estimating a model is called “learning” in neural network analysis.

To better understand neural networks, we shall discuss several classical neural networks such as Hebbian Linear Associator, Perceptron, ADALINE (ADAptive Linear) and MLP (Multi-Layer Perceptron). Nowadays, the first three are rarely used in real-life applications as non-linear, more versatile, complex and powerful neural network models are available. Nevertheless, these models are still the building blocks of the larger networks. Moreover, there are more similarities between those linear neural network models and econometric models (i.e., multiple linear regression model, logistic model and LPM).
Hebbian Linear Associator is shown in Figure 5.6. The output $y$ can be presented as follow:

$$y = \text{purelin}(b + w^T x) = b + w^T x$$  \hspace{1cm} (5.11)

where $y$ is scalar, which is the output of the model. $b$ is scalar, and it is the bias. $w$ is the vector of weights (i.e., $w_1, w_2, \ldots, w_k$), and $x$ is vector of external inputs (i.e., $x_1, x_2, \ldots, x_k$). This mathematic presentation is exactly the same as linear regression model in econometrics.

The architecture of one-Perceptron Network is shown in Figure 5.7. The mathematical presentation is as follows

$$y = \text{hardlim}(b + w^T x)$$ \hspace{1cm} (5.12)

where $y$ is the vector of output. $b$ is bias. $w$ is the vector of weights and $x$ is vector of external inputs. The transfer function is the hardlimiter function, which is defined as:

$$y_i = \text{hardlim}(a_i) = 1 \text{ if } a_i \geq 0 \hspace{1cm} (5.13)$$

$$0 \text{ otherwise}$$

Perceptron may remind us of the Logit model and Probit model in econometrics.

The ADALINE can be presented as follow
$$y = \text{purelin}(b + w^T x) = b + w^T x$$ \hfill (5.14)$$

Equation (5.14) is identical to equation (5.11), except that the target values are continuous values in equation (5.11), whereas target values are binary for ADALINE so that it can be used to classify objects into two categories. ADALINE model could be thought as the Linear Probability Model (LPM) in the regime of econometrics, because both of ADALINE and LPM are used to model the dummy dependent (output) variable.

A network such as single Perceptron, ADALINE model, is able to divide the input space into regions by the linear boundary (hyperplane). However, they can not solve the linear non-separability issues. A Multilayer Perceptron (MLP) method is a far more flexible mechanism to model data. We can use the following formula to present a multilayer Perceptron model with one hidden layer.

$$y = f' (2w_2f'(Wx + b_1) + b_2)$$ \hfill (5.15)$$

where $y$ is the output; $f'$ is the transfer function of the output layer; $w_2$ is the weight vector of the output layer; $b_2$ is the bias of output layer; $f'$ is the transfer function of the hidden layer; $W$ is the weight matrix of the hidden layer; $b_1$ is the bias vector of the hidden layer.

Suppose the transfer function $f'$ is a pure linear function, Eq. (5.15) can then be rewritten as

$$y = 2w_2f'(Wx + b_1) + b_2$$ \hfill (5.16)$$
Eq. (5.16) is a class of additive/linear models equivalent to the statistic models, projection pursuit models (Friedman and Stuetzle, 1981). Another statistic model for nonlinear regression, the generalized additive model, is analogous to Eq. (5.15). Hastie and Tibshirani (1990) proposed a series of generalized additive models. These models assume that the mean of the dependent variable depends on an additive predictor through a nonlinear link function. Generalized additive models permit the response probability distribution to be any member of the exponential family of distributions. Many widely used statistical models belong to this general class, including additive models for Gaussian data, nonparametric logistic models for binary data, and nonparametric log-linear models for Poisson data. The class of general additive models takes the form

$$y = zf(y(x) + zb)$$  \hspace{1cm} (5.17)

where the $f$ are nonlinear function and $zf$ represents the logistic sigmoid function. From above discussion, we shall find the similarities between the models of neural networks and econometrics/statistics. Jordan (1995) even believes that the discriminant methods from the two fields are two sides of the same coin. Theoretically, econometric/statistical discriminant methods can be categorized into two general types, the sampling paradigm and the diagnostic paradigm (Riley, 1992). The sampling paradigm estimates the distribution of the predictor variable separately for each class and combines these with the prior probabilities of each class occurring. Then using Bayes theorem, the posterior probabilities of belonging to each class are calculated. On the other hand, diagnostic paradigm uses the information from the samples to estimate the
conditional probability of an observation belonging to each class, based on predictor variables. Crudely put, the regression model (i.e. Logit model) has tended to focus on sampling paradigm whereas the neural network is one of the techniques based on diagnostic paradigm.

2.3 Empirical design aspects of neural networks

A mathematical theorem developed in 1963 by Kolmogorov, has been suggested as the guide to determine the number of layers and number of neurons in each layer. According to the Kolmogorov’s theorem (1957), one hidden layer is adequate for most practical issues. The Kolmogorov’s mapping neural network existence theorem states that any continuous function from the \( n \) dimensional cube \([0, 1]^n\) to the real numbers \( \mathbb{R} \) can be implemented by only one hidden layer network with \( 2^n + 1 \) neurons, and \( m \) neurons in the output layer (Fu, 1994, Bishop, 1995, Reed and Marks, 1998). The problem is that the proof for this theorem is less constructive, and the designed number of neurons is not suitable for many practical cases. Much of theoretical work regarding neural network model construction is perhaps more appropriately called as “quasi-theoretical” (Kastens and et al., 1996). Ad hoc criteria, such as observing data distribution/pattern, using training, testing and/or validation and other empirical methods, still remain as effective way to design neural network models.

In practice, if the relationship among the data is linear, a hidden layer may not be required. For a non-linear problem, however, a hidden layer is crucial. Even a network with only one hidden layer can effectively “learn” the patterns contained within the data. The fewer neurons in a network, the smaller the number of algebraic operations needed,
and the shorter the amount of computing time required. Further, a complex network model may simulate in-sample data very well, but not perform satisfactorily in out-of-sample projections. This is analogous to a regression model that contains enough variables so that nearly every observation is intersected by the regression hyperplane. Such a model would not likely be useful for making out-of-sample forecasts. Thus, an elaborate neural network with many hidden layers, neurons and weights may simply “memorize” the information contained in the sample, and not forecast well beyond the data set.

In this application, our neural network consists of three layers (14-m-1) as illustrated in Figure 5.8, an input layer containing 14 inputs, a hidden layer composed of m neurons (for comparison purpose, we shall do multiple experiment with various number of neurons in the hidden layer), and an output layer consisting of one unit (economic growth). Thus, the neural network model is

\[ y = f^2(f^1(W^d x + b^I) + b^2), \]
\[ b^I = (b_1, b_2, ..., b_m)^T, \]

In this study, \( y \) is the economic growth rate, \( x \) is the vector of inputs \( x=(x_1, x_2, ..., x_{14})^T \) representing exogenous factors or determinants of economic growth, \( W^d \) denotes a 14 × m matrix of weights of the hidden layer, \( b^I \) is the vector of bias of the hidden layer (a bias is equivalent to an intercept in a linear regression model), \( w^2 \) is the vector of weights of the output layer \( w^2=(w_1, w_2, ..., w_m) \), and \( b^2 \) is the bias of the output layer. Because there is only one output node in our study, \( b^2 \) is a scalar.
The notation \( f^1(\cdot) \) denotes an activation function of a hidden layer, which is assumed to be differentiable and bounded. A sigmoid function such as a logistic, tanh or Gaussian function is a common choice. A logistic function was employed in this study, largely because its derivative is easily found. And, \( f^2(\cdot) \) is the activation function of the output layer, which is also a step function.

One of the most powerful algorithms for estimating eq. 5.18 is backpropagation in neural networks, which has been widely used in other studies (Hagan, Demuth and Beale). In this study, we use a backpropagation algorithm which follows the least squares criterion of minimizing the sum of squared errors (SSE). The backpropagation algorithm has three iterative steps, which are executed until the error reaches a preset tolerance level.

a. Propagating the inputs forward through the net. In this step, the hidden layer receives external inputs and then determines corresponding output values. The outputs of the hidden layer become the inputs of the output layer. The outputs from the output layer are the net’s primary simulation or prediction values. The difference (error) between target values and output values is calculated.

b. Propagating the error backward through the network and calculating the correction terms of the weights and biases. This step is crucial in the process. Based on the feedback information of the difference (error) from the previous step, the correction terms of the weights and biases between the output layer and the hidden layer are calculated according to the steepest descent rule.

c. Updating the weights and biases. After the weights and biases are updated, return to step \((a)\).
To carry out those steps, we adopt a variation of the traditional backpropagation algorithm, the Levenberg-Marquardt backpropagation (LMBP), because it speeds up the convergence of neural network training.

3. Evaluation and Metrics for Prediction Accuracy

The essential criterion for evaluating the performance of forecasting tools is the accuracy of the forecast. However, the evaluation of forecast accuracy is not trivial, because accuracy can be measured in many ways, and the evaluation results may be inconsistent if different criteria are used. There are many ways to measure forecast accuracy. The evaluation results may not concur if different measures are used. The measures can be placed in three categories: (1) absolute errors, (2) squared errors and (3) directional errors. Both absolute error and squared error measures treat the positive and negative errors the same. The absolute error method gives all errors the same weights no matter if the error is small or large. The squared error method gives most weight to the large errors, and the least weight to the small errors. Because of this, the measures obtained from the forecast techniques cannot always be the same. To measure the directional errors is to assess the forecast techniques from another angle. Directional errors occur when the sign of forecast values is not consistent with the corresponding sign of the values in the data set. A measure of directional error is the number of pairs in which the forecast values and actual values have different signs. Since each category of measure emphasizes a specific property of errors, different measures of forecast accuracy could lead different conclusions.
To better understand the properties of the forecast methods, we used series forecasting error measures.

3.1 Absolute error measures

The common absolute error measures are (1) the sum absolute error (SAE), (2) the mean absolute error (MAE), and (3) the mean absolute percentage error (MAPE).

Suppose that the forecast error of point $k$ is

$$ e_k = t_k - o_k $$  \hspace{1cm} (5.20)

where $o_k$ is the forecast value and $t_k$ is the actual value at point $i$.

Then the sum absolute error (SAE) is

$$ SAE = \sum_{k=1}^{s} |e_k| $$  \hspace{1cm} (5.21)

where $s$ is the number of observations.

The mean absolute error (MAE) is

$$ MAE = \frac{SAE}{s} $$  \hspace{1cm} (5.22)

Both the SAE and the MAE measure the magnitude of the errors obtained to form the forecast. Forecast errors may sometimes appear large in an absolute index, but are acceptable relative to the magnitude of the observations contained in the data set. In this sense, relative index is more informative. Thus, mean absolute percentage error (MAPE), measures the magnitude of absolute errors in relative terms.
The indices of squared errors

The most popular index for measuring squared errors is the sum of squared errors (SSE). The SSE is employed in many neural network algorithms, because its quadratic form is useful to finding the optimum solution. The sum of squared errors is

$$SSE = \sum_{k=1}^{s} e_k^2$$

(5.24)

The mean square error (MSE) refers to average square error for a set of prediction points.

$$MSE = \frac{SSE}{s}$$

(5.25)

The root mean square error (RMSE) is the square root of MSE. It re-scales the errors in order to keep the errors’ dimension as the predicted value.

$$RMSE = \left( MSE \right)^{\frac{1}{2}}$$

(5.26)

Theil decomposed mean square error as
\[
\frac{1}{s} \sum_{k=1}^{s} (t_k - o_k)^2 = (\bar{t} - \bar{o})^2 + (d_t - d_o)^2 + 2(1 - r)d_od_t \tag{5.27}
\]

Where, \(\bar{o}\) and \(\bar{t}\) are means of the predicted values and the actual values, respectively, and

\[
\bar{o} = \frac{\sum_{k=1}^{s} o_k}{s} \tag{5.28}
\]

\[
\bar{t} = \frac{\sum_{k=1}^{s} t_k}{s} \tag{5.29}
\]

\(d_o\) and \(d_t\) are the standard deviations of the predicted values and the actual values, which are defined as

\[
d_o^2 = \frac{\sum_{k=1}^{s} (o_k - \bar{o})^2}{s} \tag{5.30}
\]

\[
d_t^2 = \frac{\sum_{k=1}^{s} (t_k - \bar{t})^2}{s} \tag{5.31}
\]

and \(r\) is the correlation coefficient of predicted and actual values:

\[
r = \frac{\frac{1}{s} \sum_{k=1}^{s} (o_k - \bar{o})(t_k - \bar{t})}{d_od_t} \tag{5.32}
\]
For convenience in this study, the error components are

\[ U^M = s(\bar{i} - \bar{o})^2 \]  

(5.33)

\( U^M \) is called the bias proportion, which indicates errors in central tendency.

\[ U^D = s(d_t - d_o)^2 \]  

(5.34)

\( U^D \) is called variance proportion, which reflects the difference between the variances.

\[ U^C = 2s(1 - r)d_td_o \]  

(5.35)

\( U^C \) is called covariance proportion, which is due to incomplete covariance.

Normally, if \( U^M \) is large, this means that the average of predicted values deviates from the mean of actual values. This is a serious error. The other two errors may be difficult to eliminate.

3.3 Direction Errors

Another index for measuring errors, is called “direction errors” \((DE)\). Direction error is a “turning point” error in time series data. Direction errors occur when the predicted value and its corresponding value in the data set have different sign. For example, in our case of forecasting economic growth, suppose that the actual growth rate
is negative at a specific point. If our prediction value is positive, a direction error occurs in this situation. Therefore, direction errors are often considered as serious prediction errors.

4. Description of the Data

We used county-level data from 3,005 continental US counties for the empirical application. The data are divided into two groups, one of which is used for in-sample estimation/training, and the other to conduct out-of-sample tests. The out-of-sample group is randomly selected from the data by choosing counties which have a FIPS code ending with the number ‘5’. The advantages of this sampling method are that it guarantees that both sample groups consist of observations from each region and state proportionally, and it also maintains the property of randomness of the sample. The out-of-sample group consists of 599 observations, which account for about one-fifth of all observations, and the in-sample group includes 2,406 observations.

To begin, we re-scaled all inputs by adding or multiplying by a constant, because re-scaling the inputs can not only improve the training speed, but also reduce the chances of reaching a local optimum that is not globally optimal. All input data values lie between 0 to 1 after they re-scaled. It is not necessary to re-scale the target variable, so we leave $D_Y$ unchanged.

As mentioned earlier, it is commonly agreed that although many hidden layers are useful in some applications, one hidden layer is generally both sufficient and efficient. Hence, we use only one hidden layer in our neural network model. To select an
appropriate number of neurons in the hidden layer, we experiment with 1 to 10, and 15 neurons, respectively, and use only one neuron in the output layer in all experiments.

5. Results for the Econometric and Neural Network Methods

Ordinary Least Squares (OLS) estimates of equation (1) are reported in table 5.1. All the coefficient estimates are significant at the 5 percent level or lower, except for $MW$ and $HWY$, and the signs are generally consistent with expectations. At the sample average, the effect of the initial income level on income growth is negative; this result is consistent with findings in the literature, and is known as conditional convergence. For education, the effect on income growth is close to zero at the sample means, but increases as income rises above the sample average. The interesting implication is that, in low-income counties, higher levels of education failed to increase the rate of income growth over the period 1990 to 1995. Urban counties (Beale codes 0-3) grew more rapidly than counties with codes 4 through 7, while rural counties (codes 8 and 9) grew significantly less rapidly. In addition, less densely populated areas grew less rapidly than more densely populated areas, a finding that is consistent with Ciccone and Hall.

To evaluate the performance of OLS and neural network methods, we compare the magnitude of the errors in the in-sample simulation and the out-of-sample prediction. There is only one set of predictions for the OLS method. Neural networks generate distinct predictions or forecasts every time the number of neurons contained in the hidden layer changes. In this analysis, all the neural networks have one hidden layer, and eleven different experiments were conducted: with 1 to 10 and 15 neurons in the hidden layer, respectively. Results are presented in table 5.2 for errors of in-sample simulations, and in
table 5.3 for errors of out-of-sample predictions. The first row shows indices applied to 
OLS errors. The second to eleventh rows represent results obtained from the neural 
networks with 1 to 10 neurons, respectively. The last row is the result of the neural 
network with 15 neurons. The purpose of this additional experiment with 15 neurons is to 
verify the trend observed in the experiments with 1 to 10 neurons. Patterns in these errors 
are readily apparent, as discussed in the next section.

6. Comparison of Econometric and Neural Network Methods

First, we assess the forecast accuracy of the two methods (Tables 5.2 and 5.3). Here, the advantage of neural networks over OLS is clear. Nearly all of the error indices 
for the OLS method are larger than those for the neural networks, even though some of 
the results of the neural network experiments are sub-optimal.

The advantage of the neural network estimates is most obvious in the experiments 
of in-sample simulations (Table 5.2). The SSE from OLS is 10.73, whereas the smallest 
SSE for the neural networks is 5.95 (row 12). The latter is 55.5 percent of the former. The 
SAE from OLS is 113.64, whereas the smallest SAE for the neural networks is 87.79 (or 
77.3 percent of the former). The number of directional errors for the OLS model (472) is 
greater than that of all the experiments of neural networks. Most importantly, perhaps, the 
bias error ($U^M$) from OLS is considerably larger than that obtained from the neural 
networks (1.93E-04 versus 2.22E-09 for the neural net with 5 neurons). This indicates 
that neural networks perform an in-sample prediction that is superior to OLS. In fact, if 
more neurons were used in the hidden layer, we could obtain predicted values from the 
in-sample simulation that are even closer to the actual data.
Neural networks have advantages over *OLS* in out-of-sample prediction performance as well (Table 5.3). The *SSE* from *OLS* is 4.04, whereas the smallest *SSE* for the neural networks is 3.31 (81.9 percent of the former). The *SAE* from *OLS* is 32.03. The smallest corresponding *SAE* for the neural networks is 28.53 (89.1 percent of the former). Hence, we conclude that neural networks not only have an advantage over the *OLS* method in the in-sample simulation, but also provide better out-of-sample forecasts for this data set.

Second, consider the effect of the number of neurons used in the hidden layer of the neural network. The *SAEs*, *SSEs*, *DEs* of in-sample simulations obtained from the neural networks are plotted in Figures 5.9, 5.10 and 5.11, respectively. With few exceptions, the errors generally decline as the number of neurons in the hidden layer increases. This means that the net better simulates patterns of data as the number of neurons increases. But, there are several exceptions. For example, in Figure 5.10 of the sum of square error, the *SSEs* of experiments with 4 and 5 neurons are no smaller than that of experiment with 3 neurons in the hidden layer. The reason for this is that the initial weights were not appropriate, and the neural network “training” converged at a local minimum. Therefore, it is helpful to train the model more than once with different initial weights to find a global optimum.

Thus, the errors of the in-sample predictions generally decline as the number of neurons in the hidden layer becomes larger. The only cost involved here is computational. In this study, 15 was the highest number of neurons used, and the model with 15 neurons illustrates the power of the in-sample simulation capability of the neural network method.
The plots showing the $SAE$ (Figure 5.12) and $SSE$ (Figure 5.13) for out-of-sample forecasts follow a ‘V’ shape. The error initially declines as the number of neurons increases. After reaching a minimum, the error rises, however. As indicated earlier, too many neurons are detrimental to out-of-sample forecasting, and lead to over-training or over-fitting. A net with too many neurons in the hidden layer simply memorizes the information contained in the sample, and is useless for out-of-sample forecasts. This shows that neural networks are a powerful tool when the input to output vector space mapping contains both regularities and exceptions (Refenes, 1995, 1997), but a model which learns too many specific details about the data performs poorly in out-of-sample forecasts.

Third, we further investigate measures of absolute versus squared error, and absolute versus relative error. Normally, larger sums of absolute errors lead to larger sums of squared errors, but this is not always the case. Because each measure uses different weights, the conclusions drawn from each case are not necessarily consistent. For example, the value of $SSE$ in row 5 is smaller than that of its adjacent rows (rows 4 and 6) in Table 5.3, but the corresponding $SAE$ value is larger than those of its adjacent rows. This is a direct result of the fact that the SSE applies a weight equal to the value of the error itself (and larger errors have a larger weight while smaller errors have a smaller weight), while the SAE gives the same weight (1) to all errors. However, the value of $SSE$, in row 8 of Table 5.3, is greater than that of row 7, but the corresponding $SAE$ value in row 8 is less than that in row 7.

Measures of absolute and relative error may not necessarily concur either. This argument is verified in our study. There clearly is a declining pattern in Figure 5.9 for
SAE, but Figure 5.14 for MAPE shows a zig-zig shape with no consistent pattern. Small absolute errors do not always coincide with a small relative error. In practice, the measures related to relative errors are usually considered to be more informative than those related to absolute errors.

7. Conclusion

The neural network method overwhelmingly outperforms the OLS method in terms of minimizing the errors of in-sample predictions and out-of-sample forecasts. This conclusion is valid regardless of the measure discussed before that is selected as an evaluation index.

However, the individual evaluation measures may influence our evaluation, because all the measures do not concur. Neural network experts commonly employ SSE, as do many commercial software packages such as Matlab. If we follow this rule, experiment 9 in our study, which uses 8 neurons, is the optimal design for our data set, because the out-of-sample SSE is the smallest. Although some other measures, including the SAE and RMSE support this choice, the MAPE, DE and \( U^M \) do not. As discussed earlier, the MAPE, DE and \( U^M \) are more informative than the other measures, and MAPE, DE and \( U^M \) imply that the network with 8 neurons in the hidden layer in experiment 9 is not an optimal design. Only a comprehensive and careful comparison that includes various measures of forecast errors will reveal this nuance. From this, researchers will better understand the properties of the forecast errors, and they can then make more informed choices, depending on which kind of error they wish to emphasize or minimize. Although neural networks tend to attain higher forecast accuracy than OLS, some weaknesses need to be highlighted. The main weakness is that a neural network model is
a non-parametric method. The quantitative relationship between the dependent and independent variables is not as immediately obvious as in an econometric model. Sensitivity analysis can be conducted for each input in the net, but this is time-consuming, and the results may be unstable because of randomness of the initial weights. Another potential disadvantage is that neural network analysis is itself time-consuming. The model’s training time increases exponentially with the number of neurons selected. In this study, the Tool Box of Matlab was used to train the models. Only a few minutes are needed to achieve convergence when the number of neurons is under 3. But, it takes hours of computing time to complete 250 iterations with 10 neurons, and more time for 15 neurons. More computing time would be required for an even greater number of neurons.

Neural network methods at this time are perhaps best viewed as supplements to econometric methods in studying economic issues, and not necessarily as substitutes. In future research, we intend to use neural network analysis in an attempt to improve the specification of economic growth models.
Chapter Six

Feedforward Neural Networks in Prediction of Food Manufacturing Establishment Growth

1. General Introduction

Manufacturing industries are very important to rural economic development, and among manufacturing industries, food processing has been considered as the most promising recruitment target for rural communities (Barkema, Drabenstott and Stanley, 1990). The benefits of food manufacturing industries to rural communities include employment opportunities, off-farm employment opportunities, higher wage rates, and backward and forward linkages within the rural economy. In fact, the benefits even extend to other economic sectors and the society. For example, besides providing various processed foods to consumers and tax income to the government, the food manufacturing industries generate considerable revenue to the national economy.

According to Connor and Schiek (1997), the commercial food processing industry is defined as a “branch of manufacturing that transforms raw animal, or marine materials into intermediate foodstuffs or edible products through the application of labor, machinery, energy, and scientific knowledge”. In this study, we will investigate the food and kindred products sector (named “SIC 2000” category in the Office of Management and Budget’s Standard Industrial Classification (SIC) manual). The sector contains many sub-industries within SIC 2000. We will model the determinants of industrial establishment location decisions for the aggregate industry (SIC 2000) and nine subcategories. These include meat products (SIC 2010), dairy products (SIC 2020),
preserved fruits and vegetables (SIC 2030), grain mill products (SIC 2040), bakery products (SIC 2050), sugar and confectionery products (SIC 2060), fats and oils (SIC 2070), beverages (SIC 2080) and miscellaneous food and kindred products (SIC 2090). The descriptions of the nine industries are found in Table 6.1.

An establishment is defined in the County Business Patterns document (1999) as “a single physical location at which business is conducted or services or industrial operations are performed. It is not necessary identical with a company or enterprise, which may consist of one or more establishments. When two or more activities are carried on at a single location under a single ownership, all activities generally are grouped together as a single establishment”.

Numerous studies exist of the food manufacturing industry and related non-food industries. Specifically, researchers have classified the determinants for manufacturing industries into two categories: 1. traditional factors, such as accessibility to market, the cost and availability of factors of production and others, and 2. non-traditional factors, including government policies (e.g. tax and expenditure), unionization, unemployment and others. In earlier 1980s and before, researchers emphasized the traditional factors, and most of the studies had consistent findings and conclusions. Morgan (1964) found that traditional locational factors had a dominant influence on industrial location. The influence of non-economic factors was negligible. Kieschnick (1981) used a survey method to identify which factors were more important to industrial location. He found that access to market, management, labor supply and cost and other traditional factors were important to firm’s location decision. Plaut and Pluta (1983 and 1987) analyzed the relationship between business climate, taxes and expenditures, and state industrial
growth. They found that the high local taxes might not be a deterrent to industrial growth if the benefits from such as taxes were perceived to accrue locally rather than statewide. They concluded that business climate, tax, and expenditure were important for industrial growth, but the importance of these factors was still considered to be less than that of the traditional variables.

In late 1980s and 1990s, economists not only continued their efforts to investigate the effects of traditional factors on manufacturing industries, but also shed light on the influences of government behavior and other non-traditional factors on the manufacturing industrial growth. Bartik (1985 and 1989) examined the effects of unionization, taxation, and state characteristics on location decisions in the U.S. He found that the traditional factors had influence on location decision, however, non-traditional factors such as unionization, tax and public services affected plant location as well.

In 1990, Fox and Murray completed a study dealing with the effect of public policies on local business decision. He concluded that long-run policies such as transportation infrastructure, education, water routes, urbanization, hospitals and wages were the most important factors in business development. The short-run policies including government expenditure and taxes had modest influence on industry development.

Goetz (1996) was the first economist who systematically investigated economic determinant of the location criteria for food manufacturing establishments both at state level and at county level in the US. He followed the assumption (Blair and Premus, Calzonetti and Walker, Moore et al., Wheat, Woodward) that food processing firms were assumed to follow a two-stage process in making the location choice, in which first the
geographic region and then the specific US state was chosen, and finally the choice was to a specific county within a state. Therefore, his studies could be divided into at least two stages, including state level and county level. For example, in his county-level study, market access variables, labor force characteristics policy variables, raw materials and agglomeration economies variables were included in the regression models. His results illustrated that the factors might have different effects depending on the specific sub-industry under study.

This study can be thought of as an extension to Goetz’s research. However, more emphasis is given to predicting establishment growth for the aggregate industry and each sub-industry at county level. Moreover, a discrete binary independent variable was used in place of the continuous variable. Another purpose was to compare the effectiveness of econometrics methods such as the linear probability model and the Logit model and feedforward neural networks in forecasting establishment growth by comparing in-sample prediction and out-of-sample forecast.

2. Location Determinants and Hypotheses

Although the process of choosing an establishment location is very complex (Bohm, Herzog and Schlottman, 1983; Knapp and Graves 1989; Goetz, 1996), the primary objective for location decision is profit maximization. Specifically, the assumed economic goal of the firm is to maximize profit in the long term (Wheat, 1973; Bartik, 1985 and 1989; Woodward, 1992), since a new establishment indicates long-run capital investment (Blair and Premus, 1987). Given the profit maximization of industrial location decision, there are a number of constraints that affect the objective. Following Goetz’s
variable classification, the five categories include a market access variable, a raw material (inputs) variable, labor force characteristics, a policy variable and an agglomeration variable are discussed below.

2.1 Market Access Variables

Market access encompasses two basic ideas. One is to access to the input market, and the other is to access the final products market. A measure for market access is transportation infrastructure (Woodward and Glichman, 1991). Bartik (1989) and Woodward (1992) mentioned that the presence of adequate transportation systems increases plant profitability and the probability of plant attraction. We selected three variables to capture the characteristics of market access, including Interstate highway access (HWY), railway access (RAIL) and seaport access (PORT). The hypothesis was that if a county is accessible by Interstate highway, railway and/or seaport, it would be in a better position to attract food processors. Therefore, the signs of the three variables (HWY, RAIL, PORT) were positive.

2.2 Raw Material (Inputs) Variables

In general, processing industries are often raw material-oriented industries, and so are food processing industries. Raw materials used in food processing such as vegetables, fruits and milk are often perishable. Processing firms for perishable food commodities are generally located close to sources. Furthermore raw material for non-perishables such as corn and wheat are bulky so that the cost for transportation of material can be high, and firms processing those raw materials have more incentive to
close to the raw material source, in order to decrease the transportation costs (Goetz, 1996). Two variables representing raw materials are included in our models. They were cash receipts from livestock (LIVE) and crops (CROP) per capita. They were proxies for the availability of raw materials. The signs of the two variables should be positive. That is, the more of these a county produces, the more attractive the county might be to food processors.

2.3 Labor Force Characteristics Variables

Labor is a critical food processing input. Specifically, the cost, quantity and quality of labors are considered as determinants to plant-location decisions. This category was represented by four variables in our model. First was the hourly wage rate (WAGE) per capita. Normally, lower wage rate could attract food manufacturing industrial firms. A second variable was the unemployment rate (UNEM). The sign of this variable was expected to be negative. A higher unemployment rate indicated that more potential workers were available for the manufacturing firms. A county was attractive to the food processing firms if many labors are unemployed in that specific location. The sign of this variable was expected to be positive. The third factor was the number of high school or above graduates (HS_GRD). If the unemployment rate reflected the quantity of potential workers, HS_GRD could capture the quality of the workers. Higher-quality workers could improve the productivity of manufacturing industries. Therefore, it was expected that the sign of the HS_GRD would be positive. The remaining two variables in this category were HNDL and OPRT. HNDL was a variable representing percentage of the workers, such as handlers, equipment cleaners, helpers and labors, in total workers in a
county. OPRT represents the percentage of workers such as machine operators, assemblers and inspectors. Since the food processing industry frequently hires semi-skilled and even unskilled labors (Goetz, 1996), the sign on HNDL should be positive and the sign of OPRT should be negative.

2.4 Policy Variables

There was just one policy variable in the models, the property tax rate (PROPT). We used property tax rate per capita to represent this category since the property tax policy could directly affect the profitability and long-term viability of food processing firms. A higher tax rate may discourage food processors to extend their businesses into a particular county. Thus, the expected sign of the variable was negative.

2.5 Agglomeration Economy Variables

Agglomeration economies are collective benefits (cost reductions) that accrue to firms that locate in close proximity to one another (Herzog and Schlottmann, 1991). In certain sense, agglomeration economies are scale economies. This variable was measured by the number of existing plants in a county (Woodward, 1992). Population can also be a proxy for capturing the agglomeration economies. Therefore, two variables in this category were included in our study. They were POP87 and EST87. POP87 presented the total population in the county in 1987. EST87 presented the number of establishments in the county in 1987. Both of them were assumed having positive relationship with food processing industrial growth.
3. Empirical Methodology and Data Description

3.1 The Empirical Methodology

For the purpose of comparison of the effectiveness in prediction of establishment growth, both traditional econometric methods and neural networks were employed. The two econometric methods employed were Ordinary Least Squares and Logit. The Neural Networks model employed the feedforward backpropagation method. The same data set was used for all methods.

The dependent variable was limited into two possible values. If the number of establishments in 1997 was larger than for 1987, then the value of the variable was assumed to be 1 (net growth in the sub-industry). Otherwise, the value of the variable is 0 (a stable or declining sub-industry).

We could interpret the probability of food processing establishment growth by the generic forms as follows (Greene, 1996):

\[
\text{Prob}(Y=1) = F(\beta'x), \quad (6.1)
\]
\[
\text{Prob}(Y=0) = 1 - F(\beta'x), \quad (6.2)
\]

where \(x\) is a vector of independent (explanatory) variables as outlined above. The set of parameters \(\beta\) reflects the impact of changes in independent variables on the probability. \(F(\cdot)\) present the cumulative probability distribution function. Different assumptions for the probability distribution function therefore lead to different methods.

\[ \]

a. The Linear Probability Model (PLM)
A linear probability model assumes the cumulative distribution function of probability is linear. This means that.

\[ F(x, \beta) = \beta'x. \quad (6.3) \]

Practically, the PLM model for the food processing establishment growth can be simply presented as

\[ y_k = \beta_k'x_k + \epsilon_k, \text{ } k=0, \ldots, 9. \quad (6.4) \]

Where \( k \) represents SIC number of the industry \((k=0)\) and sub-industries \((k=1, \ldots, 9)\). \( y_k \) is the net growth of establishments between 1987 and 1997. The variable being forecast is a binary variable \((0\text{ or }1)\). \( x_k \) is the vector of the independent variables. \( \epsilon_k \) is the error term.

The parameters in above equation can be estimated by the standard OLS procedures.

\[ b. \text{ The Logit Model} \]

In the Logit model, the cumulative probability is assumed to follow the pattern of the logistic function. The equation can be written as

\[ p_k = E(y_k = 1|x_k) = \frac{1}{1 + e^{-z_k}}, \quad (6.5) \]

\[ z_k = \ln\left(\frac{y_k}{1-y_k}\right) = \beta_k x_k + \epsilon_k, \text{ } k=0, \ldots, 9. \quad (6.6) \]
where $e$ is the base of the natural logarithm. $k$ represents SIC number of the industry ($k=0$) and sub-industries ($k=1, \ldots, 9$). $p_k$ is the predicted probability. $y_k$ is the net growth of establishments between 1987 and 1997. $x_k$ is the vector of the independent variables. $\epsilon_k$ is the error term.

c. The Feedforward Neural Networks

A linear probability model has obvious drawbacks. It assumed that the probability of growth was distributed in a linear pattern, and errors were normally distributed. Those assumptions may not be valid in practice. The probability prediction may lie out of the range of $[0, 1]$. Although the Logit model solved partial problems of the linear probability model such as limiting the prediction probability in the range of $[0, 1]$, it simply assumed that the cumulative probability function was logistic function. This assumption was lack of flexibility and even not reasonable as well, in practice.

A feedforward neural network model is non-linear mathematical function without any prior hypothesis on the pattern of cumulative probability distribution or errors’ distribution. A feedforward neural network consists of three types of layers such as input layer, hidden layers and output layer. Each layer is composed of a number of units (so called as “neurons”), which can be interconnected between different layers. Input layer is considered as a lower layer, since each node contains a raw input (e.g. an economic variable). The principle of the feedforward is that lower layer propagate information to next higher layer through the connection channels associated with weights. The functions in a single unit can be presented as follows:
\[ I_j = \sum_{s=1}^{n} w_{js} O_s, \quad (6.7) \]
\[ O_j = \frac{1}{1 + e^{I_j}}, \quad (6.8) \]

where \( i \) represents the \( i^{th} \) layer; \( j \) represents the \( j^{th} \) layer; \( n \) is the number of inputs (including the bias) from previous layer; \( s \) represents \( s^{th} \) input from previous layer. \( O_{is} \) is the \( s^{th} \) node’s output from \( i^{th} \) layer; \( w_{js} \) is the weight associated with the connection between the \( s^{th} \) node in \( i^{th} \) layer and current node in \( j^{th} \) layer. \( I_j \) and \( O_j \) are input and output of the \( j^{th} \) layer, respectively.

Intuitively, there are some similarities between above equations and the Logit model. However, the transfer function is not necessarily a logistic function, but it could be other form of sigmoid function. Since there are many layers with a number of units in a feedforward network, these computations could happen many times within in a single layer and its output will be propagated to the next layer for further computations. The final-form function of the network for a feedforward network with one hidden layer is as follows:

\[ z_{0k} = f(z_{W_1f(W_2x_k)}), \quad k=1, \ldots, 9, \quad (6.9) \]

where \( k \) represents SIC number of the industry (\( k=0 \)) and sub-industries (\( k=1, \ldots, 9 \)). \( x \) is the vector of independent variables. The subscripts at the left of a letter present layers. \( f(\cdot) \)
is transfer function. The two transfer functions are not required to be the same. \( W \) is the matrix of weights. \( o \) is the output of the network.

A major algorithm used to estimate the weights associated with the network is the backpropagation. This algorithm computes the gradient of the case-wise sum of squared error function with respect to the weights (and biases) for a feedforward network. After a number of iterations of learning, the error will converge at a tolerant level.

### 3.2 Data Description

We used county-level data from 3,049 continental US counties of the food processing industry for the 9 sub-industries. For the purpose of examining the accuracy of prediction, the data were divided into two groups, one of which was used for in-sample estimation/training, and the other to conduct out-of-sample tests. The out-of-sample group was randomly selected from the data by choosing the counties that had a FIPS code ending with the number ‘5’. The advantages of this sampling method were that it guaranteed that both sample groups consisted of observations from each region and state proportionally, and it also maintained the property of randomness of the sample. The out-of-sample group consisted of 655 observations, which accounted for about one-fifth of all observations, and the in-sample group included 2,444 observations.

As to the dependent variable, it was based on the change in the number of the each food processing industry as a whole or each sub-industry. We calculated the change value by using the number of establishment in 1997 less the number of establishments in 1987. Since the models required binary independent variable, counties with negative growth were grouped together with zero growth. It meant that the values of both negative
growth and zero growth were set to 0s. The values of the independent variable were set to 1s, if the counties had a positive growth.

The independent variables had continuous values. However, we re-scaled all the values of the independent variables by divided by the maximum value of the corresponding observations. All independent variable values lied between 0 and 1, after they were re-scaled. Re-scaling the inputs could not only improve the training speed, but also reduced the chances of reaching a local optimum that was not globally optimal.

4. Results and Discussion

4.1 The Results of the Linear Probability Models

a. The Estimated Parameters Description

The estimated parameters of the LPM were presented in Tables 6.5 and 6.6. As expected, the sign on the highway access variable (HWY) was positive and statistically significant at the one-percent level in all sub-industrial models except the aggregate one (SIC2000), in which it is significant at the five-percent level. The study found that the presence or absence of an Interstate highway was a very important factor to food industrial growth. However, although railway (RAIL) had positive sign in most of the models, it was not significant in eight out of ten models, except in models of meat products industry (SIC 2010) and grain mill products industry (SIC 2040). The variable of seaports (PORT) was significant at one-percent level in the models of dairy products (SIC 2020), Bakery products (SIC 2050) and sugar and confectionery products (SIC 2060) industries. It was statistically significant at five-percent level in the miscellaneous industry (SIC 2090 model). All the signs were the expected positive values. In short, the
railway and seaport variables also played the very important role in food industrial development. However, there was no evidence that railway plays an important role for the food processing industries.

The sign of wage rate (WAGE) was surprising. The sign in all models was positive instead of expected negative. It was statistically significant at the one or ten percent levels in most of the specifications, except for meat products (SIC 2010), fats and oils (SIC 2070) and miscellaneous industries (SIC 2090). Nor was the sign on the unemployment rate (UNEM) as expected, and it was negative not positive in all the models. Hence, the food processing industry could bring new employment opportunities for the counties that already had a considerable number of food processing establishments. The variable reflecting labor quality, HS_GRD, did not have effect on most of the industries except grain mill industry, according to the regression results. The percentages of categories of labors (HNDL and OPRT) did affect some of the food processing industries, however, the effects were negative.

Tax policy had negative impacts on the industrial growth and was significantly different from zero in the aggregate industry (SCI 2000), dairy products (SIC 2020), preserved fruits and vegetables (SIC 2030) and miscellaneous industries (SCI 2090). The signs for variables of livestock (LIVE) and crop (CROP), the two raw materials variables, were also contrary to our expectations. CROP had negative sign and was significant in most of models except that of meat (SIC 2010) and fats and oils (SIC 2070) industries. LIVE had a negative sign and was statistically significant only in three specifications including sugar and confectionery (SIC 2060), beverages (SIC 2080) and misc. (SIC 2090) industries. Perhaps the major component of raw material shipping costs
is the time and labor involved in loading and unloading, and once the materials are loaded, it is of little consequence cost-wise as to how far geographically they have to be shipped—within reason of course.

As proxy variable for agglomeration economy, population (POP87) was positively associated with plant attraction, and the existing establishments (ES87) deterred food processing industrial location.

**b. Prediction Accuracy Discussion**

As mentioned before, one objective of this study was to compare the effectiveness of the different methods in forecasting establishment growth. Therefore, the indices of prediction accuracy and errors were reported in Table 6.9 and 6.10. There were two result groups, including those of in-sample prediction and out-of-sample forecast.

The index of “CC” was referred as to the correctness count (CC) for prediction. If the prediction (or forecast) was the same as the actual situation in a county, “CC” accrued by 1. Hence, the number of “CC” could tell us how many predictions were correct. During calculating the prediction index, if the predicted probability was higher than 0.5, the county was considered as one with positive establishment growth, and vice versa. The index of “CR” stands for Correctness Ratio presented in percentage. MSE stands for Mean Squared Errors. MABSE is Mean Absolute Error.

The correctness count and the correctness ratio were very important indices for the binomial regression model. Since the traditional R-squared was not well suited for testing “goodness of fit” for these models, the correctness count and correctness ratio are alternative methods for evaluating the goodness of fit. The results revealed that the values
of correctness ratios were very high. They were about 67% for the aggregate industry (SIC 2000) for in-sample prediction and out-of-sample forecast. As to the sub-industries, the correct percentages were over 80 percent, and three of them were above 90 percent. We can conclude that the specifications for the models were acceptable.

4.2 The Results for the Logit Model

a. The Estimated Parameters Description

The results for the Logit models were reported in Tables 6.7 and 6.8. The sign and statistical significance of the co-efficiencies for the variables were similar to those in the linear probability models except those of HS_GRD, LIVE and CROP. In the linear probability models, almost all of the estimated co-efficiencies for HS_GRD were not statistically significant even at the level of 10 percent. However, the co-efficiencies were positive and significant at 5 or 10 percent levels in the aggregate industry (SIC 2010), dairy products (SIC 2020), grain mill product (SIC 2040), beverage (SIC 2080) and miscellaneous industries (SIC 2090) in the Logit models. This result confirmed the accuracy of our prior expectations that the quality of labors is positively associated with the industrial location decision.

Some positive signs for the variables LIVE and CROP appeared in the Logit models for the sub-industries. This indicated that raw material availability led to positive establishment growth in some sub-industries. However, the signs of these two variables were still negative in most of the sub-industries.

b. Prediction Accuracy Discussion
The indices of prediction accuracy and errors were reported in Table 6.11 and 6.12. The values of CC and CR were larger than or at least equal to those for the linear probability models. The values of MSE and MABSE were less than those of linear probability models. It indicated that the assumption of cumulative probability of distribution function was more nearly proper than that for the linear function.

4.3 Results of Neural Networks

a. The Estimated Weights Description

The neural network deployed is a feedforward model with one hidden layer. There are 13 input nodes in the input layer, 5 neurons in the hidden layer, and 1 neuron in the output layer. Each node in this layer is connected with all nodes of the next layer. These connections are known as weights and are numeric in nature.

For observing the weights (sign and magnitude), two experiments have been done for SIC2000, and the weight matrices are shown in Tables 6.13, 6.14, 6.15 and 6.16, respectively.

As shown in Table 6.13, the weight collection between the input and hidden layer are 14 x 5 matrix, which includes 1 x 5 bias. The weight set in Table 6.14 is 6 x 1 matrix including 1 bias, which are the connections between the hidden layer and the output layer.

An artificial neural network weight governs how much activity, output or influence is associated with the signal from a neuron. The higher the weight assigned to a connection (synapse in biological neural networks), the higher the activity or strength associated with the connection. In the example of Exclusive OR (XOR) in the previous
chapter, the relationships between the thresholds, weights, inputs and outputs can be intuitively analyzed in detail. But for the network with large numbers of inputs and hidden neurons, it may be practically impossible to determine weight and bias values required to achieve desired outputs from given inputs. However, the rule of the training process is that artificial neural network models attempt to train networks and adjust their weights so that they can produce desired outputs from given inputs. If biological memory and learning are the result of synapse strengths and modifications of synapse strengths, then the artificial network models can be very instructive (Best, 2005). In fact, the weights of neural networks could not be interpreted in terms of any economic meaning. More specifically, even the signs of the weights between an input and neurons in the hidden layer are not the same. For example, in the estimated Linear Probability Model, the sign on the highway access variable (HWY) was positive and statistically significant at the one-percent level in all sub-industrial models except the aggregate one (SIC2000), in which it is significant at the five-percent level. However, there two negative and three positive weights for the variable of HWY in Table 6.13.

Table 6.15 and Table 6.16 present the weights of the same network and data set but different training experiment. This experiment has as good performance as experiment one in terms of the predication accuracy and errors. However, the weights are almost totally different from those of Table 6.13 and 6.14. The reason these two sets of weights are different is because there are infinite solutions for a certain problem of classification. Just as Simon (1999) states that a neural network rarely converges on the best and unique solution to a classification problem. But often, a neural network does not need to find the absolute best solution, only one of many good solutions. A quickly
converging good solution, rather than a slowly converging best solution, usually is sufficient for most problems.

\textit{b. Prediction Accuracy Discussion}

The prediction accuracy and errors of neural networks were reported in Table 6.17 and 6.18. Just as the results of LPM and Logit models, the correct count percentages are very high in all the sub-industries, which are over 80 percent for in-sample predication and out-of-sample forecasting.

For comparison purpose, in Table 6.19 of the in-sample prediction, 6 out of 10 of the values of CC and CR were larger than or equal to the index values in the linear probability model, whereas 5 out of 10 of the values of CC and CR were larger than or equal to the index values and Logit models. However, as shown in Table 6.21, most of the Mean Squared Errors (MSE) are slightly greater than those of LPM and Logit models, whereas not all of the Mean Absolute Errors are less than those of the linear probability models and the Logit models.

As illustrated in Table 6.20, the results of out-of-sample forecast of neural networks were as good as or better than that of linear probability in terms of the correctness counts. But not all of the count values of neural networks are greater than those of the Logit models. Especially, in Table 6.22, almost all of the Mean Squared Errors (MSE) of neural networks are less than those of linear probability models and Logit models. Although neural networks have less error values than LPM and Logit models in table 6.24 of Mean Absolute Errors (MABSE), the advantage of neural networks in this study was not obvious as in previous application. This result is consistent
to Gavidia and Gupta’s research (2004). In their research, they compared the prediction of occupational attainment using a backpropagation neural network model and a multinomial logit model. The dependent variable was discrete one as well. The empirical results indicated that there was no significant difference in the number of accurate forecasts between the neural network and the multinomial logit model. Likewise, there was no significant difference in mean square error between both models.

5. **Summary and Conclusions**

The purpose of the study was to use conventional econometric and neural network techniques to determine the factors that affect food manufacturing establishment growth. Specifically, the study two econometric methods employed were the linear probability model and the Logit model. Results from these models were compared with feedforward neural network forecasts for food manufacturing establishment growth using both in-sample prediction and out-of-sample forecasts. The aggregate food manufacturing sector and its 9 sub-industries were investigated. Cross-sectional data from 3049 continental U.S. counties were used and divided into two sets. The first set had 2444 county-level observations for estimating the econometric models and training the neural network model. The other set had 655 observations for out-of-sample forecast so as to test the performance of the estimated models when confronted with a new data set.

Both the linear probability model and the Logit models confirmed that the highway access was extremely important in the location decision. It was evident that agglomeration economies for food manufacturing industries could be represented by the population size in a county but not by the existing number of establishments. Wage rate
might be higher if there were lots of food processing establishments in a county. This indicated that food manufacture might bring a high wage rate and off-farm income for farmers.

The unemployment rate was also negatively related to the food manufacturing establishment growth. A higher quality of labor could attract the food processors. A high average skill level among workers in a county could deter food manufacturing industry development since food manufacturing industries employed a significant share of low-skilled workers (i.e. raw materials handlers, equipment cleaner and helpers). Tax policy could also deter food manufacturing establishment growth. Livestock and crop marketing cash receipts per capita did not always support the hypothesis that food processors tended to locate near sources of raw agricultural commodities as we assessed this from sub-industry to sub-industry. These two variables might not fully capture the characteristics of raw material availability.

Empirically, we shall conclude that neural networks have as good as, or even better performance than Linear Probability Model in terms of the comparison results shown in Tables 6.19 to 6.24. Especially, the indices of prediction accuracy (Correctness Counts) are higher than those of the LPM model for many sub-industries, in both in-sample predication and out-of-sample forecast. However, neural networks did not show their strong advantage over the Logit model because that the dependent variable is discrete values.

Theoretically, neural networks have important advantages over econometric models. Neural networks do not impose any assumptions or hypotheses on the cumulative probability distribution functions and the pattern of error distribution. In a
certain sense, neural networks had the flexibility to “let the data speak” (Joerding and others, 1994). With respect to the comparative predictive ability of econometric methods versus feedforward neural networks, this research revealed that feedforward network models had better performance in in-sample prediction and they were at least as good as linear probability models and Logit models in doing out-of-sample forecasts. Although neural networks are a promising prediction and forecasting tool for economists, some disadvantages of neural network methods were also obvious. Neural network models have some weaknesses. A major and inherent problem of neural networks is that the internal structure cannot tell researchers how it processed the input information or reached a conclusion. That process is represented only in the matrix of connection weights, which cannot be reliably translated into meaning or interpretation (Hawley, Johnson and Raina, 1990). In other words, the weights in the model do not have economic meanings whereas the coefficients in econometric models can often be interpreted in an economic context. This is because econometric models generally have theoretical foundations whereas neural networks are based on emulation or simulation. The latter is much weaker from the standpoint of identification of structure and explanatory power. However, some economists such as Joerding (1993) and others consider that this weakness of neural networks should not be overstated since neural networks can address many of the same questions for which other flexible nonlinear models are sometimes applied.

Secondly, related to the first weakness, although there are weights corresponding to each input, they do not indicate to the researcher anything about the inputs. Thus, that the quantitative relationship between the dependent and independent variables (inputs) is
not as obvious as in an econometric model. Furthermore, the methods of significance testing cannot be applied to determine which independents (inputs) could be incorporated into the model. A sensitivity analysis can be carried out for each input to determine its marginal effect, but sometimes it is often misleading (Sarle, 2002). Hence, just as DeTienne and Lewis (2003) mentioned, “When results are sought without an extraordinary need to understand the parameters behind them, then neural networks are most beneficial. When researchers need both an accurate prediction and an understanding of the predictive parameters, both ANNs and Regression should be used in conjunction.”

As we have understood the advantages and disadvantages of neural networks, we can wisely deploy them in our economic studies. One thing for sure is that neural networks will never replace conventional econometric methods, but they are useful tools to our tool box. Just as what we have done in this application, by using both traditional econometric (linear regression, Logit) and neural network approaches, we obtain the best from both approaches. Both methods could compliment each other. We believe that economic researchers could gain significant benefits in using both methods in applied problems.
Chapter Seven

Summary and Conclusions

1. Thesis Summary and Conclusions

The objectives of this dissertation were to examine the fundamental components, concepts and theory of neural network methods from econometric and statistical perspective, with particular focus on econometrically and statistically relevant models, and to apply neural network methods to model agricultural economic issues in a fashion that would permit comparisons. The dissertation is divided into two parts, 1. a neural network theory and methods review and 2., two essays illustrating applications in agricultural economics data.

   In part one, we reviewed neural network theory and methods from econometric and statistical perspective. We analyzed three major classical linear neural networks, including Hebbian Linear Associator Network, Perceptron and ADALINE, by comparing them with the counterparts in econometrics. We found that the mathematical presentation of Hebbian Linear Associator Network is exactly the same as linear regression model in econometrics; The Perceptron network shares the same architecture as those of the Logit model Probit models in econometrics; The ADALINE model could be thought as the Linear Probability Model (LPM) in econometrics. Both neural networks and econometrics have common problems of modeling and interference. Neural networks and econometrics statistics, particularly in discriminant methods, are two sides of the same coin in terms of the nature the statistical and modeling issues. On one side, econometric models are sampling paradigm-oriented methods that estimate the distribution of the
predictor variable separately for each class and combine them with the prior probabilities of the occurrence of each class. However, neural networks are based on a diagnostic paradigm, which use the information from the samples to estimate the conditional probability of an observation belonging to each class, based on predictor variables. Neural network and econometrical/statistical methods have the same properties, except that the natural parameterizations differ.

For the nonlinear neural networks, we focused on the Multilayer Perceptron Network. We found that the mathematic formula for the Multilayer Perceptron model with one hidden layer was remarkably similar to certain nonlinear statistical models such as projection-pursuit regression and generalized additive models. This may be why statisticians see neural networks as nonlinear regression methods that are estimated by optimizing some measure of fit to the training data.

In part two, we evaluated the relative effectiveness of econometric and neural network methods by two empirical applications in agricultural economics.

The first application was to forecast economic growth by comparing in-sample prediction and out-of-sample forecast errors from models estimated using cross-sectional county-level income growth data. In this application, we observed the ability of data modeling for neural network and the Ordinary Least Squares (OLS) methods. Since the criteria used to evaluate forecast accuracy can profoundly alter the ranking of the two methods that generated the forecasts, we devoted considerable effort to testing a variety of measures of forecast errors, including absolute, squared and directional errors. From the experiment, we found that the neural network method outperformed the Ordinary Least Squares (OLS) method in terms of minimizing the errors of in-sample
predictions. This is not a surprising result in that neural networks, especially multilayer feedforward networks, are “Universal Approximators” (Hornik and White, 1989). The goodness of fit of the neural networks to the set of in-sample data could be extremely good. The error minimized by the training process is not the error measure of ultimate interest (Smith, 1993). Just as Bishop (1995) states, our goal of training a neural network is to find the model having the best performance on new data. In our experiment, the out-of-sample forecasts for the neural networks are better than those for OLS methods as well.

The second application used feedforward neural networks to predict growth or decline in food manufacturing establishment. This study is an extension of Goetz’s research. However, more emphasis was given to predicting establishment growth for the aggregate industry and each sub-industry at the county level. Moreover, a discrete binary independent variable was used in place of the continuous variable. We compared the effectiveness of econometrics methods such as the linear probability model and the Logit model and feedforward neural networks in forecasting establishment growth by evaluating both the in-sample predictions and the out-of-sample forecasts.

The empirical results revealed that feedforward network models were superior performers with respect to in-sample prediction and may be superior to linear probability models, but there was no significant difference in the number of accurate forecasts between the neural network and the Logit model.

In general, the empirical applications suggested that neural network methods outperformed the traditional econometric models including Multiple Regression Analysis, Linear Probability Model (LPM), and their performance is as good as that of
the Logit model in terms of minimizing the errors of in-sample predictions and out-of-sample forecasts. Although neural networks have some advantages (i.e., nonlinearity, low biased error and others) over econometric methods, limitations include a lack of tests for statistical significance. Hence, neural networks are perhaps best viewed as supplements to econometric methods in studying economic issues and drawing cause-and-effect relationships, and not as substitutes. Just as Martin and Morris (1999) states, neural network modeling is no replacement for the process and the generated data. Careful initial study employing traditional statistical methods provides a preliminary insight into the process and cause-and-effect and is an essential prerequisite for dynamic modeling employing neural networks.

2. Future Work

First, we need to continue to undertake an analogous comparison study with neural networks, for other econometric methodologies such as time-series analysis. Time series analysis is a very important part of econometrics from a methodological perspective. Time series issues are also of critical importance in agricultural economics. However, we did not have an empirical study in this thesis, which employed time series analysis methods such as AR and ARMA models and neural networks.

Second, there is no structured methodology available for choosing, developing, training and verifying neural networks. We need to continue to study and standardize the paradigm for neural network deployment in agricultural economic research so that the quality of neural network may be predictable regardless of network design and learning algorithm. Refenes, Burgess and Bentz (1997) had a study on this regard in financial
engineering. They described a number of ways to deal with the problem of variable
selection, show how to use model misspecification tests, deploy a way based on
cointegration to deal with the problem of nonstationarity, and generally describe
approaches to predictive neural modeling which are more in tune with the requirements
for modeling financial data series. It is necessary for us to conduct similar studies in
agricultural economics to overcome the weakness of neural network modeling including
the lack of established procedures for performing tests for misspecified models, and test
of statistical significance for the various parameters.
Figure 1.1 The structure of a biological cell body and synaptic terminals.

Figure 1.2 The structure of a simulated neuron with excitatory and inhibitory signals.
\[ a = b + w^T x \]
\[ y = f(a) \]

Figure 1.3. The structure of a neuron

\[ a = b + W x \]
\[ y = f(a) \]

Figure 1.4. The structure of a layer of neurons
Figure 1.5 The structure of a neural network

Figure 2.1 The Architecture of linear network
\[ a = b + \mathbf{w}^T \mathbf{x} \]
\[ y = f(a) \]

**Figure 2.2** The architecture of Perceptron network

\[ b + \mathbf{w} \mathbf{x} = 0 \]

**Figure 2.3** Decision boundary for Two-input Perceptron
Figure 2.4 Decision boundary for two-input ADALINE

Figure 2.5 General representation of a neural network and econometric model
Figure 3.1 Logic “OR”

Figure 3.2 Two input/single-output neuron
Figure 3.3 Surface of transfer function

Figure 3.4 Surface of transfer function (scaled by 10)
### Figure 3.5 Neuron of Logic “OR”

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$y_I$</th>
<th>OR</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0.9994</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0.9999</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0.0007</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1.0</td>
<td>1</td>
</tr>
</tbody>
</table>

The neuron diagram shows the inputs $x_1$ and $x_2$, the sum ($\Sigma$), and the output $y_I$. The decision boundary is marked for the logic "OR".

### Figure 3.6 Decision boundary of logic “OR”

The diagram illustrates the decision boundary for the logic "OR" with points $(0,0)$, $(0,1)$, $(1,0)$, and $(1,1)$. The boundary line separates the regions for different logic outcomes.
\[ \begin{array}{ccc} x_1 & x_2 & y_1 \\ 0 & 1 & 0.0002 \\ 1 & 0 & 0.0002 \\ 0 & 0 & 0.0000 \\ 1 & 1 & 0.9993 \end{array} \]

**Figure 3.7** Neuron of Logic “AND”

\[ x_2 \]
\[ x_1 \]
\[ \Sigma \]
\[ 15.6236 \]
\[ 15.8425 \]

\[ \begin{array}{c} \text{AND} \\ \text{and}(0, 0) = 0 \\ \text{and}(1, 0) = 0 \\ \text{and}(1, 1) = 1 \end{array} \]

**Figure 3.8** Decision boundary of logic “AND”
<table>
<thead>
<tr>
<th>( x_1 )</th>
<th>( x_2 )</th>
<th>( y )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0.9994</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0.9996</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1.0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0.0005</td>
</tr>
</tbody>
</table>

**Figure 3.9** Neuron of Logic “NAND”

**Figure 3.10** Decision boundary of logic “NAND”
xor(0, 0) = 0
xor(0, 1) = 1
xor(1, 0) = 1
xor(1, 1) = 0

**Figure 3.11** The linear non-separability (XOR)

\[ \text{not}(x_1 \text{ AND } x_2) \]

\[ x_1 \text{ OR } x_2 \]

\[ 1 \text{ AND } 2 \]

**Figure 3.12** The structure of a network for solving XOR
Figure 3.13 Network of "XOR"
Figure 3.14 Surface of XOR neural network (StatSoft, 2003)

Figure 3.15 Decision boundaries of logic “XOR”

Note: Dash line 1: Decision boundary of one neuron
Dash line 2: Decision boundary of another neuron
Figure 3.16 The model of classifier of neural
Figure 3.17 The flow chart of the Backpropagation algorithm
Figure 4.1. Discontinuous function

Figure 4.2 The relationship between error and training time or number of neurons/layers
Figure 5.1 Structure of a biological neuron

Figure 5.2 Structure of an artificial neuron
**Figure 5.3** Bio-neuron versus artificial neuron
Figure 5.4. The structure of a layer of neurons

Figure 5.5 The structure of a neural network
Figure 5.6 The structure of a linear network

\[ y = f(a) = a \]

Figure 5.7 The structure of Perceptron

\[ y = \begin{cases} 1 & a \geq 0 \\ 0 & a < 0 \end{cases} \]
$1a = 1b + 1Wx$

$1y = \text{logistic}(1i)$

$2y = f(2a) = \begin{cases} 
1 & 2a \geq 0 \\
0 & 2a < 0 
\end{cases}$

Figure 5.8 MPL with one hidden layer
**Figure 5.9** Sum of absolute error of in-sample prediction

Note: horizontal axis represents experiment numbers:
- 1 — is result of the *OLS* method;
- 2 to 11 — are experiments involving 1 to 10 neurons in the hidden layer;
- 12 — is experiment involving 15 neurons in the hidden layer.

**Figure 5.10** Sum of square error of in-sample prediction

Note: horizontal axis represents experiment numbers:
- 1 — is result of the *OLS* method;
- 2 to 11 — are experiments involving 1 to 10 neurons in the hidden layer;
- 12 — is experiment involving 15 neurons in the hidden layer.
Figure 5.11 Direction error of in-sample prediction
Note: horizontal axis represents experiment numbers:
1— is result of the OLS method;
2 to 11 — are experiments involving 1 to 10 neurons in the hidden layer;
12 — is experiment involving 15 neurons in the hidden layer.

Figure 5.12 Sum absolute error of out-of-sample forecast
Note: horizontal axis represents experiment numbers:
1— is result of the OLS method;
2 to 11 — are experiments involving 1 to 10 neurons in the hidden layer;
12 — is experiment involving 15 neurons in the hidden layer.
Figure 5.13 Sum of Square error of out-of-sample forecast
Note: horizontal axis represents experiment numbers:
1— is result of the OLS method;
2 to 11 — are experiments involving 1 to 10 neurons in the hidden layer;
12 — is experiment involving 15 neurons in the hidden layer.

Figure 5.14 Mean absolute percentage error of in-sample prediction
Note: horizontal axis represents experiment numbers:
1— is result of the OLS method;
2 to 11 — are experiments involving 1 to 10 neurons in the hidden layer;
12 — is experiment involving 15 neurons in the hidden layer.
Table 1.1 The Common Transfer Functions

<table>
<thead>
<tr>
<th>Name</th>
<th>Input/output Relation</th>
</tr>
</thead>
</table>
| Hard Limit                    | \( o = 0 \quad i < 0 \)  
                                 | \( o = 1 \quad i \geq 0 \)  |
| Symmetrical Hard Limit        | \( o = -1 \quad i < 0 \)  
                                 | \( o = +1 \quad i \geq 0 \)  |
| Linear                        | \( o = I \)            |
| Saturating Linear             | \( o = 0 \quad i < 0 \)  
                                 | \( o = 0 \quad 0 \leq i \leq 1 \)  
                                 | \( o = 1 \quad i > 1 \)  |
| Symmetric Saturating Linear   | \( o = -1 \quad i < -1 \)  
                                 | \( o = i \quad -1 \leq i \leq 1 \)  
                                 | \( o = 1 \quad i > 1 \)  |
| Logistic Sigmoid              | \( o = 1/(1+e^i) \)    |
| Hyperbolic Tangent Sigmoid    | \( o = (e^i + e^{-i})/(e^i + e^{-i}) \) |
| Positive Linear               | \( o = 0 \quad i < 0 \)  
                                 | \( o = i \quad i \geq 0 \)  |

Note: \( o \) is output; \( i \) is input.
### Table 1.2 The Comparison of econometric and Neural Network Terms

<table>
<thead>
<tr>
<th>Econometrics</th>
<th>Neural networks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Independent variables,</td>
<td>Inputs</td>
</tr>
<tr>
<td>explanatory variable,</td>
<td></td>
</tr>
<tr>
<td>regressors</td>
<td></td>
</tr>
<tr>
<td>Dependent variable</td>
<td>Target values, or training values</td>
</tr>
<tr>
<td>Predicted values</td>
<td>Outputs</td>
</tr>
<tr>
<td>Residues</td>
<td>Errors</td>
</tr>
<tr>
<td>Observations</td>
<td>Patterns or training pairs</td>
</tr>
<tr>
<td>Parameters</td>
<td>Weights</td>
</tr>
<tr>
<td>Estimation</td>
<td>Training, learning or self-organization</td>
</tr>
<tr>
<td>Intercept</td>
<td>Bias</td>
</tr>
<tr>
<td>Model</td>
<td>Architecture</td>
</tr>
<tr>
<td>Prediction</td>
<td>Forward propagation</td>
</tr>
<tr>
<td>Forecast</td>
<td>Prediction</td>
</tr>
<tr>
<td>Iteration</td>
<td>Epoch</td>
</tr>
<tr>
<td>Sample</td>
<td>Training set</td>
</tr>
<tr>
<td>Hold-out sample</td>
<td>Test set</td>
</tr>
<tr>
<td>Out-of-sample</td>
<td>Test set, validation set</td>
</tr>
<tr>
<td>Regression</td>
<td>Supervised learning</td>
</tr>
<tr>
<td>Interpolation, extrapolation</td>
<td>Generalization</td>
</tr>
</tbody>
</table>
### Table 3.1 Output of Logic “OR”

<table>
<thead>
<tr>
<th>Input 1 ((x_1))</th>
<th>Input 2 ((x_2))</th>
<th>Output ((t))</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

### Table 3.2 Output of Logic “AND”

<table>
<thead>
<tr>
<th>Input 1 ((x_1))</th>
<th>Input 2 ((x_2))</th>
<th>Output ((t))</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

### Table 3.3 Output of Logic “NAND”

<table>
<thead>
<tr>
<th>Input 1 ((x_1))</th>
<th>Input 2 ((x_2))</th>
<th>Output ((t))</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
### Table 3.4 Output of Logic “XOR”

<table>
<thead>
<tr>
<th>Input 1 ($x_1$)</th>
<th>Input 2 ($x_2$)</th>
<th>Output ($t$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Table 5.1 Regression Model for Economic Growth

<table>
<thead>
<tr>
<th>Variable</th>
<th>Parameter Estimate</th>
<th>Mean</th>
<th>Std. Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constant</td>
<td>0.1577***</td>
<td>0.187</td>
<td></td>
</tr>
<tr>
<td>IN (Income)</td>
<td>-0.000014***</td>
<td>11,462.5</td>
<td>2,687.3</td>
</tr>
<tr>
<td>HS (Education)</td>
<td>-0.0008***</td>
<td>69.54</td>
<td>10.40</td>
</tr>
<tr>
<td>IN×ED (×10^6)</td>
<td>0.069***</td>
<td>0.81</td>
<td>0.28</td>
</tr>
<tr>
<td>UR (Urban)</td>
<td>0.0129***</td>
<td>0.27</td>
<td>0.44</td>
</tr>
<tr>
<td>RU (Rural)</td>
<td>-0.0324***</td>
<td>0.25</td>
<td>0.43</td>
</tr>
<tr>
<td>LD (Land)</td>
<td>-0.0538***</td>
<td>962.27</td>
<td>1,281.63</td>
</tr>
<tr>
<td>HW (Highway spending)</td>
<td>-0.0460***</td>
<td>6,656.31</td>
<td>22,928.08</td>
</tr>
<tr>
<td>CRP (Tax)</td>
<td>0.0027***</td>
<td>6.44</td>
<td>3.64</td>
</tr>
<tr>
<td>HWY (Highway access)</td>
<td>0.0044</td>
<td>0.12</td>
<td>0.12</td>
</tr>
<tr>
<td>RTW (Right-to-Work)</td>
<td>0.0093**</td>
<td>0.76</td>
<td>0.43</td>
</tr>
<tr>
<td>UN (Union affiliation)</td>
<td>0.0007***</td>
<td>19.70</td>
<td>12.17</td>
</tr>
<tr>
<td>WT (West)</td>
<td>0.0234***</td>
<td>0.15</td>
<td>0.35</td>
</tr>
<tr>
<td>MW (Midwest)</td>
<td>0.0048</td>
<td>0.35</td>
<td>0.48</td>
</tr>
<tr>
<td>SO (South)</td>
<td>0.0342***</td>
<td>0.45</td>
<td>0.50</td>
</tr>
</tbody>
</table>

R^2: 0.282
Adj. R^2: 0.278
SSE: 10.73

Note: ** significant at the 5% level; *** significant at the 1% level.
Table 5.2 In-Sample Prediction Errors

<table>
<thead>
<tr>
<th>Row Number</th>
<th>Method</th>
<th>SAE</th>
<th>MAE</th>
<th>MAPE</th>
<th>SSE</th>
<th>MSE</th>
<th>RMSE</th>
<th>U^a</th>
<th>U^b</th>
<th>U^c</th>
<th>DE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>OLS</td>
<td>113.64</td>
<td>0.0472</td>
<td>4.27</td>
<td>10.73</td>
<td>0.0045</td>
<td>0.0668</td>
<td>1.93E-04</td>
<td>3.26</td>
<td>7.42</td>
<td>472</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>112.30</td>
<td>0.0467</td>
<td>5.34</td>
<td>10.35</td>
<td>0.0043</td>
<td>0.0656</td>
<td>1.79E-06</td>
<td>2.96</td>
<td>7.38</td>
<td>431</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>104.40</td>
<td>0.0434</td>
<td>4.64</td>
<td>8.74</td>
<td>0.0036</td>
<td>0.0603</td>
<td>1.55E-06</td>
<td>1.90</td>
<td>6.84</td>
<td>437</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>102.67</td>
<td>0.0427</td>
<td>4.24</td>
<td>8.76</td>
<td>0.0036</td>
<td>0.0603</td>
<td>1.31E-06</td>
<td>1.90</td>
<td>6.86</td>
<td>418</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>101.55</td>
<td>0.0422</td>
<td>4.43</td>
<td>8.73</td>
<td>0.0036</td>
<td>0.0602</td>
<td>9.34E-07</td>
<td>1.89</td>
<td>6.84</td>
<td>418</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>98.78</td>
<td>0.0411</td>
<td>4.53</td>
<td>7.81</td>
<td>0.0032</td>
<td>0.0570</td>
<td>2.22E-09</td>
<td>1.43</td>
<td>6.39</td>
<td>388</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>95.73</td>
<td>0.0398</td>
<td>5.16</td>
<td>7.50</td>
<td>0.0031</td>
<td>0.0558</td>
<td>4.18E-07</td>
<td>1.30</td>
<td>6.19</td>
<td>390</td>
</tr>
<tr>
<td>8</td>
<td>7</td>
<td>95.62</td>
<td>0.0397</td>
<td>4.90</td>
<td>7.41</td>
<td>0.0031</td>
<td>0.0555</td>
<td>7.33E-08</td>
<td>1.26</td>
<td>6.15</td>
<td>373</td>
</tr>
<tr>
<td>9</td>
<td>8</td>
<td>95.18</td>
<td>0.0396</td>
<td>5.47</td>
<td>7.12</td>
<td>0.0030</td>
<td>0.0544</td>
<td>2.71E-07</td>
<td>1.15</td>
<td>5.98</td>
<td>377</td>
</tr>
<tr>
<td>10</td>
<td>9</td>
<td>92.80</td>
<td>0.0386</td>
<td>4.91</td>
<td>6.91</td>
<td>0.0029</td>
<td>0.0536</td>
<td>2.01E-07</td>
<td>1.07</td>
<td>5.84</td>
<td>373</td>
</tr>
<tr>
<td>11</td>
<td>10</td>
<td>91.41</td>
<td>0.0380</td>
<td>4.61</td>
<td>6.76</td>
<td>0.0028</td>
<td>0.0530</td>
<td>3.02E-09</td>
<td>1.02</td>
<td>5.75</td>
<td>375</td>
</tr>
<tr>
<td>12</td>
<td>15</td>
<td>87.79</td>
<td>0.0365</td>
<td>3.29</td>
<td>5.95</td>
<td>0.0025</td>
<td>0.0497</td>
<td>1.89E-07</td>
<td>0.75</td>
<td>5.20</td>
<td>360</td>
</tr>
</tbody>
</table>

^aOLS— is result of the OLS method;  
1 to 10 — are experiments involving 1 to 10 neurons in the hidden layer;  
15 — is experiment involving 15 neurons in the hidden layer.
## Table 5.3 Out-of-Sample Forecast Errors

<table>
<thead>
<tr>
<th>Row Number</th>
<th>Method</th>
<th>SAE</th>
<th>MAE</th>
<th>MAPE</th>
<th>SSE</th>
<th>MSE</th>
<th>RMSE</th>
<th>U_M</th>
<th>U_S</th>
<th>U_C</th>
<th>DE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>OLS</td>
<td>32.03</td>
<td>0.0535</td>
<td>9.65</td>
<td>4.04</td>
<td>0.0067</td>
<td>0.0819</td>
<td>0.0224</td>
<td>1.70</td>
<td>2.32</td>
<td>124</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>31.28</td>
<td>0.0522</td>
<td>9.10</td>
<td>3.82</td>
<td>0.0064</td>
<td>0.0799</td>
<td>0.0191</td>
<td>1.46</td>
<td>2.35</td>
<td>112</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>29.83</td>
<td>0.0498</td>
<td>7.23</td>
<td>3.71</td>
<td>0.0062</td>
<td>0.0787</td>
<td>0.0243</td>
<td>1.05</td>
<td>2.63</td>
<td>112</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>29.38</td>
<td>0.0490</td>
<td>7.61</td>
<td>3.56</td>
<td>0.0059</td>
<td>0.0771</td>
<td>0.0238</td>
<td>1.03</td>
<td>2.51</td>
<td>115</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>29.49</td>
<td>0.0492</td>
<td>8.39</td>
<td>3.49</td>
<td>0.0058</td>
<td>0.0763</td>
<td>0.0195</td>
<td>1.00</td>
<td>2.46</td>
<td>117</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>29.47</td>
<td>0.0492</td>
<td>9.45</td>
<td>3.62</td>
<td>0.0060</td>
<td>0.0777</td>
<td>0.0188</td>
<td>0.69</td>
<td>2.91</td>
<td>106</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>29.24</td>
<td>0.0488</td>
<td>6.94</td>
<td>3.68</td>
<td>0.0061</td>
<td>0.0784</td>
<td>0.0154</td>
<td>0.80</td>
<td>2.87</td>
<td>108</td>
</tr>
<tr>
<td>8</td>
<td>7</td>
<td>29.10</td>
<td>0.0486</td>
<td>5.28</td>
<td>3.77</td>
<td>0.0063</td>
<td>0.0793</td>
<td>0.0345</td>
<td>0.95</td>
<td>2.79</td>
<td>120</td>
</tr>
<tr>
<td>9</td>
<td>8</td>
<td>28.53</td>
<td>0.0476</td>
<td>7.95</td>
<td>3.31</td>
<td>0.0055</td>
<td>0.0744</td>
<td>0.0222</td>
<td>0.61</td>
<td>2.68</td>
<td>115</td>
</tr>
<tr>
<td>10</td>
<td>9</td>
<td>29.89</td>
<td>0.0499</td>
<td>7.28</td>
<td>3.83</td>
<td>0.0064</td>
<td>0.0800</td>
<td>0.0206</td>
<td>0.72</td>
<td>3.09</td>
<td>125</td>
</tr>
<tr>
<td>11</td>
<td>10</td>
<td>29.46</td>
<td>0.0492</td>
<td>6.26</td>
<td>3.87</td>
<td>0.0065</td>
<td>0.0804</td>
<td>0.0336</td>
<td>0.86</td>
<td>2.98</td>
<td>122</td>
</tr>
<tr>
<td>12</td>
<td>15</td>
<td>31.66</td>
<td>0.0529</td>
<td>6.15</td>
<td>5.47</td>
<td>0.0091</td>
<td>0.0956</td>
<td>0.0230</td>
<td>0.10</td>
<td>5.35</td>
<td>117</td>
</tr>
</tbody>
</table>

*aOLS— is result of the OLS method; 
1 to 10 — are experiments involving 1 to 10 neurons in the hidden layer; 
15 — is experiment involving 15 neurons in the hidden layer.*
<table>
<thead>
<tr>
<th>SIC</th>
<th>Industries</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000</td>
<td>Food and kindred products</td>
<td></td>
</tr>
<tr>
<td>2010</td>
<td>Meat products</td>
<td>Sausages, poultry slaughtering and processing, and other prepared meats.</td>
</tr>
<tr>
<td>2020</td>
<td>Dairy products</td>
<td>Creamery butter, cheese, natural, processed, dry, condensed, evaporated products, ice cream, frozen desserts and fluid milk.</td>
</tr>
<tr>
<td>2030</td>
<td>Preserved fruits and vegetables</td>
<td>Canned specialties, canned fruits and vegetables, dehydrated fruits, vegetables, soups, pickles, sauces, salad dressings, frozen fruits and vegetables, frozen specialties.</td>
</tr>
<tr>
<td>2040</td>
<td>Grain mill products</td>
<td>Flour and other grain mill products, cereal breakfast foods, rice milling, prepared flour mixes and boughs, wet corn milling, dog and cat food, prepared feeds.</td>
</tr>
<tr>
<td>2050</td>
<td>Bakery products</td>
<td>Bread, cake, and related products, cookies and crackers, frozen bakery product, except bread.</td>
</tr>
<tr>
<td>2060</td>
<td>Sugar and confectionery products</td>
<td>Raw cane sugar, cane sugar refining, beet sugar, candy and other confectionery products, chocolate and cocoa products, chewing gum, salted and roasted nuts and seeds.</td>
</tr>
<tr>
<td>2070</td>
<td>Fats and oils</td>
<td>Cottonseed oils soybean oil mills, vegetable oil mills, animal and marines fats and oils, edible fats and oils.</td>
</tr>
<tr>
<td>2080</td>
<td>Beverages</td>
<td>Malt beverages, malt, wines, brandy spirits, distilled and blended liquors, bottled and canned soft drinks, flavoring extracts and syrups.</td>
</tr>
<tr>
<td>2090</td>
<td>Misc. food and kindred products</td>
<td>Canned and cured fish seafoods, fresh and frozen fish, roasted coffee, potato chips and similar snacks, manufactured ice, macaroni and spaghetti, food preparation.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
<th>Source</th>
<th>Expected sign</th>
</tr>
</thead>
<tbody>
<tr>
<td>EST</td>
<td>Net growth variable of SIC 2000 (EST=1, if net growth is larger than 0; EST=0, else )</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>EST10</td>
<td>Net growth variable of SIC 2010 (EST10=1, if net growth is larger than 0; EST10=0, else )</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>EST20</td>
<td>Net growth variable of SIC 2020 (EST20=1, if net growth is larger than 0; EST20=0, else )</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>EST30</td>
<td>Net growth variable of SIC 2030 (EST30=1, if net growth is larger than 0; EST30=0, else )</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>EST40</td>
<td>Net growth variable of SIC 2040 (EST40=1, if net growth is larger than 0; EST40=0, else )</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>EST50</td>
<td>Net growth variable of SIC 2050 (EST50=1, if net growth is larger than 0; EST50=0, else )</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>EST60</td>
<td>Net growth variable of SIC 2060 (EST60=1, if net growth is larger than 0; EST60=0, else )</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>EST70</td>
<td>Net growth variable of SIC 2070 (EST70=1, if net growth is larger than 0; EST70=0, else )</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>EST80</td>
<td>Net growth variable of SIC 2080 (EST80=1, if net growth is larger than 0; EST80=0, else )</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>EST90</td>
<td>Net growth variable of SIC 2090 (EST90=1, if net growth is larger than 0; EST90=0, else )</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>HWY</td>
<td>Interstate highway access (HWY=1 if entry/exit ramp presents; HWY=0, else)</td>
<td>b</td>
<td>(+)</td>
</tr>
<tr>
<td>RAIL</td>
<td>Railway access (RAIL=1 if node known to exist; RAIL =0, else)</td>
<td>b</td>
<td>(+)</td>
</tr>
<tr>
<td>PORT</td>
<td>Seaport access (PORT=1 if seaport exists; PORT=0, else)</td>
<td>b</td>
<td>(+)</td>
</tr>
<tr>
<td>WAGE</td>
<td>Average wage for employee ($/hour, 1987)</td>
<td>c</td>
<td>(-)</td>
</tr>
<tr>
<td>HS_GRD</td>
<td>High school or above graduates (percent of adults, 1990)</td>
<td>d</td>
<td>(+)</td>
</tr>
<tr>
<td>UNEM</td>
<td>Unemployment rate (percent of labor force, 1986)</td>
<td>e</td>
<td>(+)</td>
</tr>
</tbody>
</table>
Table 6.3 Variable description (continued)

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
<th>Source</th>
<th>Expected sign</th>
</tr>
</thead>
<tbody>
<tr>
<td>HNDL</td>
<td>Handlers, etc. (percent of labor force, 1990)</td>
<td>D</td>
<td>(+)</td>
</tr>
<tr>
<td>OPRT</td>
<td>Operators, etc. (percent of labor force, 1980+1990 average)</td>
<td>d</td>
<td>(-)</td>
</tr>
<tr>
<td>PROPT</td>
<td>Property taxes ($/capita, 1987)</td>
<td>f</td>
<td>(-)</td>
</tr>
<tr>
<td>LIVE</td>
<td>Livestock marketing cash receipts ($/capita, 1987)</td>
<td>g</td>
<td>(+)</td>
</tr>
<tr>
<td>CROP</td>
<td>Crop marketing cash receipts ($/capita, 1986-1987 average)</td>
<td>g</td>
<td>(+)</td>
</tr>
<tr>
<td>POP87</td>
<td>Population (in thoughts, 1987)</td>
<td>g</td>
<td>(+)</td>
</tr>
<tr>
<td>ES8700</td>
<td>SIC 2000 establishments (# of 100, 000 persons, 1987)</td>
<td>a</td>
<td>(+)</td>
</tr>
<tr>
<td>ES8710</td>
<td>SIC 2010 establishments (# of 100, 000 persons, 1987)</td>
<td>a</td>
<td>(+)</td>
</tr>
<tr>
<td>ES8720</td>
<td>SIC 2020 establishments (# of 100, 000 persons, 1987)</td>
<td>a</td>
<td>(+)</td>
</tr>
<tr>
<td>ES8730</td>
<td>SIC 2030 establishments (# of 100, 000 persons, 1987)</td>
<td>a</td>
<td>(+)</td>
</tr>
<tr>
<td>ES8740</td>
<td>SIC 2040 establishments (# of 100, 000 persons, 1987)</td>
<td>a</td>
<td>(+)</td>
</tr>
<tr>
<td>ES8750</td>
<td>SIC 2050 establishments (# of 100, 000 persons, 1987)</td>
<td>a</td>
<td>(+)</td>
</tr>
<tr>
<td>ES8760</td>
<td>SIC 2060 establishments (# of 100, 000 persons, 1987)</td>
<td>a</td>
<td>(+)</td>
</tr>
<tr>
<td>ES8770</td>
<td>SIC 2070 establishments (# of 100, 000 persons, 1987)</td>
<td>a</td>
<td>(+)</td>
</tr>
<tr>
<td>ES8780</td>
<td>SIC 2080 establishments (# of 100, 000 persons, 1987)</td>
<td>a</td>
<td>(+)</td>
</tr>
<tr>
<td>ES8790</td>
<td>SIC 2090 establishments (# of 100, 000 persons, 1987)</td>
<td>a</td>
<td>(+)</td>
</tr>
</tbody>
</table>

b. Constructed using TransCad GIS software and geo-coded from the U.S. Transportation Sampler CD-ROM.
c. Census of Manufacturing, 1987, Department of Commerce.
e. U.S.A. counties on CD-ROM, Department of Commerce.
g. Regional Economic Information System (REIS) CD-ROM, Department of Commerce.
### Table 6.4 The Summary Statistics for variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Number</th>
<th>Mean</th>
<th>Std Dev</th>
<th>Minimum</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>EST</td>
<td>2444</td>
<td>0.337</td>
<td>0.473</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>EST10</td>
<td>2444</td>
<td>0.184</td>
<td>0.388</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>EST20</td>
<td>2444</td>
<td>0.063</td>
<td>0.242</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>EST30</td>
<td>2444</td>
<td>0.108</td>
<td>0.311</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>EST40</td>
<td>2444</td>
<td>0.154</td>
<td>0.361</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>EST50</td>
<td>2444</td>
<td>0.141</td>
<td>0.348</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>EST60</td>
<td>2444</td>
<td>0.089</td>
<td>0.285</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>EST70</td>
<td>2444</td>
<td>0.049</td>
<td>0.215</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>EST80</td>
<td>2444</td>
<td>0.115</td>
<td>0.319</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>EST90</td>
<td>2444</td>
<td>0.178</td>
<td>0.382</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>HWY</td>
<td>2444</td>
<td>0.434</td>
<td>0.496</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>RAIL</td>
<td>2444</td>
<td>0.762</td>
<td>0.426</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>PORT</td>
<td>2444</td>
<td>0.042</td>
<td>0.201</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>WAGE</td>
<td>2444</td>
<td>8.507</td>
<td>1.106</td>
<td>6.849</td>
<td>11.251</td>
</tr>
<tr>
<td>HS_GRD</td>
<td>2444</td>
<td>69.541</td>
<td>7.321</td>
<td>56.800</td>
<td>81.900</td>
</tr>
<tr>
<td>UNEM</td>
<td>2444</td>
<td>8.685</td>
<td>4.098</td>
<td>1.800</td>
<td>37.900</td>
</tr>
<tr>
<td>HNDL</td>
<td>2444</td>
<td>0.045</td>
<td>0.013</td>
<td>0</td>
<td>0.137</td>
</tr>
<tr>
<td>OPRT</td>
<td>2444</td>
<td>0.191</td>
<td>0.067</td>
<td>0.034</td>
<td>0.438</td>
</tr>
<tr>
<td>PROPT</td>
<td>2444</td>
<td>0.428</td>
<td>0.363</td>
<td>0</td>
<td>5.694</td>
</tr>
<tr>
<td>LIVE</td>
<td>2444</td>
<td>1.886</td>
<td>4.888</td>
<td>0</td>
<td>127.213</td>
</tr>
<tr>
<td>CROP</td>
<td>2444</td>
<td>1.128</td>
<td>1.896</td>
<td>0</td>
<td>24.380</td>
</tr>
<tr>
<td>POP87</td>
<td>2444</td>
<td>0.081</td>
<td>0.277</td>
<td>0</td>
<td>8.466</td>
</tr>
<tr>
<td>ES87</td>
<td>2444</td>
<td>10.457</td>
<td>12.559</td>
<td>0</td>
<td>118.182</td>
</tr>
<tr>
<td>ES8710</td>
<td>2444</td>
<td>2.457</td>
<td>5.078</td>
<td>0</td>
<td>50.000</td>
</tr>
<tr>
<td>ES8720</td>
<td>2444</td>
<td>1.300</td>
<td>4.372</td>
<td>0</td>
<td>65.000</td>
</tr>
<tr>
<td>ES8730</td>
<td>2444</td>
<td>0.837</td>
<td>2.749</td>
<td>0</td>
<td>50.000</td>
</tr>
<tr>
<td>ES8740</td>
<td>2444</td>
<td>2.446</td>
<td>5.685</td>
<td>0</td>
<td>66.667</td>
</tr>
<tr>
<td>ES8750</td>
<td>2444</td>
<td>0.483</td>
<td>1.426</td>
<td>0</td>
<td>22.222</td>
</tr>
<tr>
<td>ES8760</td>
<td>2444</td>
<td>0.356</td>
<td>1.527</td>
<td>0</td>
<td>23.077</td>
</tr>
<tr>
<td>ES8770</td>
<td>2444</td>
<td>0.358</td>
<td>1.780</td>
<td>0</td>
<td>50.000</td>
</tr>
<tr>
<td>ES8780</td>
<td>2444</td>
<td>0.742</td>
<td>2.115</td>
<td>0</td>
<td>40.000</td>
</tr>
<tr>
<td>ES8790</td>
<td>2444</td>
<td>1.443</td>
<td>5.506</td>
<td>0</td>
<td>100.000</td>
</tr>
</tbody>
</table>
Table 6.5 The Results of the Linear Probability Models

<table>
<thead>
<tr>
<th></th>
<th>EST</th>
<th>EST10</th>
<th>EST20</th>
<th>EST30</th>
<th>EST40</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTERCEP</td>
<td>0.1568</td>
<td>0.0694</td>
<td>-0.0987</td>
<td>-0.0571</td>
<td>-0.1422</td>
</tr>
<tr>
<td></td>
<td>(1.211)</td>
<td>(0.644)</td>
<td>(-1.487)</td>
<td>(-0.670)</td>
<td>(-1.443)</td>
</tr>
<tr>
<td>HWY</td>
<td>0.0500**</td>
<td>0.0513***</td>
<td>0.0402***</td>
<td>0.0521***</td>
<td>0.0464***</td>
</tr>
<tr>
<td></td>
<td>(2.425)</td>
<td>(2.984)</td>
<td>(3.808)</td>
<td>(3.853)</td>
<td>(2.953)</td>
</tr>
<tr>
<td>RAIL</td>
<td>0.0252</td>
<td>0.0466**</td>
<td>0.0064</td>
<td>0.0235</td>
<td>0.0391**</td>
</tr>
<tr>
<td></td>
<td>(1.107)</td>
<td>(2.442)</td>
<td>(0.550)</td>
<td>(1.567)</td>
<td>(2.246)</td>
</tr>
<tr>
<td>PORT</td>
<td>-0.0434</td>
<td>0.0341</td>
<td>0.0674***</td>
<td>-0.0286</td>
<td>0.0275</td>
</tr>
<tr>
<td></td>
<td>(-0.868)</td>
<td>(0.817)</td>
<td>(2.627)</td>
<td>(-0.871)</td>
<td>(0.720)</td>
</tr>
<tr>
<td>WAGE</td>
<td>0.0214*</td>
<td>-0.0092</td>
<td>0.0183***</td>
<td>0.0267***</td>
<td>0.0156*</td>
</tr>
<tr>
<td></td>
<td>(1.782)</td>
<td>(-0.914)</td>
<td>(2.962)</td>
<td>(3.362)</td>
<td>(1.695)</td>
</tr>
<tr>
<td>HS_GRD</td>
<td>0.0028</td>
<td>0.0022</td>
<td>0.0013</td>
<td>0.0011</td>
<td>0.0027*</td>
</tr>
<tr>
<td></td>
<td>(1.343)</td>
<td>(1.256)</td>
<td>(1.218)</td>
<td>(0.779)</td>
<td>(1.675)</td>
</tr>
<tr>
<td>UNEM</td>
<td>-0.0084***</td>
<td>-0.0032</td>
<td>-0.0046***</td>
<td>-0.0035**</td>
<td>-0.0057***</td>
</tr>
<tr>
<td></td>
<td>(-3.181)</td>
<td>(-1.434)</td>
<td>(-3.416)</td>
<td>(-2.004)</td>
<td>(-2.817)</td>
</tr>
<tr>
<td>HNDL</td>
<td>1.1948</td>
<td>1.3702</td>
<td>-0.2496</td>
<td>-0.9867</td>
<td>-0.3527</td>
</tr>
<tr>
<td></td>
<td>(1.175)</td>
<td>(1.619)</td>
<td>(-0.485)</td>
<td>(-1.497)</td>
<td>(0.460)</td>
</tr>
<tr>
<td>OPER</td>
<td>-0.7106***</td>
<td>-0.1416</td>
<td>-0.1781</td>
<td>-0.3657**</td>
<td>-0.0184</td>
</tr>
<tr>
<td></td>
<td>(-3.250)</td>
<td>(-0.776)</td>
<td>(-1.591)</td>
<td>(-2.552)</td>
<td>(-0.111)</td>
</tr>
<tr>
<td>PROPT</td>
<td>-0.0808**</td>
<td>-0.0264</td>
<td>-0.0498***</td>
<td>-0.0475**</td>
<td>-0.0414</td>
</tr>
<tr>
<td></td>
<td>(-2.303)</td>
<td>(-0.904)</td>
<td>(-2.780)</td>
<td>(-2.065)</td>
<td>(-1.545)</td>
</tr>
<tr>
<td>LIVE</td>
<td>-0.0007</td>
<td>0.0029</td>
<td>0.0005</td>
<td>-0.0019</td>
<td>-0.0012</td>
</tr>
<tr>
<td></td>
<td>(-0.309)</td>
<td>(1.557)</td>
<td>(0.391)</td>
<td>(-1.262)</td>
<td>(-0.669)</td>
</tr>
<tr>
<td>CROP</td>
<td>0.0013</td>
<td>-0.0105**</td>
<td>-0.0070**</td>
<td>-0.0099**</td>
<td>0.0078*</td>
</tr>
<tr>
<td></td>
<td>(0.215)</td>
<td>(-2.055)</td>
<td>(-2.244)</td>
<td>(-2.452)</td>
<td>(1.683)</td>
</tr>
<tr>
<td>POP87</td>
<td>0.1124***</td>
<td>0.0479</td>
<td>0.0344*</td>
<td>0.0483**</td>
<td>0.1658***</td>
</tr>
<tr>
<td></td>
<td>(3.001)</td>
<td>(1.532)</td>
<td>(1.792)</td>
<td>(1.964)</td>
<td>(5.802)</td>
</tr>
<tr>
<td>ES87</td>
<td>-0.0055***</td>
<td>-0.0048***</td>
<td>-0.0002</td>
<td>0.0009</td>
<td>-0.0030**</td>
</tr>
<tr>
<td></td>
<td>(-7.029)</td>
<td>(-3.047)</td>
<td>(-0.208)</td>
<td>(0.388)</td>
<td>(-2.242)</td>
</tr>
<tr>
<td>R-square</td>
<td>0.0554</td>
<td>0.0208</td>
<td>0.0544</td>
<td>0.0589</td>
<td>0.0551</td>
</tr>
<tr>
<td>Adj R-squ</td>
<td>0.0504</td>
<td>0.0155</td>
<td>0.0493</td>
<td>0.0539</td>
<td>0.0501</td>
</tr>
<tr>
<td>SSE</td>
<td>516.215</td>
<td>360.142</td>
<td>135.619</td>
<td>222.349</td>
<td>300.615</td>
</tr>
</tbody>
</table>

Note:  ** significant at the 5% level;  
       *** significant at the 1% level.
Table 6.6 The Results of the Linear Probability Models (continued)

<table>
<thead>
<tr>
<th>Variable</th>
<th>EST50</th>
<th>EST60</th>
<th>EST70</th>
<th>EST80</th>
<th>EST90</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTERCEP</td>
<td>0.0335</td>
<td>0.0920</td>
<td>-0.0181</td>
<td>-0.0432</td>
<td>0.1016</td>
</tr>
<tr>
<td>HWY</td>
<td>0.0541***</td>
<td>0.0510***</td>
<td>0.0228**</td>
<td>0.0440***</td>
<td>0.0588***</td>
</tr>
<tr>
<td>RAIL</td>
<td>0.0202</td>
<td>0.0199</td>
<td>0.0133</td>
<td>-0.0027</td>
<td>0.0155</td>
</tr>
<tr>
<td>PORT</td>
<td>0.1339***</td>
<td>0.0778***</td>
<td>0.0120</td>
<td>0.0351</td>
<td>0.0846**</td>
</tr>
<tr>
<td>WAGE</td>
<td>0.0403***</td>
<td>0.0189***</td>
<td>0.0035</td>
<td>0.0462***</td>
<td>0.0130</td>
</tr>
<tr>
<td>HS_GRD</td>
<td>-0.0005</td>
<td>-0.0007</td>
<td>0.0000</td>
<td>-0.0003</td>
<td>0.0017</td>
</tr>
<tr>
<td>UNEM</td>
<td>-0.0111***</td>
<td>-0.0038**</td>
<td>-0.0021*</td>
<td>-0.0056***</td>
<td>-0.0010</td>
</tr>
<tr>
<td>HNDL</td>
<td>-1.7377**</td>
<td>-1.1678**</td>
<td>1.3951***</td>
<td>-1.1753*</td>
<td>-0.5924</td>
</tr>
<tr>
<td>OPER</td>
<td>-0.2861*</td>
<td>-0.3111**</td>
<td>-0.1586</td>
<td>-0.5368***</td>
<td>-0.6482***</td>
</tr>
<tr>
<td>PROPT</td>
<td>-0.0208</td>
<td>-0.0190</td>
<td>-0.0112</td>
<td>-0.0256</td>
<td>-0.0582**</td>
</tr>
<tr>
<td>LIVE</td>
<td>-0.0019</td>
<td>-0.0023*</td>
<td>0.0006</td>
<td>-0.0031**</td>
<td>-0.0037**</td>
</tr>
<tr>
<td>CROP</td>
<td>-0.0177***</td>
<td>-0.0081**</td>
<td>0.0011</td>
<td>-0.0169***</td>
<td>-0.0118**</td>
</tr>
<tr>
<td>POP87</td>
<td>0.2413***</td>
<td>0.1247***</td>
<td>0.0488***</td>
<td>0.0772***</td>
<td>0.1665***</td>
</tr>
<tr>
<td>ES87</td>
<td>-0.0063</td>
<td>0.0010</td>
<td>0.0021</td>
<td>0.0005</td>
<td>-0.0024*</td>
</tr>
<tr>
<td>R-square</td>
<td>0.1675</td>
<td>0.0892</td>
<td>0.0163</td>
<td>0.1067</td>
<td>0.0794</td>
</tr>
<tr>
<td>Adj R-squ</td>
<td>0.1630</td>
<td>0.0844</td>
<td>0.0111</td>
<td>0.1020</td>
<td>0.0745</td>
</tr>
<tr>
<td>SSE</td>
<td>245.680</td>
<td>180.837</td>
<td>111.358</td>
<td>222.147</td>
<td>329.175</td>
</tr>
</tbody>
</table>

Note: ** significant at the 5% level; *** significant at the 1% level.
Table 6.7 The Results of the Logit Models

<table>
<thead>
<tr>
<th></th>
<th>EST</th>
<th>EST10</th>
<th>EST20</th>
<th>EST30</th>
<th>EST40</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTERCEP</td>
<td>-1.7736***</td>
<td>-2.4048***</td>
<td>-6.3708***</td>
<td>-4.2813***</td>
<td>-4.3723***</td>
</tr>
<tr>
<td></td>
<td>(-2.896)</td>
<td>(-3.293)</td>
<td>(-5.091)</td>
<td>(-4.149)</td>
<td>(-5.322)</td>
</tr>
<tr>
<td>HWY</td>
<td>0.2202**</td>
<td>0.3438***</td>
<td>0.7518***</td>
<td>0.5137***</td>
<td>0.3517***</td>
</tr>
<tr>
<td></td>
<td>(2.268)</td>
<td>(2.973)</td>
<td>(3.671)</td>
<td>(3.111)</td>
<td>(2.738)</td>
</tr>
<tr>
<td>RAIL</td>
<td>0.1522</td>
<td>0.3598***</td>
<td>0.2347</td>
<td>0.3612*</td>
<td>0.3975**</td>
</tr>
<tr>
<td></td>
<td>(1.371)</td>
<td>(2.574)</td>
<td>(0.960)</td>
<td>(1.755)</td>
<td>(2.478)</td>
</tr>
<tr>
<td>PORT</td>
<td>-0.2212</td>
<td>0.2022</td>
<td>0.6318**</td>
<td>-0.2465</td>
<td>0.0992</td>
</tr>
<tr>
<td></td>
<td>(-0.942)</td>
<td>(0.795)</td>
<td>(2.044)</td>
<td>(-0.758)</td>
<td>(0.369)</td>
</tr>
<tr>
<td>WAGE</td>
<td>0.0947*</td>
<td>-0.0670</td>
<td>0.2826***</td>
<td>0.2043**</td>
<td>0.0990</td>
</tr>
<tr>
<td></td>
<td>(1.678)</td>
<td>(-0.984)</td>
<td>(2.644)</td>
<td>(2.198)</td>
<td>(1.355)</td>
</tr>
<tr>
<td>HS_GRD</td>
<td>0.0167*</td>
<td>0.0169</td>
<td>0.0401*</td>
<td>0.0269</td>
<td>0.0275**</td>
</tr>
<tr>
<td></td>
<td>(1.670)</td>
<td>(1.413)</td>
<td>(1.916)</td>
<td>(1.557)</td>
<td>(2.031)</td>
</tr>
<tr>
<td>UNEM</td>
<td>-0.0395***</td>
<td>-0.0230</td>
<td>-0.1052***</td>
<td>-0.0368</td>
<td>-0.0519***</td>
</tr>
<tr>
<td></td>
<td>(-3.048)</td>
<td>(-1.477)</td>
<td>(-3.326)</td>
<td>(-1.602)</td>
<td>(-2.770)</td>
</tr>
<tr>
<td>HNDL</td>
<td>6.4267</td>
<td>10.3104*</td>
<td>1.5368</td>
<td>-7.8574</td>
<td>-1.8774</td>
</tr>
<tr>
<td></td>
<td>(1.300)</td>
<td>(1.756)</td>
<td>(0.136)</td>
<td>(-0.822)</td>
<td>(-0.274)</td>
</tr>
<tr>
<td>OPER</td>
<td>-3.2036***</td>
<td>-1.0421</td>
<td>-4.7089*</td>
<td>-5.0781**</td>
<td>-0.1283</td>
</tr>
<tr>
<td></td>
<td>(-3.014)</td>
<td>(-0.831)</td>
<td>(-1.949)</td>
<td>(-2.527)</td>
<td>(-0.088)</td>
</tr>
<tr>
<td>PROPT</td>
<td>-0.4050**</td>
<td>-0.2066</td>
<td>-1.3147***</td>
<td>-0.6195*</td>
<td>-0.4043*</td>
</tr>
<tr>
<td></td>
<td>(-2.356)</td>
<td>(-0.952)</td>
<td>(-2.972)</td>
<td>(-1.883)</td>
<td>(-0.165)</td>
</tr>
<tr>
<td>LIVE</td>
<td>-0.0038</td>
<td>0.0197*</td>
<td>0.0183</td>
<td>-0.1163**</td>
<td>-0.0126</td>
</tr>
<tr>
<td></td>
<td>(-0.327)</td>
<td>(1.713)</td>
<td>(0.703)</td>
<td>(-2.129)</td>
<td>(-0.688)</td>
</tr>
<tr>
<td>CROP</td>
<td>0.0113</td>
<td>-0.0838**</td>
<td>-0.2167**</td>
<td>-0.1071</td>
<td>0.0649*</td>
</tr>
<tr>
<td></td>
<td>(0.386)</td>
<td>(-2.115)</td>
<td>(-2.353)</td>
<td>(-1.578)</td>
<td>(1.814)</td>
</tr>
<tr>
<td>POP87</td>
<td>0.6463**</td>
<td>0.2094</td>
<td>0.0309</td>
<td>0.1008</td>
<td>1.1121***</td>
</tr>
<tr>
<td></td>
<td>(2.538)</td>
<td>(1.182)</td>
<td>(0.160)</td>
<td>(0.512)</td>
<td>(3.976)</td>
</tr>
<tr>
<td>ES87</td>
<td>-0.0328***</td>
<td>-0.0433***</td>
<td>0.002</td>
<td>0.0124</td>
<td>-0.0309**</td>
</tr>
<tr>
<td></td>
<td>(-6.872)</td>
<td>(-3.066)</td>
<td>(0.114)</td>
<td>(0.411)</td>
<td>(-2.222)</td>
</tr>
</tbody>
</table>
Table 6.8 The Results of the Logit Model (continued)

<table>
<thead>
<tr>
<th></th>
<th>EST50</th>
<th>EST60</th>
<th>EST70</th>
<th>EST80</th>
<th>EST90</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTERCEP</td>
<td>-3.7431***</td>
<td>-2.8610***</td>
<td>-4.5087***</td>
<td>-4.2929***</td>
<td>2.3545***</td>
</tr>
<tr>
<td></td>
<td>(-3.865)</td>
<td>(-2.702)</td>
<td>(-3.438)</td>
<td>(-4.335)</td>
<td>(-2.942)</td>
</tr>
<tr>
<td>HWY</td>
<td>0.3505**</td>
<td>0.6186***</td>
<td>0.5487***</td>
<td>0.3489**</td>
<td>0.3483***</td>
</tr>
<tr>
<td></td>
<td>(2.163)</td>
<td>(3.384)</td>
<td>(2.604)</td>
<td>(2.140)</td>
<td>(2.683)</td>
</tr>
<tr>
<td>RAIL</td>
<td>0.2629</td>
<td>0.4080*</td>
<td>0.4219</td>
<td>0.0556</td>
<td>0.1343</td>
</tr>
<tr>
<td></td>
<td>(1.348)</td>
<td>(1.795)</td>
<td>(1.526)</td>
<td>(0.293)</td>
<td>(0.875)</td>
</tr>
<tr>
<td>PORT</td>
<td>0.5202</td>
<td>0.4399</td>
<td>0.2874</td>
<td>0.0567</td>
<td>0.3414</td>
</tr>
<tr>
<td></td>
<td>(1.773)</td>
<td>(1.585)</td>
<td>(0.690)</td>
<td>(0.208)</td>
<td>(1.334)</td>
</tr>
<tr>
<td>WAGE</td>
<td>0.2821***</td>
<td>0.1357</td>
<td>0.0946</td>
<td>0.3058***</td>
<td>0.0243</td>
</tr>
<tr>
<td></td>
<td>(3.071)</td>
<td>(1.365)</td>
<td>(0.794)</td>
<td>(3.375)</td>
<td>(0.323)</td>
</tr>
<tr>
<td>HS_GRD</td>
<td>0.0170</td>
<td>0.0135</td>
<td>-0.0002</td>
<td>0.0284*</td>
<td>0.0255*</td>
</tr>
<tr>
<td></td>
<td>(1.022)</td>
<td>(0.737)</td>
<td>(-0.008)</td>
<td>(1.671)</td>
<td>(1.890)</td>
</tr>
<tr>
<td>UNEM</td>
<td>-0.1193***</td>
<td>-0.0481*</td>
<td>-0.0594*</td>
<td>-0.0543**</td>
<td>-0.0034</td>
</tr>
<tr>
<td></td>
<td>(-4.639)</td>
<td>(-1.910)</td>
<td>(-1.929)</td>
<td>(-2.407)</td>
<td>(-0.202)</td>
</tr>
<tr>
<td>HNDL</td>
<td>-12.5169</td>
<td>-14.1636</td>
<td>31.2295***</td>
<td>-2.7790</td>
<td>-0.5610</td>
</tr>
<tr>
<td></td>
<td>(-1.341)</td>
<td>(-1.296)</td>
<td>(3.204)</td>
<td>(-0.285)</td>
<td>(-0.078)</td>
</tr>
<tr>
<td>OPER</td>
<td>-2.3007</td>
<td>-4.9765**</td>
<td>-4.2057*</td>
<td>-7.9475***</td>
<td>-5.5481***</td>
</tr>
<tr>
<td></td>
<td>(-1.224)</td>
<td>(-2.273)</td>
<td>(-1.780)</td>
<td>(-3.869)</td>
<td>(-3.630)</td>
</tr>
<tr>
<td>PROPT</td>
<td>-0.3315</td>
<td>-0.2787</td>
<td>-0.3072</td>
<td>-0.3387</td>
<td>-0.4914</td>
</tr>
<tr>
<td></td>
<td>(-1.102)</td>
<td>(-0.837)</td>
<td>(-0.736)</td>
<td>(-1.139)</td>
<td>(-1.916)</td>
</tr>
<tr>
<td>LIVE</td>
<td>-0.0584</td>
<td>-0.2892***</td>
<td>0.0109</td>
<td>-0.4422***</td>
<td>-0.1307***</td>
</tr>
<tr>
<td></td>
<td>(-1.286)</td>
<td>(-3.439)</td>
<td>(0.537)</td>
<td>(-4.668)</td>
<td>(-3.196)</td>
</tr>
<tr>
<td>CROP</td>
<td>-0.2360***</td>
<td>-0.0681</td>
<td>0.0101</td>
<td>-0.1856**</td>
<td>-0.0662</td>
</tr>
<tr>
<td></td>
<td>(-3.018)</td>
<td>(-0.912)</td>
<td>(0.173)</td>
<td>(-2.229)</td>
<td>(-1.376)</td>
</tr>
<tr>
<td>POP87</td>
<td>2.4574***</td>
<td>0.4217*</td>
<td>0.3425*</td>
<td>0.0798</td>
<td>1.0306***</td>
</tr>
<tr>
<td></td>
<td>(5.039)</td>
<td>(1.805)</td>
<td>(1.663)</td>
<td>(0.435)</td>
<td>(3.389)</td>
</tr>
<tr>
<td>ES87</td>
<td>-0.1465**</td>
<td>0.0128</td>
<td>0.0324</td>
<td>0.0029</td>
<td>-0.0446*</td>
</tr>
<tr>
<td></td>
<td>(-2.049)</td>
<td>(0.225)</td>
<td>(0.856)</td>
<td>(0.082)</td>
<td>(-1.908)</td>
</tr>
</tbody>
</table>
Table 6.9 The Prediction Accuracy and Errors of the LPMs (in-sample)

<table>
<thead>
<tr>
<th>SIC</th>
<th>CC</th>
<th>CR(%)</th>
<th>MSE</th>
<th>MABSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000</td>
<td>1647</td>
<td>67.33</td>
<td>0.211</td>
<td>0.424</td>
</tr>
<tr>
<td>2010</td>
<td>1995</td>
<td>81.56</td>
<td>0.147</td>
<td>0.295</td>
</tr>
<tr>
<td>2020</td>
<td>2292</td>
<td>93.70</td>
<td>0.055</td>
<td>0.116</td>
</tr>
<tr>
<td>2030</td>
<td>2180</td>
<td>89.13</td>
<td>0.091</td>
<td>0.186</td>
</tr>
<tr>
<td>2040</td>
<td>2069</td>
<td>84.59</td>
<td>0.123</td>
<td>0.247</td>
</tr>
<tr>
<td>2050</td>
<td>2130</td>
<td>87.08</td>
<td>0.101</td>
<td>0.213</td>
</tr>
<tr>
<td>2060</td>
<td>2225</td>
<td>90.96</td>
<td>0.074</td>
<td>0.153</td>
</tr>
<tr>
<td>2070</td>
<td>2327</td>
<td>95.13</td>
<td>0.046</td>
<td>0.092</td>
</tr>
<tr>
<td>2080</td>
<td>2164</td>
<td>88.47</td>
<td>0.091</td>
<td>0.192</td>
</tr>
<tr>
<td>2090</td>
<td>2015</td>
<td>82.38</td>
<td>0.135</td>
<td>0.272</td>
</tr>
</tbody>
</table>

Table 6.10 The Prediction Accuracy and Errors of the LPMs (out-of-sample)

<table>
<thead>
<tr>
<th>SIC</th>
<th>CC</th>
<th>CR (%)</th>
<th>MSE</th>
<th>MABSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000</td>
<td>397</td>
<td>67.50</td>
<td>0.217</td>
<td>0.429</td>
</tr>
<tr>
<td>2010</td>
<td>504</td>
<td>81.56</td>
<td>0.137</td>
<td>0.282</td>
</tr>
<tr>
<td>2020</td>
<td>568</td>
<td>93.74</td>
<td>0.055</td>
<td>0.113</td>
</tr>
<tr>
<td>2030</td>
<td>540</td>
<td>89.17</td>
<td>0.088</td>
<td>0.181</td>
</tr>
<tr>
<td>2040</td>
<td>510</td>
<td>84.63</td>
<td>0.127</td>
<td>0.250</td>
</tr>
<tr>
<td>2050</td>
<td>531</td>
<td>87.37</td>
<td>0.099</td>
<td>0.207</td>
</tr>
<tr>
<td>2060</td>
<td>571</td>
<td>91.01</td>
<td>0.051</td>
<td>0.128</td>
</tr>
<tr>
<td>2070</td>
<td>580</td>
<td>95.09</td>
<td>0.038</td>
<td>0.084</td>
</tr>
<tr>
<td>2080</td>
<td>522</td>
<td>88.35</td>
<td>0.105</td>
<td>0.203</td>
</tr>
<tr>
<td>2090</td>
<td>496</td>
<td>82.46</td>
<td>0.139</td>
<td>0.275</td>
</tr>
</tbody>
</table>
Table 6.11 The Prediction Accuracy and Errors of the Logit Models (in-sample)

<table>
<thead>
<tr>
<th>SIC</th>
<th>CC</th>
<th>CR (%)</th>
<th>MSE</th>
<th>MABSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000</td>
<td>1651</td>
<td>65.62</td>
<td>0.211</td>
<td>0.421</td>
</tr>
<tr>
<td>2010</td>
<td>1995</td>
<td>83.31</td>
<td>0.147</td>
<td>0.295</td>
</tr>
<tr>
<td>2020</td>
<td>2293</td>
<td>93.88</td>
<td>0.055</td>
<td>0.110</td>
</tr>
<tr>
<td>2030</td>
<td>2181</td>
<td>89.26</td>
<td>0.090</td>
<td>0.180</td>
</tr>
<tr>
<td>2040</td>
<td>2070</td>
<td>84.30</td>
<td>0.123</td>
<td>0.245</td>
</tr>
<tr>
<td>2050</td>
<td>2137</td>
<td>87.77</td>
<td>0.096</td>
<td>0.193</td>
</tr>
<tr>
<td>2060</td>
<td>2226</td>
<td>94.38</td>
<td>0.073</td>
<td>0.146</td>
</tr>
<tr>
<td>2070</td>
<td>2326</td>
<td>95.87</td>
<td>0.046</td>
<td>0.091</td>
</tr>
<tr>
<td>2080</td>
<td>2161</td>
<td>86.28</td>
<td>0.087</td>
<td>0.175</td>
</tr>
<tr>
<td>2090</td>
<td>2017</td>
<td>81.98</td>
<td>0.133</td>
<td>0.266</td>
</tr>
</tbody>
</table>

Table 6.12 The Prediction Accuracy and Errors of the Logit Models (out-of-sample)

<table>
<thead>
<tr>
<th>SIC</th>
<th>Count</th>
<th>Correct(%)</th>
<th>MSE</th>
<th>MABSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000</td>
<td>405</td>
<td>66.94</td>
<td>0.215</td>
<td>0.425</td>
</tr>
<tr>
<td>2010</td>
<td>504</td>
<td>83.31</td>
<td>0.137</td>
<td>0.282</td>
</tr>
<tr>
<td>2020</td>
<td>568</td>
<td>93.88</td>
<td>0.055</td>
<td>0.107</td>
</tr>
<tr>
<td>2030</td>
<td>540</td>
<td>89.26</td>
<td>0.088</td>
<td>0.176</td>
</tr>
<tr>
<td>2040</td>
<td>509</td>
<td>84.13</td>
<td>0.127</td>
<td>0.248</td>
</tr>
<tr>
<td>2050</td>
<td>535</td>
<td>88.43</td>
<td>0.094</td>
<td>0.187</td>
</tr>
<tr>
<td>2060</td>
<td>571</td>
<td>94.38</td>
<td>0.051</td>
<td>0.122</td>
</tr>
<tr>
<td>2070</td>
<td>579</td>
<td>95.70</td>
<td>0.039</td>
<td>0.084</td>
</tr>
<tr>
<td>2080</td>
<td>523</td>
<td>86.45</td>
<td>0.100</td>
<td>0.186</td>
</tr>
<tr>
<td>2090</td>
<td>499</td>
<td>82.48</td>
<td>0.138</td>
<td>0.271</td>
</tr>
</tbody>
</table>
Table 6.13 The Weight Matrix of Hidden Layer of the Neural Network (SIC2000):
Experiment One

<table>
<thead>
<tr>
<th></th>
<th>Neuron 1</th>
<th>Neuron 2</th>
<th>Neuron 3</th>
<th>Neuron 4</th>
<th>Neuron 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bias</td>
<td>15.118</td>
<td>-1.5089</td>
<td>-15.3078</td>
<td>12.6094</td>
<td>15.2349</td>
</tr>
<tr>
<td>HWY</td>
<td>15.2676</td>
<td>-19.5936</td>
<td>3.2961</td>
<td>4.7762</td>
<td>-10.7377</td>
</tr>
<tr>
<td>RAIL</td>
<td>-16.246</td>
<td>-14.0484</td>
<td>1.2352</td>
<td>3.1089</td>
<td>14.3512</td>
</tr>
<tr>
<td>PORT</td>
<td>-2.1303</td>
<td>-19.7436</td>
<td>-1.819</td>
<td>-3.7851</td>
<td>1.4083</td>
</tr>
<tr>
<td>WAGE</td>
<td>5.6259</td>
<td>17.3894</td>
<td>-25.9349</td>
<td>-41.6511</td>
<td>-4.2947</td>
</tr>
<tr>
<td>HS_GRD</td>
<td>-2.5707</td>
<td>-16.9169</td>
<td>-11.1023</td>
<td>5.8751</td>
<td>1.5131</td>
</tr>
<tr>
<td>UNEM</td>
<td>0.1423</td>
<td>7.7039</td>
<td>5.9002</td>
<td>5.6703</td>
<td>-6.1285</td>
</tr>
<tr>
<td>HNDL</td>
<td>7.4119</td>
<td>-1.3151</td>
<td>-3.4011</td>
<td>0.6143</td>
<td>-2.7526</td>
</tr>
<tr>
<td>OPER</td>
<td>-4.7441</td>
<td>12.4022</td>
<td>6.7398</td>
<td>6.9394</td>
<td>0.1004</td>
</tr>
<tr>
<td>PROPT</td>
<td>-2.294</td>
<td>-3.7382</td>
<td>7.1805</td>
<td>6.8658</td>
<td>3.5793</td>
</tr>
<tr>
<td>POP87</td>
<td>-11.7699</td>
<td>9.3062</td>
<td>-0.4312</td>
<td>8.1872</td>
<td>18.671</td>
</tr>
</tbody>
</table>

Table 6.14 The Weight Vector of Output Layer of the Neural Network (SIC2000):
Experiment One

<table>
<thead>
<tr>
<th></th>
<th>Neuron 1</th>
<th>Neuron 2</th>
<th>Neuron 3</th>
<th>Neuron 4</th>
<th>Neuron 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bias</td>
<td>-11.6958</td>
<td>15.118</td>
<td>-1.5089</td>
<td>-15.3078</td>
<td>12.6094</td>
</tr>
</tbody>
</table>
Table 6.15 The Weight Matrix of Hidden Layer of the Neural Network (SIC2000): Experiment Two

<table>
<thead>
<tr>
<th></th>
<th>Neuron 1</th>
<th>Neuron 2</th>
<th>Neuron 3</th>
<th>Neuron 4</th>
<th>Neuron 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bias</td>
<td>4.8018</td>
<td>-253.25</td>
<td>64.5099</td>
<td>-171.178</td>
<td>-0.9333</td>
</tr>
<tr>
<td>HWY</td>
<td>-0.1145</td>
<td>0.1182</td>
<td>-15.0355</td>
<td>1.3819</td>
<td>-112.171</td>
</tr>
<tr>
<td>RAIL</td>
<td>-0.9189</td>
<td>-1.3853</td>
<td>14.9368</td>
<td>181.7961</td>
<td>1.283</td>
</tr>
<tr>
<td>PORT</td>
<td>-0.693</td>
<td>-5.0573</td>
<td>-58.1518</td>
<td>-4.6232</td>
<td>21.1651</td>
</tr>
<tr>
<td>WAGE</td>
<td>-0.6473</td>
<td>33.0202</td>
<td>-74.3113</td>
<td>9.8596</td>
<td>50.2424</td>
</tr>
<tr>
<td>HS_GRD</td>
<td>0.133</td>
<td>264.593</td>
<td>103.4268</td>
<td>-9.2037</td>
<td>-103.219</td>
</tr>
<tr>
<td>UNEM</td>
<td>-0.5755</td>
<td>-38.1395</td>
<td>-5.6788</td>
<td>-20.4941</td>
<td>-4.9867</td>
</tr>
<tr>
<td>HNDL</td>
<td>-0.4326</td>
<td>20.2596</td>
<td>-49.5239</td>
<td>75.6458</td>
<td>2.0991</td>
</tr>
<tr>
<td>OPER</td>
<td>0.3087</td>
<td>-29.3327</td>
<td>-9.1021</td>
<td>-73.8944</td>
<td>37.9469</td>
</tr>
<tr>
<td>PROPT</td>
<td>0.4711</td>
<td>-63.1108</td>
<td>-15.9809</td>
<td>33.3057</td>
<td>-0.9137</td>
</tr>
<tr>
<td>LIVE</td>
<td>-0.8399</td>
<td>86.8525</td>
<td>2.8744</td>
<td>673.1599</td>
<td>340.2703</td>
</tr>
<tr>
<td>CROP</td>
<td>3.5358</td>
<td>-39.7897</td>
<td>-7.7764</td>
<td>84.9715</td>
<td>-484.296</td>
</tr>
<tr>
<td>POP87</td>
<td>963.2186</td>
<td>-59.082</td>
<td>-30.0893</td>
<td>13.8549</td>
<td>461.7065</td>
</tr>
<tr>
<td>ES87</td>
<td>-3.3673</td>
<td>0.0995</td>
<td>105.4396</td>
<td>-168.751</td>
<td>20.4985</td>
</tr>
</tbody>
</table>

Table 6.16 The Weight Vector of Output Layer of the Neural Network (SIC2000): Experiment Two

<table>
<thead>
<tr>
<th></th>
<th>Neuron 1</th>
<th>Neuron 2</th>
<th>Neuron 3</th>
<th>Neuron 4</th>
<th>Neuron 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bias</td>
<td>-48.2900</td>
<td>113.0719</td>
<td>1.2050</td>
<td>-65.6447</td>
<td>1.3878</td>
</tr>
</tbody>
</table>

180
Table 6.17 The Prediction Accuracy and Errors of the Neural Networks (in-sample)

<table>
<thead>
<tr>
<th>SIC</th>
<th>CC</th>
<th>CR(%)</th>
<th>MSE</th>
<th>MABSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000</td>
<td>1729</td>
<td>70.74</td>
<td>0.200</td>
<td>0.397</td>
</tr>
<tr>
<td>2010</td>
<td>1991</td>
<td>81.46</td>
<td>0.148</td>
<td>0.306</td>
</tr>
<tr>
<td>2020</td>
<td>2302</td>
<td>94.19</td>
<td>0.05</td>
<td>0.094</td>
</tr>
<tr>
<td>2030</td>
<td>2179</td>
<td>89.16</td>
<td>0.098</td>
<td>0.18</td>
</tr>
<tr>
<td>2040</td>
<td>2069</td>
<td>84.66</td>
<td>0.133</td>
<td>0.303</td>
</tr>
<tr>
<td>2050</td>
<td>2128</td>
<td>87.07</td>
<td>0.100</td>
<td>0.204</td>
</tr>
<tr>
<td>2060</td>
<td>2226</td>
<td>91.08</td>
<td>0.081</td>
<td>0.162</td>
</tr>
<tr>
<td>2070</td>
<td>2325</td>
<td>95.13</td>
<td>0.049</td>
<td>0.049</td>
</tr>
<tr>
<td>2080</td>
<td>2193</td>
<td>89.73</td>
<td>0.077</td>
<td>0.137</td>
</tr>
<tr>
<td>2090</td>
<td>2028</td>
<td>82.98</td>
<td>0.130</td>
<td>0.256</td>
</tr>
</tbody>
</table>

Table 6.18 The Prediction Accuracy and Errors of the Neural Networks (out-of-sample)

<table>
<thead>
<tr>
<th>SIC</th>
<th>CC</th>
<th>CR (%)</th>
<th>MSE</th>
<th>MABSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000</td>
<td>403</td>
<td>66.61</td>
<td>0.236</td>
<td>0.426</td>
</tr>
<tr>
<td>2010</td>
<td>499</td>
<td>82.48</td>
<td>0.144</td>
<td>0.301</td>
</tr>
<tr>
<td>2020</td>
<td>568</td>
<td>93.88</td>
<td>0.058</td>
<td>0.119</td>
</tr>
<tr>
<td>2030</td>
<td>540</td>
<td>89.26</td>
<td>0.096</td>
<td>0.165</td>
</tr>
<tr>
<td>2040</td>
<td>511</td>
<td>84.46</td>
<td>0.134</td>
<td>0.303</td>
</tr>
<tr>
<td>2050</td>
<td>533</td>
<td>88.10</td>
<td>0.097</td>
<td>0.184</td>
</tr>
<tr>
<td>2060</td>
<td>571</td>
<td>94.38</td>
<td>0.056</td>
<td>0.135</td>
</tr>
<tr>
<td>2070</td>
<td>580</td>
<td>95.87</td>
<td>0.041</td>
<td>0.041</td>
</tr>
<tr>
<td>2080</td>
<td>522</td>
<td>86.28</td>
<td>0.123</td>
<td>0.202</td>
</tr>
<tr>
<td>2090</td>
<td>498</td>
<td>82.31</td>
<td>0.141</td>
<td>0.249</td>
</tr>
</tbody>
</table>
Table 6.19 Comparison of Correctness Counts among the Three Methods (in-sample)

<table>
<thead>
<tr>
<th>SIC</th>
<th>LPM</th>
<th>LOGIT</th>
<th>NN</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000</td>
<td>1647</td>
<td>1651</td>
<td>1729</td>
</tr>
<tr>
<td>2020</td>
<td>2292</td>
<td>2293</td>
<td>2302</td>
</tr>
<tr>
<td>2030</td>
<td>2180</td>
<td>2181</td>
<td>2179</td>
</tr>
<tr>
<td>2040</td>
<td>2069</td>
<td>2070</td>
<td>2069</td>
</tr>
<tr>
<td>2050</td>
<td>2130</td>
<td>2137</td>
<td>2128</td>
</tr>
<tr>
<td>2060</td>
<td>2225</td>
<td>2226</td>
<td>2226</td>
</tr>
<tr>
<td>2070</td>
<td>2327</td>
<td>2326</td>
<td>2325</td>
</tr>
<tr>
<td>2080</td>
<td>2164</td>
<td>2161</td>
<td>2193</td>
</tr>
<tr>
<td>2090</td>
<td>2015</td>
<td>2017</td>
<td>2028</td>
</tr>
</tbody>
</table>

Table 6.20 Comparison of Correctness Counts among the Three Methods (out-of-sample)

<table>
<thead>
<tr>
<th>SIC</th>
<th>LPM</th>
<th>LOGIT</th>
<th>NN</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000</td>
<td>397</td>
<td>405</td>
<td>403</td>
</tr>
<tr>
<td>2010</td>
<td>504</td>
<td>504</td>
<td>499</td>
</tr>
<tr>
<td>2020</td>
<td>568</td>
<td>568</td>
<td>568</td>
</tr>
<tr>
<td>2030</td>
<td>540</td>
<td>540</td>
<td>540</td>
</tr>
<tr>
<td>2040</td>
<td>510</td>
<td>509</td>
<td>511</td>
</tr>
<tr>
<td>2050</td>
<td>531</td>
<td>535</td>
<td>533</td>
</tr>
<tr>
<td>2060</td>
<td>571</td>
<td>571</td>
<td>571</td>
</tr>
<tr>
<td>2070</td>
<td>580</td>
<td>579</td>
<td>580</td>
</tr>
<tr>
<td>2080</td>
<td>522</td>
<td>523</td>
<td>522</td>
</tr>
<tr>
<td>2090</td>
<td>496</td>
<td>499</td>
<td>498</td>
</tr>
</tbody>
</table>
Table 6.21 Comparison of Mean Squared Errors (MSE) among the Three Methods (in-sample)

<table>
<thead>
<tr>
<th>SIC</th>
<th>LPM</th>
<th>LOGIT</th>
<th>NN</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000</td>
<td>0.211</td>
<td>0.211</td>
<td>0.2</td>
</tr>
<tr>
<td>2010</td>
<td>0.147</td>
<td>0.147</td>
<td>0.148</td>
</tr>
<tr>
<td>2020</td>
<td>0.055</td>
<td>0.055</td>
<td>0.05</td>
</tr>
<tr>
<td>2030</td>
<td>0.091</td>
<td>0.09</td>
<td>0.098</td>
</tr>
<tr>
<td>2040</td>
<td>0.123</td>
<td>0.123</td>
<td>0.133</td>
</tr>
<tr>
<td>2050</td>
<td>0.101</td>
<td>0.096</td>
<td>0.1</td>
</tr>
<tr>
<td>2060</td>
<td>0.074</td>
<td>0.073</td>
<td>0.081</td>
</tr>
<tr>
<td>2070</td>
<td>0.046</td>
<td>0.046</td>
<td>0.049</td>
</tr>
<tr>
<td>2080</td>
<td>0.091</td>
<td>0.087</td>
<td>0.077</td>
</tr>
<tr>
<td>2090</td>
<td>0.135</td>
<td>0.133</td>
<td>0.13</td>
</tr>
</tbody>
</table>

Table 6.22 Comparison of Mean Squared Errors(MSE) among the Three Methods (out-of-sample)

<table>
<thead>
<tr>
<th>SIC</th>
<th>LPM</th>
<th>LOGIT</th>
<th>NN</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000</td>
<td>0.217</td>
<td>0.215</td>
<td>0.236</td>
</tr>
<tr>
<td>2010</td>
<td>0.137</td>
<td>0.137</td>
<td>0.144</td>
</tr>
<tr>
<td>2020</td>
<td>0.055</td>
<td>0.055</td>
<td>0.058</td>
</tr>
<tr>
<td>2030</td>
<td>0.088</td>
<td>0.088</td>
<td>0.096</td>
</tr>
<tr>
<td>2040</td>
<td>0.127</td>
<td>0.127</td>
<td>0.134</td>
</tr>
<tr>
<td>2050</td>
<td>0.099</td>
<td>0.094</td>
<td>0.097</td>
</tr>
<tr>
<td>2060</td>
<td>0.051</td>
<td>0.051</td>
<td>0.056</td>
</tr>
<tr>
<td>2070</td>
<td>0.038</td>
<td>0.039</td>
<td>0.041</td>
</tr>
<tr>
<td>2080</td>
<td>0.105</td>
<td>0.1</td>
<td>0.123</td>
</tr>
<tr>
<td>2090</td>
<td>0.139</td>
<td>0.138</td>
<td>0.141</td>
</tr>
</tbody>
</table>
Table 6.23 Comparison of Mean Absolute Errors (MABSE) among the Three Methods (in-sample)

<table>
<thead>
<tr>
<th>SIC</th>
<th>LPM</th>
<th>LOGIT</th>
<th>NN</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000</td>
<td>0.424</td>
<td>0.421</td>
<td>0.397</td>
</tr>
<tr>
<td>2010</td>
<td>0.295</td>
<td>0.295</td>
<td>0.306</td>
</tr>
<tr>
<td>2020</td>
<td>0.116</td>
<td>0.11</td>
<td>0.094</td>
</tr>
<tr>
<td>2030</td>
<td>0.186</td>
<td>0.18</td>
<td>0.18</td>
</tr>
<tr>
<td>2040</td>
<td>0.247</td>
<td>0.245</td>
<td>0.303</td>
</tr>
<tr>
<td>2050</td>
<td>0.213</td>
<td>0.193</td>
<td>0.204</td>
</tr>
<tr>
<td>2060</td>
<td>0.153</td>
<td>0.146</td>
<td>0.162</td>
</tr>
<tr>
<td>2070</td>
<td>0.092</td>
<td>0.091</td>
<td>0.049</td>
</tr>
<tr>
<td>2080</td>
<td>0.192</td>
<td>0.175</td>
<td>0.137</td>
</tr>
<tr>
<td>2090</td>
<td>0.272</td>
<td>0.266</td>
<td>0.256</td>
</tr>
</tbody>
</table>

Table 6.24 Comparison of Mean Absolute Errors (MABSE) among the Three Methods (out-of-sample)

<table>
<thead>
<tr>
<th>SIC</th>
<th>LPM</th>
<th>LOGIT</th>
<th>NN</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000</td>
<td>0.429</td>
<td>0.425</td>
<td>0.426</td>
</tr>
<tr>
<td>2010</td>
<td>0.282</td>
<td>0.282</td>
<td>0.301</td>
</tr>
<tr>
<td>2020</td>
<td>0.113</td>
<td>0.107</td>
<td>0.119</td>
</tr>
<tr>
<td>2030</td>
<td>0.181</td>
<td>0.176</td>
<td>0.165</td>
</tr>
<tr>
<td>2040</td>
<td>0.25</td>
<td>0.248</td>
<td>0.303</td>
</tr>
<tr>
<td>2050</td>
<td>0.207</td>
<td>0.187</td>
<td>0.184</td>
</tr>
<tr>
<td>2060</td>
<td>0.128</td>
<td>0.122</td>
<td>0.135</td>
</tr>
<tr>
<td>2070</td>
<td>0.084</td>
<td>0.084</td>
<td>0.041</td>
</tr>
<tr>
<td>2080</td>
<td>0.203</td>
<td>0.186</td>
<td>0.202</td>
</tr>
<tr>
<td>2090</td>
<td>0.275</td>
<td>0.271</td>
<td>0.249</td>
</tr>
</tbody>
</table>
References


Vita

Author’s Name – Jianhua Chen

Birthplace – Hunan, People’s Republic of China

Birth Date – November 21, 1965

Education

Master of Science in Systems Engineering
Chinese Academy of Agricultural Sciences
July – 1990

Bachelor of Science in Political Science
Beijing Science and Technology University
September – 1987

Work Experience

Tentoe Inc.
Pleasanton, CA
11/00 – present
Senior Engineer of Data Analysis and Management

University of Kentucky
Lexington, KY
09/96 – present
Research Assistant

Chinese Academy of Agricultural Sciences
Beijing, China
08/90 – 08/96
Deputy Chief of the Department of Technological Economics, Assistant Professor

Selected Publications


Shun-Jia-Wa Village.” Technical Progress and Agricultural Development.


