TESTING A NEURAL NETWORK'S ABILITY TO PREDICT FUTURES PRICES

By

LONNIE HAMM

Bachelor of Science

Oklahoma State University

Stillwater, Oklahoma

1991

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
Master of Science
July, 1995

TESTING A NEURAL NETWORKS ABILITY

TO PREDICT FUTURES PRICES

Thesis Approved:

Wade Brorsen

Thesis Advisor

Stephen R. Koontz

Jim Krehbiel

Thomas C. Collins

Dean of the Graduate College

ACKNOWLEDGEMENTS

I wish to express my sincere gratitude to Dr. B. Wade Brorsen, Chairman of my graduate committee, for his guidance in conducting this research. This research would have been considerably more difficult without his constructive guidance, encouragement, and friendship. I would also like to thank Dr. Stephen Koontz and Dr. Tim Krehbiel for their helpful comments and numerous suggestions for further research. A special thanks also goes to Dr. Ramesh Sharda who was instrumental in helping me to gain the knowledge I have about neural networks. Finally, I would like to thank the Department of Agricultural Economics for their financial support during my graduate studies.

TABLE OF CONTENTS

Chapter Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

CHAPTER 1

INTRODUCTION

Much interest has been generated by a new class of computer programs called neural networks. Neural networks have been successfully applied in fields as diverse as task coordination, optical pattern recognition and sonar signal processing. A recent article in *U.S. News and World Report* states that most financial organizations are experimenting with neural networks, but only a few have actually put them to use (Egan). Much of the excitement surrounding neural networks is due to their unique ability to handle nonlinear data. Neural networks are universal approximators capable of approximating any nonlinear function (White 1989). This means the functional form of the model need not be made explicit.

Many people believe that neural networks hold great promise for predicting futures prices. However, the predictability of futures prices has been a source of controversy within the academic community for many years. It has been the contention of many academics that the markets are efficient in the sense that the current price reflects all information that can be known. In direct contradiction to this view, many traders have indicated they use technical analysis to aid in their predictions of future prices. Brorsen and Irwin report that 80 percent of commodity investment pools use computerized technical trading systems. Two highly regarded books by Schwager (1989,1992) give the trading testimonials of a large number of traders who have used technical analysis to produce significant trading profits over a

long period of time.

Market efficiency is not merely a trivial or academic matter. Stocks, commodities, currencies, and other financial instrument are traded for various reasons: reduction of business risks, purchases and sales of raw materials, and the investment of personal or corporate wealth. Thus the efficiency of the economy depends in part on the efficiency of the markets which operate in the economy. As Fama argued (pg. 383):

> The primary role of the capital market is allocation of ownership of the economy's capital stock. In general terms, the ideal is a market in which prices provide accurate signals for research allocation: that is, a market in which firms can make production-investment decisions, ...

Agricultural economists have long been interested in the efficiency of the futures markets. If producers and other market participants use futures prices in their production and marketing decisions and futures prices give erroneous signals about future spot prices then a misallocation of resources may occur (Stein).

Futures markets offer an opportunity to speculate on the price changes of commodities and various financial instruments. In recent years money invested in managed futures has increased dramatically. The managed money segment of the futures industry has grown to an estimated $20-billion plus business (Futures). If the current futures price reflects all available information, as some market efficiency theories stipulate, then it would be impossible for speculators to extract any profits from this market.

Technical Trading Systems and Market Efficiency

In light of the growing evidence that there is some predictability in futures prices, many in academia have modified their views on the usefulness of technical

may sluggishly adjust to new information. Profitable trading systems would then be possible because of the price trends that exist as a result of the adjustment process caused by information shocks (Beja and Goldman; Nawrocki). If market participants are not heterogeneously informed of new or difficult to obtain information, then prices tend to be an imperfect aggregator of information. Using a two period noisy rational expectations model, Brown and Jennings claim that because of this imperfect aggregation, the current price is not a sufficient statistic for private information possessed by market participants. As a result, historical prices add information that is not available with the current price alone. In other words, technical analysis provides additional information to market participants forming an expectation of future prices.

Researchers have begun to argue that asset prices exhibit nonlinear dynamics (LeBaron). The particular type of nonlinear structure which asset prices may possess has been an area of debate. Much empirical research has been devoted to investigating the possibility of chaos in economic systems. A chaotic system exhibits complex dynamical properties and has limited forecastabiltiy. The property of limited forecastabilty has particular relevance to the study of market efficiency. However, empirical research generally shows little support for chaos but strong evidence for nonlinear dependence (Blank; Brock, Hsieh, and Lebaron; De Grauwe Dewachter, Embrechts; Decoster, Labys, and Mitchell; Frank and Stengos; Hinich and Patterson; Hsieh; Mayfield and Mizrach; Peters; Scheinkman and LeBaron).

Neftci argued that if price dynamics are nonlinear, technical analysis may be capturing information contained in higher-order moments of asset prices. This information would not be captured by traditional linear models. Mechanical trading rules based on technical analysis have some advantage over traditional statistical tests

in detecting nonlinear dependence. Mechanical trading rules have been able to detect nonlinear dependence and can be used to test the quality of the dependence (Nawrocki). Logue and Sweeney used a mechanical trading rule to detect dependence in foreign exchange rates whereas spectral analysis detected no dependence. Brock, Lakonishok, and LeBaron examined the stock returns generated from some technical trading rules. Their results suggest that the return-generating process of stocks is more complicated than the returns suggested by linear models.

Neural Networks and Market Efficiency

A new class of computer programs, called neural networks, is particularly capable of handling nonlinear data. Neural networks are capable of approximating any nonlinear function (White 1989). The nonlinear forecasting ability of neural networks has been demonstrated in a wide variety of applications. Among them was Lapedes and Farber who demonstrated that neural networks are capable of decoding deterministic chaos. Neural networks have been successfully applied in fields as diverse as task coordination, optical pattern recognition and sonar signal processing.

Oldfield, Rogalski, and Jarrow claim that the arrival of information is best described by a sporadic jump process. When information arrives in this way and there are price adjustment delays, i.e. the market is in disequilibrium, then any mechanical trading rule should by adaptive in response to changing information and dependence levels (Nawrocki). Neural networks may be one of the best tools currently available to extract complicated and changing dependencies in the price structure of a time series of futures prices. Consequently, many people feel that neural networks hold great promise for predicting financial variables.

Testimonials can be unreliable, nonetheless, specific examples of firms applying neural networks to trading are being reported. Gerber Baby Foods is using neural advice to trade cattle futures (Computing Canada). Neural networks are also being used to trade the S&P index (Business Week, November 2, 1992). Shearson Lehman is reported to be using neural networks to predict the performance of stocks and bonds (Business Week, March 2, 1992). Nuwave Investment Corporation uses a neural network as their primary tool to forecast price changes in a variety of futures prices.

Some documented examples of applying neural networks to building commodity trading systems have been published. Some of the earliest studies, including those of Collard (1991, 1992) and Bergerson and Wunsch showed promising results. All of these studies, however, were lacking in both the short sample period as well as absence of any statistical tests on the profits. More recently, academic studies have begun to use neural network based trading systems to answer questions of market efficiency. Tsibouris was unable to reject the null hypothesis of a random walk in exchange rate returns by using a neural network based prediction model. Studies by Grudnitski and Osburn and Lee and Huh have shown better results. Lee and Huh forecasted the futures prices of treasury bonds, the S&P 500, and Oil and rejected the random walk. Grudnitski and Osborn produced trading returns of 17.04% and 16.36% trading the S&P 500 and Gold futures respectively.

There are some potential advantages in using a neural network to test market efficiency and investigate for potential nonlinearities. Traditional tests may not be able detect nonlinearities as a more direct method such as a neural network trading model (LeBaron). Market efficiency theories are realistically mitigated by bounded

rationality arguments. Such arguments hold that humans are inherently limited in their ability to process information, so that efficiency can hold only to the limits of human information processing (White 1988). Neural Networks are a relatively new technology which theoretically could aid market participants in processing information.

The popular press is filled with claims about the usefulness of neural networks to commodities trading, but little academic research is available to support these claims. This research will investigate usefulness of a neural network for trading a previously neglected futures series, Kansas City Wheat, as well as a more commonly analyzed futures price series, the deutsche mark currency contract.

Objectives

General Objective:

Determine the applicability of neural networks to trading futures.

Specific Objective:

Determine the profitability of a trading system based on a feedforward neural network with one hidden layer and inputs consisting of eight lags of the continuous weekly returns, traded on the hard red winter wheat and deutsche mark futures contract.

Procedure

The objectives given above are addressed by developing neural network trading models to simulate trading of the hard red winter wheat and deutsche mark futures contract. The neural network trading models produce weekly trading signals. The trading signals associated with a particular week are executed on the opening of the first trading day of the week because the trading signals are based upon

information up to and including the last trading day of the previous week. The positions are then held for the duration of the trading week and reversed on the opening of the first trading day of the following week if the network signals so indicate. Therefore the models are in the market at all times.

The dependent variable in both the wheat and deutsche mark trading model is a binary variable which represents whether a long or short position should be take for the following week. The independent variables of the deutsche mark trading model consist of eight lags of the weekly returns. The weekly returns are the changes in the logs of the closing prices on the last trading day of the weeks. The independent variables of the wheat models are defined analogously except that quarterly dummy variables are added to account for any seasonality.

Out-of-sample trading simulations are performed for the time period from 1985-1992. The trading simulations for each commodity are performed by eight neural network trading models, each of which corresponds to a specific time frame of trading. The time frame of trading for each of the models is one year long and begins on the first trading day of the year and ends on the last trading day of the year. The trading models are chosen a priori to have one hidden layer. The number of hidden neurons for the networks which provide out-of-sample trading results are chosen by examining the out-of-sample trading performance of various network configurations in previous testing periods. The network configuration, i.e. 4, 6, 8, or 10 hidden neurons, for a particular testing period is chosen based upon which configuration produced the highest average out-of-sample net trading profits in the four previous testing periods. In order to provide true out-of-sample trading simulations on the first year of the eight years of simulations, it is necessary to

analyze the performance of networks on 4 testing periods prior to this first year. Thus the trading results of 12 years are analyzed, but only 8 are true out-of-sample results.

Both gross and net trading returns are calculated for the trading simulations. Net returns are calculated by taking into account transaction costs which are composed of both commission costs and skid error or bid-ask spread. Commission costs are $35 and skid costs are $30 giving a round-turn transaction cost of $65. The null hypotheses that net and/or gross trading returns are less than zero are testing using a bootstrapping type of methodology. The bootstrapping type of methodology was first applied to the analysis of trading returns by Brock, Lackonishock, and LeBaron. Others using this methodology were Allen and Karjalainen, and Levich.

In this research, the objective of the bootstrapping procedure is to simulate the distribution of the gross and net trading returns under two separate null models. The null models are that the data follows a random walk with drift or a GARCH(1, 1) process. The data in this study are the weekly returns as described above. For the random walk with drift null model, 500 simulated sets of weekly returns are constructed by "scrambling" the actual weekly returns from each testing period. The process of "scrambling" refers to resampling with replacement. For the GARCH(1, 1) null hypothesis a GARCH(1, 1) model is fit to the actual weekly returns. The residuals are then standardized by the estimated standard deviation for each observation. These standardized residuals are then "scrambled". The estimated standard deviations are then multiplied by the scrambled residuals and added to the predictions from the GARCH(1,1) to form a new data set. The standardized residuals for each testing period are "scrambled" 500 times to generate 500 data sets. The 500

simulated data sets from both null models are used to construct 500 testing sets for each null model and each testing period. The neural network trading models which are chosen to generate true out-of-sample testing results are tested on each of the 500 testing sets corresponding to the null models. The simulated distribution distributions are composed of the trading profits obtained by this procedure. Simulated p-values for a given net or gross trading profit figure can then be calculated by computing the percentage of items in the simulated distribution which exceed the level being tested.

Organization of Thesis

Chapter 2 discusses the use of technical analysis and reviews the efficient market hypothesis and evidence against it. The evidence for nonlinearity of commodity prices is discussed as well as its implications for market efficiency and tests thereof. Noisy rational expectations and disequilibrium theory are presented as alternatives to the traditional theories of market efficiency. The relationship of these alternative theories to technical analysis is also discussed.

Chapter three discusses the theory of neural networks. The history and develop of neural networks is briefly discussed and some applications of neural networks are presented. The details of the feedforward type of neural network are presented in detail. The chapter concludes by presenting the specific method used in this research to estimate the parameters of the neural network.

Chapter 4 presents the details of the methods used to accomplish the research objectives. The first section discusses the data and procedures used to develop the neural network trading models. Next, the trading rules and assumptions are presented. In the last section, the procedures used to test the statistical significance of

the trading returns are presented.

Chapter 5 presents and discusses the results of the study. The results of the estimation of the parameters of the neural network trading models is presented. The trading profits of the trading models for all 12 testing periods are provided. The trading profits of the 8 out-of-sample trading simulations are presented. Next, the statistical analysis of these profits is presented. The results of these statistical tests are discussed in the context of the research objectives.

The last chapter summarizes the study's results and conclusions. General conclusions on the applicability of neural networks in trading futures is presented. Conclusions on market efficiency in the context of the specific models tested in this research are presented.

CHAPTER 2

TECHNICAL ANALYSIS AND MARKET EFFICIENCY

Much theory on asset price determination has been developed which seeks to explain market behavior or how some specific attribute or function of the market operates. However, according to some of the theory, the techniques that some traders use to make trading decisions should be of little or no value in making money. More specifically, the efficient market hypothesis implies that statistically significant profits can not be made from trading techniques that only use past prices as information, i.e. technical analysis. This chapter discusses the use of technical analysis and reviews the efficient market hypothesis and evidence against it. Alternative theories of market efficiency and their relation to technical analysis are also presented.

Technical Analysis

Technical analysis is an integral part of the trading decisions of many speculators in futures markets. Irwin and Brorsen reported that over 80 percent of public futures funds had trading advisors which relied entirely on technical analysis. The Chicago Board of Trade in a 1983 survey found that 50% of all speculators consulted charting services. Taylor and Allen reported that 90% of the chief foreign exchange dealers in London place some weight on Technical analysis in forming their expectations. Two highly regarded books by Schwager (1989, 1992) give the trading testimonials of a large number of traders who have used technical analysis to produce

significant trading profits over a long period of time. Admittedly, trading testimonials may be unreliable (see Edwards and Ma; Elton, Gruber, and Rentzler; Irwin), however, it is widely accepted that technical analysis plays a role in the trading decisions of many traders.

In spite of the apparent widespread use of technical analysis, the academic community has been less than receptive to its concepts. For example, Malkiel (pg. 132) stated:

> Obviously, I am biased against the chartist. This is not only a personal predilection, but a professional one as well. Technical analysis is anathema to the academic world. We love to pick on it. Our bullying tactics are prompted by the two considerations: (1) the method is patently false; and (2) it's easy to pick on. And while it may seem a bit unfair to pick on such a sorry target, just remember: it is your money we are trying to save.

Malkiel's use of the term chartist was meant to refer to those traders who look at charts as well as those who make trading decisions based on mathematical measures, i.e. moving averages, relative strength indexes, etc. Indeed, because of the proliferation of the personal computer, the 1980's witnessed the increasingly widespread use of mathematical measures in making trading decisions. The results to be obtained in this thesis are more applicable to this type of trading as opposed to "charting".

Traditionally, academics have pointed to the weak form of the efficient market hypothesis, proposed by Fama, which says that prices, both past and present, fully reflect all available information. There is an obvious contradiction between the

widespread use of technical analysis, and some economic theories which say that there is no economic justification for its use.

Efficient Market Hypothesis

Fama defined an efficient market as one in which asset prices fully reflect available information. Following the notation of Fortune, let s be defined to be a set of "circumstances" which describes the state-of-the-world. The set s would consist of variables such as interest rates, dividends, and inflation. Suppose also that there are N such possible states of the world, i.e. s=1, 2, ..., N, and that Ω_t represents the information set available at time t. If we let $\pi(s|\Omega_t)$ be the probability that state s will occur and we calculate the value of the asset for each state of nature s, $P^*(s)$, then the expected value is

(1) $$E(P_t^*|\Omega_t) = \sum_s P^*(s)\pi(s|\Omega_t).$$

The efficient market hypothesis states that

(2) $$P_t = E(P_t^*|\Omega_t)$$

where P_t is the current price of the asset. Three different forms of the efficient market hypothesis are associated with three different levels of information:

a. The weak form in which the information set Ω_t is just historical prices.
b. The semi-strong form in which Ω_t is made up of all publicly available information at time t.
c. The strong form in which Ω_t is taken to be all information, both public and private, at time t.

Thus there are three testable forms of the efficient market hypothesis. Note that the

information available in the weak form is a subset of the information available in the semi-strong form, and the information available in the semi-strong form is a subset of the information available in the strong form. Technical trading systems use past prices and thus are only a test of the weak form of the efficient market hypothesis. Thus, failing to reject the weak form according to a technical trading system test does not imply that the market is inefficient according to the semi-strong and strong forms of the efficient market hypothesis. However, if we conclude that the market is weak-form inefficient, then we must conclude that the market is also inefficient according to the semi-strong and strong forms of the efficient market hypothesis.

One method of empirically testing the efficient market hypothesis is with the random walk model. The random walk model states that successive price changes are independent and identically distributed. A second way to test market efficiency is by simulating a trading system and comparing the trading profits obtained to the profits that are theoretically possible under the efficient market hypothesis. These two types of tests are discussed in the following two sections

Tests of the Random Walk Model

The efficient market hypothesis has implications for the sequence of prices over time. If we are in time t and want to forecast the price for time $t+1$, then we can only utilize information available at time t to make our forecast. Thus the best prediction of P_{t+1} is $E(P^*_{t+1}|\Omega_t)$ where Ω_t is the information set available at time t.

Any new information that arrives between time t and t+1 is random and thus its effect on P_t creates a random deviation from $E(P^*_{t+1}|\Omega_t)$.

From equation (2), the efficient market hypothesis implies that

(3) $$P_{t+1} = (1 + r)P_t + \varepsilon_{t+1}$$

where $E(\varepsilon_{t+1}) = 0$, $\varepsilon_{t+1} \sim i.i.d.$, and r is the expected rate of return on the asset under consideration. Thus a sequence of prices will be a random walk with drift. Therefore, the random walk test of market efficiency involves searching for dependence in the residuals of (3) or in the prices. The random walk model is sometimes stated as

(4) $$f(r_{t+1}|\Omega_t) = f(r_{t+1}), \qquad t \in \mathbf{I}$$

where f is a density function and r_t is the one-period percentage return $(P_{t+1} - P_t)/(P_t)$. Equation (4) states that the conditional and marginal probability distributions are equal. The random walk test is a test of the weak form of the efficient market hypothesis because the information set is composed of past prices. Fama admitted that the random walk model is a stringent restriction and that market efficiency does imply that prices follow a random walk, or in other words are identically independently distributed. However, a random walk does imply market efficiency.

The earliest tests of the random walk hypothesis tended to favor prices following a random walk. Several commodity futures prices were shown to resemble a random walk (Working). Kendall made similar conclusions about wheat prices, cotton prices, and share indices. Fama(1965) concluded that the 30 stocks comprising

the Dow Jones Industrial Average followed something similar to a random walk. Fama admitted there was some serial correlation but concluded that it was insignificant from an economic viewpoint. Fama's research had a significant impact on the academic community and many researchers thereafter assumed that security and commodity prices follow a random walk.

Some later research on commodity futures as well as the underlying commodities concluded that the random walk model is not a good description of commodity price behavior. Leuthold used spectral analysis to study the live cattle futures market and found non-randomness. Cargill and Rausser tested for serial correlation in the cash prices of corn, oats, wheat, soybeans, copper, live cattle, and pork bellies. Cargill and Rausser's results did not support the random walk model. Taylor (1985) found evidence of price trends in several futures contracts. Price trends would not exist if prices followed a random walk.

However, researchers have not consistently rejected the random walk (Mussa, Meese, and Rogoff). Cornell, and Mussa suggest that exchange rates appear to follow a random walk. Mussa (1982) argues that this stochasticity supports the ideas of rational expectations and market efficiency. Brock, Hsieh, and LeBaron analyzed five major currencies and found little evidence of serial correlation. They also failed to detect any linear dependence using the runs test. These results are consistent with the findings of many others (e.g. Giddy and Dufey; Burt, Kaen and Booth; Cornell; Logue and Sweeney; Logue, Sweeney and Willett; Rogalski and Vinso). This led

Brock, Hsieh and LeBaron to say that there has been no strong statistical evidence confirming or refuting the random walk hypothesis.

The mixed results could be traced back to the methods used to test the random walk hypothesis. The traditional methods to test the random walk hypothesis are autocorrelation, spectral analysis, and runs tests. Each of these methods has been criticized as a test of independence (Taylor 1985). In particular, autocorrelation tests are inappropriate if the stochastic process X generating observed returns x_t is linear (Taylor 1986). The reason for this is that the standard errors of the coefficients need not be $1/\sqrt{n}$ as they are if the X is linear. Taylor (1986) showed that the standard errors frequently exceed $3/\sqrt{n}$. The autocorrelation results of Brock, Hsieh and Lebaron addressed this problem by using heteroskedasticity-consistent standard errors. They stated that at best, exchange rate changes are linearly independent. More recently, there is evidence that stock and futures prices are nonlinearly dependent.

New statistical tools have recently been developed to test for the existence of potentially forecastable structure, nonstationarity, or hidden patterns. The BDS test developed by Brock, Dechert, and Scheinkman has been one of the most popular tests. Other tests include the Keenan, Tsay, Ramsey RESET, White dynamic information matrix (White 1987), and the McLeod-Li tests.

The BDS tests the null hypothesis that a series is identically independently distributed. The alternative includes not only nonlinear dependence but linear

dependence. However, the linear dependence can be ruled out if the data are first transformed by removing any possible linear dependence. Brock, Hsieh and Lebaron looked at the five major exchange rates and found evidence of nonlinearity using the BDS statistic. Brock, Hsieh and Lebaron along with Scheinkman and Lebaron found evidence that stock returns follow a nonlinear dynamic system. Frank and Stengos found similar evidence for the silver and gold markets.

Nonlinear dependence can arise in two ways. Nonlinearities can arise because of nonlinear dependence in the return series themselves referred to as mean nonlinearity. Nonlinearities can also arise because of dependence in the variance of returns. Brock, Hsieh, and LeBaron claim that these two types nonlinearity encompass all nonlinear stochastic models discussed in the time series literature.

Changes in the level of trading activity and/or the arrival of information will most certainly cause changes in the return series as market participants adjust to the changing trading environment. This changing trading environment can also lead to heteroskedasticity of daily returns. Variances can be non-stationary or conditional upon past observations and other variables. The most obvious question is how does heteroskedasticity in returns make a return series nonlinear? Changes in variance or conditional variance will cause more autocorrelation between x_t^2 and $x_{t+\tau}^2$ than there is between x_t and $x_{t+\tau}$, where x_t is the return for day t. This is the nonlinear characteristic of a return series as described by Taylor (1986).

Mandelbrot in 1963 was the first to suggest that the variance of stock returns

was not constant over time. Mandelbrot thought that stock returns were uncorrelated, but noticed that large changes tended to be followed by large changes, and similarly for small changes. This characteristic was observed in other economic series and eventually led to Engle's development of the ARCH model and Bollerslev's GARCH model. In the ARCH model,variance is function of past errors. The GARCH process generalizes the ARCH process by modeling the variance as a function of lagged values of itself as well as past errors. GARCH models have been popular with researchers investigating the dynamics of returns because GARCH models can explain the unconditional leptokurtic distributions.

If price changes are independent, then the law of large numbers implies that the distribution of price changes should be normal. A great deal of evidence exists which suggests that the distribution of price changes for futures prices is leptokurtic. A leptokurtic distribution has more observations around the mean and fatter tails than a normal distribution. It has long been noted that stock market returns have more observations in the tail than predicted by the normal distribution (Osborne, 1964; Fama, 1965). Mandelbrot (1964) suggested that stock returns may belong to a family of "Stable Paretian" distributions.

The stock market crash of October 19, 1987 brought more attention and interest to the issue of the distribution of security prices as well as other financial instruments. The most recent studies of the distribution of returns has only confirmed the observations of earlier researchers. Turner and Weigel (1990) found that returns

for the S&P 500 were consistent with a leptokurtic distribution. Similar results for foreign exchange rates have been found by Hsieh (1988) and Friedman and Vandersteel (1982). Many researchers have found that futures prices are also leptokurtic (Hudson, Leuthold, and Sarassoro; Cornew, Town, and Crowson; Gordon; Hall, Brorsen, and Irwin).

Two competing explanations exist which would explain the heavy tails of the distribution of returns: the data are independently drawn from a leptokurtic distribution which remains fixed over time, and the data are drawn from a distribution which varies over time. Hsieh (1986) analyzed foreign exchange rates and found evidence in favor of changing distributions as the explanation for leptokurtic distributions. Furthermore, there was evidence that changing means and variances was the cause of the changing distributions. Brock, Hsieh, and LeBaron used a test involving the third order moments to distinguish between mean and variance nonlinearity in exchange rate changes. They found evidence indicating that changing variances are responsible for the nonlinearities in exchanges rates. These results are similar to what Yang and Brorsen found for 7 out of 15 commodities tested. However, Brock, Hsieh, and Lebaron also found evidence that the nonlinearities may be more complicated.

From the preceding discussions, we would conclude that evidence exists which indicates that returns for futures prices, as well as other financial instruments, are nonlinear. The implications of this finding for market efficiency and technical

analysis are numerous. The efficient market hypothesis implies that investors react to information as it is received, or in other words, in a linear fashion. Nonlinear and leptokurtic returns imply that investors react in a cumulative fashion. The next section puts forth some reasons why technical analysis may be useful, and the last section presents some economic theory which says that technical analysis may be useful.

Tests of Market Efficiency Using Simulated Trading Returns

Ample evidence suggests that daily returns do not follow a random walk. However, a random walk implies market efficiency but market efficiency does not imply a random walk, thus we can draw no conclusion about market efficiency. The random walk model assumes perfect competition. Perfect competition assumes (1) zero transaction costs, (2) all traders are risk neutral, (3) information is transmitted to all traders instantaneously, (4) all traders agree about the influence of new information on current prices, and (5) the cost of information is zero. Thus a test of the random walk model is simultaneously testing the efficient market hypothesis and the extent to which market behavior conforms to the ideal of perfect competition.

Trading system tests of market efficiency can provide a stronger test of market efficiency because some of the assumptions such as zero transaction costs can be eliminated. The Jensen test of the efficient market hypothesis states that a market is efficient with respect to information set Ω_t if it is impossible to earn economic trading profits between time's t and $t+\tau$ by using the information in Ω_t.

The information sets Ω_t usually considered are those sets associated with the weak, semi-strong, and strong forms of the efficient market hypothesis as discussed in the first section of this chapter.

Economic profits are assumed to be net of all costs and adjusted for risk. Accounting for transaction costs is relatively straight forward, however, adjusting for risk is less obvious. Risk adjusted returns for stock investments can be analyzed with capital asset pricing model (CAPM) (Sharpe). The CAPM says that the risk premium for accepting a share's undiversifiable risk is proportional to the covariance between share return and the return on the market portfolio. The market portfolio in stock market analysis is a portfolio made up of all shares or some market index such as the S&P 500 made up of a large number of shares. Some researchers have concluded that the CAPM satisfactorily describes the relationship between risk and return (Black, Jensen, and Scholes; Miller and Scholes).

To apply the CAPM to futures markets, an appropriate market portfolio must be found. The S&P 500 and various weighted averages of the S&P 500 and the Dow Jones Commodity Cash Index have been used as the market portfolio in the context of futures markets (Dusak; Carter, Rausser and Schmitz). However, each of the market portfolios proposed for use with futures markets has been criticized for various reasons. To date, no resolution of this matter has been found

It has been argued that tests of market efficiency using trading rules is desirable (Friend and Westerfield, Logue and Sweeney). Mechanical trading rules

can detect nonlinear dependence (Alexander; Cheng and Deets). Mechanical trading rules have detected nonlinear dependence in foreign exchange rates while spectral analysis failed to detect dependence (Friend and Westerfield). The economic quality of the dependence can also be investigated with mechanical trading rules whereas statistical tests can only test for the existence of the information (Friend and Westerfield).

Trading rules have been used to analyze market efficiency in a variety of markets. Some of the earliest studies concerning futures markets include Leuthold and Peterson who found that statistically significant returns were attainable in live hogs from 1973 to 1977. Stevenson and Bear tested corn and soybean futures prices from 1951 through 1968 and reported impressive dollar profits, however, due to inadequate methodology, the statistical significance of the profits can not be interpreted easily.

More contemporary studies include Irwin and Brorsen (1984) who tested weak form efficiency by testing trading rules on a portfolio of fourteen commodities from 1963 through 1983. Lukac and Brorsen simulated trading of 23 technical trading systems on 30 different markets and found that 22 of the systems produced statistically significant returns. Boyd and Brorsen tested five technical trading systems on seven commodities and found that significant annual net returns were possible. Irwin, Krukemyer, and Zulaf found that returns from public commodity pools were sufficient to cover the costs and risks involved in futures trading. In

addition, excess net returns significantly greater than zero were found for institutional commodity pools.

Recently, there has seen a great deal of interest in applying neural networks to commodities trading. In two papers, Collard reported promising results using a neural network as a commodity trading model. A study by Bergerson and Wunsch used the hindsight of a market technician (chartist) to train a neural network when to buy and sell a commodity. They then used the network along with some traditional money management techniques and obtained good results. Grudnitski and Osburn used a neural network to attain annual returns of 16 percent per year for Gold and 17 percent per year for the S&P 500. Trippi and DeSieno found that a neural network based trading strategy outperformed a buy and hold strategy for the S&P 500. Claussen and Uhrig used a neural network to predict the directional movements in cash soybean prices and obtained from 86 to 97% accuracy with trading horizons ranging from 5 to 30 days. Other studies showing promising results include Kimoto for stocks and Refenes for exchange rates.

The last two sections have shown the inadequacy of the weak form of the efficient market hypothesis to explain securities and futures prices. In response to the evidence, academics are beginning to change their view that technical analysis is of no use to market participants. Alternative theories of market behavior have been put forth in an effort to improve on the efficient market hypothesis and to explain the usefulness of technical analysis.

Noisy Rational Expectations and Disequilibrium Theory

Noisy rational expectations was first proposed by Robert Lucas. Information flows between traders have been studied in the context of the noisy rational expectations model by Grossman (1976, 1978), Kihlstrom and Mirman, and Green. Noisy rational expectations removes the assumption that all transactions are made with complete information. Each trader has their own expectation of the future price of some asset based on that traders available information. More precisely, the *i*th trader observes y_i, where $y_i = P_1 + \varepsilon_i$ and P_1 is the price of some asset one period in the future. The noise term, ε_i prevents any trader from learning the true value of P_1. Thus the current equilibrium price P_0, is a function of $(y_1, y_2, \ldots, y_n)$ where n is the number of agents in the market and can be written as $P_0(y_1, y_2, \ldots, y_n)$.

Much research on market efficiency focuses on the degree to which $P_0(y)$ aggregates the information of the diversely informed traders. The concern is to what degree is P_0 a sufficient statistic for P_1. In the model proposed by Grossman (1976), there exists a $P_0^*(y)$ which is so efficient in aggregating information that there is no incentive for individuals to collect information. The end result is that equilibrium is not possible if information is costly. Thus only an imperfect information equilibrium can be an equilibrium in an economy in which information is costly.

One of the major criticisms of the efficient market hypothesis is that it assumes that information is costless. Fama's definition of the efficient market hypothesis says

that costless information is a sufficient condition for prices to fully reflect all available information. Grossman and Stiglitz argued that costless information is also a necessary condition. If information is costly then those agents who arbitrage must receive a return on their costly activity or arbitrage will cease. Grossman and Stiglitz questioned whether a competitive equilibrium could exist whereby arbitrage profits are eliminated.

Grossman and Stiglitz proposed a model in which there is an equilibrium degree of disequilibrium. The model describes a market in which prices only partially reflect the information of informed individuals so that agents who expend resources to obtain information are compensated for their efforts. Grossman and Stiglitz's model differs from that of Grossman (1976) in that in Grossman and Stiglitz's model there are two distinct sets of traders, informed and uninformed. The informed traders observe the future price of an asset with some degree of uncertainty and thus the current price is an imperfect aggregator of the available information. Uninformed traders, however, do infer some amount of information from the current price. Diamond and Verrecchia analyzed the degree to which the informed traders infer information from the current price that is not redundant with respect to the private information they already possess. Grossman and Stiglitz's model only allows for private information concerning one piece of information. Diamond and Verrecchia's model allows for more diverse information. In this framework, they conclude that informed market participants are able to infer information from prices

that is not redundant with respect to the private information they possess.

The models proposed by Grossman, Grossman and Stiglitz, and Diamond and Verrecchia were based upon a two-period economy. These models, in contradiction to the efficient market hypothesis, say that the current price does not reveal all information. However the models do not say anything about the usefulness of past prices in forming expectations about future prices. Hellwig, Singleton, and Grundy and McNichols developed models in which past prices are useful. In Hellwig's model, investors use the most recent price as a substitute for the current price which is constrained to be unusable. Singleton studied the time series properties of asset prices across alternative economies and Grundy and McNichols studied the time series properties of information in the form of private signals.

Brown and Jennings extended the models of Diamond and Verrecchia as well as Hellwig. Brown and Jennings analyzed a three-period economy in which the payoff occurs in the last period. In their model, an information set containing only the second period price is dominated by an information set containing a weighted average of the first and second period prices. Historical prices are useful in determining investors demands because the current price does not reveal all information. The first period price is not perturbed by the random variation in the second period supply. Thus past prices are useful in making inferences about private signals. In addition, past prices contain information that is not redundant to the private signals of informed traders. A market that behaves according to Brown and

Jennings model is not weak-form efficient in the sense defined by Fama.

In the preceding discussion, various models, i.e. Grossman, Grossman and Stiglitz, Diamond and Verrecchia, imply that the current price does not reveal all information. This is in direct contradiction to the efficient market hypothesis as proposed by Fama. In fact, an equilibrium price that reveals all information would cause the market to break down (Grossman and Stiglitz). These models led to the development of the model of Brown and Jennings in which not only does the current price not fully reveal all information, but past prices are useful in predicting future prices and investors can use technical analysis to their benefit.

The preceding models propose a price which could be said to be in a equilibrium degree of disequilibrium (Grossman and Stiglitz). The models assume that informed individuals absorb, process and act on their private information immediately. Disequilibrium theory argues that markets do not immediately adjust to information shocks. A market that does not immediately adjust to new information could exhibit trends or dependence. The implications for the usefulness of technical analysis are obvious.

One of the major factors affecting how information is processed is the nature in which the information arrives. Black, Cohen et al., and Beja and Goldman argue that information arrives in the securities market as large random infusions. Fama had originally assumed that information arrives as small random doses while disequilibrium models argue that information arrives spontaneously and is associated

with a jump in price (Oldfield, Rogalski, and Jarrow). Larson, Mann and Heifner, and Stevenson and Bear argued that information arrives in the futures markets as large infusions or shocks. Large amounts of information could take time to process and act upon. Bounded rationality theory argues that humans are inherently limited in their ability to process information (Simon 1955, 1982). In addition the information may contain large amounts of noise which make it hard to determine how the information may effect future prices. Disequilibrium theory argues that the major reason that prices cannot immediately adjust to large and possibly noisy amounts of information is because of market frictions. These market frictions may include the cost of acquiring information, transaction costs, taxes, and noisy information channels (Beja and Goldman).

Nawrocki stated that the degree or characteristics of the disequilibrium may change over time. This changing disequilibrium would have a nonstationary dependence structure. However, Nawrocki classifies this type of disequilibrium as stationary disequilibrium if the market structure, i.e. laws, regulations, etc., is stable. A disequilibrium that shows a stable amount of disequilibriium and dependence structure is referred to as a continuous disequilibrium. Nawrocki used mechanical trading rules to test for the type of disequilibrium that may have been present in 50 different securities. He found evidence that stationary disequilibrium best describes the disequilibrium that may have been present. Consistent with this finding is that adaptive trading rules performed better than nonadaptive rules. These

findings would certainly have implications for the type of technical analysis based trading system that may be profitable. A neural network may be one of the best tools available to build an adaptive trading system.

by Minsky and Papert. It should be noted that Werbos in 1974 developed the mathematical framework for the backpropagation neural network, however, his work went unnoticed at the time.

The first section of this chapter briefly discusses some applications of neural networks and alternative neural network paradigms. The next section presents the feedforward type of neural network in detail. The last chapter discusses and presents some methods for estimating the parameters of feedforward neural network.

Applications and Types of Neural Networks

Neural networks are flexible and have been used to solve many different problems. Some of the applications have been to perform coordination tasks (Selfridge), decode deterministic chaos (Lapedes and Farber, Gallant and White), and recognize hand-printed characters (Fukushima and Miyake). Trippi has assembled various papers which use neural networks in financial market forecasting, macro economic prediction, credit risk classification, exchange rate prediction and other applications related to finance and economics.

The most common uses of neural networks can be classified into the following categories: classification, associative memory, and autoassociative memory. An example of classification would be to classify sonar signals as those reflecting from a submarine or from a naturally occurring underwater object. Another example would be that used in this research. Namely given some inputs composed of forecasting variables, outputs are produced which indicate a buy, sell or hold position in a particular commodity or financial instrument. An example of an associative memory application would be any time series model or a price prediction model. An

autoassociative network is one in which some pattern that has been corrupted by noise is presented to the network and the network reproduces the original uncorrupted pattern. In general a neural network can be viewed as estimating a map $f: X \rightarrow Y$ where X is the space of inputs or independent variable and Y is the space of outputs or dependent variables. In the case of classification Y is a $n \times 1$ vector of variables, each of which indicate inclusion or exclusion in one of n different categories. In an associative memory application Y is a vector containing that which is to be predicted, e.g. the price of corn one month from now. In an auto associative application $Y=X$, where X is the uncorrupted version of the input pattern.

The terminology of neural networks has not become standardized and hence the term "neural network" can mean different things to different people. The term neural network defined in its most general sense is an architecture in which its operations are distributed among many relatively simple processors (Masters). This definition suggests a great deal of flexibility in what computing paradigms can be called neural networks. Indeed, a great deal of research has been devoted to developing different types of neural networks. The literature is extensive and developing rapidly and therefore a complete review of the subject is beyond the scope of this research. However, for the interest of those readers seeking to do research in this area, several different types of neural networks are briefly discussed below.

Some models that are decades old are now receiving renewed interest because they are easily recast as a neural network. For example Donald Specht's probabilistic neural network which is used for classification is identical to kernal discriminant analysis (Sarle 1994b). Another example would be the functional link network

developed by Yoh-Han Pao. The functional link network is simply a multiple regression with a nonlinear font-end, and a nonlinear transformation applied to the output (Masters). These two types of modeling techniques suddenly attracted attention when they were presented in the context of a neural network.

Other types of neural networks such as feedforward neural networks and radial basis (RBF) networks are more unique. However, there are some similarities between these types neural networks and existing modeling techniques. It will be shown later that the standard feedforward type network could be though of as a form of nonlinear regression. Xu, Krzyzak, and Yuille have established some useful connections between kernel regression estimators and RBF networks. Feedforward neural networks are the focus of this research and are discussed in detail in the next section.

Feedforward Neural Networks

In light of the considerable hype which has surrounded neural networks, it would be useful to discuss what a feedforward neural network is not before discussing what a feedforward neural network is. Neural networks were originally inspired by the way in which a group of biological neurons process information. Therefore, the development of neural networks has its roots in neuroscience. There are obvious analogies that can be drawn between the functioning of artificial neural networks and their biological counterparts. However, an artificial neural network is a much simplified model of the way a collection of brain cells operate. In fact, beyond simple analogies, the neurons in an artificial neural network share little in common with their biological counterparts.

The word neural probably leads people to sometimes write that a neural

network simulates the behavior of the human brain. The human brain contains about 1.5×10^{10} neurons of various types and each neuron receives signals from 10 to 10^{3} other neurons (Ripley). Therefore, an artificial neural network is a much simplified mathematical representation of the way a relatively small collection of biological neurons operate. The process by which biological neurons process information is complex. The communication between neurons is both electrical and chemical and each of these communication process is complex. As will become clear in the next section, the neurons or processing elements in an artificial neural network are simple nonlinear functions and the "communication" between the neurons is linear. However, even though a neural network shares little in common with the workings of biological neurons, they are powerful enough to possess the ability to "learn" from experience, develop rules, and recognize patterns in data.

If an artificial neural network is not a model of the brain, the question is what is a neural network? Before proceeding with the answer to this question, it would be useful to associate some of the terminology used in the neural network literature to the corresponding terminology used in statistics or econometrics. The neural network literature refers to (Sarle 1994b):

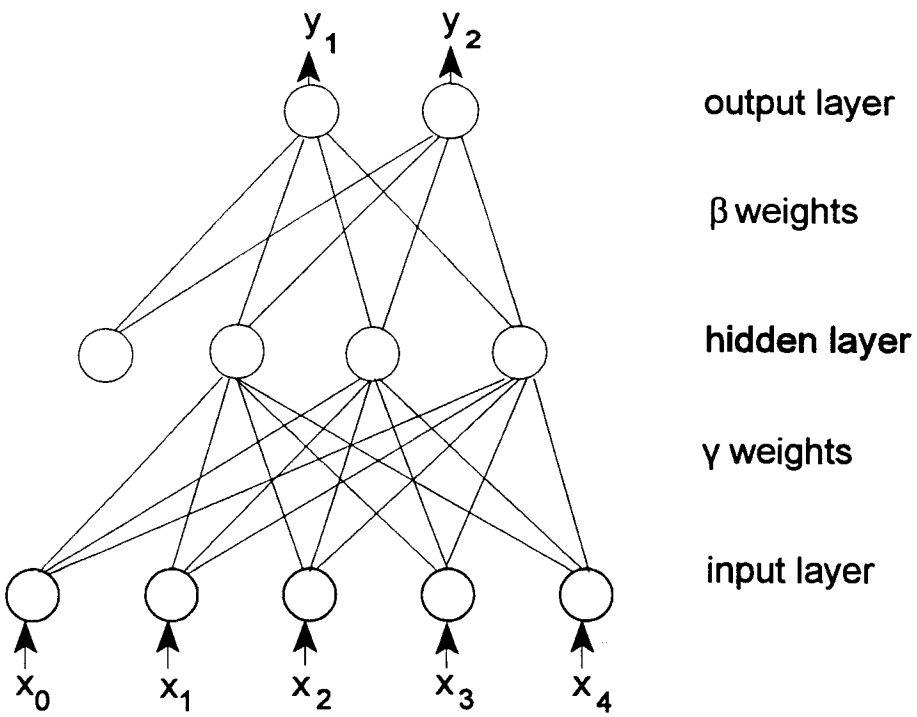
- independent variables as inputs
- dependent variables as targets
- predicted values as outputs
- individual variables as a feature
- estimation as training, learning, adaptation, or self-organization.
- observations as training patterns
- parameter estimates as synaptic weights or connection strengths.

In general, a neural network can be viewed as estimating a map $f: X \to Y$ where X is the space of inputs and Y is the space of outputs.

The following discussion describes how a feed forward type network with one hidden layer produces its output given some input. Figure 1 provides a reference for the discussion and is a visual image of equation (1) which will be developed. The neurons in a neural network are usually arranged in layers. The input layer contains the inputs (independent variables) at time t and the output layer contains the output(s) (dependent variables) at time t. Note that similar to a vector autoregression model, there could be more than one output (dependent variable).

Suppose we have n inputs and q outputs. Then for each training pattern (observation), the n Input neurons send the signals x_i, $i = 1, \ldots, n$, to the p neurons in the hidden layer via connection weights (parameters) γ_{ij}, $j = 1, \ldots, p$. Each hidden unit j then sums the input to itself yielding $\tilde{x}'\gamma_j$, where $\tilde{x} = (x_0 = 1, x_1, \ldots, x_n)'$ and $\gamma_j = (\gamma_{0j}, \gamma_{1j}, \ldots, \gamma_{nj})'$. Notice that by definition, $x_0 = 1$. The sum of each hidden neuron is then is then processed by an 'activation function' ψ which as a nonlinear mapping from $\Re$ to $\Re$. The output or activation of hidden neuron j is $\psi_j(\tilde{x}'\gamma_j)$, $j = 1, \ldots, p$. In other words, each hidden neuron is a nonlinear single (scalar) valued function whose input is the dot product of the input vector and the weight vector which is associated with itself. The signals from the hidden neurons are then passed to the output neuron(s) in an analogous manner as from the input layer to the hidden layer. The hidden layer sends the signal $\tilde{\psi} = (\psi_0 = 1, \psi_1, \ldots, \psi_p)'$ to the q neurons in the output layer via connection weights (parameters) β_{jk}, $j = 0, 1, \ldots, p$, $k = 1, \ldots, q$. The term ψ_0 will serve the same purpose as x_0 in

Figure 1. Feedforward Neural Network With One Hidden Layer
(4 input variables and 2 output variables)



the input layer. The output neurons then process the signals from the hidden layer in the same way that the hidden neurons process the signals from the input layer. Therefore, assuming an output 'activation function' F, the output from neuron k would be $y_k = F(\Psi'\beta_k)$ where $\beta_k = (\beta_{0k}, \beta_{1k}, ..., \beta_{pk})'$. Thus using all of the functional relationships that have been established, we can write the functional relationship between the input features (variables) and a particular output feature (variable) y_k:

$$(1) \qquad y_k = F(x,\theta) = F_k\left(\beta_{0k} + \sum_{j=1}^{p} \beta_{jk}\psi_j(\tilde{x}'\gamma_j)\right), \qquad k,p \in \mathbb{N}$$

where $\theta = (\beta_1', ..., \beta_k', \gamma_1', ..., \gamma_p')'$. If we assume that the activation function F is the identity function $F(a) = a$ and there is only one output or dependent variable, as is the case in this research, then equation (1) reduces to

$$(2) \qquad y = f(x,\theta) = \beta_0 + \sum_{j=1}^{p} \beta_j\psi_j(\tilde{x}'\gamma_j), \qquad p \in \mathbb{N}$$

where $\theta = (\beta_1', \gamma_1', ..., \gamma_p')'$. From the preceding discussion, it is clear that the neurons in a neural network need not be thought of as mysterious. All neurons in a neural network are mearly "processing elements". neurons in the input layer, they only serve as "input terminals" for the inputs to the network. It can be seen readily from equation (2) that a feedforward neural network can be considered a nonlinear regression.

The form of the activation functions, sometimes called transfer functions, ψ and F can be chosen quit freely. However, the functions are generally monotonically increasing. The two most common functions are the sigmoid and the hyperbolic tangent given by $f(x) = 1/(1 + \exp(-x))$ and $f(x) = (\exp(x) - \exp(-x))/(\exp(x) + \exp(-x))$ respectively. A nonlinear transfer function is required in the hidden layer a required because they are responsible for the

nonlinear approximation capabilities of the feedforward type neural network. A transfer function in the output neuron(s) is not required and thus is sometimes set to be the identity function as it is in equation (2). Different transfer functions can be used in different layers and even within layers. In this research, the hyperbolic tangent transfer function is used for the hidden layer neurons and the identity function for the output layer. The use of the hyperbolic tangent transfer function, as apposed to the sigmoid, has been shown to produce better convergence behavior in certain circumstances.

The input neuron $x_0 = 1$ and $\psi_0 = 1$ are commonly referred to as bias neurons. The parameters connecting the bias neurons to the respective neurons in the hidden or output layer are analogous to the intercept in a regression. These parameters for the hidden layer are γ_{0j}, $j = 1, ..., p$ and for the output layer β_{0k}, $k = 1, ..., q$. Since the bias terms in the hidden layer in effect produces an extra input whose value is always 1, care must be taken when using dummy variables as inputs. For example, four quarterly dummy variables as inputs would lead to a "redundant input". One of the inputs would be an exact linear combination of the other inputs, including the "bias input". Redundant Inputs cause "flats" in the parameter space, and thus can lead to numerical difficulties (White).

An obvious question to ask is, what are the properties of the mapping obtained by a neural network? It has been shown that the single hidden layer feedforward networks of the type given in equation 1 and depicted in figure 1 are "universal approximators". In other words, given sufficiently many hidden units and properly adjusted parameters, a neural network can approximate an arbitrary mapping arbitrarily well for a large class of functions. The theoretical function approximation

capabilities of feedforward neural networks have been explored by Hornik, Stinchcombe, and White and Cybenko. Barron showed that the approximation capabilities of feedforward neural networks require that the number of parameters grow linearly. Other function approximation methods, e.g. polynomial, spline, and trigonometric expansions, require that the number of parameters grow exponentially for comparable approximation. The universal approximation properties of neural networks are the key to the demonstrated usefulness of neural networks in many applications as well the potential usefulness of neural networks in economics. With a neural network there is no need to explicitly identify the functional form. Only the variables relevant to the particular problem need be identified.

Although not the focus of this research, a type of neural network called feedback or recurrent networks is worth mentioning. Feedback networks include one or more direct or indirect loops or connections. A common procedure is to include the values of hidden neurons in the current prediction or time period as inputs to the network in the next prediction or time period. If a feedback network is making a prediction a variable for time t it also has information on errors, predictions, or some other variable(s) whose value was determined at times prior to time t. Feedback networks have been shown to have interesting properties. for those readers interested, good references are Hertz, Krogh, and Palmer, Zurada, and Kuan and White.

Learning (Estimation) In Neural Networks

Until recently, and perhaps still, the generalized delta rule, commonly known as backpropagation, was the most commonly used training algorithm and was some

times viewed with Mystique. As Kuan and White write

> For a period, artificial neural network models coupled with the method of backpropagation came to be viewed as magic, with considerable accompanying hype and extravagant claims.

Backpropagation is the gradient descent method familiar to anyone with experience in nonlinear optimization. The analytical derivatives of the function for given in (1) are easily obtained. Those familiar with nonlinear optimization know that gradient descent is inferior to many other algorithms which are available. Gradient Descent is very slow to converge and in addition, if the error surface has "valleys", it can suffer from a condition known as hemstitching (Avriel). Hemstitching is a condition were the weight changes "bounce" from wall to wall, making little progress down the valley.

It should be noted that the traditional implementation of backpropagation is actually a quasi gradient descent whereby the parameters are adjusted after presentation of each observation. However, for most implementations that economists would be interested in, traditional nonlinear least squares is probably the preferred criterion. The preferable optimization routines would be Marquardt, various Newton methods, and conjugate gradient methods. In this research, the Marquardt algorithm will be used. Hagan and Menhaj give a detailed presentation of using this method for a neural network.

CHAPTER 4

DATA AND PROCEDURE

This chapter explains the methods used to accomplish the research objectives. The first section discusses the data and procedures used to develop the neural network trading models. Next, the trading rules and assumptions are presented. In the last section, the procedures used to test the statistical significance of the trading returns are presented. The chapter concludes with a summary of the data and procedures presented.

The Data

A trading system is constructed which uses neural network based trading models to produce trading signals. Simulated trading returns are generated using this model to trade the hard red winter wheat (hereafter referred to as KCBT wheat or simply wheat) and Deutsche mark futures contracts. To estimate the parameters of the neural network, inputs and outputs (independent and dependent variables respectively) are needed to implement the Marquardt estimation algorithm. The inputs and outputs of the network are a collection of pairs

$$\{(x_1, y_1), \ldots, (x_T, y_T)\} \tag{1}$$

where $x_t = [x_{1,t}\ x_{2,t}\ \ldots\ x_{n,t}]'$ is a $n \times 1$ vector of inputs and y_t is the desired output of the network for observation t. All data needed to calculate the inputs and outputs as defined in this section were obtained from the data vendor Technical Tools.

For both the wheat and deutsche mark trading models, the y_t's are defined by

(2) $$y_t = \begin{cases} 1, \text{ if } r_{t+1} \geq 0 \\ -1, \text{ otherwise} \end{cases}$$

where

(3) $$r_t = \ln (P_t / P_{t-1})$$

and P_t is the closing price of the specified commodity on the last trading day of week t. Thus the trading models will produce weekly trading signals. Note also that the values of the outputs are either 1 or -1 as opposed to the typical 1 or 0. The reason for this is because both inputs and outputs are scaled between 1 and -1. If a transfer function other than the identity function is used in the output layer, scaling of the outputs to the range of the transfer function is necessary. Scaling of the inputs is never required. However, to avoid numerical difficulties, it is common to scale the data to the range of the transfer functions in the hidden layer. The hyperbolic tangent transfer function used in this study has a range of -1 to 1, thus all data in this study are scaled to that range.

The inputs for both the wheat and deutsche mark trading models are composed of eight lags of the weekly returns with three additional quarterly dummy variables added to the inputs of the wheat model. Specifically, for the deutsche mark trading model, the inputs are defined by an 8×1 vector with the individual elements defined by

(4) $$x_{i,t} = r_{t-i-1} \qquad i = 1, \ldots, 8$$

where r_t is as defined in (3) with P_t being the weekly closing price for the Deutsche mark. The inputs to the wheat trading model are defined by an 11×1 vector with the first eight elements defined as in (4) for the Deutsche mark model with P_t

replaced by the weekly closing price for KCBT wheat. The final three elements of the input vector for the wheat trading model are defined by

(5) $$x_{(i+8)t} = \begin{cases} 1, & \text{if quarter is } i \\ -1, & \text{otherwise} \end{cases} \qquad i = 1,2,3$$

The quarterly dummy variables defined in (5) were added because of seasonality effects that may exist in the KCBT wheat contract. Quarterly dummy variables were excluded from the deutsche mark model because of the unlikelihood of any seasonality in the Deutsche mark contract. Note that dummy variables are included for only 3 quarters. Because of the bias term, defined in chapter 3, a 4th dummy variable would be a redundant input. Redundant inputs will produce numerical difficulties as described in White and Hecht-Nielsen.

A difficulty in using futures prices to construct a set of variables is that futures price series are "discontinuous" in the following sense. Each commodity is represented by several different futures prices representing different months of delivery. Because of liquidity costs, most commodity pool operators hold the majority of their positions in the "nearby" contract (Brorsen and Irwin). Therefore, it has become common practice to construct a price series using the contract closest to expiration. Constructing a price series in this manner assumes that on a given date, a position that is held in a contract nearing expiration is rolled into the next nearest contract month. The specific date on which a contract is rolled over into the next contract is called the rollover date. For this research, the rollover date is the 20th of the month prior to expiration for both wheat and the deutsche mark. If this date falls on a weekend or holiday, the rollover date is the next nearest trading day. For this research, a continuous price series around the rollover date would be

(6) $$\{\ldots, P_{t-1}^{O}, P_{t}^{N}, P_{t+1}^{N}, \ldots\}.$$

where P_t^N is the weekly closing price of the new contract being rolled into and P_{t-1}^O is the weekly closing price of the old contract. In (6), P_{t-1}^O is a weekly closing price that occurs before the 20th and P_t^N occurs on or after the 20th of the month before expiration of the contract.

The price series that would result from the procedure described above would still tend to be "discontinuous" on the rollover date. This is because it is common for the price of the next nearest contract to be at a discount or premium to the price of the nearest contract. Therefore, the weekly return series calculated from the price series in (6)

(7) $$\{\ldots, r_{t-1} = \ln(P_{t-1}^O/P_{t-2}^O), r_t = \ln(P_t^N/P_{t-1}^O), r_{t+1} = \ln(P_{t+1}^N/P_t^N), \ldots\}$$

would be inaccurate for the observation $r_t = \ln(P_t^N/P_{t-1}^O)$. To resolve this problem, the return series is calculated as

(8) $$\{\ldots, r_{t-1} = \ln(P_{t-1}^O/P_{t-2}^O), r_t = \ln(P_t^N/P_{t-1}^N), r_{t+1} = \ln(P_{t+1}^N/P_t^N), \ldots\}$$

if P_t falls on or after the 20th of the month prior to expiration and P_{t-1} before the 20th.

Neural Network Trading Model

The major advantage of neural networks is their nonlinear approximation capabilities. However, these capabilities are also a source of difficulty in developing a neural network that forecasts satisfactorily out-of-sample. Neural networks can memorize or overfit the data. If the network memorizes the data, it will fail to make generalizations about the relationship between the inputs and outputs of the network. The network would then perform poorly in out-of-sample tests. The degree to which a neural network "fits" the data is proportional to the number of hidden neurons. Therefore, a goal in building a neural network model is to estimate the number of

hidden neurons that will maximize forecasting ability.

To accomplish this goal, neural networks with different numbers of hidden neurons are trained on a training set and evaluated on an out-of-sample test set. The data in the training set and test sets are mutually exclusive. Thus networks containing different numbers of hidden neurons can be tested on test set to evaluate the ability of the different network configurations to generalize on a data set they have not been trained on. The configurations (number of hidden neurons) of the networks chosen to provide true out-of-sample testing results will be chosen based upon the configuration which produced the highest average net trading profits for the previous four testing periods. Thus, true out-of-sample testing will only be available after four preliminary testing periods. The procedure for estimating the number of hidden neurons may be made clearer by looking at table 1.

Table 1 presents the specific time periods for estimation and testing of the trading models for the wheat and deutsche mark trading models. All time periods given in the table are assumed to begin with the first trading day and end with the last trading day of the years given. The dashed line in the table indicates when true out-of-sample testing will begin. The testing periods from 1981-1984 can be considered preliminary testing periods since their purpose is to estimate the number of hidden neurons to use for the trading model estimated over the training period 1979-1984 and tested on the data from 1985. This procedure continues for all test periods. For example, the number of hidden neurons for the model tested on the data from 1986, are chosen based on the performance of networks with different configurations tested from 1982-1985.

Before the data discussed above is used to train or test a network, it is scaled

Table 1. Time Periods for Data Sets Pertaining to Estimation and Testing of the Neural Network Trading Models.[a]

Training Period	Testing Period
1975-1980	1981
1976-1981	1982
1977-1982	1983
1978-1983	1984
1979-1984	1985
1980-1985	1986
1981-1986	1987
1982-1987	1988
1983-1988	1989
1984-1989	1990
1985-1990	1991
1986-1991	1992

[a]All time periods listed in the table begin with the first trading day and end with the last trading day of the years given. Statistical tests of trading profits are performed on time periods after the dashed line.

or mapped between -1 and 1. The output of the network given in (2) is defined to be -1 or 1, thus no scaling of the output is necessary. The inputs are scaled according to

$$x'_{i,t} = 2(x_{i,t} - \min(\check{x}_i))/(\max(\check{x}_i) - \min(\check{x}_i)) - 1 \quad (8)$$

where $x_{i,t}$ is observation t of variable i, $x'_{i,t}$ is the scaled observation, and $\check{x}_i = [x_{i,1}, .., x_{i,T}]$. As an example, to scale the training set composed of data from 1979-1984, the maximum and minimum values are determined over the same range. The same maximum and minimum values are then used to scale the testing set for 1985.

The goal in estimation of the parameters of a neural network is to find the global minimum of the objective function. This can be difficult because of the presence of multiple local minimums in the objective function. To increase the probability of finding the global minimum, for each training period and network configuration, the estimation procedure is performed 10 different times using 10 different randomly chosen starting values. For the 10 estimations, the weights associated with the network with the lowest objective function value after convergence (defined in (9)) are retained.

The following summarizes the procedures described above. A total of 480 models are estimated for each of the two commodities, 10 for each network configuration times 4 configurations times 12 training periods. Based upon the objective function values, 4 networks out of each training period are retained. These 4 networks represent each of the four possible network configurations chosen for this research, i.e. 4, 6, 8, and 10 hidden neurons. These four networks for each training period, are used in the procedures described above to pick the configuration of the networks on which statistical tests of trading profits are calculated in future testing

periods.

The training of the networks was done on an IBM 3090 mainframe using the vector facility and SAS software. In particular, the training was done using the proc NLIN procedure available in the STAT module of SAS software. The Marquardt algorithm was used to estimate the parameters. The macro used to implement the proc NLIN procedure is listed in the appendix. The macro is similar to those given in Sarle (1994a). The estimation algorithm is determined to have converged if the following conditions are met

$$(\mathrm{SSE}^{i-1} - \mathrm{SSE}^{i})/(\mathrm{SSE}^{i} + 10^{-6}) < 10^{-16}$$

and

$$\max\left(\max_m(|B_m^{i-1} - B_m^{i}| / |B_m^{i-1}|,\ \max_n(|\theta_n^{i-1} - \theta_N^{i}| / |\theta_N^{i-1}| \right) < 10^{-16} \tag{9}$$

$$m = 1, .., k, \quad n = 1, .., p$$

where SSE^i is the value of the sum-of-squared error objective function for the ith iteration and B_m^i and γ_n^i are the parameters or weights for the neural network given in (1) Of chapter 3. Iteration refers to the iteration of the Marquardt estimation algorithm. The first statement in (9) is that the change in the objective function is small and the second condition is that the maximum change in the parameters is small.

Trading Rules and Assumptions

The trading rule is

$$\begin{array}{l} \text{long if } \hat{y}_t \geq 0, \\ \text{short if } \hat{y}_t < 0 \end{array} \tag{10}$$

where $\hat{y}_t$ is the output of the neural network trading models in the testing periods. Since trading signals occur at the end of the trading week, it is assumed that trades are entered on the opening of the next trading day, typically Monday. The trading

models are in the market at all times, therefore, positions are held until the trading model indicates a change in position. Transaction costs are composed of the commission costs and the skid error, or bid-ask spread, of each trade. Commission costs are $35 per round-turn trade, since this is the maximum cost that most discount futures brokers charge. An additional $30 is added to the transaction cost to account for skid error. This leaves transaction costs of $65 per round-turn trade. Transaction costs will also be incurred on rollover dates when a trade is rolled over into the next contract month.

Evaluation procedure

Gross daily trading returns are computed for the neural network trading model assuming one contract is being traded. The gross daily trading returns are the actual dollar returns. From the gross daily returns, net daily trading returns are calculated by taking into account transaction costs. The hypotheses to be tested are

$$H_0: GTR \leq 0 \qquad H_a: GTR > 0 \tag{11}$$

and

$$H_0: NTR \leq 0 \qquad H_a: NTR > 0 \tag{12}$$

where GTR and NTR are gross and net trading returns respectively.

Some studies have sought to test the hypotheses above by using standard t-ratios. The shortcoming of this procedure is that the use of a t-statistic assumes that the underlying distribution of returns is normally and independently distributed. However, research on futures prices indicates that the distribution of futures prices is leptokurtic (Yang and Brorsen; Hudson, Leuthold, and Sarassoro; Hall, Brorsen, and Irwin; Gordon). A leptokurtic distribution has more observations around the mean

and fatter tails than a normal distribution. If futures prices are leptokurtic then the returns to trading systems are likely to be leptokurtically distributed. These departures from normality will be addressed using simulated distributions of trading returns generated from bootstrapping procedures. The bootstrapping type of methodology was first applied to the analysis of trading returns by Brock, Lackonishock, and LeBaron. Others using this methodology are Allen and Karjalainen, and Levich.

The bootstrapping procedure consists of fitting a hypothesized null model to the data. In this research the data set is the log of weekly price changes. Two different null models will be tested in this research: a random walk with drift, and a GARCH(1, 1) model. The null models are used to generate a large number of simulated data sets. The data sets are created by first estimating the null model and then scrambling, i.e. resampling with replacement, the residuals. The data sets are then used to create new testing sets for the neural network trading models. These simulated testing sets are used with the neural networks which were chosen to generate true out-of-sample testing results to obtain a simulated distribution of trading profits under the null model.

If we wish to generate B simulated return series from a specific null model, a resample is obtained by drawing T items with replacement from the residual series of the estimated null model

(13) $$e = \{e_1, \ldots, e_T\}$$

to form a resampled residual series

(14) $$e_b^* = \{e_{b,1}^*, \ldots, e_{b,T}^*\}.$$

The series in (14) is then substituted for the residuals in (13). That is, the series in

(14) is added to the predicted values from the estimated null model to create a new set of weekly returns

$$r_b^* = \{r_{b,1}^*, \ldots, r_{b,T}^*\}, \qquad b = 1, \ldots, B. \tag{15}$$

Thus, a new set of weekly returns is created each time the residuals are "scrambled".

The random walk with drift null model is defined to be

$$r_t = e_t \tag{16}$$

where e_t ~ IID with non-zero mean. The simulated return series are created by "scrambling" the original return series. If the null model is true, the simulated series will have the same drift, volatility, and unconditional distribution as the original series. In other words, the simulated price series will replicate the statistical properties of the original series.

The GARCH(1, 1) null model is defined to be

$$\begin{aligned} r_t &= \mu + \epsilon_t \\ \epsilon_t &= \sqrt{h_t}\, e_t \\ h_t &= \omega + \alpha \epsilon_{t-1}^2 + \gamma h_{t-1} \\ e_t &\sim \mathrm{N}(0,1) \end{aligned} \tag{17}$$

The GARCH model introduced by Bollerslev is a generalization of the ARCH model introduced by Engle. The GARCH model addresses both nonlinear dependence and leptokurtosis. The residuals e_t in (17) can be standardized as

$$z_t = e_t / \sqrt{h_t}, \qquad z_t \sim \mathrm{N}(0,1). \tag{18}$$

Bootstrapping of the standardized residuals in (18) generates the resampled series

$$z_b^* = \{z_{b,1}^*, \ldots, z_{b,T}^*\} \qquad b = 1, \ldots, B \tag{19}$$

which are used to generate the resampled residual series

(20) $$e_b^* = \{\sqrt{h_1}z_{b,1}^*, \ldots, \sqrt{h_T}z_{b,T}^*\}, \qquad b = 1, \ldots, B.$$

The series in (20) is used to generate B new sets of weekly returns.

For the random walk with drift null model defined in (16), 500 simulated return series are generated for each of the eight out-of-sample test periods listed in table 1, i.e. 1985-1992. That is, each of the eight return series associated with the eight true out-of-sample testing periods is "scrambled" 500 times. These simulated weekly return series are then used to create 500 testing sets containing the neural network inputs discussed earlier in this chapter. Each neural network trading model associated with each of the out-of-sample testing periods is tested on the 500 simulated test sets associated with each of the testing periods. Net trading profits computed from testing the network on the simulated testing sets is used to calculate the simulated p-values

(21) $$p(\text{MTR} > c) = \sum_{b=1}^{500} n_b / 500$$

where $p(\cdot)$ represents probability, c is the value being tested, GTR is as defined in (11), and

(22) $$n_b = \begin{cases} 1, & \text{if } \text{GTR}_b > c \\ 0, & \text{otherwise} \end{cases}$$

where GTR_b is the gross trading returns for the neural network trading model on the bth simulated testing set. Simulated p-values for net trading returns (NTR) are similarly defined.

The procedure for the GARCH(1, 1) null model is analogous to that above. The testing periods are only a year long and thus heteroskedasticity may fail to be significant in some of the individual time periods. Therefore, the GARCH(1, 1) model is estimated over the entire test period from 1985-1992. However, to create the simulated testing sets for a specific testing period, only the residuals from that

testing period are bootstrapped.

CHAPTER 5

RESULTS AND IMPLICATIONS

This chapter presents the results of the estimation of the neural networks as well as the trading simulations performed with those neural networks. The chapter is divided into 3 sections. The first section presents the results for the estimation of the parameters of the neural network trading models. Next, the results of the trading simulations are presented. The last section presents the results of the statistical tests of the profits obtained from the trading simulations.

Results of Neural Network Estimation

Multiple local minima can cause an estimation algorithm to reach a solution which is very different from the solution at the global minimum. As was discussed in chapter 4, to increase the probability of obtaining a solution close to the global minimum, the estimation for each training period was performed 10 different times, each time from a different random starting point. Figures 2 and 3 are representative examples of the results of this reestimation. The figures show in histogram form the sum-of-squared error (SSE) after convergence across the possible numbers of hidden neurons for the different starting values. Figure 2 shows the results for the deutsche mark trading model estimated on the data from training period 1. The reduction in SSE from the highest to the lowest was as much as 43% for 4 hidden neurons, from 470 to 264. The reduction was around 30% for 6 and 8 hidden neurons. In General,

Figure 2. Histogram of Objective Function Values Across Hidden Neurons For Different Starting Values, for the Deutsche Mark Trading Model and Training Period 1.

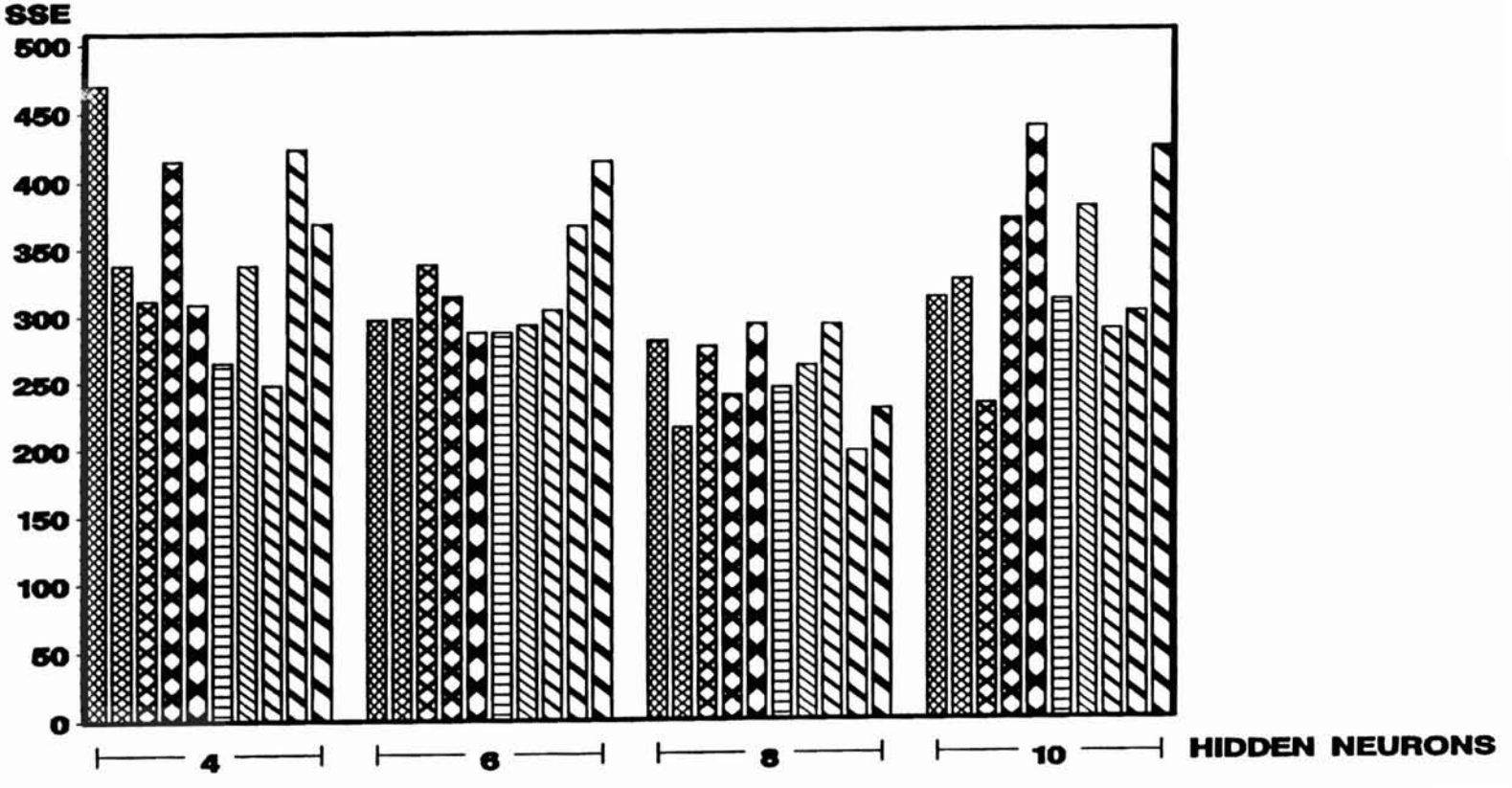
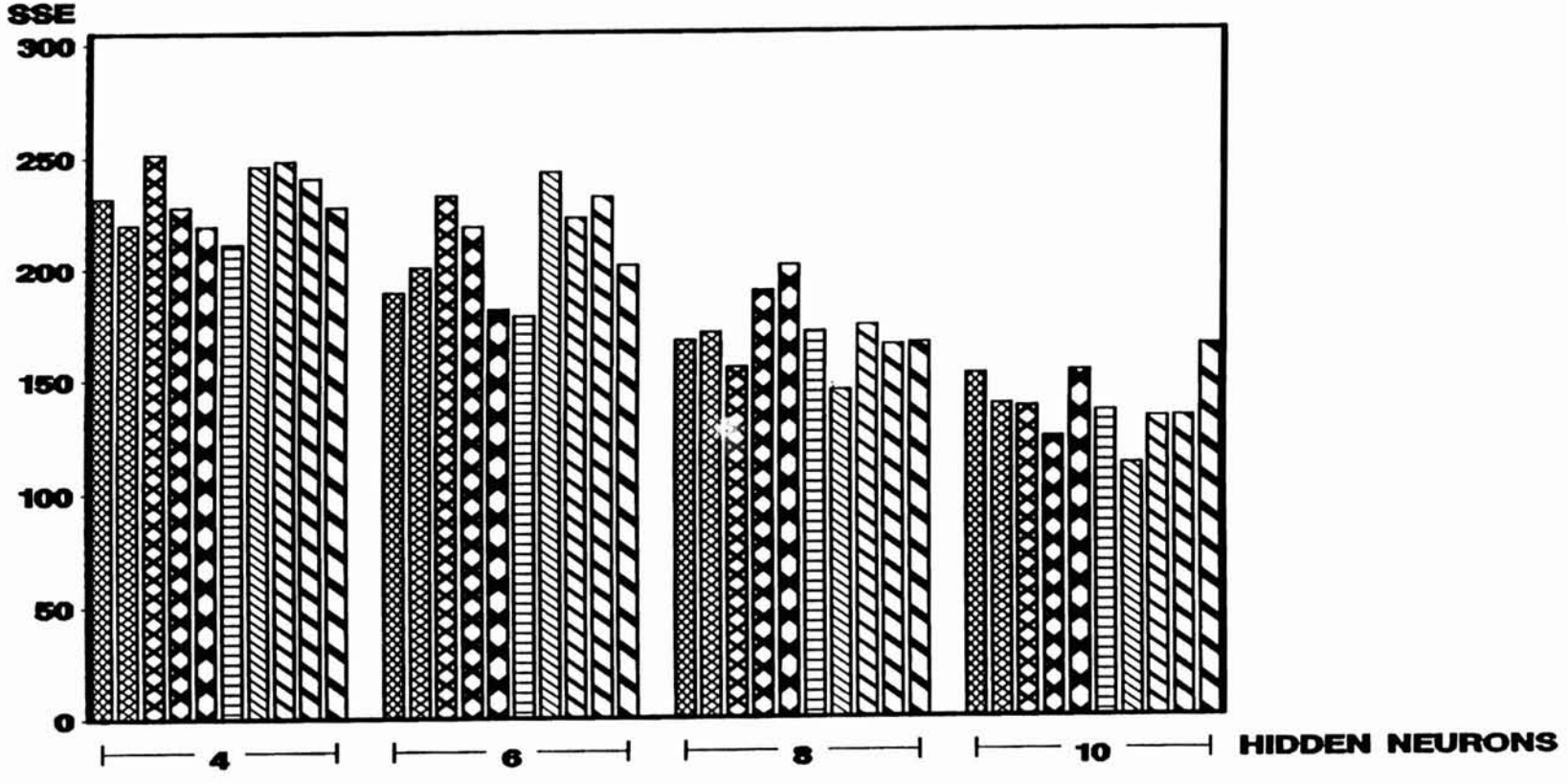


Figure 3. Histogram of Objective Function Values Across Hidden Neurons For Different Starting Values, for the KCBT Wheat Trading Model and Training Period 8.



The results shown in figures 2 and 3 show that at least with this particular data, the objective function value can vary greatly. The common practice of one estimation, with no additional effort to reach a lower objective function value may not be adequate. The variation in objective function values might indicates the need for multiple reestimations. Another, possibly superior, solution to the challenge of finding the global minimum may involve the application of a global optimization algorithm. Global optimization algorithms have been applied to the estimation of neural network parameters by Baba et al., and Styblinski and Tang. Global optimization algorithms have shown some promise and are an important area of research, however, they are beyond the scope of this research.

Trading Simulation Result

As discussed in chapter 4, the neural networks were chosen a priori to have one hidden layer. What remained to be chosen was the number of hidden neurons in the single hidden layer. Following the procedures outlined in chapter 4, the number of hidden neurons chosen for the networks which provide out-of-sample trading results are chosen by examining the out-of-sample trading performance of various network configurations in previous testing periods. The network configuration, i.e. 4, 6, 8 or 10 hidden neurons, for a particular testing period is chosen based upon which configuration produced the highest average out-of-sample net trading profits in the four previous testing periods. Therefore, out-of-sample trading results will be calculated for all testing periods and all possible network configurations but true out-of-sample trading simulation results are only available after the first 4 testing periods.

Tables 2 and 3 show the out-of-sample net trading profits for 4, 6, 8 and 10

Table 3. Net Trading Profits For All Numbers of Hidden Neurons for trading KCBT Wheat Futures Based on a neural network.

Training Period[a]	Testing Period[b]	Hidden Neurons			
		4	6	8	10
1975-1980	1981	-400	990	-240	-695
1976-1981	1982	-2637	-102	5202	-92
1977-1982	1983	-2070	-470	-3335	-3830
1978-1983	1984	-1145	-2750	-2020	-785
1979-1984	1985	120	-1295	-135[c]	-2775
1980-1985	1986	-4927	1862	-5722[c]	-4737
1981-1986	1987	-1892	-1097[c]	-4457	-6147
1982-1987	1988	257	-3192[c]	4572	-2507
1983-1988	1989	-6365	-2545[c]	-1350	135
1984-1989	1990	-3260	-2700[c]	-7025	-1685
1985-1990	1991	-2175	1790[c]	5015	-2725
1986-1991	1992	-12	582[c]	-2082	-1842
	Average[d]	-2282	-824	-1398	-2786

[a]The training period starts on the first trading day of the first year given and ends on the last trading day of the second year given.
[b]Testing period begins on the first trading day and ends on the last trading day of the year given.
[c]Indicates a net trading profit figure for which statistical tests of significance will be calculated. The number of hidden neurons associated with this net trading profit figure produced the highest average out-of-sample net trading profits in the four previous testing periods. Statistical tests of trading profits are only calculated for testing periods after the dashed line, i.e. 1985-1992.
[d]Average is calculated from 1985-1992.

hidden neurons for the deutsche mark and wheat trading models. The dashed lines indicate the time at which true out-of-sample trading results are available. For each testing period after the dashed line a c by a net trading profit figure indicates that statistical tests of significance will be calculated for that number. These net trading profits are associated with the network configuration which was chosen for out-of-sample trading simulations.

The results presented in tables 2 and 3 show that on average the neural network trading models were not profitable. It can also be seen that there is a large variation in profits. Assuming one contract is traded, the highest individual profit for a single testing period was $12,232 for the deutsche mark and $5015 for wheat. However, it is obvious from viewing all of the results that large trading profits for a single year are probably spurious. One can see the dangers of evaluating a trading system on the basis of one year's profits.

Tables 4 and 5 provide other statistics for the trading models. The gross profit and trading costs given in tables 4 and 5 are associated with the network configurations which were indicated in tables 2 and 3 to be the networks chosen for out-of-sample trading simulations. As we would expect from the net profits in tables 1 and 2, the gross trading profits are not impressive. The lack of profitability of the neural network trading models is reflected in the fact that the turning point accuracy is around 50 percent. With a transaction cost of $65 per round-turn trade, the trading costs would indicate that the average trade is being held for around 2 weeks for both the deutsche mark and wheat trading models.

Table 4. Various Statistics for Deutsche Mark Trading Models.

Testing Period	Trading Costs[a]	Gross Profit	%TPA[b]	Number of Weeks Long[c]	Number of Weeks Short
1985	2275	3750	56	21	31
1986	1495	-10762	44	18	34
1987	1885	1487	51	20	33
1988	2080	3600	52	19	33
1989	1755	13987	62	14	38
1990	1690	-5812	40	24	28
1991	1885	9100	56	32	20
1992	1495	-362	55	42	11
Average	1820	1873	52	23.75	28.5

[a]Trading costs assume one contract is being traded. Trading costs are composed of $35 for commission costs and $30 for skid error.
[b]Percentage of correct turning point forecasts (%TPA) are according to the output variable given in chapter 4.
[c]Indicates the number of weeks in the testing period which in which the trading model indicated a long position.

Table 5. Various Statistics for KCBT Wheat Trading Models.

Testing Period	Trading Costs[a]	Gross Profit	%TPA[b]	Number of Weeks Long[c]	Number of Weeks Short
1985	1885	1750	54	25	27
1986	2210	-3512	42	28	24
1987	1885	787	57	33	20
1988	2080	-1112	44	30	22
1989	1820	-725	52	30	22
1990	1625	-1075	54	30	22
1991	1885	3675	52	28	24
1992	2080	2662	51	31	22
Average	1933	306	51	29.38	22.88

[a]Trading costs assume one contract is being traded. Trading costs are composed of $35 for commission costs and $30 for skid error.
[b]Percentage of correct turning point forecasts (%TPA) are according to the output variable given chapter 4.
[c]Indicates the number of weeks in the testing period which in which the trading model indicated a long position.

Statistical Evaluation

The hypotheses to be tested are that gross and/or net trading returns are less than zero. To statistically test these hypotheses, a bootstrapping procedure is used to simulate the distribution of the gross and net trading returns under two separate null models. The two null models are that the weekly returns follow a random walk with drift or a GARCH(1, 1) process. In this study, the bootstrapping procedure consists of generating 500 simulated testing sets from each of the null models for each of the testing periods. The procedures used to generate these testing sets are given in detail in chapter 4, thus they will not be repeated here. The neural network trading models which were chosen to generate true out-of-sample testing results for a specified testing period are tested on each the 500 testing sets corresponding to the null models. The simulated distributions are composed of the trading profits obtained by this procedure.

Table 6 contains the parameter estimates as well as their t-values for the GARCH null model. All parameters, except the intercept for the wheat data, are significant. Tables 7 and 8 present the simulated p-values under both null models for both net and gross trading profits. For the deutsche mark trading models only the net and gross profits for the testing period 1989 are significant at the .05 level. For the wheat models, none of the testing periods showed significant net or gross profits.[1]

---

[1]To test whether the poor results for the neural network models were due to a poor neural network model or the lack of any dependence (linear or nonlinear) in the weekly returns, a Donchian trading system was tried (Donchian). For this study, the trade signals for the Donchian system were defined by: If closing price for this week is higher than the highest weekly closing price for the last eight weeks then go long and hold the position until a sell signal occurs, Else if closing price is lower than the lowest weekly closing price for the last eight weeks than go short and hold the position until a buy signal occurs. The Donchian system averaged $-386 per year for KCBT wheat and $2424 per year for the deutsche mark over 1985-1992. Bootstrapping analogous to that used for the neural network trading models was used to test the null hypothesis of zero profits from the Donchian system. No trading profit from the Donchian system for an individual year for either commodity was significant at the .05 level. The Donchian system only performed marginally better than the neural network models. However, it shows that the reasons for the poor trading performance reported in this study may lie with the neural network trading models. Further research to compare the neural network models to more traditional technical trading rules is needed.

Table 6. GARCH(1, 1) Parameters Estimated Over 1985-1992 for KCBT Wheat and Deutsche Mark Weekly Returns.[a]

	μ	ω	α	γ
Deutsche Mark	0.15168 (1.871)	0.60236 (1.955)	0.15527 (2.774)	0.65120 (4.886)
KCBT Wheat	-0.00230 (-0.024)	0.34896 (2.105)	0.20694 (4.247)	0.73722 (13.628)

[a]T-statistics are given in parentheses.

Table 7. Simulated P-Values for Gross and Net Profits for Deutsche Mark.[a]

Testing Period	Net Profit[b]	GARCH P-Value[c]	Random Walk P-Value[d]	Gross Profit[e]	GARCH P-Value	Random Walk P-Value
1985	1475	.288	.288	3750	.244	.260
1986	-12257	.904	.916	-10762	.908	.920
1987	-397	.356	.418	1487	.360	.426
1988	1520	.386	.392	3600	.378	.374
1989	12232	.026	.034	13987	.024	.034
1990	-7502	.782	.734	-5812	.794	.738
1991	7215	.204	.186	9100	.200	.182
1992	-1857	.538	.488	-362	.536	.486

[a]Simulated p-values are fraction of net or gross profits from bootstrap simulations which are greater than the profit value being tested. A simulated p-value greater than .05 indicates significantly positive returns at the .05 level.
[b]The net profit figures in this column are reproduced from table 2.
[c]The GARCH(1, 1) null model is described in equation 17 of chapter 4.
[d]The Random walk with drift null model is described in equation 15 of chapter 4.
[e]The gross profit figures in this column are reproduced from table 4.

Table 8. Simulated P-Values for Gross and Net Profits for KCBT Wheat.[a]

Testing Period	Net Profit[b]	GARCH P-Value[c]	Random Walk P-Value[d]	Gross Profit[e]	GARCH P-Value	Random Walk P-Value
1985	-135	.142	.138	1750	.150	.142
1986	-5722	.962	.958	-3512	.932	.934
1987	-1097	.304	.382	787	.338	.398
1988	-3192	.668	.724	-1112	.654	.704
1989	-2545	.930	.588	-725	.722	.622
1990	-2700	.656	.500	-1075	.684	.522
1991	1790	.118	.138	3675	.118	.136
1992	582	.170	.254	2662	.040	.242

[a]Simulated p-values are fraction of net or gross profits from bootstrap simulations which are greater than the profit value being tested. A simulated p-value greater than .05 indicates significantly positive returns at the .05 level.
[b]The net profit figures in this column are reproduced from table 3.
[c]The GARCH(1, 1) null model is described in equation 17 of chapter 4.
[d]The Random walk with drift null model is described in equation 15 of chapter 4.
[e]The gross profit figures in this column are reproduced from table 5.

The large trading profits obtained for the deutsche mark model for 1989 could be the result of a neural network model which has truly captured some of the behavior of the deutsche mark futures contract. As discussed in chapter 4 and the first section of this chapter, the globally optimal set of parameters for a neural network are difficult to estimate. The large trading profits in 1989 could be the result of finding an optimal set of parameters. To investigate this possibility, the original unscaled data for the testing periods 1990-1992 were scaled using the maximums and minimums that were used to scale the testing set for 1989. The trading model which produced the large trading profits in 1989 was then tested on the testing sets for 1990-1992. The trading profits obtained were -8657, 420, and -19912 in the years 1990, 1991, and 1992 respectively. Therefore, we can assume that the large profits obtained for 1989 were spurious. Indeed, tables 2 and 3 show multiple net trading profits of a large amount, both positive and negative. However, the positive trading profits with a relatively large magnitude appear randomly scattered across hidden neurons and test periods. It is possible that for this problem, 500 bootstrap samples are not enough to accurately measure p-values for profits with a large magnitude.

CHAPTER 6

SUMMARY AND CONCLUSIONS

The first section of this chapter presents a summary of the results reported in the previous chapter. Conclusions to be drawn from these results are also reported. The last two sections discuss the limitations of the study and give suggestions for further research.

Summary of Results and Conclusions

The reestimation from different starting points of the neural networks produced widely varying sum-of-squared error (SSE) values after convergence. The reduction in SSE from the highest to the lowest was as much as 43% for the deutsche mark and 30% for wheat. The results showed that the common practice of one estimation, with no additional effort to reach a lower objective function value may not be adequate. The variation in SSE in this study indicates the need for using alternative sets of starting values.

Out-of-sample trading results for all possible network configurations, i.e. 4, 6, 8 and 10 hidden neurons, showed a large variation in profits. The highest individual profit for a single testing period, i.e. 1 year, was $12,232 for the deutsche mark and $3675 for wheat. However, the trading profits were on average zero or negative and thus the large trading profits for a single year and a specific model are probably spurious.

A researcher who wished to report positive results for neural-network-based

trading models could report the trading results of the network configuration which produced the highest trading profits each testing period. These results would not be truly out of sample. To avoid this problem, the trading results which were chosen to perform statistical tests on were produced by a network with a configuration that was chosen based upon information available up until the beginning of the specific testing period. These true out-of-sample trading results showed that the neural networks trading models were not on average profitable. Assuming one contract is being traded, average net trading profits for the eight testing periods were $53 for the deutsche mark and $-1,627 for wheat. Average gross trading profits were $1873 and $306 for the deutsche mark and wheat respectively.

Bootstrapping procedures were carried out to test the trading profits for each testing period. Only the testing period of 1989 for the deutsche mark showed significant profits at the .05 level. The net trading profits for this testing period were $12,232. To investigate whether these trading results were the result of finding a globally optimal set of parameters, the network which produced the large profits in 1989 was tested on the testing periods 1990-1992. The average trading profits obtained from this were negative. Therefore, we can assume that the large profits obtained for 1989 were spurious.

For the specific trading models in this study, we cannot reject the null hypothesis that gross or net trading returns are less than or equal to zero. Consequently, using a feedforward neural network with one hidden layer, we cannot conclude that the weekly returns for deutsche mark or KCBT wheat futures are predictable using an information set consisting of eight lags of the weekly returns. Therefore, using the specific trading models in this study and the commodities tested, we cannot conclude that neural networks would be useful in futures trading. Based on the information obtained from this study,

we cannot reject the hypothesis that futures prices are weak-form efficient.

The results of this research should only be interpreted in the context of the specific trading model used in this research. In addition, there are many limitations to this research. Numerous other studies have reported significant trading profits using more traditional technical trading rules. Among these studies are Lukac and Brorsen and Boyd and Brorsen. Further research is needed before rigorous conclusions about the dependence, possibly nonlinear, which may exist in the commodities studied in this research.

Limitations of Study

The greatest limitation of this study is that we do not know why the neural network trading models performed poorly. It is possible that given the information set that was available to the networks, there is no forecastable structure in the weekly returns of the two commodities tested. It is also possible that the feedforward type of neural network structure is not adequate for the forecasting problem in this study. Or a feedforward neural network may be adequate, but we might have been unable to find a globally optimal set of parameters. An additional limitation is that any conclusions drawn from this study are weakened by the fact that only two commodities are tested.

Directions for Further Study

The limitations of this study provide many opportunities for further study. The most potential for further research is in the improvement of the neural network trading models. The feedforward type of neural network with one hidden layer may not be adequate to model the behavior of the two commodities in this study with the inputs that were used. One hidden layer is theoretically capable of finding almost any mapping

function between the inputs and outputs. However, as the complexity of the mapping function increases, the difficulty of finding the globally optimal set of parameters may also increase. In some cases it may be easier to estimate the parameters of a network with two hidden layers than a network with one hidden layer. Alternatives to the feedforward type of neural network architecture may be more capable of modeling commodity prices. For example, recurrent neural networks are capable of rich dynamic behavior, exhibiting memory and context sensitivity (Kuan and White).

Better methods for estimating the parameters of the neural network trading models also offer opportunities for further research. Global optimization algorithms such as those given in Baba et al., and Styblinski and Tang may be superior to the reestimation technique used in this study. Robust estimation techniques may also be an area of further research. Related to robust estimation techniques, alternatives to the sum-of-squared error objective function could be examined.

The inputs to the neural network trading models were relatively simple. It is likely that another set inputs could be found which produced better results. However, the set of inputs to choose from is very large. The use of economic theory or statistical techniques would be needed to narrow the set of possible inputs to a size that could actually be examined. Statistical tests of nonlinear structure may also be useful in finding a better set of inputs as well as the methods used to model the commodity price behavior using these inputs. The neural network test for neglected nonlinearity presented in Kuan and White is one such test.

BIBLIOGRAPHY

Allen, Franklin, and Risto Karjalainen. "Using Genetic Algorithms to Find Technical Trading Rules." unpublished working paper, Rodney L. White Center for Financial Research, University of Pennsylvania, 1993.

Avriel, Mordecai. *Nonlinear Programming Analysis and Methods.* Englewood Cliffs, New Jersey: Prentice-Hall Inc., 1976.

Baba, Norio, Yoshio Mogami, Motokazu Kohzaki, Yasuhiro Shiraishi, and Yutaka Yoshida. "A Hybrid Algorithm for Finding the Global Minimum of Error Function of Neural Networks and Its Applications." *Neural Networks* 7(1994):1253-1265.

Barron, A.R. "Universal Approximation Bounds for Superpositions of a Sigmoidal Function." Manuscript, Department of Statistics, University of Illinois, IL 61820, 1991.

Beja, Avraham and M. Barry Goldman. "On the Dynamic Behavior of Prices in Disequilibrium." *Journal of Finance* 34(May 1980):235-247.

Bergerson, Karl and Donald C. Wunsch II. "A Commodity Trading Model Based on a Neural Network-Expert System Hybrid." *IEEE International Conference on Neural Networks,* 1991.

Blank, S.C., ""Chaos" in Futures Markets? A Nonlinear Dynamical Analysis." *Journal of Futures Markets* 11(1982):711-728.

Boyd, Milton S. and B. Wade Brorsen. "Factors Related to Futures Market Disequilibrium." *Canadian Journal of Agricultural Economics* 39(Dec 1991):769-778.

Brock, W.A., D. Hsieh, and B. LeBaron. *Nonlinear Dynamics, Chaos, and Instability: Statistical Theory and Economic Evidence.* MIT Press, Cambridge, MA, 1991.

Brock, William, Josef Lakonishok, and Blake LeBaron. "Simple Technical Trading Rules and the Stochastic Properties of Stock Returns." *The Journal of Finance* 5(December 1992):1731-1764.

Brock, W.A., and C.L. Sayers. "Is the Business Cycle Characterized by Deterministic Chaos?" *Journal of Monetary Economics* 22(1988):71-90.

Brorsen, B. Wade, and Scott H. Irwin. "Futures Funds and Price Volatility." *Review of Futures Markets* 6(1987):118-135.

Brown, David P., and Robert H. Jennings. "On Technical Analysis." *The Review of Financial Studies* 2(Dec 1989):527-551.

Collard, J.E. "A Commodity Trader." *IEEE Conference on Neural Networks*, 1990.

Collard, J.E. "Commodity Trading with a Two Year Old." *ORSA/TIMS Meeting*, Nashville, 1991.

Cybenko, G. "Approximation by Superpositions of a Sigmoid Function." *Mathematics of Control, Signals, and Systems* 2(1989):303-314.

De Grauwe, P., H. Dewachter, and M. Embrechts. *Exchange Rate Theory: Chaotic Models of Foreign Exchange Markets*. Blackwell, Oxford, 1993.

Decoster, G.P., and W.C. Labys, and D.W. Mitchell. "Evidence of chaos in commodity futures prices." *Journal of Futures Markets* 12(1992):291-305.

Donchian, R.D. "Trends Following Methods in Commodity Analysis." *Commodity Year Book - 1957*, Commodity Research Inc., 1957, pp. 35-47.

Edwards, Franklin R., and Cindy Ma. "Commodity Pool Performance : Is the Information Contained in Pool Prospectuses Useful?" *Journal of Futures Markets* 8(Oct 1988):589-616.

Egan, Jack. "Artificially Intelligent investing." *U.S. News & World Report* 56(March 1993):73.

Elton, Edwin J., Martin J. Gruber, and Joel Rentzler. "New Public Offerings, Information, and Investor Rationality : The Case of Publicly Offered Commodity Funds." *Journal of Business* 62(Jan 1989):1-15.

Fama, Eugene F., "Efficient Capital Markets: A Review of Theory and Empirical Work." *Journal of Finance* 25(May 1970):383-417.

Frank, M., and T. Stengos. "Measuring the Strangeness of Gold and Silver Rates of Return." *Review of Economic Studies* 56(1989):553-68.

Fukushima, K. And S. Miyake. "Neocognition: A New Algorithm for Pattern Recognition Tolerant of Deformations and Shifts in Position." *Pattern Recognition* 15(1984):455-469.

Gordon, J.D. "The Distribution of Daily Changes in Commodity Futures Prices." Technical Bulletin No. 1702, ERS, USDA.

Grudnitski, G. and L. Osburn. "Forecasting S&P 500 and Gold Futures Prices: An Application of Neural Networks." *Journal of Futures Markets* 13(1993):631-638.

Hagan, Martin T., and Mohammad B. Menhaj. "Training Feedforward Networks with the Marquardt Algorithm." *Neural Networks* 5(Nov 1994):989-993.

Hall, J.A., B. Wade Brorsen, Scott H. Irwin. "The Distribution of Futures Prices: A Test of the Stable Paretian and Mixture of Normals Hypothesis." *Journal of Financial and Quantitative Analysis* 24(March 1989):105-116.

Hecht-Nielsen, R. "Theory of the Back-Propagation Neural Network." in *Proceedings of the International Joint Conference on Neural Networks,* Washington, D.C., 1989, IEEE, New York.

Hertz, John, Anders Krogh, Richard G. Palmer. *Introduction to the Theory of Neural Computation.* Redwood City, California: Addison-Wesley Publishing Co., 1991.

Hinich, M.J., and D.M. Patterson. "Evidence of Nonlinearity in Daily Stock Returns." *Journal of Business and Economic Statistics* 3(1985):69-77.

Hornik, K., M. Stinchcombe, and H. White. "Multi-Layer Feedforward Networks are Universal Approximators." *Neural Networks* 2(1989):359-366.

Hsieh, D. "Chaos and Nonlinear Dynamics: Applications to Financial Markets." *Journal of Finance* 46(1991):1839-1878.

Hudson, M.A., R.M. Leuthold, and G.F. Sarassoro. "Commodity Futures Price Changes: Recent Evidence for Wheat, Soybean and Live Cattle." *Journal of Futures Markets* 7(June 1987):287-301.

Irwin, Scott H., Terry R. Krukemyer, Carl R. Zulauf. "Investement Performance of Public Commodity Pools : 1979-1990. *Journal of Futures Markets* 13(Oct 1993):799-820.

Irwin, Scott H. "Advances in Futures and Options Research." Chance and Trippi, eds., Greenwich: JAI Press, 1994.

Jensen, Michael C., "Some Anomalous Evidence Regarding Market Efficiency." *Journal of Financial Economics* 6(June 1978):95-102.

Kuan, C.M., and Halbert White. "Artificial Neural Networks: an Econometric Perspective." *Econometric Reviews* 13(1994):1-91.

Lapedes, A., and R. Farber. "Nonlinear Signal Processing Using Neural Networks: Prediction and System Modeling." Los Alamos National Laboratory Technical Report.

LeBaron, Blake. "Chaos and Nonlinear Forecastability in Economics and Finance." unpublished working paper, Department of Economics, University of Wisconsin-Madison, 1994.

Lee, Sang Bin, and Sung Moo Huh. "Futures Market Timing Ability with Neural Networks."

Levich, Richard M., and Lee R. Thomas, III. "The Significance of Technical Trading-rule Profits in the Foreign Exchange Market: a Bootstrap Approach." *Journal of International Money and Finance* 12(1993):451-474.

Lukac, Louis P., and B. Wade Brorsen. "A Comprehensive Test of Futures Market Disequilibrium." *Financial Review* 25(Nov 1990):593-622.

Masters, T. *Practical Neural Network Recipes in C++*, New York: Academic Press, 1993.

Mayfield, E.S., and B. Mizrach. "On Determining the Dimension of Real-Time Stock Price Data." *Journal of Business and Economic Statistics* 10(1992):367-374.

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

Nawrocki, David. "Adaptive Trading Rules and Dynamic Market Disequilibrium." *Applied Economics* 16(Feb 1984):1-14.

Neftci, Salih N. "Naive Trading Rules in Financial Markets and Wiener-Kolmogorov Prediction Theory: A Study of 'Technical Analysis'." *Journal of Business* 64(Oct 1991):549-571.

Oldfield, G., R. Rogalski, and R. Jarrow. "An Autoregressive Jump Process for Common Stock Returns." *Journal of Financial Economics* 5(1977):389-418.

Peters, E., "A Chaotic Attractor for the S+P 500." *Financial Analyst Journal* (March-April 1991):55-81.

Ripley, B.D. "Statistical Aspects of Neural Networks." in Barndorff-Nielsen, O.E., J.L. Jensen, and W.S. Kendall, eds., *Networks and Chaos: Statistical and Probabilistic Aspects*, London: Chapman & Hall, 1993.

Rumelhart, D.J. McClelland, and the PDP Group. "Parallel Distributed Processing." *Explorations in the Microstructure of Cognition, Vol. 1: Foundation*, Cambridge, MIT Press, 1986.

Sarle, Warren S. "Neural Network Implementation in SAS Software." *in Proceedings of the Nineteenth Annual SAS Users Group International Conference.* April, 1994a.

Sarle, Warren S. "Neural Networks and Statistical Models." *in Proceedings of the Nineteenth Annual SAS Users Group International Conference.* April 1994b.

SAS Institute Inc. *SAS/STAT User's Guide, Version 6, Fourth Edition, Volume 2.* Cary NC:SAS Institute Inc., 1989.

Scheinkman, J., and B. Lebaron. "Nonlinear Dynamics and Stock Returns." *Journal of Business* 62(1989):311-338.

Schwager, Jack D. *The New Market Wizards*. New York: Harper Collins Publishers Inc., 1992.

Schwager, Jack D. *Market Wizards*. New York: Harper & Row Publishers Inc., 1989.

Selfridge, O., R. Sutton, and A. Barto. "Training and Tracking in Robotics." *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, Los Angeles: Morgan Kaufman, 1, pp. 670-672.

Specht, D.F. "A Generalization Regression Neural Network." *IEEE Transactions on Neural Networks.* Nov. 2, 199, pp. 568-576.

Styblinski, M.A., and T.S. Tang. "A More Efficient Global Optimization Algorithm Based on Styblinski and Tang." *Neural Networks* 7(1994):573-574.

Trippi, Robert R., and Turban Efraim. *Neural Networks in Finance and Investing.* Probus, Chicago, Illinois, 1992.

Tsibouris, George C. "Essays on Nonlinear Models of Foreign Exchange." Unpublished PhD Dissertation, Department of Economics, University of Wisconsin-Madison, 1992.

Turner, Andrew L., and Eric J. Weigel. "Daily Stock Market Volatility : 1928-1989." *Management Science* 38(Nov 1992):1586-1609.

Werbos, P. "Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences." unpublished Ph.D. Dissertation, Harvard University, Department of Applied Mathematics, 1974.

White, Halbert. "Learning In Neural Networks: A Statistical Perspective." *Neural Computation* 4(Aug 1989):425-464.

White, Halbert. "Learning in Artificial Neural Networks: A Statistical Perspective." *Neural Computation* 1(1989):425-464.

White, Halbert. "Economic Prediction Using Neural Networks: The Case of IBM Daily Stock Returns." in *Proceedings of the IEEE International Conference on Neural Networks*, July 1988, pp. II451-II458.

Xu, Lei, Adam Krzyzak, and Alan Yuille. "On Radial Basis Function Nets and Kernal Regression: Statistical Consistency, Convergence Rates, and Receptive Field Size." *Neural Networks* 7(1994):609-628.

Yang, Seung-Ryong, and B. Wade Brorsen. "Nonlinear Dynamics of Daily Futures Prices: Conditional Heteroskedasticity or Chaos." *Journal of Futures Markets* 13(Apr 1993):175-191.

Zurada, Jacek M. *Introduction to Artificial Neural Systems*. St. Paul, MN:West Publishing Co., 1992.

APPENDIX

SAS MACRO USED TO ESTIMATE NEURAL NETWORK MODELS

```
/*-------------------------------------------------------------
```

By default, the model is:

x(nx)	independent variables (input)
a(nh)	bias for hidden layer
b(nx,nh)	weights from input to hidden layer
h(nh)	hidden layer
c	bias for output
d(nh)	weights from hidden layer to output
p(ny)	predicted values (output values)
y(ny)	dependent variables (training values)
r(ny)	residuals

$$\text{act}(z) = \frac{1}{1+\exp(z)} = \frac{\tanh(x/2)+1}{2}$$

$$h(j) = \text{act}\left(a(j) + \sum_{i=1}^{nx} b(i,j) * x(i)\right)$$

$$p(k) = c(k) + \sum_{j=1}^{nh} d(j) * h(j)$$

$$r(k) = y(k) - p(k)$$

note: k=1 above

The function act() is an _activation_ or _squashing_ function. The logistic function shown above is the most commonly used activation function, but many other functions can be used. Note that the output p[k] is _not_ squashed in this model. It is common practice to squash the output, but this often leads to various practical difficulties. In particular, people often forget to scale their dependent variables to lie between 0 and 1, and optimization tends to be more difficult when the output is squashed.

Initial parameter estimates (weights) may be read from a data set or generated randomly. Since there are no general methods for computing rational initial estimates, it is usually advisable to try several sets of random initial estimates to reduce the chance of being

trapped by a local minimum. This can be accomplished by running the
macro %itermodl.

----------*/

```
%macro nlinmodl( /* Simple example of fitting a neural network
                 (multilayer perceptron) using PROC MODEL */
  data=DATA,  /* Data set containing inputs and training values */
  xvar=XVAR,  /* List of variables that are inputs.
                 Do not use abbreviated lists with -, -- or :.
                 Do not use the variable name _o. */
  yvar=YVAR,  /* Output variable that has training values.
                 Do not use abbreviated lists with -, -- or :.
                 Do not use the variable name _o. */
  inest=,     /* Data set containing initial parameter estimates */
  hidden=2,      /* Number of hidden nodes */
  acthid=LOGISTIC, /* activation function applied to hidden nodes */
                /* the values can be LOGISTIC, or TANH */
  actout=LINEAR,   /* activation function applied to output nodes */
                /* the values can be LINEAR, LOGISTIC, or TANH */
  random=0,      /* Seed for random numbers for initial weights */
  bound=30,      /* Bound on absolute values of weights */
  output=_EST_,  /* Data set containing estimated weights and other
                 variables */
  outvars=SSE=SSE,
                /* Additional variables to put included in the
                  output data set */
  id=HIDD _ITER_, /* Variables to be included in the ouput data set
                  that are calculated within the proc nlin */
  maxiter=200,   /* Maximum number of training iterations allowed */
  nlinopt=METHOD=MARQUARDT G4SINGULAR NOPRINT CONVERGEOBJ=10E-16
        CONVERGEPARM=10E-16);
                /* Options to be passed to PROC NLIN */

  %if %index(&xvar &yvar,-:) %then %do;
    %put %qcmpres(ERROR: Abbreviated variable lists are not allowed
       by the NLINMODL macro.);
    %goto exit;
  %end;

  %****** check if activations are correct;
  %if %qupcase(&acthid) ^= LOGISTIC & %qupcase(&acthid) ^= TANH %then %do;
    %put %qcmpres(ERROR: Incorrect activation has been specified for the
       hidden layer.);
    %goto exit;
  %end;

  %if %qupcase(&actout) ^= LOGISTIC & %qupcase(&actout) ^=TANH &
     %qupcase(&actout) ^= LINEAR %then %do;
    %put %qcmpres(ERROR: Incorrect activation has been specified for the
       output layer.);
    %goto exit;
  %end;
```

```
%****** find number of inputs and put them in macro array;
%let nx=%xlstarr(xvar,_x);
%let ny=%xlstarr(yvar,_y,1);
%if &nx=0 OR &ny=0 %then %goto exit;
%global nh;
%if &hidden %then %let nh=&hidden;

%****** get default data set name;
%if %qupcase(&data)=_LAST_ %then %let data=&syslast;

%****** calculate or set initial estimates for parameters;
%if %bquote(&inest)= %then %do;   %** if no initial parameter data set;
  data _null_;
    %do _ih=1 %to &nh;
    call symput("n_a&_ih",ranuni(&random)-.5);
      %do _ix=1 %to &nx;
        call symput("n_b&_ix._&_ih",ranuni(&random)-.5);
      %end;
    %end;

    call symput('n_c',ranuni(&random)-.5);
    %do _ih=1 %to &hidden;
      call symput("n_d&_ih",ranuni(&random)-.5);
    %end;
  run;
%end;


%else %do;
  /* Make sure that inest data set contains only one observation */
  data temp; set &inest;
    if _n_=2 then stop;
    run;

  data &inest; set temp;
    run;

  proc datasets nolist;
    delete temp;
    run;

  /* Now set initial values for weights */
  data _null_; set &inest;
    %do _ih=1 %to &nh;
      call symput("n_a&_ih",_a&_ih);
      %do _ix=1 %to &nx;
        call symput("n_b&_ix._&_ih",_b&_ix._&_ih);
      %end;
    %end;
    call symput('n_c',_c);
    %do _ih=1 %to &hidden;
      call symput("n_d&_ih",_d&_ih);
    %end;
    run;
```

```
  %end;


%****** training;
proc nlin data=&data maxiter=&maxiter &nlinopt;

  %* parameter statement and random initial values;
  parameters %initparm ;

  %* define model;
  %* compute hidden layer;
  %do _ih=1 %to &nh;
    _sum1_&_ih=_a&_ih
    %do _ix=1 %to &nx;
      +&&_x&_ix*_b&_ix._&_ih
    %end;
    ;
    %* apply activation function to hidden node;
    %actfunc( _h&_ih,_sum1_&_ih,&acthid)
  %end;

  %* compute output;
  _sum2=_c
  %do _ih=1 %to &hidden;
    +_h&_ih*_d&_ih
  %end;
  ;
  %actfunc(_o,_sum2,&actout)

  %* apply activation function to output and define model statement;
  model &yvar=_o;

  bounds %bounds

  %* analytical derivatives;
  %deriv

  %* compute a few variables to be included in output data set;
  hidd=&hidden;

  %* include calculated variables (or others) in output data set;
  id &id;

  %* output statement;
  output out=&output parms=%parmout &outvars;

  run;

  /* output is to large, reduce to one observation */
  data temp(drop=date &xvar &yvar); set &output;
    if _n_=2 then stop;
    run;

  data &output; set temp;
```

```
    run;

  proc datasets nolist;
    delete temp;
    run;

%exit:
%mend nlinmodl;



%macro nlinmod2(  /* This macro executes the nlinmodl macro by calling it the
                  first time with gradient descent to find good starting
                  values and then calling it the second time using the Marquardt
                  algorithm */
  data=DATA,  /* Data set containing inputs and training values */
  xvar=XVAR,  /* List of variables that are inputs.
                  Do not use abbreviated lists with -, -- or :.
                  Do not use the variable name _o. */
  yvar=YVAR,  /* Output variable that has training values.
                  Do not use abbreviated lists with -, -- or :.
                  Do not use the variable name _o. */
  inest=,     /* Data set containing initial parameter estimates */
  hidden=2,      /* Number of hidden nodes */
  acthid=LOGISTIC, /* activation function applied to hidden nodes */
                /* the values can be LOGISTIC, or TANH */
  actout=LINEAR,   /* activation function applied to output nodes */
                /* the values can be LINEAR, LOGISTIC, or TANH */
  random=0,       /* Seed for random numbers for initial weights */
  bound=30,       /* Bound on absolute values of weights */
  output=_EST_,  /* Data set containing estimated weights and other
                 variables */
  outvars=SSE=SSE,
                /* Additional variables to put included in the
                  output data set */
  id=HIDD _ITER_, /* Variables to be included in the ouput data set
                  that are calculated within the proc nlin */
  initit1=50,
  initit2=50,
  maxiter1=150,
  maxiter2=200,
  initopt1=METHOD=GRADIENT NOPRINT SMETHOD=HALVE NOHALVE
        CONVERGEOBJ=10E-16 CONVERGEPARM=10E-16,
  initopt2=METHOD=GRADIENTNOPRINTCONVERGEOBJ=10E-16CONVERGEPARM=10E-16,
  nlinopt1=METHOD=MARQUARDT G4SINGULAR NOPRINT CONVERGEOBJ=10E-16
        CONVERGEPARM=10E-16,
  nlinopt2=METHOD=MARQUARDT G4SINGULAR NOPRINT CONVERGEPARM=10E-16
        CONVERGEOBJ=10E-16
        );


  /* First do gradient descent without line search to
    obtain better better starting values for Marquardt */
  data _null_;
```

```
  file print;
  put 'Beginning initialization with gradient descent';
  run;

%nlinmodl(data=&data, xvar=&xvar, yvar=&yvar, inest=&inest,
        hidden=&hidden, acthid=&acthid, actout=&actout,
        random=&random, bound=&bound, output=&output, outvars=&outvars,
        id=&id, maxiter=&initit1, nlinopt=&initopt1);

/* Now do marquardt minimization */
data _null_;
  file print;
  put 'Now starting training with marquardt algorithm';
  run;

%nlinmodl(data=&data, xvar=&xvar, yvar=&yvar, inest=&output,
        hidden=&hidden, acthid=&acthid, actout=&actout,
        random=&random, bound=&bound, output=&output, outvars=&outvars,
        id=&id, maxiter=&maxiter1, nlinopt=&nlinopt1);

/* check to see if converged */
data _null_; set &output;
  call symput('chkiters',_iter_);
  run;

/* if not converged and options call for second optimization */
%if &chkiters^=. & &nlinopt2^= & &initopt2^= %then %do;
  /* Do gradient descent with line search */
  data _null_;
    file print;
    put 'Have not reached convergence; beginning 2nd gradient descent';
    run;

  %nlinmodl(data=&data, xvar=&xvar, yvar=&yvar, inest=&output,
          hidden=&hidden, acthid=&acthid, actout=&actout,
          random=&random, bound=&bound, output=&output,
          outvars=&outvars, id=&id, maxiter=&initit2,
          nlinopt=&initopt2);

  /* Do marquardt minimization */
  data _null_;
    file print;
    put 'Beginning marquardt minimization second time';
    run;

  %nlinmodl(data=&data, xvar=&xvar, yvar=&yvar, inest=&output,
          hidden=&hidden, acthid=&acthid, actout=&actout,
          random=&random, bound=&bound, output=&output,
          outvars=&outvars, id=&id, maxiter=&maxiter2,
          nlinopt=&nlinopt2);

%end;

%else %do;
```

```
    data _null_;
      file print;
      put 'Model converged the first time';
      run;
  %end;

%mend nlinmod2;



%macro itermodl(    /* This macro calls the nlinmod2 macro repeatedly to
                    find the lowest minimum  */
  data=DATA,        /* Data set containing inputs and training values */
  xvar=XVAR,        /* List of variables that are inputs.
                  Do not use abbreviated lists with -- or :. */
  yvar=YVAR,        /* List of variables that have training values.
                  Do not use abbreviated lists with -- or :. */
  hidden=2,         /* Number of hidden nodes */
  acthid=logistic, /* activation function applied to hidden nodes */
  actout=linear,    /* activation function applied to output nodes */
                  /* the values can be LINEAR, LOGISTIC, or TANH */
  random=0,         /* Seed for random numbers for initial weights */
  bound=30,         /* Bound on absolute values of weights */
  output=_EST_,     /* Data set containing estimated weights */
  outvars=SSE=SSE,
                  /* Additional variables to put included in the
                    output data set */
  id=hidd _iter_, /* Variables to be included in the ouput data set
                    that are calculated within the proc nlin */
  initit1=50,
  initit2=50,
  maxiter1=150,
  maxiter2=200,
   initopt1=METHOD=GRADIENT    NOPRINT    SMETHOD=HALVE    NOHALVE
CONVERGEOBJ=10E-8
          CONVERGEPARM=10E-8,
  initopt2=METHOD=GRADIENT NOPRINT CONVERGEOBJ=10E-8 CONVERGEPARM=10E-8,
  nlinopt1=METHOD=MARQUARDT G4SINGULAR NOPRINT CONVERGEOBJ=10E-8
          CONVERGEPARM=10E-8,
  nlinopt2=METHOD=MARQUARDT G4SINGULAR NOPRINT CONVERGEPARM=10E-8
          CONVERGEOBJ=10E-8,
  tempdat=TEMPDAT, /* Data name for temporary output data set */
  endo=5,           /* number of restarts */
  outhist=_ITHIST_); /* data set containing iteration history */


   %let n=&endo;
   %let bestsse=;
   %let nextsse=;

   %do i=1 %to &endo;

   data _null_;
     iter=symget('i');
```

```
  file print;
  put 'Beginning iteration ' iter ' of itermodel';
  run;

%nlinmod2(data=&data, xvar=&xvar, yvar=&yvar, hidden=&hidden,
        acthid=&acthid, actout=&actout, random=&random, bound=&bound,
        output=&tempdat, outvars=&outvars, id=&id, initit1=&initit1,
        initit2=&initit2, maxiter1=&maxiter1, maxiter2=&maxiter2,
        nlinopt1=&nlinopt1, nlinopt2=&nlinopt2);

%if &i=1 %then %do;
  %* entering first do ;
  data &output(drop=&xvar &yvar); set &tempdat;
    if _n_=2 then stop;
    run;

  data _null_; set &tempdat;
    call symput('bestsse',sse);
    run;

  data temp1(keep=sse1-sse&n); set &tempdat;
    if _n_=2 then stop;
    sse1=sse;
    run;
%end;

%else %do;
  %* entering second do ;
  data _null_; set &tempdat;
    call symput('nextsse',sse);
    run;
  %if &nextsse < &bestsse %then %do;
    data &output; set &tempdat;
      if _n_=2 then stop;
      call symput('bestsse',sse);
      run;
  %end;

  %let ib= %eval(&i-1);
  data temp&i(keep=sse1-sse&n); merge temp&ib &tempdat;
    if _n_=2 then stop;
    sse&i=sse;
    run;

  %if &i=&endo %then %do;
    data &outhist; set temp&i;
      run;
  %end;
%end;

%end;

proc datasets nolist;
  %do _ov=1 %to &i;
```

```
      delete temp&_ov;
    %end;
    run;

%mend itermodl;



%macro netrun( /* this macro uses a data step to run a network that is already
                estimated */
  data=DATA,
  xvar=XVAR,
  yvar=YVAR,
  hidden=,
  acthid=LOGISTIC, /* activation function applied to hidden nodes */
  actout=LINEAR,   /* activation function applied to output nodes */
                   /* the values can be LINEAR, LOGISTIC, or TANH */
  predict=predout,
  inest=_EST_,
  out=_DATA_);

  %if %bquote(&hidden)= %then %do;
    %put ERROR: Number of hidden neurons is blank.;
    %goto exit;
  %end;

  %let nx=%xlstarr(xvar,_x);
  %let ny=%xlstarr(yvar,_y,1);

  %if &nx=0 OR &ny=0 %then %goto exit;

  %global nh;
  %if &hidden %then %let nh=&hidden;

  data temp1(drop=date); set &inest;
    if _n_=2 then stop;
    run;

  data temp2;
    if _n_=1 then do;
      do;
        set temp1;
        end;
      end;
    set &data;
    run;

  data &out(keep=date &xvar &yvar &predict); set temp2;
    %* compute hidden layer;
    %do _ih=1 %to &nh;
      _sum=_a&_ih
      %do _ix=1 %to &nx;
        +&&_x&_ix*_b&_ix._&_ih
      %end;
```

```
      ;
      %* apply activation function to hidden node;
      %actfunc( _h&_ih,_sum,&acthid);
    %end;

    %* compute output;
    _sum=_c
    %do _ih=1 %to &hidden;
      +_h&_ih*_d&_ih
    %end;
    ;
    %* apply activation function to output;
    %actfunc(&predict,_sum,&actout)
    run;

    proc datasets nolist;
      delete temp1 temp2;
      run;

%exit:
%mend netrun;



%*************************** UTILITY MACROS ******************************;

%macro xlstarr(_list,_array,size); %* convert list to array,
                                   return count;
  %local __i __n __temp;
  %if %bquote(&&&_list)= %then %do;
    %put ERROR: %UPCASE(&_list) is blank.;
    %let __i=0;
    %goto exit;
  %end;

  %if %bquote(&size)= %then %let __n=2000000000;
                       %else %let __n=&size;
  %do __i=1 %to &__n;
    %let __temp=%qscan(&&&_list,&__i,%str( ));
    %if %bquote(&__temp)= %then %goto break;
    %global &_array&__i;
    %let &_array&__i=&__temp;
    %*put &_array&__i=&&&_array&__i;
  %end;
%break:
  %let __i=%eval(&__i-1);

  %if %bquote(&size)^= %then %do;
    %if &__i>&size %then %put
      WARNING: More than &size items in %UPCASE(&_list)=&&&_list..;
    %else %do;
      %let __temp=&&&_array&__i;
      %do __i=&__i+1 %to &size;
        %global &_array&__i;
```

```
        %let &_array&__i=&__temp;
        %*put &_array&__i=&&&&_array&__i;
      %end;
    %end;
    %let __i=&size;
  %end;

%exit:
  &__i
%mend xlstarr;

%macro actfunc(out,in,act);        %* activation function;
  %if %qupcase(&act)=LINEAR %then %do;
    &out=&in;
  %end;
  %else %if %qupcase(&act)=LOGISTIC %then %do;
    if &in<-45 then &out=0;       %* avoid overflow;
    else if &in>45 then &out=1;  %*    or underflow;
    else &out=1/(1+exp(-&in));   %* logistic function;
  %end;
  %else %if %qupcase(&act)=TANH %then %do;
    if &in<-22.5 then &out=-1;    %* avoid overflow;
    else if &in>22.5 then &out=1; %*    or underflow;
    else &out=1-2/(1+exp(2*&in)); %* tanh function;
  %end;
  %else %do;
    %put ERROR: Unrecognized activation function "&act".;
    ACTIVATION???;
  %end;
%mend actfunc;

%macro initparm;
  %do _ih=1 %to &nh;
    _a&_ih=&&n_a&_ih
    %do _ix=1 %to &nx;
      _b&_ix._&_ih=&&n_b&_ix._&_ih
    %end;
  %end;

  _c=&n_c
  %do _ih=1 %to &hidden;
    _d&_ih=&&n_d&_ih
  %end;

%mend initparm;


%macro deriv;
  %if %qupcase(&actout)=LINEAR %then %do;
    der._c=1;
    %do _ih=1 %to &hidden;
      der._d&_ih=_h&_ih;
    %end;
  %end;
```

```
%if %qupcase(&actout)=LOGISTIC %then %do;
  der._c=_o*(1-_0);
  %do _ih=1 %to &hidden;
    der._d&_ih=_h&_ih*_o*(1-_0);
  %end;
%end;

%if %qupcase(&actout)=TANH %then %do;
  der._c=1-(_o)**2;
  %do _ih=1 %to &hidden;
    der._d&_ih=_h&_ih*(1-_0**2);
  %end;
%end;

%if %qupcase(&actout)=LINEAR & %qupcase(&acthid)=LOGISTIC %then %do;
  %do _ih=1 %to &nh;
    der._a&_ih=_h&_ih*(1-_h&_ih)*_d&_ih;
    %do _ix=1 %to &nx;
      der._b&_ix._&_ih=&&_x&_ix*_h&_ih*(1-_h&_ih)*_d&_ih;
    %end;
  %end;
%end;

%if %qupcase(&actout)=LINEAR & %qupcase(&acthid)=TANH %then %do;
  %do _ih=1 %to &nh;
    der._a&_ih=(1-(_h&_ih)**2)*_d&_ih;
    %do _ix=1 %to &nx;
      der._b&_ix._&_ih=&&_x&_ix*(1-(_h&_ih)**2)*_d&_ih;
    %end;
  %end;
%end;

%if %qupcase(&actout)=LOGISTIC & %qupcase(&acthid)=LOGISTIC
  %then %do;
  %do _ih=1 %to &nh;
    der._a&_ih=_h&_ih*(1-_h&_ih)*_d&_ih*_o*(1-_o);
    %do _ix=1 %to &nx;
      der._b&_ix._&_ih=&&_x&_ix*_h&_ih*(1-_h&_ih)*_d&_ih*_o*(1-_o);
    %end;
  %end;
%end;

%if %qupcase(&actout)=LOGISTIC & %qupcase(&acthid)=TANH %then %do;
  %do _ih=1 %to &nh;
    der._a&_ih=(1-(_h&_ih)**2)*_d&_ih*_o(1-_o);
    %do _ix=1 %to &nx;
      der._b&_ix._&_ih=&&_x&_ix*(1-(_h&_ih)**2)*_d&_ih*_o*(1-_o);
    %end;
  %end;
%end;

%if %qupcase(&actout)=TANH & %qupcase(&acthid)=LOGISTIC %then %do;
  %do _ih=1 %to &nh;
    der._a&_ih=_h&_ih*(1-_h&_ih)*_d&_ih*(1-_o**2);
```

```
      %do _ix=1 %to &nx;
        der._b&_ix._&_ih=&&_x&_ix*_h&_ih*(1-_h&_ih)*_d&_ih*(1-_o**2);
      %end;
    %end;
  %end;

  %if %qupcase(&actout)=TANH & %qupcase(&acthid)=TANH %then %do;
    %do _ih=1 %to &nh;
      der._a&_ih=(1-(_h&_ih)**2)*_d&_ih*(1-_o**2);
      %do _ix=1 %to &nx;
        der._b&_ix._&_ih=&&_x&_ix*(1-(_h&_ih)**2)*_d&_ih*(1-_o**2);
      %end;
    %end;
  %end;

%mend deriv;


%macro parmout;
  %do _ih=1 %to &nh;
    %str( _a&_ih)
    %do _ix=1 %to &nx;
      %str( _b&_ix._&_ih)
    %end;
  %end;

  %str( _c)
  %do _ih=1 %to &hidden;
    %str( _d&_ih)
  %end;

%mend parmout;


%macro bounds;
  -&bound< _c<&bound

  %do _ih=1 %to &hidden;
    ,-&bound< _d&_ih<&bound
  %end;

  %do _ih=1 %to &nh;
    ,-&bound< _a&_ih<&bound
    %do _ix=1 %to &nx;
      ,-&bound< _b&_ix._&_ih<&bound
    %end;
  %end;
  ;
%mend bounds;
```

VITA

Lonnie Hamm

Candidate for the Degree of

Master of Science

Thesis: TESTING A NEURAL NETWORK'S ABILITY TO PREDICT FUTURES PRICES

Major Field: Agricultural Economics

Biographical:

Personal Data: Born in Liberal, Kansas, On February 12, 1968, the son of Abie and Verna Hamm.

Education: Graduated from Hooker High School, Hooker, Oklahoma in May 1986; received Bachelor of Science degree in Accounting from Oklahoma State University, Stillwater, Oklahoma in May 1991. Completed the requirements for the Master of Science degree with a major in Agricultural Economics at Oklahoma State University in July 1995.

Experience: Raised on a farm near Hooker, Oklahoma; employed as a farm laborer during summers; employed by Oklahoma State University, Department of Agricultural Economics as a graduate research assistant, June 1992 to present.

Professional Memberships: American Agricultural Economics Association.