DEVELOPING A NEURAL NETWORK SYSTEM FOR FINANCIAL PREDICTION AND AN APPLICATION ON ISE

by

Oğuz ÇALIŞKAN

A thesis submitted to

Institute of Social Sciences

in partial fulfillment of the requirements

for the degree of

Master of Business Administration

Boğaziçi University

1997

DEVELOPING A NEURAL NETWORK SYSTEM FOR FINANCIAL PREDICTION AND AN APPLICATION ON ISE

by

Oğuz ÇALIŞKAN

APPROVED BY:

Doç. Dr. Vedat AKGİRAY

Prof. Dr. Özer ERTUNA

Doç. Dr. Atilla ODABAŞI

February 1997

ABSTRACT

DEVELOPING A NEURAL NETWORK SYSTEM FOR FINANCIAL PREDICTION AND AN APPLICATION ON ISE

by Oğuz ÇALIŞKAN

After a number of successful applications in image processing and recognition, neural networks have gained a wide-spread use and an admirable place in data processing field. Their ability in data mining, classification, generalisation and trend prediction is the key factor of which they have been extensively used in finance. As neural networks are good at learning non-linear relationships and at predicting non-random movements, they have been utilised especially in forecasting stock price movements. Even though most of the researches and the models gave unsatisfactory results, there are a few successful applications that encourage the use of neural networks in the prediction of stock price movements. In this study, it's explained how artificial neural networks are used as a method for financial forecasting. After giving a brief description of neural networks, every step of developing a neural network forecasting system is explained in detail. In this study, four models have been developed for the prediction of stock price movements, using multi-layer feed-forward neural networks. By presenting the long and short term trends of the end-of-week closing prices, weekly trade volume and the end-of-week market index to the network, the models are expected to predict the future price movements by analysing the past trends. At last, a model has been developed for the prediction of market index.

KISA ÖZET

YAPAY SİNİRSEL AĞLAR KULLANILARAK FİNANSAL ÖNGÖRÜ SİSTEMİ GELİŞTİRME VE İMKB'YE UYARLANMASI

Oğuz ÇALIŞKAN

Yapay sinirsel ağlar özellikle görüntü işleme ve tanıma alanlarında geliştirilen başarılı uygulamalardan sonra pekçok alanda kullanılmaya başlanılmıştır. Veri inceleme, sınıflandırma, genelleme ve yönelim öngörmedeki yetenekleri, sinirsel ağların finans alanında kullanılmasını sağlamıştır. Özellikle düzenli olmayan ilişkileri betimleyebilme ve rastgele olmayan hareketleri tahmin edebilme özellikleri yapay sinir ağlarının hisse senedi fiyat hareketlerinin öngörümünde kullanılmasını yaygınlaştırmıştır. Bu alanda yapılan araştırmaların ve geliştirilen modellerin çoğunlukla tatmin etmeyen sonuçlar vermesine rağmen az da olsa başarılı çalışmalar yapay sinir ağlarının hisse senedi fiyat hareketlerinin öngörümünde kullanılmasını cesaretlendirmektedir. Bu çalışmada bir finansal öngörme yöntemi olarak yapay sinir ağlarının nasıl kullanılacağı anlatılıyor. Sinirsel ağlar hakkında genel bir bilgi verildikten sonra, bu teknikle geliştirilen finansal öngörü sistemlerinin başarılı olmasındaki en önemli faktörün sağlıklı bir girdi-çıktı ilişkisi kurmak ve iyi bir ağ yapısı kurmak olduğu düşünülerek tasarınm aşamaları detaylı olarak açıklanıyor. Bu çalışmada yapay sinirsel ağlar kullanılarak dört model geliştiriliyor. Kapanış fiyatları, işlem hacmi ve endeks verilerinin uzun ve kısa vadeli yönelimleri girdi olarak verilerek modellerin eski yönelimleri kullanarak gelecekteki fiyat hareketlerini ve yönelimlerini öngörebileceği belirtiliyor. Son olarak endeksin haraketlerinin öngörümü için bir model geliştiriliyor.

TABLE OF CONTENTS

Page of Approval ii
Abstract iii
Kısa Özet iv
Table of Contents v
Table of Figures vi
1.Introduction 1
2.Neural Networks 4
2.1 Hidden Layers 5
2.2 A simple Weighted Network..The Perceptron 6
2.3 Perceptron Learning 7
2.4 Learning as gradient descent and backpropagation 8
2.5 Learning in Multilayer Networks 9
2.6 Second Order Algorithms 12
2.6.1 Quickprop 13
3.NN Design for Financial Forecasting 14
3.1 Defining the Relationship 15
3.2 Testing for Profitability 16
3.3 Organizing Input Data 17
3.4 Transformig And Preprocessing the Data 21
3.5 Neural Network Architecture 22
3.6 Training the Model 23
3.7 Validating The Model 24
3.8 Creating an Operational System 26
3.9 Retraining the Model 27
4.Literature Survey 28
5.Methodology and Analysis 31
5.1 Model 1 41
5.2 Model 2 43
5.3 Model 3 45
5.4 Model 4 47
5.5 A Model for the Prediction of Market Index 49
6.Conclusion 51
References 54

LIST OF FIGURES

Page

Figure 1 The structure of a simple multilayer neural network 4
Figure 2 A simple model for hidden nodes 5
Figure 3 A biological approach for neural networks 7
Figure 4 Learning process in a parametric system 7
Figure 5 Extended neural network to compute the error function 10
Figure 6 Local approximation of Quickprop 13
Figure 7 The output of model 1. 42
Figure 8 The output of model 2. 44
Figure 9 The output of model 3. 46
Figure 10 The output of model 4. 48
Figure 11 The output of Market Index Prediction Model. 50

LIST OF TABLES

Page

Table 1 The statistics of closing price 39
Table 2 The statistics of market index 39
Table 3 The statistics of trade volume..... 39
Table 4 The statistics of Model 1 41
Table 5 The statistics of Model 2..... 43
Table 6 The statistics of Model 3..... 45
Table 7 The statistics of Model 4 47
Table 8 The statistics of Market Index Prediction Model 49

1. INTRODUCTION

Today's business world is faced with the task of getting more from less, often without the resources to develop the improvements needed. This includes the use of information technology. Although lack of information is seldom a problem, real measurable improvements in knowledge gained from the information and the resulting decisions based on the information are hard to find.[1]

A powerful emerging technology can be used to efficiently process information to achieve greater knowledge and improved decision making. This technology is the field of artificial neural networks, commonly referred to as simply neural networks. Neural networks self-adapt to learn from information, providing powerful models representing knowledge about a specific problem.

Artificial neural networks are the result of academic investigations that involve using mathematical formulations to model nervous system operations. The resulting techniques are being successfully applied in a variety of everyday business applications.

Neural networks have received a lot of publicity recently with all the talk about being mathematical models that emulate the way a brain works. A more meaningful way to think of a neural network is a statistical method of modeling non-linear relationships.

Neural networks represent a meaningfully different approach to using computers in the workplace. A neural network is used to learn patterns and relationships in data. The data may be the results of a market research effort, the results of a production process given varying operational conditions, or the

decisions of a loan officer given a set of loan applications. Regardless of the specifics involved, applying a neural network is a substantial departure from traditional approaches.

Traditionally a programmer or an analyst specifically codes every facet of the problem in order for the computer to understand the situation. Neural networks do not require the explicit coding of the problem. For example, to generate a model that performs a sales forecast, a neural network only needs to be given raw data related to the problem. The raw data might consist of: history of past sales, prices, competitors' prices, and other economic variables. The neural network sorts through this information and produces an understanding of the factors impacting sales. The model can then be called upon to provide a prediction of future sales given a forecast of the key factors.

These advancements are due to the creation of neural network learning rules, which are the algorithms used to learn the relationships in the data. The learning rules enable the network to "gain knowledge" from available data and apply that knowledge in making key decisions.

Applications of neural networks are numerous. Many receive their first introduction by reading about the applications of the techniques in financial market predictions. Claims are made by several well-known investment groups that at least some of their technical analysis of financial markets and portfolio selection is performed with neural networks.

Optical Character Recognition (OCR) is another wide-spread (and growing) application that will soon touch all of our lives. Other successful applications of the techniques include: analysis of market research data, industrial process control, forecasting applications, and credit card fraud identification. A number of banks are using neural network - based systems to

control credit card fraud. These systems are able to recognize fraudulent use based on past charge patterns with greater accuracy than other available methods. [7]

Another example of using neural networks to improve decisions is in medical diagnosis. A neural network can be shown a series of case histories of patients, with a number of patient characteristics, symptoms, and test results. The network is also given the diagnosis for the case from the attending physician. The network can then be shown information regarding new patients and the network will provide a diagnosis for the new cases. This essentially creates a system containing the expertise of numerous physicians which can be called upon to give an immediate, real-time initial diagnosis of a case to medical personnel. [4]

As a general rule neural networks should be applied in situations where traditional techniques have failed to give satisfactory results, or where a small improvement in modeling performance can make a significant difference in operational efficiency. In specific terms, should use neural networks in cases where believed that more detail can be learned from data and where the information needs to be determined faster and with more efficiency. [1]

2. NEURAL NETWORKS

A neural network is a mathematical model that processes information and generates some form of response based upon the relationship or pattern identified within the data. Higher forms of life, such as humans, possess complex neural networks in the form of a nervous system. Neural networks also exist as computer-based systems. A neural network consists of a group of relatively simple units, called neurons, that work cooperatively toward solving complex tasks such as classification or prediction.

Neural Network techniques use a set of processing elements analogous to neurons in the brain. These processing elements are interconnected in a network that can then identify patterns in data as it is exposed to the data. In a sense, the network learns from experience just as people do. This distinguishes neural networks from traditional computing programs, that simply follow instructions in a fixed sequential order.

The structure of a neural network looks something like the following:

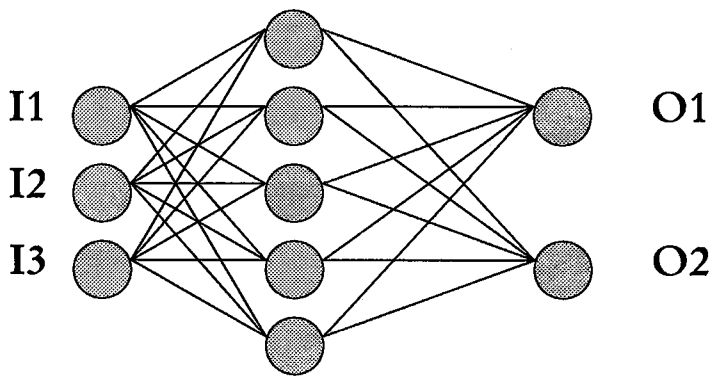
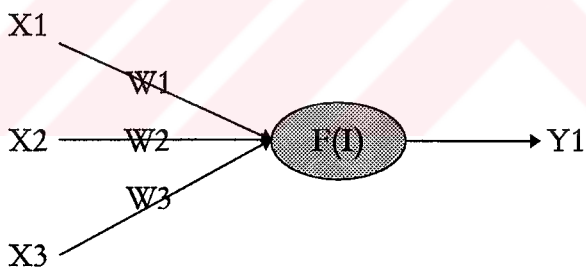


Figure 1 The structure of a simple multilayer neural network

The leftmost layer represents the input layer, in this case with 3 inputs, labels I1 through I3. In the middle is something called the hidden layer, with a variable number of nodes. It is the hidden layer that performs much of the work of the network. The output layer in this case has two nodes, O1 and O2 representing output values we are trying to determine from the inputs. For example, predict sales (output) based on past sales, price and season (input).

2.1. Hidden Layers

Each node in the hidden layer is fully connected to the inputs. That means that what is learned in a hidden node is based on all the inputs taken together. This hidden layer is where the network can learn interdependencies in the model. The following diagram provides some detail into what goes on inside a hidden node.



W1,W2,W3 are weights and X1,X2,X3 are inputs.

I = X1*W1 + X2*W2 + X3*W3

Y1 = Nonlinear transform of F(I)

Figure 2 A simple model for hidden nodes

Simply speaking a weighted sum is performed: X1 times W1 plus X2 times W2 plus X3 times W3. This weighted sum is performed for each hidden

node and each output node. This shows how interactions are represented in the network.

2.2. A Simple Weighted Network... The Perceptron

In 1958 Frank Rosenblatt proposed the perceptron, a more general computational model than McCulloch-Pitts units. The essential innovation was the introduction of numerical weights and a special interconnection pattern. In the original Rosenblatt model the computing units are threshold elements and the connectivity is determined stochastically. Learning takes place by adapting the weights of the network with a numerical algorithm. Rosenblatt's model was refined and perfected in the 1960s and its computational properties were developed by Minsky and Papert. [3]

The classical perceptron is a whole network for the solution of certain pattern recognition problems. A projection surface called the retina transmits binary values to a layer of computing units in the projection area. The connections from the retina to the projection units are deterministic and non-adaptive. The connections to the second layer of computing elements and from the second to the third are stochastically selected in order to make the model biologically plausible. The idea is to train the system to recognize certain input patterns in the connection region, which in turn leads to the appropriate path through the connections to the reaction layer. The learning algorithm must derive suitable weights for the connections. [29]

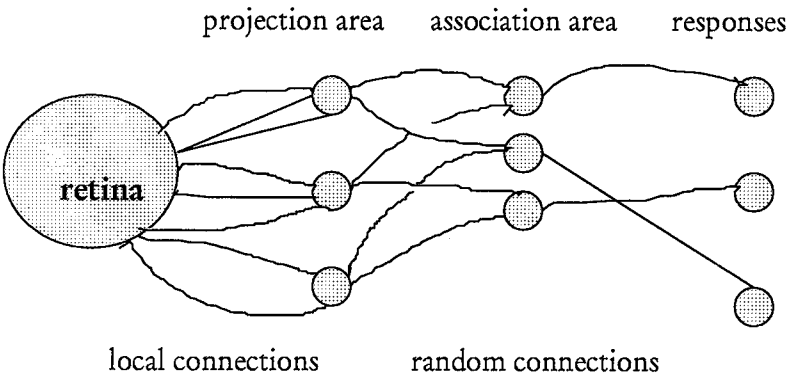


Figure 3 A biological approach for neural networks

2.3. Perceptron Learning

A learning algorithm is an adaptive method by which a network of computing units self-organizes to implement the desired behavior. This is done in some learning algorithms by presenting some examples of the desired input-output mapping to the network. A correction step is executed iteratively until the network learns to produce the desired response. The learning algorithm is a closed loop of presentation of examples and of corrections to the network parameters as shown in Figure 4. [3]

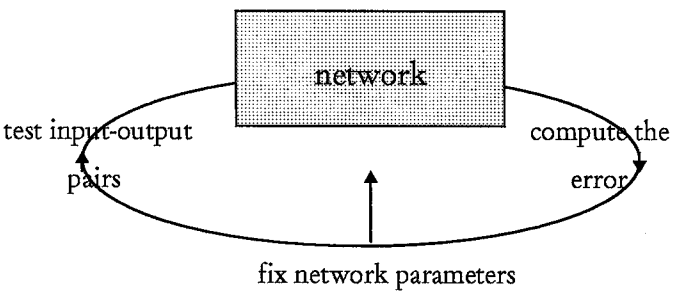


Figure 4 Learning process in a parametric system

2.4. Learning as Gradient Descent and Backpropagation Algorithm

When more parameters and more complicated topologies are considered, the computational effort needed for finding the correct combination of weights increases substantially. Backpropagation algorithm is a very popular learning method and capable of handling such large learning problems.

The backpropagation algorithm looks for the minimum of the error function in weight space using the method of gradient descent. The combination of weights which minimizes the error function is considered to be a solution of the learning problem. Since this method requires computation of the gradient of the error function at each iteration step, the error function must be continuous and differentiable. Obviously a kind of activation function other than the step function used in perceptrons must be used, because the composite function produced by interconnected perceptrons is discontinuous, and therefore the error function too. [17] One of the most popular activation functions for backpropagation networks is the *sigmoid* , a real function defined by the expression

$$s_c = 1/(1 + e^{-cx})$$

The constant c can be selected arbitrarily and its reciprocal $1/c$ is called the temperature parameter in stochastic neural networks. The shape of the sigmoid changes according to the value of c. Higher values of c bring the shape of the sigmoid closer to that of the step function.

The sigmoid's output range contains all numbers strictly between 0 and 1. Both extreme values can only be reached asymtotically . If the weights are $w_1,..,w_n$ and a bias $-\theta$, a sigmoidal unit computes for the input $x_1,..,x_n$ the output

$$\frac{1}{1 + \exp\left(\sum_{i=1}^{n} w_i x_i - \theta\right)}$$

A higher net amount of excitation brings the unit's output nearer to 1. The continuum of output values can be compared to a division of the input space in a continuum of classes. A higher value of c makes the separation in input space sharper. [2]

2.5. Learning in Multilayer Networks

A feed-forward neural network is a computational graph whose nodes are computing units and whose directed edges transmit numerical information from node to node. [8] Each computing unit is capable of evaluating a single primitive function of its input. In fact the network represents a chain of function compositions which transform an input to an output vector. The network is a particular implementation of a composite function from input to output space, called as network function. The learning problem consists of finding the optimal combination of weights so that the network function φ approximates a given function f as closely as possible. [29]

Lets consider a feed-forward network with n input and m output units. It can consist of any number of hidden units and can exhibit any desired feed-forward connection pattern and with a training set $\{(x_1,t_1),....,(x_p,t_p)\}$ consisting of p input-output pairs. When the input pattern x_i from the training set is presented to this network, it produces an output o_i different in general from the target t_i. Here the aim is to make the o_i and t_i identical for i=1,...,p, by using a learning algorithm, that is to minimize the error function of the network defined as

$$E = \frac{1}{2}\sum_{i=1}^{p}\|o_i - t_i\|^2$$

After minimizing this function for the training set, new unknown input patterns are presented to the network. The network must recognize whether a new input vector is similar to learned patterns and produce a similar output.

The backpropagation algorithm is used to find a local minimum of the error function. The network is initialized with randomly chosen weights. The gradient of the error function is computed and used to correct the initial weights. [17]

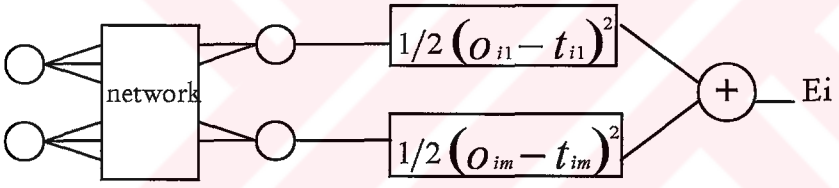


Figure 5 Extended neural network to compute the error function

The first step of the minimization process is to extend the network so that it computes the error function automatically. Each one of the j output units of the network is connected to a node which evaluates the function $\frac{1}{2}(o_{ij} - t_{ij})^2$, where o_{ij} and t_{ij} denote the jth component of the output o_i and of the target t_i. The output of the additional m nodes are collected at a node which adds them up and gives the sum Ei as its output. The network extension has to be built for each pattern t_i. A computing unit collects all quadratic errors and

outputs their sum $E_1+.....+E_p$. The output of this extended network is the error function E.

The weights in the network are the only parameters that can be modified to make the quadratic error E as low as possible. Because E is calculated by the extended network exclusively through composition of the node functions. Thus E can be minimized by using an iterative process of gradient descent,

$$\nabla E = \left(\frac{\partial E}{\partial w_1}, \ldots, \frac{\partial E}{\partial w_l} \right)$$

Each weight is updated using the increment

$$\nabla w_i = -\gamma \frac{\partial E}{\partial w_i} \qquad \text{for } i=1,\ldots,l$$

where γ represents a learning constant, i.e., a proportionality parameter which defines the step length of each iteration in the negative gradient direction.

In the backpropagation this increment is calculated in the same manner. Here the information fed into the subnetwork in the feed-forward step was $o_i w_{ij}$, where o_i is the stored output of unit i. The backpropagation step computes the gradient of E with respect to this input, i.e., $\partial E/\partial o_i w_{ij}$. Since in the backpropagation step o_i, is treated as a constant, it results in

$$\frac{\partial E}{\partial w_{ij}} = o_i \frac{\partial E}{\partial o_i w_{ij}}$$

If the backpropagated error at the j-th node is denoted by δ_j, then the partial derivative of E with respect to w_{ij} as

$$\frac{\partial E}{\partial w_{ij}} = o_i \delta_j$$

Once all partial derivatives have been computed, the gradient descent can be performed by adding to each weight w_{ij} the increment

$$\nabla w_i = -\gamma o_i \delta_j$$

This correction step is needed to transform the backpropagation algorithm into a learning method for neural networks. [29]

2.6. Second Order Algorithms

Although the backpropagation algorithm is a very popular and widely used learning method, it's a rather slow learning algorithm, which through malicious selection of parameters be made even slower. By using any of the well-known techniques of nonlinear optimization, it's possible to accelerate the training method with practically no effort. In this study Quickprop-a fast learning method- is used and here is explained.

Second order algorithms gets more information from the shape of the error function than the mere value of the gradient. A better iteration can be performed if the curvature of the error function is also considered at each step. [2]

2.6.1. Quickprop

This algorithm tries to take second order information into account but follows a rather simple approach: only one dimensional minimization steps are taken and information about the curvature of the error function in the update directions is obtained from the current and the last partial derivative of the error function in this direction.

Quickprop is based on independent optimization steps for each weight. A quadratic one-dimensional approximation of the error function is used. The update term for each weight at the k-th step is

$$\Delta^{(k)} w_i = \Delta^{(k-1)} w_i \left(\frac{\nabla_i E^{(k)}}{\nabla_i E^{(k-1)} - \nabla_i E^{(k)}} \right)$$

where it's assumed that the error function has been computed at steps (k-1) and k using the weight difference $\Delta^{(k-1)} w_i$, obtained from a previous Quickprop or a standard gradient descent step.

According to the value of the derivatives, Quickprop updates may become very large. This is avoided by limiting $\Delta^{(k)} w_i$ to a constant times $\Delta^{(k-1)}$.[29]

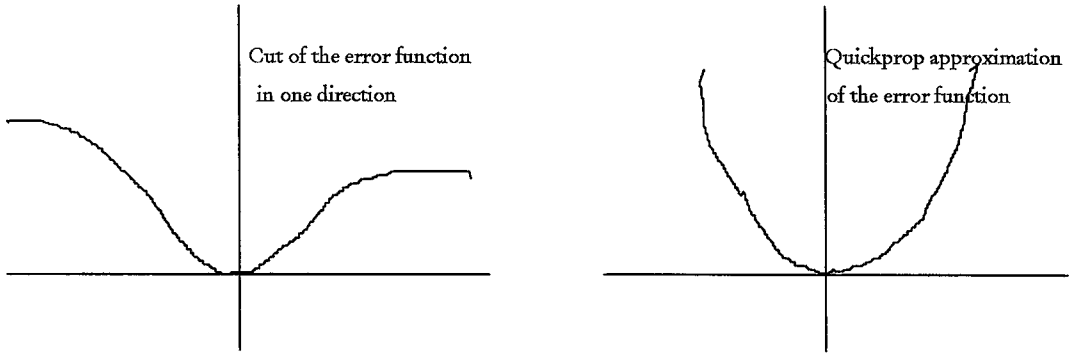


Figure 6 Local approximation of Quickprop

3. NN DESIGN FOR FINANCIAL PREDICTION

Several decisions need to be made before the neural network model is designed. First, it is necessary to select which market to trade stocks, bonds, mutual funds. Then the time frame for trading should be determined, whether intra-day; short-term positions, with decisions made intra-day; short-term positions, with decisions made after the market close; intermediate-term positions, with decisions made after the market close; or long-term positions of months to years. The trading style is the third decision to make. It might be chosen to be invested most of the time and use the model to determine which alternative has the most potential. Or timing buy, hold, and sell signals for a specific issue may be preferred. A third option might be to identify situations that occur infrequently, but that have very high reward-to-risk ratios. [32]

Each decision will result in unique requirements for network design, data collection, trading system signal generation, and trading system operation.

The basic processing unit of a neural network is the neuron (or node). Each neuron receives inputs, sometimes along with feedback, and produces an output signal.

A neural network typically has several layers. The input layer contains a neuron corresponding to each input (independent) variable. The output layer contains a neuron for each variable being predicted (dependent variable). While it is possible for a single neural network to predict more than one thing, it is usually better to design a network with exactly one output neuron.

The hidden layer contains neurons that are connected to both the input layer and the output layer. It is possible for a neural network to contain

multiple hidden layers, but a three-layer network will be adequate for most financial forecasting applications.

The development of a neural network model has three basic steps. First, a portion of the historical data available is processed by the neural network modeling tools as they search for and learn the general relationships among the variables. Next, a (usually different) portion of the historical data is tested to see how well the relationships hold. Finally, the network is used to generate investment signals from current data.

3.1. Defining the Relationship

A neural network is a mathematical model of the relationships between a dependent variable (output) and several independent variables (inputs).

Creating a neural network model consists of making many guesses about the underlying real-world relationship, then testing to see if the guess is sufficiently accurate or profitable. One implication of the statistical nature of financial relationships is that it will be difficult or impossible to know when the best model has been found. The dynamic nature of financial relationships also complicates the statistical modeling since market relationships can change over time. [34]

Design issues include the definition of what is to be predicted (the dependent variable) as well as the determination of which inputs (independent variables) will be used to make the prediction. These two areas should be defined as precisely as possible. For example, the dependent variable might be defined as the percentage change in ISE index over a one-week period of time, and the independent variables might be defined as volume and 5-day exponential moving averages of closing price.

If possible, all variables, independent and dependent, should be expressed in dimensionless terms. For example, it's better to use the price-to-earnings ratio, which is a dimensionless number, rather than earnings. Also, changes are usually better expressed as percentages than as differences.

In defining the dependent variable to be predicted, It's recommended that it be continuous (such as percentage of price change) rather than discrete (such as buy, hold, sell). Discrete variables, which can be arranged in a rank order, can sometimes be coded in such a way that they can be treated as continuous variables. For example, buy, hold, sell could be translated to +1, 0, -1.

It is easier to forecast the change in a moving average of daily closing prices than to forecast the change in raw daily closing prices. However, the signals that result from forecasting the change in a moving average will need to be carefully tested against the actual trading prices to see if the signaled trades are profitable. [5]

3.2. Testing for Profitability

After becoming familiar with the neural networks, it's a good idea to model a system that it's known works. This approach has several major advantages. First, the system its inputs, its outputs, some of its interrelationships, and the feel it has in given situations are well known. Second, the system is known to work, so it can concentrated on implementing the neural network version. Third, the rate of return under realistic conditions is known, so there is a benchmark against which to measure. [33]

It should be decided before proceeding any further if the results of trading this model are acceptable and consistent with the trading and investment plans.

These results are the upper limit of profitability for this system and model. Because of the high amount of short-term variability in the price of the issue being traded, the correlation coefficient between the ideal output of the model and the typical neural net model will probably be less than 0.50, and the associated profitability will be much less than the upper limit. [33]

3.3. Organizing Input Data

End-of-day data will be sufficient for models used to trade all time frames except intra-day. Intra-day trading will require tick-by-tick or very short duration bars of series such as price, volume, and open interest of the issue being traded and also any of the intermarket variables which are being used[34]

It is important that the same type of data should be used for predicting as for training. For example, long-term market predictions may use economic indicators. If the model includes monthly housing starts, use initial release figures for both training and prediction, do not use final revisions for training and initial releases for prediction.

The time period being modeled is limited to the period that has valid data for all independent variables. If most of the data starts in 1989, but one particular item begins in 1991, then there are three choices. One, the missing data for that one item should be collected. Two, the variable with the short data from the set of independent variables should be dropped. Three, everything should begin from 1991. The short data set should not be extended backward with simulated or constant data.

As necessary, data reported at longer intervals (say, monthly) and shorter intervals (say, daily) must be converted to the interval length which will be used in the model (say, weekly). All data must be arranged in time sequence

and be verified that all variables have exactly the same number of data points and that all dates agree. There must be no extra or missing values.

How many data points should be used? The short answer is as many as possible. The longer answer requires a bit of analysis and a word of caution. Before any modeling runs begin, the historical data must be partitioned into three data sets. A suggested approach is to partition the data as follows: 60% for training the model, 20% for testing the model periodically during training, and 20% for verifying the model with out-of-sample data. This last set is sometimes called the production data set. [9]

During the training of the neural network, the data in the training set will be thoroughly examined and the network will learn or generalize the relationship between the dependent and independent variables. At intervals, under the control of the network development system (the program) , the partially trained network will be presented with the independent variables in the test set and will make predictions of the dependent variable, which is known. The goodness of fit of the predictions will be measured. Based upon the fit, modifications to the network might be made and training continued using the training set. The fact that the test set was examined (perhaps many times) before the model reached its final state implies that the test set is partially incorporated into the training set.

Consequently, a final test of the finished network is made using the production data set. The production set should include at least one block of data longer than the prediction period. If, for example, the model predicts five days ahead, the production data set might include one or more 10-day periods. Providing that the goodness of fit on these periods is adequate, then the model might be useful. If not, further modifications to the model will be required, as will a new production data set.

Think of the total number of input neurons plus hidden neurons as similar to the number of degrees of freedom in a regression model. By keeping this total low relative to the number of observations in the data set, you reduce the likelihood that the model will memorize a relationship.

Since it is difficult to know specifically which variables will be important (statistically significant), there is a tendency to create a lot of independent variables and let the network sort them out. Just as in regression analysis, when the number of independent variables (columns of inputs) becomes large relative to the number of observations (rows of data in the training set), the fit for the training set increases and the memorization of details dominates the generalization. Try to have at least 10 to 20 times as many observations in the training set as the total number of input neurons plus hidden neurons. [19]

Two issues come into play in determining how many data points one should have: completeness of data and proper time frame. Data must be organized such that a balance exists between these two aspects.

On the one hand, try to gather as many observations into the training set as possible so that the model does not overfit to the data. The training set should include instances of the entire range of possible observations that are anticipated in the future. For example, in order to have the model signal an extraordinary situation similar to April 1994, there must be observations similar to that period in the training set. Another example would be if the network models asset allocation throughout periods of economic expansion and contraction. In that case, there must be data representing at least one (and preferably more) full cycle.

On the other hand, too much data might lead the network to reach the wrong conclusion. If the relationship being modeled changes as time passes, older observations contribute nothing (or, worse, wrongly) to the model. In order for the modeling process to be successful, the underlying real-world relationship must hold for the entire period being modeled.

In order to use many independent variables, there must be many observations so the model does not overfit the data. If many observations are used, the underlying relationship must remain unchanged. If that relationship is found, the model will be valid for a long period of time until something in the relationship changes. On the other hand, if the relationship is thought to change quickly, then the number of observations must cover a period of time no longer than the period over which the relationship can be profitably approximated. Limiting the number of observations limits the number of independent variables. And the guess regarding the relationship must be good. If the model approximates the underlying relationship for a short period of time, then it must be periodically retrained or rebuilt.

Consider the following example to get a feel for the kind of numbers a network might incorporate. Assume the following: daily data will be used, with about 260 trading days in a year; the underlying relationship being modeled has not changed greatly in 10 years; the next few years will not differ greatly from the past 10 years; daily data for all variables is available for the past 10 years; and one of the independent variables will be a 200-day moving average.

The total data set consists of 2,600 observations. Reserving the first 200 to initialize the moving average leaves 2,400 to partition into training, test, and production sets of 1,440, 480, and 480, respectively. The production set could be a single block of the most recent 22 months, or it could be selected, scattered

blocks. The test set could also be either a single block or scattered smaller blocks (as small as a single day).

If the number of observations in the training set should be 10 to 20 times the total number of neurons, about 70 to 140 neurons can be safely used. Assuming they are in the ratio of I : 2 x (SQRT I) : 1 for input layer, hidden layer, and output layer, respectively, where I is the number of input neurons, the number of neurons in the input layer can be up to about 55 to 115. [5]

3.4. Transforming & Preprocessing the Data

Neural networks rarely process raw data. The raw data typically must be transformed before giving it to the network. The purpose of a transformation is to increase the usefulness of the information contained in the time series being transformed. Typical transformations include calculating various ratios, smoothing the series by computing moving averages, scaling, and normalizing.

It is helpful to plot and visually examine the distribution of each variable. If the distribution has outliers, they will strongly affect the model. Trimming the distribution so that outliers are brought toward the mean will usually improve the model. One way this can be accomplished is by calculating the mean and standard deviation of each variable, then creating a new variable whose upper and lower limits are some number of standard deviations from the mean. Values such as 1.50, 1.67, or 2.00 standard deviations are useful. Recall from statistics that 2 standard deviations plus or minus from the average include about 95% of the range of a normally distributed variable. Try several values and look at the resulting distributions.

Because of the manner in which they are calculated, some data series that always fall within a well-defined range are self-trimming and will not require

further transformation. An example is the stochastic oscillator that always moves within a range of 0.0 to 1.0. [16]

Neural networks are extremely good at detecting explicit relationships calculated into the transformed variables. For example, if price levels are used rather than percentage changes and simple moving averages are included as transformed variables and if the test data set consists of a random selection of observations, then the network will learn the relationship between the raw price and the average price. The fit on the training set and the short periods in the test set will be excellent, but the predictions for longer periods of unseen data will be poor.

The process of imagining and creating transformations of time series is called preprocessing. The majority of the modeling effort is spent in this area.

Neural networks are tools that can be used equally well with fundamental data (such as commodity prices, interest rates, profit margins, and market share) and technical data (such as share price relative to its moving average). They are particularly powerful for combining fundamental, technical, and inter-market factors together in a single model. In some ways, neural networks can be thought of as generalizations of regression analysis. [31]

3.5. Neural Network Architecture

Determining the optimal number of layers and hidden neurons will require some trial and error. For most applications, one hidden layer is sufficient. The network requires enough hidden neurons to learn the general features of the relationship. With too many hidden neurons, it will begin to learn idiosyncrasies in the training set and will predict the test set poorly; with too few, it will fail to learn the general features and will fail to train on the

training set. The goal is to use as few neurons in the hidden layer as possible while still retaining the network's ability to learn the relationships among the data.

Assuming that there is only one output neuron, it is suggested to begin by setting the number of hidden layer neurons to the square root of the number of input layer neurons. If the model fails to train, increase the number of hidden layer neurons and try again. If the model trains well and validates poorly, decrease the number of hidden layer neurons and try again. When the model trains well and validates well, then the right number is found. If the model is successful, it will probably not be extremely sensitive to the number of hidden neurons. For example, if the model works well with 12 hidden neurons, it will probably also work nearly as well with 10, 11, 13, or 14. If the model to both train well and validate well is not found , then either additional (or different) independent variables are needed, or the dependent variable cannot be modeled (as would be the case if it were random).

It is popularly thought that neural networks can be presented with large amounts of data in many independent data streams and that the networks will determine what is important and ignore the rest. Unfortunately, it is very difficult to determine which variables are important and which are not. Consequently, it is difficult to prune out unnecessary input variables in order to make room for others. And, since cross-variable interactions are often important, it is difficult to test blocks of variables. [9]

3.6. Training the Model

During the training process, the goodness of fit between the actual target output and the value predicted by the network is measured. The training is an iterative process incorporating both the training set and the test set. The neural

net uses the training set of data, the first 60% of the time series, to learn the relationships between the input variables and the output variable. The goodness of fit is measured by running the model against both the training set and the test set, the next 20% of data in the time series. The information resulting from this process is used to guide the process of modifying the model and repeating the training process until a successful model is found.

The goodness of fit can be judged from several different perspectives. Depending on the trading system, the model to give the lowest average error or the model to give the most conservative worst error may be selected. Which of these is best can be determined only later by testing the trading signals that result from the model. To begin with, tell the neural network development system to minimize mean square error. [9]

3.7. Validating the Model

After the model successfully learns the training set and predicts the test set, apply it to the production set (the last 20% of data in the time series) and measure the goodness of fit for this set. This process is referred to as validation.

The validation step is extremely important. The key point in this step is to compare the goodness of fit that results for each of the three data sets.

The three data sets should be analyzed separately. If a functional relationship among the independent and dependent variables exists, the three will be similar in correlation and in scatter plot. With very noisy data, such as financial data, expect the training set to have the best fit, the testing set next best, and the production set the worst fit.

Three quick methods can be used for judging the goodness of fit. The first is to compare the mean square error reported by the program as it processes

each of the three data sets. The second is to compare the correlation between the network prediction and the associated target value (the known point that the network is trying to model) for each of the three data sets.

The third is to create a scatter plot for each data set. Plot the neural network prediction along the horizontal axis and the associated target value along the vertical axis. Perfect learning (a perfect fit) will be indicated by all the points falling on a line running from lower left to upper right, while no learning (a random fit) will be indicated by the points being distributed in a circular cloud. The more likely result is partial learning, which will be indicated by a well formed ellipse.

The next decision about the model's goodness of fit will probably be based on the model's success in predicting the production set. Choosing a production set that includes a block of recent data enables the modeler to verify the current validity of the trading signals and equates to traditional paper trading of a new system.

Unlike regression models, it will be difficult to determine what the functional relationship is among the independent variables in any neural network. Some neural network tools allow the weights associated with connections between neurons to be displayed. Relatively large (positive or negative) weights correspond to relatively important contributions. But the neural network modeling process is inherently non-linear, so straightforward interpretation of the weights or of the relationship, particularly related to the hidden neurons, is difficult. Worse, from the point of personal understanding, two versions of the same model that result from different starting points in the modeling process may have very different weight sets, yet both may give accurate predictions.

The ultimate goodness of fit indicator is the bottom line in the trading account. It is important that the risk and reward aspects of a neural trading model should be evaluated.

The buy and sell prices used in the evaluation of the model should be those that will be used in actual trading. If the model predicts the percentage change in stock price one week ahead and if signals are computed over the weekend using Friday's closing data, then the buy and sell price will probably be Monday's close. Using Friday's closing prices as the trading price will probably give an inaccurate result. [34]

3.8. Creating an Operational System

After the model has been successfully developed, it must be made operational and must be set up to give useful trading signals. That is, it must be integrated with whatever system is used for data acquisition.

In operation, data must be gathered and processed with the same frequency as signals will be generated. If end-of-day signals will be generated, data can be collected after the markets close.

A prediction module that examines the newly created row of inputs is then called to produce a prediction, and the prediction interpreted into a trading signal. If intra-day signals will be generated, all of this processing must take place in real time.

If the neural network is being used with fundamental data or technical data that is gathered less frequently (such as monthly or quarterly profitability figures) there is less urgency, but the important point remains that the

operation of a neural network model will be quite different than its development.

3.9. Retraining the Model

The neural network model needs to be retrained periodically. The frequency of this retraining depends on the ongoing agreement between the predictions made by the model and the target being modeled.

If the training set includes several years and the model predicts well for production sets of several months, then retraining will be necessary only a few times a year. On the other hand, if the model predicts well only a short time beyond the prediction period, then frequent retraining will be necessary.

The retraining process is essentially the same as the model development process, but it should be easier because the previous model establishes a good starting point. [27]

4. LITERATURE SURVEY

Artificial neural networks have initiated as having the structure of biological neural systems and offered an alternative computing system by simulating the neuron's information processing system.

Although the studies on neural networks have begun many years ago, the first mathematical model of artificial neurons was presented, in 1943, by W. McCulloch and W. Pitts. This model simply has a number of input lines, a computing unit and an output. Input signals multiplied by the weights on the input lines are summed and the result is compared with a certain threshold value. If the result dominates the threshold value, then the model activates a signal through the output line. [3]

In 1958, The perceptron, an improved version of McCulloch-Pitts model, were introduced by Frank Rosenblatt. The difference was the introduction of the numerical weights and a special interconnection pattern. The Perceptron was the first neural network to be produced commercially and used in image recognition problems. [29]

Minsky and Papert analyzed the features and limitation of the classical perceptron and ,in 1960, they introduced the famous book Perceptrons which is one of the best ever written on AI and set higher standards for neural network researches. [3]

John Hopfield, in 1982, proposed a new model named as Hopfield networks. In this model, Hopfield introdued the concept of the energy

function, with which the convergence properties of the networks could be more easily analyzed.

In 1984 there occurred a development in self-organizing neural networks. Teuvo Kohonen proposed a topology-preserving map. Kohonen Networks learn to create maps of the input space in a self-organizing way.

In 1985 The PDP group reintroduced the Backpropagation algorithm, which started a new era for neural networks. The backpropagation algorithm has been used widely in the training of neural networks.

Sejnowski and Rosenberg, in 1986, developed NETtalk, a backpropagation network that was able to synthesize speech of good quality without applying linguistic transformations. The authors used a backpropagation network composed of seven groups of 29 inputs, 80 hidden and 26 output units. The text to be pronounced by the system is scanned using a sliding window of seven characters. Each of the characters is coded as one of 29 possible letters. The network was expected to produce the correct phoneme for the pronunciation of the character in the middle of the window , taking into account the three characters of the of context to the left and to the right. After training, an unknown text is scanned. The speech generated is of comparable quality to that produced by the rule-based systems. [29]

In 1988 White used neural networks in stock price prediction. In his analysis, White aimed at predicting the future movements of the prices of IBM stock by decoding the nonlinear regularities in the past history of price movements. The analysis was based on the daily stock returns in the 1972-1980 period and a neural network of five inputs and five hidden units was designed.

White found the results disappointing and concluded that "the present neural network is not a money machine." [31]

Judd, in 1990, presented the most important theorems on the complexity of learning algorithms for neural networks. It has been shown that when the number of degrees of freedom and combinetional possibilities of evaluation elements goes beyond a certain threshold, the learning problems become np-complete.[29]

March, an AI Expert, developed a stock portfolio trading system in 1995. March described a method that can be used to adjust neural network weights in situations where there is no advance knowledge about the correspondence between the network input and output. The method, which is , partially based on genetic algorithm concepts, will work in any situation in which the target objective or profit function is stepwise instead of continuous. The validity of the method is illustrated using a trading system that has a neural network as one of its components. The system is designed to maximize the profits from trading a portfolio of diverse stocks using input derived from weekly price data. A number of tables and equity graphs are used to show how the system performed when it traded the training portfolio and two other portfolios that contained stocks that were not part of the training set. For this paper, Annual Essay Award was given to March in 1995. [18][19]

5. METHODOLOGY AND ANALYSIS

Neural network design has mainly two steps. The first and the most important one is to predefine the relationship between the dependent and the independent variables. The second step is to design the network architecture that can analyze and model this relationship. An incorrectly defined relationship may lead the network to the wrong generalizations and conclusions. In the same way, a network architecture which is not suitable for the defined relationship may give incorrect results even though a sound relationship has been constructed.

The Relationship Definition:

In the stock market prediction systems neural networks are used as a method for the technical and fundamental analysis. However, because the data used in the fundamental analysis are available at relatively long intervals as compared to the technical analysis for which the data are available even in seconds, and the learning algorithms of the neural networks are similar to the technical analysis methods, the neural network systems have focused on technical data.

In this study three different types of data are used. By using the end-of-week closing price and weekly trade volume of the stock, and the end-of-week market index, it's aimed to predict the next week's return of the stock. But as mentioned before, giving to the network all the data as they are will not make sense because processing the raw data is time-exhausting and mostly unreasonable. In this study, the data are constructed in the way that the network can be aware of and analyze the past long and short term trends of the stock. The end-of week closing prices were processed and injected into the analysis in order the network to be able to take into account the previous returns. To inform the network about the past trends of the stock, six

different returns were calculated. 13 weeks return will give to the network an information about the long term trend of the stock prices and about the past performance of the stock. In the same way, 1, 2, 3, 4 and 5 weeks returns were incorporated to inform the network about the short term movements. The network, by analyzing the past trends, will predict the future return of the stock. For instance, as seen in Table 1, there is a negative correlation between 1, 3, 4, 5 and 13 weeks lagged values of the closing price and the following week's closing price whereas a positive correlation exists between two weeks lagged values of the closing price and the actual output. Most probably, the network will learn this relation ,point by point, in a more systematic way and produce more accurate and reliable results. Since this study has aimed at predicting the sharp movement of the stock prices, the short term returns were extensively used. Furthermore, the last three weeks trade volume were incorporated. Here, by using the volume information, the network is supposed to validate the trend. As seen in Table 3, there is a positive correlation between the closing prices and the trade volume. The other input is the market index which gives hints to the network about the global trend of the market. A positive correlation of 0.75 exists between the market and the stock returns.

The Network Design:

Although a general information about neural networks and their theoretical reasoning are given before, a mathematical understanding of neural networks and the learning algorithm used in the analysis is necessary to develop a robust prediction system based on neural networks.

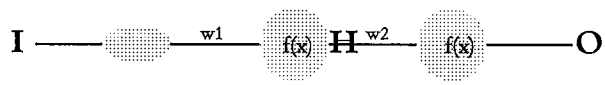
As a general concept, neural network design can be decomposed into three main parts: the network architecture, the learning algorithm and the data. All have equal importance and give a dynamism to neural net design.

The network architecture:

Neural networks contain layers which can be defined as the stages composed of identical computing units(nodes). Typically, a neural network should contain an input layer, one or more hidden layers and an output layer. In this study, the models are designed as having only one hidden layer. The number of nodes in the input and output layers are equal to the number of inputs and outputs, respectively. Finding the number of hidden nodes is not so easy and requires some trial and error. Since the number of hidden nodes affects the network's ability to learn, it's suggested, initially, to set the number of hidden nodes to twice the square root of the number of input nodes. In this study, the number of hidden nodes is determined by trial and error, that is, it's increased or decreased according to the training time and performance.

The Learning Algorithm:

The learning algorithm is an iterative process in which, input data set is presented to the network, an output is calculated and by comparing the actual output with the network output, an input-output map, which minimizes the difference between the target and the predicted values, is created. This process can be mathematically modeled as follows:



where

I : Network Input

O : Network Output

H : Hidden Node Output

f(x): Activation Function

w_1, w_2 : network weights

For each observation input I is introduced to the network and two outputs are calculated. The resulting outputs can be derived as

$H = f(w_1 I)$ for the output at the hidden node and

$O = f(w_2 H) \Rightarrow O = f(w_2 f(w_1 I))$ for the network output.

The prediction error for each observation can be computed by using the following equation

$E = \frac{1}{2} (O - T)^2$

where T denotes the target(actual) value.

To minimize the error function, the partial derivatives of the error function with respect to the weights are used together with the learning and momentum parameters. The following formulas were derived for only w_2.

$$\partial E / \partial w_2 = f(w_1 I) \cdot f'(w_2 f(w_1 I)) \cdot (f(w_2 f(w_1 I)) - T)$$
$$= H \cdot (O - T) \cdot f'(w_2 H))$$

To compute $f'(w_2 H))$ in terms of the network outputs, it's needed to specify an activation function. In this study, the activation function is the sigmoid,

$f(x) = 1 / (1 + e^{-cx})$

where 1/c is called the temperature parameter which is used to scale the x-axis and to change the input-output mapping at the nodes. Changing the value of temperature may change the speed of learning. In this study, the temperature is taken as 1. The derivative of the sigmoid is equal to

$f'(x) = f(x)\ (1 - f(x))$

This result simplifies the computation of the error function. That is,

$$\partial E/\partial w_2 = H.(O-T) \cdot f(w_2H).(1 - f(w_2H)) \quad \Rightarrow$$

$$\partial E/\partial w_2 = H.(O-T) .O. (1-O)$$

This result ,fortunately, shows that the weight updates can be done by numerical computation.

After computing all partial derivatives, the network weights are updated in the negative gradient direction by using the following equation

$$\Delta w_i = - \gamma\, \partial E/\partial w_i$$

where γ is the learning parameter.

Continuing with the above example, the weight update for w_2 is computed as

$$\Delta w_2 = - \gamma .H.O.(O-T) . (1-O)$$

The learning parameter γ is used to accelerate the learning process by improving the convergence ability of the network. The fact that, finding the best value of γ which guarantees convergence is not an easy task, leads to the utilization of another parameter, the momentum, which provides the search

process with a kind of inertia and avoids excessive oscillations in the weight updates. With momentum, the weight increment is evaluated as follows,

$$\Delta w_i = - \gamma \partial E / \partial w_i + \alpha \Delta w_i^P$$

where α is the momentum and Δw^P is the previous weight update.

As can be understood, the previous weight update is included to the calculation of the current weight update. The adjustment of both learning and momentum terms, to obtain the best possible convergence to a minimum of the error function, is done by trial and error.

The other parameters that should be mentioned are input and weight noises. A random noise increases the generalization ability of the network. Both input and weight noise helps the network avoid local minima. However, because the financial data are noisy in nature, no input and weight noise are injected into the models.

The Data:

The data set consist of three parts; training , test and production. During the training the data in the training set will be examined and the network will learn the relationship between the inputs and the output. At the same time, the test data are also presented to network at some intervals. This process provides the network with a foresight. The production is the period at which the model is used for trading in the real market.

In the analysis, a data set of 150 weekly observations is used.%67 of the data set is used for training whereas %33 is reserved for the test purposes. The

training covers the weeks in 1994 and 1995. The model were tested with the weekly data of 1996.

In the analysis part of the study ,four models were developed. The main difference between the models is their inputs. The type and the number of inputs were changed to improve the performance of the models. In the first model, trade volume, end-of-week closing price, 5-weeks return and 13 weeks return are used as independent variables to predict the dependent variable: following week's closing price of the stock. Before introducing the data to the network, they should be scaled or normalized in order to increase the utilization level of the information contained in the data. For the inputs of the first model, the following transformations were done.

i. Trade Volume of the Stock : Each observation was divided by the maximum volume in the training data. Thus, the trade volume was linearly scaled to the range 0 to -1.

$$V_{ti} = V_i / V_{maxi}$$

where V_t : trade volume after transformation

V : trade volume before transformation

V_{max}: maximum trade volume in the training set.

i : lag

ii. End-of-week closing price of the stock : The transformation of the closing price was done in two steps. First, the relative price difference was calculated for each observation.

$$R_i = (P - P_i)/P_i$$

And then each of the resulting values was divided by the maximum relative price difference.

$$R_{ti} = R_i / R_{maxi}$$

where P_i : i weeks lagged end-of-week closing price

R_{ti} : i-weeks return of the stock after transformation

R_{maxi} : maximum i-weeks return of the stock

iii. 13-weeks return of the stock : This input is scaled to the range -1 to 1 by using the following transformation

$$R_{13} = (P - P_{13})/P_{13}$$
$$R_{t13} = R_{13} / R_{max13}$$

In the second model, the inputs are trade volume, three consecutive end-of-week closing price, 13-weeks return and 13-weeks change in ISE Index

iv.13-weeks change in ISE Index : This is scaled to the range -1 to 1.

$$IR_{13} = (I - I_{13})/I_{13}$$
$$IR_{t13} = IR_{13} / IR_{max13}$$

where I_{13} : 13-weeks lagged value of ISE Index

In the third model, the number of inputs was increased to nine by adding three consecutive end-of-week values of ISE Index.

v. End-of-week closing value of ISE Index : These were scaled to the range -1 to 1 by using the same method in iv.

The last model is the most improved one. Although the type of inputs are the same as the third model the network was provided with more information about the past performance of the stock by the inputs : three consecutive trade volume, five consecutive closing price and five consecutive end-of-week value of ISE Index.

The above transformations guarantee that all the data are in the range -1 to +1. There are some outliers which are too large or small as compared to the average. Only a few of the input data points exceed the mean ±1 standard deviation area. These outliers were left unchanged because they represent the extraordinary situations which may happen in the future.

	R_t	R_{t-1}	R_{t-2}	R_{t-3}	R_{t-4}	R_{t-5}
mean	0,10	0,08	0,06	0,04	0,02	0,26
std deviation	0,25	0,23	0,21	0,17	0,12	0,46
io correlation	-0,12	-0,08	-0,06	0,03	-0,06	-0,06

Table 1 The mean and standard deviation of the transformed values of the closing price and the correlation between the inputs and the actual output .

	IR_t	IR_{t-1}	IR_{t-2}	IR_{t-3}	IR_{t-4}	IR_{t-5}
mean	0,08	0,06	0,05	0,03	0,01	0,19
std deviation	0,15	0,14	0,13	0,10	0,07	0,29
io correlation	-0,10	-0,09	-0,09	0,05	0,01	-0,10

Table 2 The mean and standard deviation of the transformed values of the market index and the correlation between the inputs and the actual output .

	V_t	V_{t-1}	V_{t-2}
mean	0,43	0,43	0,43
std deviation	0,25	0,25	0,25
io correlation	0,12	0,11	0,12

Table 3 The mean and standard deviation of the transformed values of the trade volume and the correlation between the inputs and the actual output.

Each one of the following models was designed by assuming that its predecessor's performance can be improved. So each network has different topologies and was trained using a different combination of inputs and training parameters.

5.1. Model 1

In this model, A 4-input-4-hidden layer network was used. Trade volume, closing price, 5-weeks return and 13-weeks return are used as independent variables to predict the following week's closing price of the stock. At the beginning of the training process, the model is run with different combinations of the learning parameter and momentum. A value of 0.2 for the learning parameter and 0.2 for the momentum were found as the best values.

This model assumes that the future return of the stock is related to the past returns and to the volume of the stock. 13-weeks return is taken as the long term return whereas 5-weeks return is taken as the short term return. As shown in Figure 7, the model is not successful at predicting the short term return. But it can be said that it gives a hint about long term trend. By looking at the test performance it's understood that the model is lagging the actual trend. Also can be seen that this model is not good at learning the details. The training performance can be improved by increasing the number of hidden nodes or including more inputs.

	Actual	
predicted	**up**	**down**
up	18	14
down	12	6

RMS	0.318
Correlation	0.634
Mean Predicted	0.015
Mean Actual	0.018
Predicted Stddev	0.119
Training RMS	0.085

Table 4 The statistics of model 1

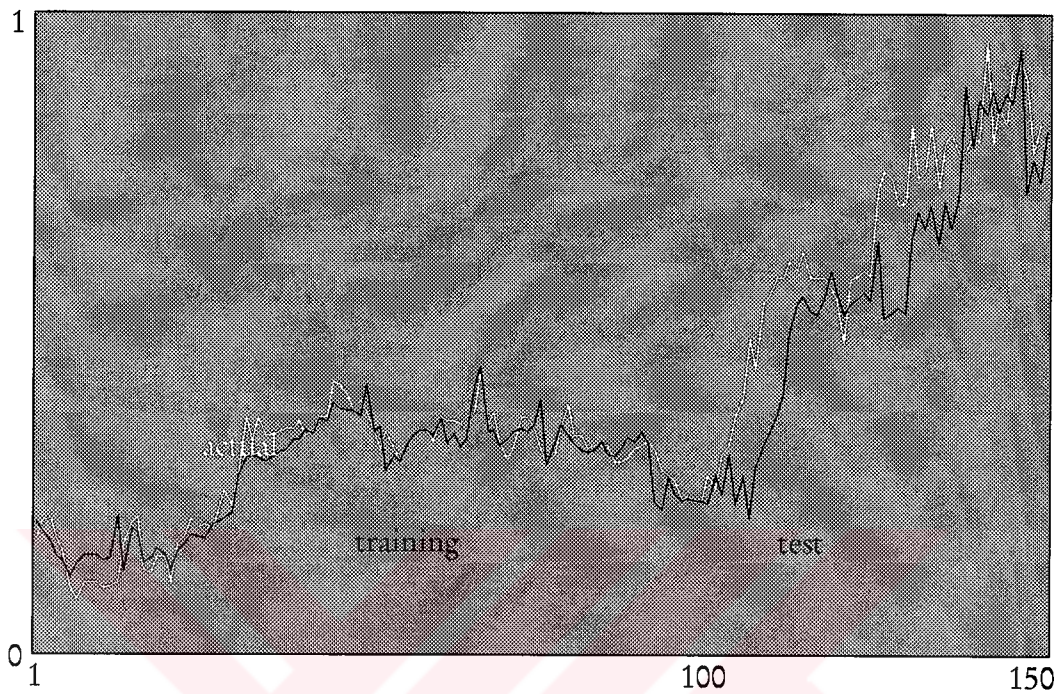


Figure 7 The output of model 1.

5.2. Model 2

In this model, A 6-input-6-hidden layer network was used. Trade volume, three consecutive closing prices, 13-weeks return of the stock price and 13-weeks return of ISE Index are used as independent variables to predict the following week's closing price of the stock. The value of the learning parameter was set to 0.3 while the momentum was left as 0.2.

This model differs from the previous one by that three more inputs are added and one is excluded. As compared to Model 1, training performance has improved. In the test period the model is fairly good at detecting upward trends. But there are some inconsistencies which may result from the inclusion of ISE Index. And also the inclusion of the previous data has improved the timing ability of the model as shown in Figure 8.

	Actual	
predicted	up	down
up	19	15
down	11	5

RMS	0.269
Correlation	0.671
Mean Predicted	0.012
Mean Actual	0.018
Stddev Predicted	0.124
Training RMS	0.071

Table 5 The statistics of model 2

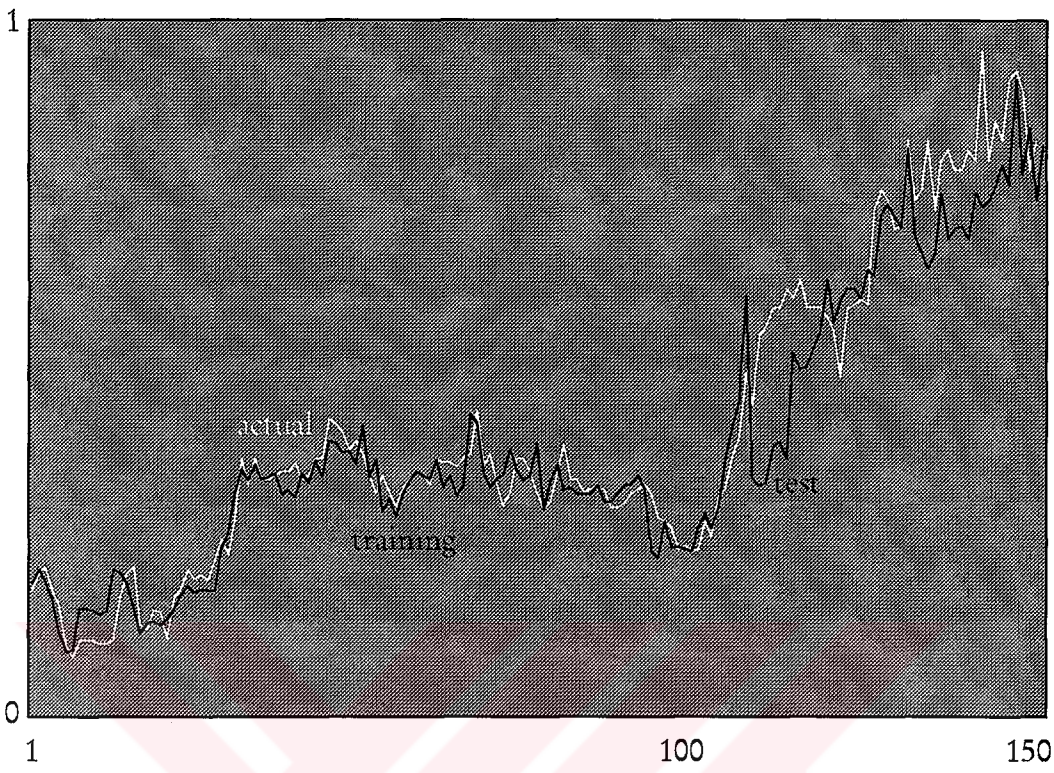


Figure 8 The output of model 2.

5.3. Model 3

In this model, A 9-input-8-hidden layer network was used. Trade volume, three consecutive end-of-week closing prices, three consecutive end-of-week value of ISE Index, 13-weeks return of the stock price and 13-weeks return of ISE Index are used as independent variables to predict the following week's closing price of the stock. The value of the learning parameter was set to 0.5 while the momentum was left as 0.2.

In this model the number of inputs are increased to nine by adding the last three weeks data of the ISE index -which may narrow the gap between the actual and the predicted data- assuming the correlation between the stock price and ISE index. As compared to Model 2, both training and test performance has improved. But the downtrends still can not be foreseen and the model cannot predict the sharp movements.

	Actual	
predicted	up	down
up	21	12
down	9	8

RMS	0.215
Correlation	0.707
Mean Predicted	0.017
Mean Actual	0.018
Stddev Predicted	0.122
Training RMS	0.058

Table 6 The statistics of model 3

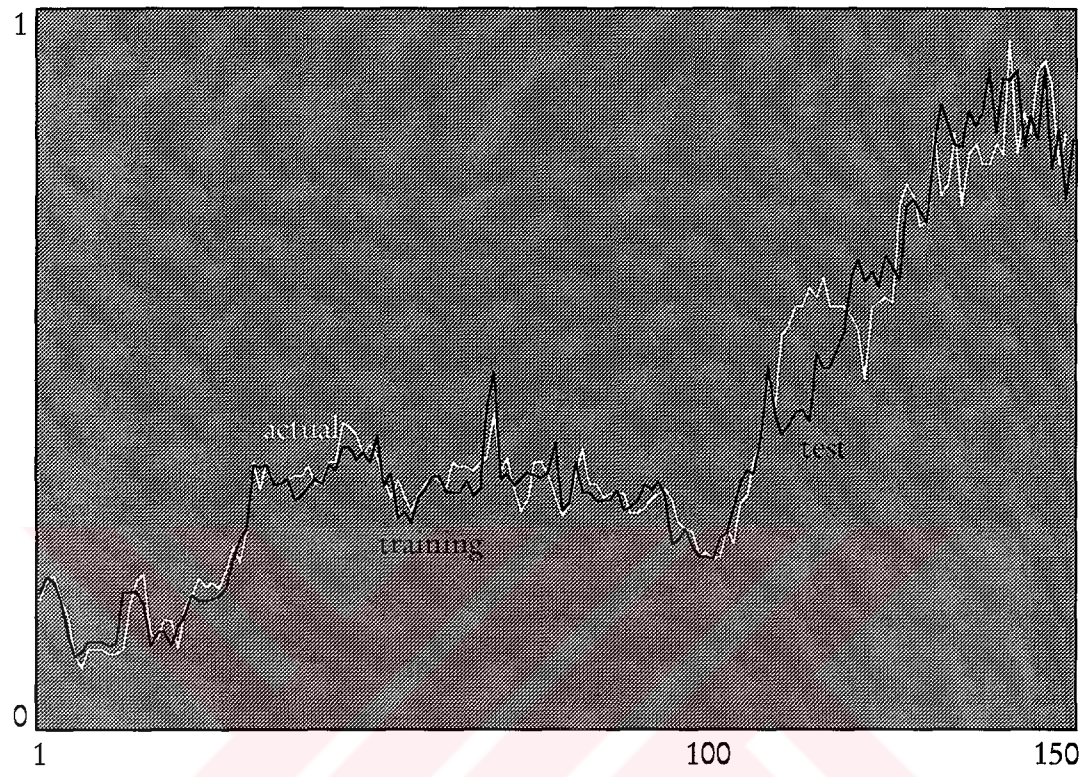


Figure 9 The output of model 3.

Model 4

In this model, A 15-input-9-hidden layer network was used. Three consecutive weekly trade volume, five consecutive end-of-week closing prices, five consecutive end-of-week values of ISE Index, 13-weeks return of the stock price and 13-weeks return of ISE Index are used as independent variables to predict the following week's closing price of the stock. The value of the learning parameter was set to 0.5 while the momentum was left as 0.4.

In the technical analysis it's believed that the movement of the stock should be supported by the change in the trade volume. In this model previous weeks' data of the volume are incorporated in order for the model to take into account the changes in the volume. And also additional four inputs (4th and 5th weeks data of the stock and ISE index) are added to give the model more information about the past performance of the stock and ISE index.

As shown in Figure 11, adding six more inputs to the network has improved the training performance resulting the best fit in the training. Although the performance of this model in test period is better than the previous models, this model is not for short term trading because of its inability to grab sharp movements, especially downs.

	Actual	
predicted	up	down
up	24	12
down	6	8

RMS	0.162
Correlation	0.801
Mean Predicted	0.016
Mean Actual	0.018
Stddev Predicted	0.097
Training RMS	0.052

Table 7 The statistics of model 4

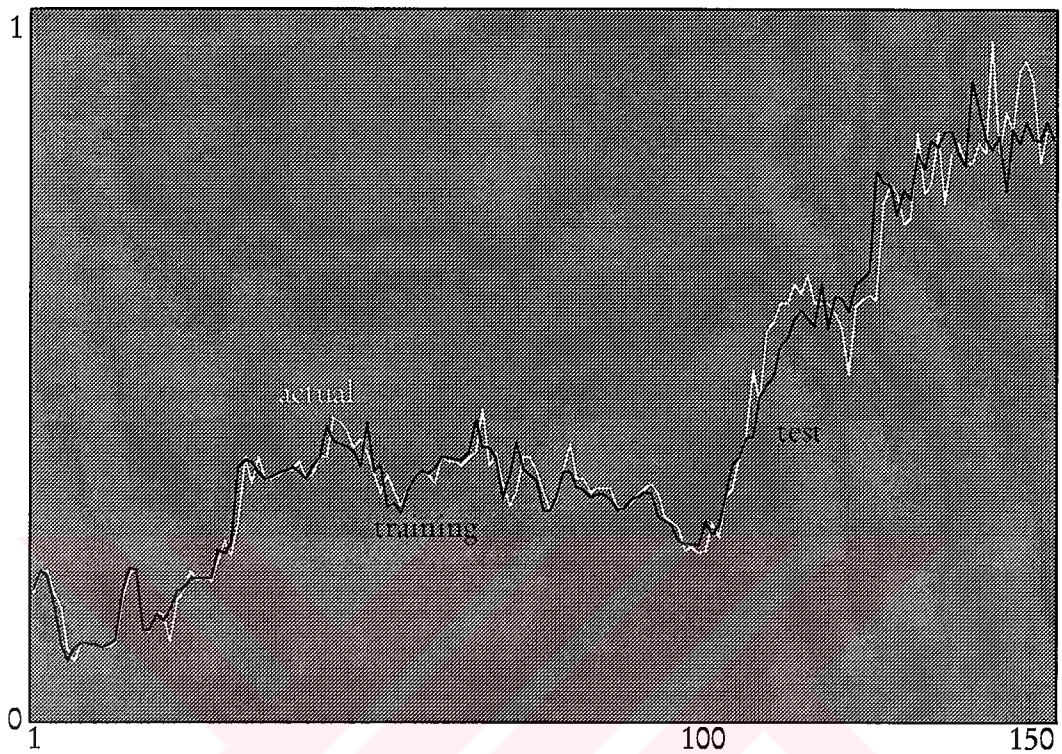


Figure 10 The output of model 4.

5.5. A Model for the Prediction of ISE Index

In this model, A 11-input-8-hidden layer network was used. Five consecutive weekly trade volumes, five consecutive end-of-week values of ISE Index and 13-weeks return of ISE Index are used as independent variables to predict the following week's ISE Index. The values of the learning parameter and the momentum were set to 0.5 and 0.2, respectively.

As shown in Table 8, The network predicted 20 of 30 up movements and only 10 of 20 down movements. As far as the means are concerned, the network can be said to perform very well. The training RMS is larger as compared to the test RMS, which can be interpreted as that the network is good at generalization. A better performance can be obtained by adding macroeconomic variables like interest rates, foreign exchange rates, inflation rate, to which the market index is highly sensitive.

	Actual	
predicted	up	down
up	20	10
down	10	10

RMS	0.183
Correlation	0.769
Mean Predicted	0.010
Mean Actual	0.011
Stddev Predicted	0.053
Training RMS	0.092

Table 8 The statistics of Market Index Prediction Model

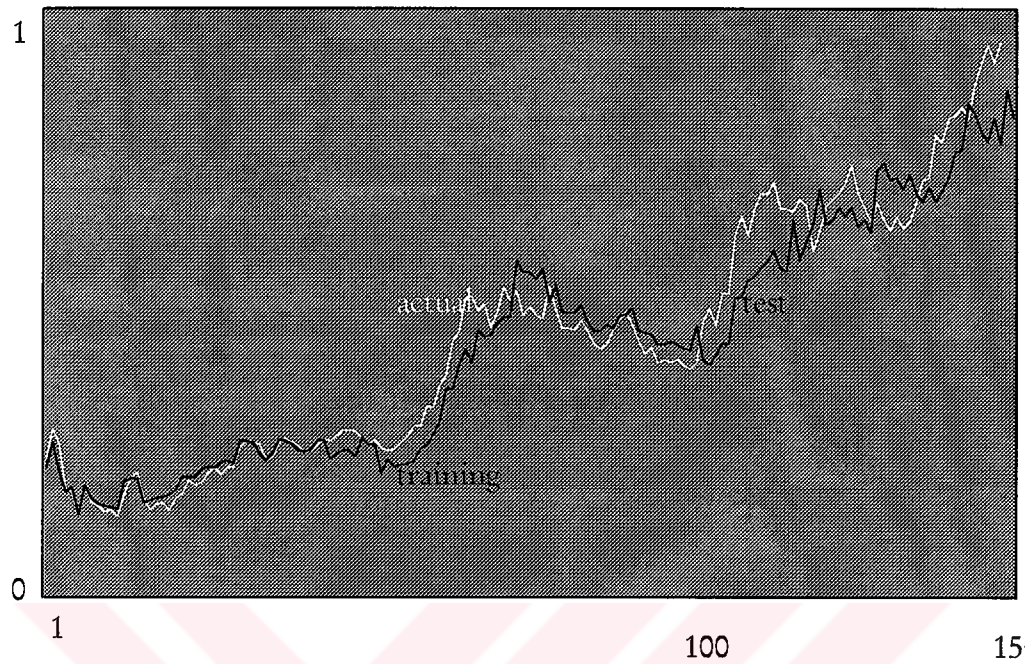


Figure 11 The output of market index prediction model.

6. CONCLUSION

The study gave a brief explanation of neural networks and Quickprop, and described the basic issues in NN design for financial prediction. In the last section four models were developed in order to predict the sharp movements in the stock price.

The first subject investigated in this study is the performance difference between standard backpropagation and quickprop learning methods. Both methods were used in the training of the model 1. It's found that, as the number of hidden nodes increase the training time difference between Quickprop and backpropagation increases.

In the analysis part of the study four models were developed. In all of the models - since this study proposes that the actions of the traders are hidden in the data and by analyzing the data these can be learned- only the past values of the stock and ISE index were used. The first model is the basic one in which short and long term returns, volume and closing price are used as input variables to predict next week's closing price. As shown in Figure 7, the model is good at generalizing, which makes sense in the long term but makes the model unreliable in the short term. The reason lying under the fact that the training performance of the model 1 is poor, is that the number of hidden nodes and inputs is less than the number of observations require. The second model is an improved version of the first. As expected a performance improvement in test and training was obtained. Also observed that the number of hidden nodes is sufficient in order for this model to grab the details. However there are some wide gaps between the predicted and actual values, that is, the prediction goes upward while the actual goes downward. In the third and the last model, the timing and test performance was concentrated on.

To narrow the gaps between the actual and the predicted values, previous values of the stock price, trade volume and ISE Index were added as new inputs. As shown in Figure 10, this is accomplished to an extent, except down movements. Although the last model is very successful as compared to first one, it cannot be used in short term trading for which it was designed. Because the model cannot learn when the stock goes down and cannot signal on time. These two factors make the model unreliable. Approximately the same results were obtained from the market index prediction model, but it can be used to predict the trend of the market if the index is not traded. Further improvement of the models was limited by the software used in the analysis. But, theoretically, a better performance can be obtained by statistically-robust data transformations and including additional inputs such as interest rates, foreign exchange rates, bond returns, earnings, technical indicators, etc. This is a part of an ongoing project- a portfolio management system- the author is currently working on.

This study showed that developing a robust and reliable neural network system for financial forecasting requires fully-dedicated time, experience, a good understanding of finance and stock market and an integrated software designed for financial forecasting. But given a steady increase in successful applications, neural networks offer substantial benefits.

Although neural networks are presently trendy — at least in some circles, the need to improve processes by doing things better and cheaper is more important than ever in today's competitive business world. Likewise, the desire to develop computer systems that can learn by themselves and improve decision-making is an ongoing goal of information technology. The neural network techniques used today may change but, the goal of developing systems that learn from past experience and lead to better business decisions will remain a high priority. Neural networks now represent one of the best practices in

achieving this goal. Furthermore, continued achievements toward this goal are likely to be inspired or generated from these technologies.

REFERENCES

1. AbuMostafa, Y., "Learning from Hints in Neural Networks", *Journal of Complexity*, Vol. 6, pp.191-198, 1996

2. Batiti, R., "First and Second Order Methods for Learning between Gradient Descent and Newton's Method", *Neural Computation*, Vol. 4, pp.141-166, 1992

3. Baum, E., "The Perceptron Algorithm is Fast for Non-malicious Distributions", *Neural Computation*, Vol. 2, pp.248-260, 1990

4. BioComp, "A Neural Indicator for S&P 500 Index", *BioComp Inc.*, FTP Document, 1996

5. California Scientific Software, "Neural Network Design and BrainMaker : A neural Net Software", *California Scientific Software* ,FTP Document ,1996

6. Connor , J. and L. Atlas , "Recurrent Neural Networks and Time Series Prediction ", *IEEE 1991*, Vol. I, pp.301-306, 1991

7. Fukushima , K.and N. Wake , "Handwritten Alphanumeric Character Recognition by the Neocognitron ", *IEEE Transactions on Neural Networks* , Vol. 2, No:3, pp.356-365, 1991

8. Harnik K. and M.Stinchcombe and H. White , "Multilayer Feedforward Networks are Universal Approximators", *Neural Networks,* Vol. 2, pp.359-366

9. Haussler, D. and E. Baum, "What Size Networks Gives Valid Generalization", *Neural Computation*, Vol. 1, pp.151-160, 1989

10. Holt, J. and T. Baker, "Backpropagation Simulations Using Limited Precision Calculations ", *IEEE 1991*, Vol. II, pp.121-126, 1991

11. Hwang, J. and M.Maechler and D.Martin and J.Schimart, "Regression Modeling in Backpropagation and Projection Pursuit Learning", *IEEE Transactions on Neural Networks*, Vol. 5, No. 3, 1994

12. Hwang, K. and F. Briggs, *Computer Architecture and Parallel Processing*, Addison Wesley ,1985

13. Johnson, L. "A Neural Network Forecasting Model for S&P 500 Index", *Computerized Investing*, FTP Document ,1996

14. Karaali, O. and M.M. Carhart, "Neural Network Forecasting of Mutual Fund Returns", The Sixteen Annual International Symposium of Forecasting, Istanbul Turkey, June 1996

15. Kung, S.Y., "Parallel Architectures for Artificial Neural Nets ", *IEEE 1988*, Vol. II, pp.165-172, 1988

16. Lawrence, Jeannette. "Designing Back Propagation Neural Networks: A Financial Predictor Example," NeuroVe$t Journal, Vol.2, No.1, Jan/Feb 1994, pp. 8-13.

17. Lee C. and T. Rostar , "Local Minima and Backpropagation", *IEEE 1991*, Vol. II, pp.173-176, 1991

18. March, David L. "Supervised Evolution of the Neural Trader Component of a Stock Portfolio Trading System (Part 1)," *NeuroVe$t Journal*, Vol.3, No.4, Jul/Aug 1995, pp. 7-12.

19. March, David L. "Supervised Evolution of the Neural Trader Component of a Stock Portfolio Trading System (Part 2)," *NeuroVe$t Journal*, Vol.3, No.5, Sep/Oct 1995, pp. 18-21.

20. Mead, C. , *Analog VLSI and Neural Systems*, Addison Wesley ,1989

21. Miller, D.P., "Better Investment Decision Making," *Investor's Advantage*, FTP Document, 1996

22. Nelson, M.N. and W.T. Illingworth, *A Practical Guide Neural Nets*, Massachusetts: Addison Wesley ,1991

23. Nicky, A., " Artificial Neural Network Algorithms in MATLAB", Kansas State University, FTP Document , 1996

24. Odom M. and R. Sharda , "A Neural Network for Bankruptcy Prediction", *IEEE 1990*, Vol. II, pp.163-168, 1990

25. Onuk, C. "Neural Network Applications to Economics and Finance", Course Papers, Bogaziçi University,1996

26. Özel, Ç., "Analysis and Synthesis of Feedforward Neural Networks Using Discrete Affine Wavelet Transformations", MS Thesis, Bogaziçi University, 1996

27. Pandy, Howard B. "Neural Network-based Trading System Design: Prediction and Measurement Tasks," *NeuroVe$t Journal*, Vol.2, No.5, Sep/Oct 1995, pp. 26-32.

28. Reeves, C.R., N.C. Steele, R.F. Albrecht, *Artificial Neural Nets and Genetic Algorithms,* Vienna : Springer Verlag,1996

29. Rojas, R., *Neural Networks: A systematic Introduction*, Berlin : Springer Verlag,1996

30. Trippi, R.R. and E.Turban , *Neural Networks in Finance and Investing*, Probus Publishing ,1993

31. White, H. , "Economic Prediction Using Neural Networks: The Case of IBM Daily Stock Returns," *IEEE 1988*, Vol. II, pp.451-458, 1988

32. Wunch, D., K. Bergerson , "A Commodity Trading Model Based on a Neural Network Expert System Hybrid," *IEEE 1991*, Vol. I, pp.289-293, 1991

33. Venetsanopoulos A. and N. Karayiannis, *Artificial Neural Networks: Learning Algorithms, Performance Evaluation and Applications,* Boston , 1993

34. Yoda, M. and T. Kimato, "Stock Market Prediction System with Modular Neural Networks, " *IEEE 1990*, Vol. I, pp.2-6, 1990

35. Z Solutions, "Introduction to Neural Networks", Z Solutions, LLC,FTP Document, 1995