FORECASTING STOCK MARKET RETURN USING ARTIFICIAL NEURAL NETWORKS

by

Volkan Doğan

B.S. in C.M.P.E, Boğaziçi University, 2002

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in System Control Engineering

Boğaziçi University

2006

ACKNOWLEDGEMENTS

I would like to thank my thesis supervisor Prof. Fikret S. Gürgen for his endless support, valuable suggestions, guidance and the patience he has shown throughout this research.

I would also like to thank my thesis co-supervisor Assoc. Prof. Nesrin Okay for her guidance in the field of economics, and for sparing her time for our continuous discussions in statistics.

I would also like to express my special thanks to Prof. H. Levent Akın, Prof. Ahmet Ademoğlu and Dr. Ayşe Başar Bener for their participation to my thesis jury.

Finally, I would like to express my deepest thanks and love to my lovely honey and my family.

ABSTRACT

FORECASTING STOCK MARKET RETURN USING ARTIFICIAL NEURAL NETWORKS

Stock market's return prediction is an important concept in emergent markets like Turkey and Brazil. In this thesis, I used artificial neural networks architecture to model and predict stock markets. Istanbul stock exchange indices National-100 and National-30 are used for domestic market, Brazilian stock exchange index, BVSP, is used as the international market.

Input space is divided into clusters with statistical clustering technique Expectation Maximization. ANN's structure Mixture of Experts is used and local experts are assigned to each cluster. While local experts are learning their region of interest, in parallel, gating experts combine the outputs of them to model overall structure. Besides, future returns are predicted based on patterns obtained from past trainings.

In financial time series modeling using ANN, I used past returns in simulations. Since we know two investigated markets are highly volatile and chaotic, the volatility factor is added to the analysis as well. Volatility calculated from RiskMetrics™ [30] is also included in the models to capture different dynamic features of the data. Results of our simulations are evaluated by using previously defined and widely accepted performance measures.

Another interesting result is gained during our simulations; two countries of different macroeconomic structures show similarities. Interaction between Turkish and Brazilian stock markets is not surprising; during the financial liberalization of the 1980-1990s, Turkey imported many constitutional laws from Latin American economies especially from Brazil.

ÖZET

HİSSE SENEDİ GETİRİSİNİN YAPAY SİNİR AĞLARI İLE TAHMİN EDİLMESİ

Hisse senedi getirilerinin tahmin edilmesi finans dünyasında özellikler Türkiye ve Brezilya gibi gelişen ekonomilerde önemli bir kavramdır. Bu tez çalışmasında, yapay sinir ağları kullanılarak hisse senedi piyasasının modellenmesi ve tahmin edilmesi yapılmıştır. Menkul Kıymetler Borsası (İMKB) Ulusal 100 ve Ulusal 30 endeksleri yerel market için, Brezilya ulusal endeksi uluslar arası piyasa için kullanılmıştır.

Girdi uzayı statiksel kümeleme tekniği olan Expectation Maximization ile kümelere bölünmüştür. Yapay sinir ağı yapısı olan Mixture of Experts kullanılmış, her kümeye yerel bir eksper atanmıştır. Yerel uzmanlar kendi ilgi alanlarını öğrenirken, geçit eksperleri lokal eksperlerin çıktılarını işleyerek esas çıktıyı oluşturacak şekilde birleştirmişlerdir. Bunun yanı sıra aynı teknik gelecek getirilerin hesaplanmasında kullanılmıştır.

Finanssal zamana serilerinin modellenmesinde sadece geçmiş getiri değerleri kullanılmamıştır. Araştırılan piyasaların fazlaca değişebilir olmasını göz önünde bulundurarak değişkenlik faktörü de modellere eklenmiştir. RiskMetrics™ [30] kullanılarak hesaplanan değişkenlik mevcut veriyi daha iyi ifade edebilmek için dahil edilmiştir. Deneylerimizin sonuçları genel olarak kabul edilmiş ve hazırlanmış performans ölçülerine göre değerlendirilmiştir.

Deneylerimizden elde ettiğimiz bir diğer ilginç sonuç ise; değişik makro ekonomilere sahip ülkelerin hisse senedi piyasalarının arasındaki ilişkidir. Bu ilişki şaşırtıcı değildir çünkü 1980-1990 yılları arasında Türkiye mali piyasaların serbestleşmesi süreci içinde önemli bir miktar düzenlemeleri Latin Amerika ülkelerinden almıştır, bunlarında başında Brezilya gelmektedir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS/ABBREVIATIONS

C_h	Context Layer units
D_t	Value of divisor at period t
$\boldsymbol{d_k}$	Desired output
$\boldsymbol{E}$	Total error of the network through all patterns
$\boldsymbol{E^n}$	Sum over output units of the squared difference between the desired output $\boldsymbol{d_k}$ and the network output $\boldsymbol{y_k}$
$E[.\|.]$	Conditional expectation operator
FW_{it}	Floatation weight of the stock i at period t
$\boldsymbol{g(x)}$	Tanh function
$\varepsilon - Increase$	Increasing trend benchmark
$\boldsymbol{H\varepsilon}$	Hit rate for $\varepsilon - Increase$
$\boldsymbol{H_0}$	Naïve predictor's hit rate performance
$\boldsymbol{H_R}$	Hit Rate
H_R^+	Positive Hit Rate
H_R^-	Negative Hit Rate
H_{RD}	Directional Hit Rate
h_t	Conditional variance (Volatility)
$\boldsymbol{net_k(t)}$	Weighted sum of the inputs
N_{it}	Total number of shares outstanding of the stock i at period t
$\boldsymbol{N_p}$	Net Profit
$\boldsymbol{pdf}$	Probability Density Function
$\boldsymbol{p_{high}}$	Highest traded price during the day
P_{it}	Closing price of the stock i at period t
$\boldsymbol{p_{low}}$	Lowest traded price during the day
$\boldsymbol{P_{\Re}}$	Mean profit per trade

p_t	Closing price at the end of the day
r_s	i.i.d. variable
r_t	One-step return (continuously compounded returns)
r_t	Return at time t
$\bar{r}_t$	Mean of the returns
$\hat{r}_t$	Predicted return at time t
$sigmoid(x)$	Sigmoid function
U_m	Bias proportion in *TIC*
U_s	Variance proportion in *TIC*
U_c	Covariance proportion in *TIC*
Vol_t	Total number of traded stocks during the day
V_t	Volatility
V_{jh}	Weight between output unit j and hidden unit h
w	A weight in the network
w_{hi}	Weight between hidden unit h and input unit i
w_{ij}	Weight to unit i from unit j
X_d	Input layer units in the network
y_j	Output of the network
y_k	Artificial neural network output
y_t	Value of the market index on time t
Z_h	Hidden units in the network
ω, α, β and γ	Constant parameters
∂_{ik}	Kronecker delta
φ_{t-1}	The information set available at time t-1
$\phi(x)$	Gaussian basis function
Δw	Change in weight
ε_t	White noise disturbance term
ε_t	i.i.d. process with zero mean and a variance equal to one

σ_t^2	Conditional variance
η	Learning rate

ACF	Autocorrelation Function
ANNs	Artificial Neural Networks
AR	Auto-Regressive
ARCH	Autoregressive Conditional Heteroskedasticity
ARMA	Autoregressive Moving Averages
BPTT	Back-Propagation Through Time
BVSP	Brazilian National Index
CMB	Capital Markets Board
CORR	Correlation
CPI	Consumer Price Index
EGARCH	Exponential GARCH
EMH	Efficient Market Hypothesis
FTSP	Financial Time Series Prediction
GARCH	Generalized Autoregressive Conditional Heteroskedasticity
GLIMs	Generalized Linear Models
GNP	Gross National Product
HME	Hierarchical Mixture of Experts
ISE	Istanbul Stock Exchange
ISE National-100	Istanbul Stock Exchange National 100 Index
ISE National-30	Istanbul Stock Exchange National 30 Index
MAPE	Mean absolute percentage error
MLP	Multilayer Perceptron
MoE	Mixture of Experts
MSE	Mean square error
PACF	Partial Autocorrelation Function
RBF	Radial Basis Functions
RMSE	Root mean square error
RNN	Recurrent Neural Networks
RSI	Relative Strength Index

RTRL	Real-time Recurrent Learning
RWH	Random Walk Hypothesis
TA	Technical Analysis
TDNN	Time Delay Neural Networks
TIC	Theil Inequality Coefficient
U.S.	United States
VaR	Value-at-Risk
XU100	Istanbul Stock Exchange National 100 Index
XU30	Istanbul Stock Exchange National 30 Index

1. INTRODUCTION

Stock market is a huge system where millions of different traders manipulate thousands of instruments every day. Obviously in such an environment predicting future returns is a highly important concept. Indeed prediction is not an eyes wide shot, it needs modeling the whole subject and requests rational methodology. The question is, is it possible to find such a logical structure to predict the stock market's return? The answer to that question is an open issue and in this thesis we have investigated how we can use artificial neural networks (ANN) to predict it.

When compared with traditional statistical approaches ANN is a new but innovative and promising methodology in this area [1]. The key factor in ANN based modeling and prediction is the ability of the system to learn the changes in the environment by itself. Besides, ANN based modeling showed a great performance in capturing the real dynamics of the underlying data [1].

Financial Time Series Prediction (FTSP) aims to find underlying patterns, trends, cycles and it also aims to forecast the future using historical and currently observable data [1]. We have used Mixture of Experts "divide and conquer" technique to analyze the underlying patterns in Istanbul and Brazilian Stock markets. National 100 and 30 indices are used from domestic market and Brazilian national index is used as the international market.

By looking at the simulation results we can be state that ANN showed a great performance in both modeling and prediction. The interaction between Brazilian and Turkish stock markets is confirmed by the simulations.

ANN is well suited for practical problems where it is easier to have data rather than knowledge. This was the case in our research. Generally, they can be viewed as one of many multivariate nonlinear and nonparametric statistical methods [2].

1.1. Aims and Objectives

This thesis aims to model and predict stock market and to investigate the relationships between these emergent markets. Istanbul Stock Exchange National 100 and 30 indices, Brazilian National index (BVSP) are studied using artificial neural network structure Mixture of Experts (MoE). For capturing dynamics of the data precisely volatility factor is taken into consideration in both modeling and prediction. The aim is to predict the short-term future returns of the underlying series. The interaction between these two markets is another consequence of the stock market's return analysis.

1.2. Organization of the Thesis

Chapters 2 and 3 give necessary information about the stock markets in associated countries. Chapters 4, 5 analyze the traditional econometrics methodologies in time series modeling and volatility concept. Sixth chapter is for the general background in artificial neural networks, different architectures from basic perceptrons to local models. Seventh chapter briefly describes the performance metrics that are used for the experiments in the thesis. Performance measures and their interpretations are given in this chapter. Experiments and associated results are given in chapter 8. The networks parameters are presented and associated experiment's results are explained in detail. Total results are given in tabular form at the end of this chapter. Conclusion and future work is summarized in chapter 9.

2. STOCK MARKET

2.1. Definition of Stock Market and Share

Stock Exchange Market which brings together people who want to buy shares and those who want to sell their shares. A share is simply a part-ownership of a company. The laws of supply and demand determine the prices buyers and sellers settle on. The companies use the service provided by the stock market if they obey the rules for buying and selling shares. In other word, a stock market is a market for the trading of company stocks and derivatives.

2.2. Istanbul Stock Exchange

The Istanbul Stock Exchange (ISE) is the only corporation in Turkey for securities exchange. It is established to provide trading in equities, bonds and bills, revenue sharing certificates, private sector bonds, foreign securities and real estate certificates as well as international securities. The ISE was founded as an autonomous, professional organization in early 1986 [3].

Takasbank is the central securities depository and chasing house of Turkey, functioning as the "Central Bank of Capital Markets" in Turkey; it aims at providing settlement, custody and related banking services to a domestic and international investors. It is also appointed as the National Numbering Agency of Turkey by the Capital Markets Board (CMB). With respect to CSD functions and responsibilities, Takasbank is regulated by the CMB. According to the CMB regulations, no institution other than Takasbank is permitted to safekeeping of physical certificates of securities in Turkey [3].

2.2.1. ISE Stock Market Indices

ISE indices are composed in order to calculate price and return performances of all shares as well as on the basis of relative markets and sectors. Until the end of 1996, the ISE used to compute only the ISE-100, Financials and Industrials price indices. As from 1997, the ISE began to calculate sector and sub-sector indices on the basis of prices and total return.

ISE price indices are computed and published throughout the trading session while the return indices are calculated and published at the close of the session only. The ISE National-100 Index is used as a main indicator of the National Market.

ISE National-All Shares Index is composed of all National Market companies except investment trusts [3].

ISE National-30 is composed of National Market companies except investment trusts and will also be used for trading in the Derivatives Market. The constituent 30 companies are selected on the basis of pre-determined criteria directed for the companies to be included in the indices [3].

ISE National-50 is composed of National Market companies except investment trusts. The constituent 50 companies are selected on the basis of pre-determined criteria directed for the companies to be included in the indices. ISE National-50 Index contains the ISE National-30 Index companies [3].

Sector and sub-sector indices are composed of National Market companies excluding investment trusts [3].

ISE Second National Market Index is composed of companies traded in the Second National Market. Regional Market's name was changed to Second National Market as of March 03, 2003 [3].

ISE New Economy Market Index is composed of companies traded in the New Economy Market [3].

This thesis discusses prediction of the ISE market index (ISE National-100) from Istanbul Stock Exchange. ISE National-100, which has been calculated since the inception of the ISE, is composed of National Market companies except investment trusts. The constituents of the ISE National-100 Index are selected on the basis of pre-determined criteria directed for the companies to be included in the indices. ISE National-100 Index contains the ISE National-50 and ISE National-30 Index companies.

The ISE indices are weighted by the market capitalization of the publicly held portion (the stocks kept in custody at Takasbank, except those kept in non-fungible accounts) of each constituent.

The basic formula for calculating ISE's float capitalization-weighted indices is as follows:

$$ISENational-100_t = \frac{\sum_{i=1}^{n} P_{it} N_{it} FW_{it}}{D_t} \tag{2.1}$$

where

p_{it} = The closing price of the stock i at period t

N_{it} = The closing price of the stock i at period t

FW_{it} = The flotation weight (publicly-held portion, i.e. the ratio of stocks kept in custody at Takasbank, except those kept in non-fungible accounts) of the stock i at period t

D_t = The value of divisor at period t (Adjusted base market value)

n = Total number of stocks included in the index.

Old and new certificate prices are considered separately and only the registered prices are taken into account in the calculation process.

2.3. The São Paulo Stock Exchange (BOVESPA)

The São Paulo Stock Exchange (BOVESPA) is the stock trade center in Brazil and Latin America's largest stock exchange, accounting for nearly 70% of the region's trade volume. BOVESPA was founded on August 23, 1890 and it has a long history of services rendered to the capital market and to the Brazilian economy. Within the process of restructuring of the Brazilian capital market, finished in 2000, BOVESPA became the single stock exchange of Brazil [31].

The securities market in Brazil is governed and controlled by the National Monetary Council (CMN), which is the highest regulatory body of the financial system, and by the Brazilian Securities and Exchange Commission (CVM) an independent federal agency [31].

2.3.1. Bovespa Stock Market Indıces

Bovespa Stock Market includes these indices [31].

- Bovespa Index Ibovespa
- Brazil Index IBrX
- Brazil Index 50 IBrX50
- Valor Bovespa Index IVBX 2
- Special Corporate Governance Stock Index IGC
- Electric Power Index IEE
- Telecommunication Sector Index ITEL
- Special Tag Along Stock Index ITAG
- Corporate Sustainability Index ISE

This thesis uses Bovespa Index, which is the main indicator of the Brazilian stock market's average performance like Turkish ISE National-100 Index. Ibovespa's basic objective is to be an average indicator of the market performance. For that purpose, its composition aims at reflecting as close as possible the real configuration of the cash market operations.

The Bovespa Index is the sum of the weights (stock theoretical quantity multiplied by its last price) of the stocks that integrates the theoretical portfolio. In this sense, it can be calculated, at any time, according to the following formula [31]:

$$Ibovespa_t = \sum_{i=1}^{n} P_{i,t} Q_{i,t} \tag{2.2}$$

$Ibovespa_t$ = Bovespa Index at moment t

n = Total number of stocks that compose the theoretical portfolio

$P_{i,t}$ =last price at stock i at moment t

$Q_{i,t}$ =Theoretical quantity of stock i in the portfolio at moment t

2.4. Data used for financial time series prediction

The information about the market comes from the study of relevant data; there are several different types of data which can be assigned to the following categories:

Technical data: This includes such figures as past stock prices, volume, volatility.

Fundamental data: These are data describing current economic activity of the company or companies whose stock prices are to be predicted. Further, fundamental data include information about current market situation as well as macroeconomic parameters (unemployment rate, inflation).

The last type of the data, derived entities, are produced by transforming and combining technical and/or fundamental data

2.4.1. Technical data

Typical technical data involved in financial time series prediction are

- Close value (price of the last performed trade during the day) (p_t)
- Highest traded price during the day (p_{high})
- Lowest traded price during the day (p_{low})
- Volume (total number of traded stock during the day) (Vol_t)

While in most cases, daily data are used for modeling stock price behavior, data for each individual trade during the day are sometimes available as well. Such data are most often used not for modeling the market, but for determining the right time of an intended trade in real trading

1. The close prices normally vary greatly and make it difficult to create a model for a longer period of time

2. The close prices for different stocks may easily differ over several decades and therefore can not be used as the same type of input in a model

2.4.2. Fundamental data

Fundamental data include the information about the activities and financial situation of companies. Their analyses provide a basis for evaluating the true value of company's stock. Fundamental analysts take into consideration the followings;

1. The general state of economy measured by inflation, interest rate, trade balance
2. The condition of the industry, to which the company belongs, measured by

- Stock price indices
- Prices of related commodities such as oil, different metals and currencies.
- The value of competitors' stocks

3. The condition of the company measured by

- P/E (price/earnings) ratio (stock price divided by the earning per share during the last 12 months)
- book value per share (net assets (assets minus liabilities) divided by the total number of shares)
- Net profit margin (net income divided by total sales)
- Debt ratio (liabilities divided by total assets)
- Prognoses of future profits
- Prognoses of future sales

2.4.3. Derived Entities

Transforming and combining technical and/or fundamental data can produce this type of data. We can summarize some examples as listed below:

2.4.3.1 Returns

In general; returns (also named as one-step return) shown by $R(t)$ is defined as the relative increase in price since the previous point in the time series y:

$$R(t) = \frac{y(t) - y(t-1)}{y(t-1)} \tag{2.3}$$

In this thesis; it is the relative increase in price since the previous point in the time series. If we calculate returns from prices of underlying financial asset between times $t-1 \text{and} t$.

$$R(t) = \frac{p(t) - p(t-1)}{p(t-1)} \tag{2.4}$$

Because of the difficulty of manipulating geometric average operations over the whole time period brings another approach, the continuously compounded return or log return $R(t)$ of an asset have also important implications on modeling of asset returns and is defined as:

$$R(t) = \ln \frac{p(t)}{p(t-1)} \tag{2.5}$$

One-step and log returns are very similar for small changes and are very often used for financial time series prediction for the following reasons:

1. $R(t)$ has a relatively constant range even if data for many years are used as input. The raw prices y vary much more and make it difficult to create a valid model for a longer period of time

2. $R(t)$ for different stocks may also be compared on an equal basis

3. It is easy to evaluate a prediction algorithm for $R(t)$ by computing the prediction accuracy of the sign of $R(t)$. Accuracy above 50 % (more precisely above the mean) indicates that a true prediction has taken place

2.4.3.2 Volatility

Volatility is the synonym for standard deviation of some value (e.g. stock price). Volatility is a measure for risk, but also for profit opportunities. In so-called delta-neutral trading strategies, the profit/loss of a trading operation depends not on the price, but on the volatility of the underlying stock price [4]. Thus, volatility is not only a measure of the risk, but also a tradable commodity. On the other hand, it is empirically known that volatility and predictability of financial time series are connected [39].

2.4.3.3 Volume

The increase of traded volume is often viewed as an indication of new information reaching the market. Therefore it may be useful to include the rate of change of volume as an additional input variable.

3. STOCK MARKET PREDICTION

Revolutions often spawn counterrevolutions and the efficient market hypothesis in finance is no exception [5]. The intellectual dominance of the efficient-market revolution has more been challenged by economists who stress psychological and behavioral elements of stock-price determination and by econometricians who argue that stock returns are, to a considerable extent, predictable. To begin with we will introduce the efficient market hypothesis.

3.1. Efficient Market Hypothesis

3.1.1 Rational Expectations

3.1.1.1 Rational Expectation – Weak Form

Whatever information people have, they make optimal use of this information in forming their expectations [6].

3.1.1.2 Rational Expectation – Strong Form

People have access to all available information about the structure of the world in which they live and they make optimal use of this information in forming their expectations. Thus, their expectations will be correct up to unsystematic errors [6].

3.1.2 Efficient Market Hypothesis – First Form

Expectations in financial markets are equal to optimal forecasts using all available information [6].

$$RET_t = \frac{P_{t+1} - P_t + C}{P_t} \tag{3.1}$$

$$RET_t^e = \frac{P_{t+1}^e - P_t + C}{P_t} \tag{3.2}$$

Rational Expectations implies: $P_{t+1}^e = P_{t+1}^{of}$ (optimal forecast), and

$$P_{t+1}^e = P_{t+1}^{of} \Leftrightarrow RET_t^e = RET_t^{of}$$

Efficient Market Hypothesis – First Form: $RET_t^e = RET_t^{of}$

3.1.3 Efficient Market Hypothesis – Second Form

Current security prices in a financial market will be set so that the optimal forecast of a security's return rate using all available information equals the security's equilibrium return rate [6].

Efficient Market Hypothesis – Second Form (Stronger):

$$RET_t^{of} = RET_t^* \left(\text{ Equilibrium Supply} = \text{Demand Return Rate } \right)$$

In case of $RET_t^{of} > RET_t^* \rightarrow P_t \uparrow, RET_t^{of} \downarrow$ until $RET_t^{of} = RET_t^*$, otherwise

$RET_t^{of} < RET_t^* \rightarrow P_t \downarrow, RET_t^{of} \uparrow$ until $RET_t^{of} = RET_t^*$

3.1.4 Efficient Market Hypothesis – Third Form

In addition to first and second form, security prices reflect true fundamental (intrinsic) value, meaning there are no price bubbles on security prices [6].

3.2. Why is stock market predictable?

A generation ago, the efficient market hypothesis was widely accepted by the financial economists in academia; Fama's influential survey article, "Efficient Capital Markets", is a good example [7]. It was generally believed that securities markets were extremely efficient in reflecting information about individual stocks and about the stock market as a whole. The accepted view was that when information arises, the news spreads very quickly and it is incorporated into the prices of securities without delay. Thus, neither technical analysis, which is the study of past stock prices in an attempt to predict future prices, nor even fundamental analysis, which is the analysis of financial information such as company earnings, asset values, etc., to help investors select undervalued stocks, would enable an investor to achieve returns greater than those that could be obtained by holding a randomly selected portfolio of individual stocks with comparable risk [5].

The efficient market hypothesis is associated with the idea of a random walk, which is a term used to characterize a price series where all subsequent price changes represent random departures from previous prices. The logic of the random walk idea is that if the flow of information is unimpeded and information is immediately reflected in stock prices, then tomorrow's price change will reflect only tomorrow's news and will be independent of the price changes today. But news is by definition unpredictable, thus, resulting price changes must be unpredictable and random [5].

As a result, prices fully reflect all known information, and even uninformed investors buying a diversified portfolio at the tableau of prices given by the market will obtain a rate of return as generous as that achieved by the experts [5].

By the start of the twenty-first century, the intellectual dominance of the efficient market hypothesis had become far less universal. Many financial economists and statisticians began to believe that stock prices are at least partially predictable. A new breed of economists' emphasized psychological and behavioral elements of stock-price determination, and came to believe that future stock prices are somewhat predictable on the basis of past stock price patterns as well as certain "fundamental" valuation metrics.

Moreover, many of these economists were even making the far more controversial claim that these predictable patterns enable investors to earn excess risk-adjusted rates of return.

In this thesis, we believe that stock prices are predictable; also we claim that it is not just predictable with traditional statistical methods but also with ANN.

3.3.Prediction Methods

The prediction of the market is indeed an interesting issue. There are several techniques in literature to accomplish this task. In this section briefly we will describe these methods. Mainly it is divided into four main categories as; technical analysis, fundamental analysis, traditional time series analysis and the last machine learning approach.

3.3.1. Technical Analysis

Technical analysis (TA) is defined as the study of *market (price) actions* for the purpose of forecasting future price trends [8]. Using technical data such as price, volume, and highest and lowest prices per trading period the technical analyst uses charts to predict future stock movements. Price charts are used to detect trends; these trends are assumed to be based on supply and demand issues which often have cyclical or noticeable patterns.

One of the reasons for TA's popularity is that it forces a discipline and control on trading by providing traders with price and profit/loss objectives before trades are made. It is also a very useful tool for short-term as well as long-term trading strategies as it does not rely on any information other than market data. Another reason for its popularity is that, while its basic ideas are easy to understand, a wide variety of trading strategies can be developed from these ideas.

Currently the major areas of technical analysis are:

- *Charting*: The study of price charts and chart patterns; e.g. trend lines, triangles, reversal patterns, and Japanese candlesticks.
- *Technical/Statistical Indicators*: The study of technical indicators; e.g., momentum, relative strength index (RSI), stochastic and other oscillators.
- *Trading Systems*: Developing computerized or automated trading systems, as well as mechanical trading systems, ranging from simple systems using technical indicators with a few basic rules (to generate trading signals such as moving averages) to complex rule based systems incorporating soft computing methods such as artificial neural networks, genetic algorithms, and fuzzy logic. The traditional trading systems are based on rigid rules for entering and exiting the market. The main advantage of these systems is that they impose discipline on traders.
- *Esoteric methods* e.g. Elliot Waves, Gann Lines, Fibonacci ratios, and astrology [17].

Technical analysis does not concern itself in studying the reasons for the price action and focuses instead on the study of the price action itself. Besides that, TA tries to identify the trend and direction of the trend. At the last point technical analysis depends on the fact that actions in any economical system repeat itself in time.

3.3.2. Fundamental Analysis

Fundamental analysis studies the effect of supply and demand on price. All relevant factors that affect the price of a security are analyzed to determine the intrinsic value of the security. If the market price is below its intrinsic value then the market is viewed as undervalued and the security should be bought. If the market price is above its intrinsic value, then it should be sold. Examples of relevant factors that are analyzed are financial ratios; e.g. Price to Earnings, Debt to Equity, Industrial Production Indices, Gross National Product (GNP) and Consumer Price Index (CPI). Fundamental analysis studies the causes of market movements, in contrast to technical analysis, which studies the effect of market movements. Interest Rate Parity Theory and Purchasing Power Parity Theory are examples of the theories used in forecasting price movements using fundamental analysis.

The problem with fundamental analysis theories is that they are generally relevant only in predicting longer trends. Fundamental factors themselves tend to lag market prices, which explain why sometimes market prices move without apparent causal factors, and the fundamental reasons only becoming apparent later on.

3.3.3. Traditional Time Series Prediction

The Traditional Time Series Prediction analyzes historic data and attempts to approximate future values of a time series as a linear combination of these historic data. Basically, this method attempts to model a nonlinear function by a recurrence relation derived from past values. The recurrence relation can then be used to predict new values in the time series, which hopefully will be good approximations of the actual values. In econometrics there are two basic types of time series forecasting: univarite and multivariate, these subjects are presented in detail in chapters four and five in this thesis.

3.3.4. Machine Learning Methods

Several methods for inductive learning have been developed under the common label *"Machine Learning"*. All these methods use a set of samples to generate an approximation of the underlying function that generated the data. The aim is to draw conclusions from these samples in such a way that when unseen data are presented to a model it is possible to infer the *explained variable* from these data. The Nearest Neighbor and Neural Networks are one of the most popular prediction techniques that are both applied to market prediction. The nearest neighbor technique is suitable for *classification tasks*. It classifies unseen data to bins by using their 'distance' from the k bin centroids. The distance is usually the Euclidean distance. In the frame of the stock market prediction this method can be applied by creating three (or more) bins. The first bin classifies the samples indicating a market rise, the second bin classifies the samples indicating a fall and the third bin is for the indication of no change in the market. Artificial Neural Networks (ANNs) can be used to perform *classification* and *regression* tasks. Detail information about ANN method will be given in chapter six of this thesis.

4. TIME SERIES MODELLING AND FORECASTING

In time series modeling, one attempts to model and predict financial variables using only information contained in their own past values and possibly current and past values of the error term. This type of modeling is different from structural models which attempt to define changes in a variable by reference to the movements in the current or past values of other variables. In our research, we try to model and predict financial time series, especially return series for Istanbul Stock Exchange and Brazilian Stock Exchange by using artificial neural networks architecture.

From the financial point of view, an important class of time series models is the family of Autoregressive Integrated Moving Average (ARIMA) models, usually associated with Box and Jenkins [9]. In order to define, estimate and use ARIMA models, it is needed to specify several important concepts of time series.

4.1. Notations and Concepts

4.1.1. Stationary Processes

A series is strictly stationary if the distribution of its values remains the same as time progresses, implying that the probability that y falls within a particular interval is the same now as at any time in the past or the future [10].

If we define stationary concept statistically; for any $t_1, t_2, ..., t_T \in Z$, any $k \in Z$ and $T = 1,2,...$, $Fx_{t_1}, x_{t_2}, ..., x_{t_T}(x_1, ..., x_T) = Fx_{t_1+k}, x_{t_2+k}, ..., x_{t_T+k}(x_1, ..., x_T)$ where F denotes the joint distribution of the set of random variables [11].

If a series satisfies the conditions such as constant mean, constant variance and constant autocovariance structure, then it is weakly or covariance stationary. If we define these conditions for the time series:

$$E(y_t) = \mu \tag{4.1}$$

$$E(y_t - \mu)(y_t - \mu) = \sigma^2 \tag{4.2}$$

$$E(y_{t_1} - \mu)(y_{t_2} - \mu) = \gamma_{t_2 - t_1} \tag{4.3}$$

What we mean by autocovariance is how a variable y is related to its previous values, for stationary series it depends only on the difference between t_1 and t_2. Autocovariance function is:

$$E(y_t - E(y_t))(y_{t-s} - E(y_{t-s})) = \gamma_s, s = 0,1,2,... \tag{4.4}$$

It is more convenient to use the autocorrelations where autocovariances normalized by dividing by the variance.

$$\tau_s = \frac{\gamma_s}{\gamma_0}, s = 0,1,2,... \tag{4.5}$$

Thus the series τ_s has the property of correlation coefficients that the values are bounded by ± 1. If τ_s is plotted against s then the graph is called autocorrelation function (acf) or correlogram.

4.1.2. White Noise Processes

A white noise process is one with no discernible structure [10]. Properties of a white noise are:

$$E(y_t) = \mu \tag{4.6}$$

$$\text{var}(y_t) = \sigma^2 \tag{4.7}$$

$$\gamma_{t-r} = \begin{cases} \sigma^2, t = r \\ 0, otherwise \end{cases} \tag{4.8}$$

White noise process has constant mean, variance and zero autocovariances, except at zero lag. In other words each observation is uncorrelated with all other values in sequence. If $\mu = 0$ then the process is called zero mean white noise process.

4.1.3. Partial Autocorrelation Function

The partial autocorrelation function, or pacf (τ_{kk}), measures the correlation between an observation k periods ago and the current observation, after controlling for observations at intermediate lags [10].

$$\tau_{kk} = \frac{\left(\tau_k - \tau_{k-1}^2\right)}{\left(1 - \tau_{k-1}^2\right)} \tag{4.9}$$

4.2. Processes

4.2.1. Moving Average Processes

The simplest type of time series model is the moving average process. If we define $u_t (t = 1,2,3,...)$ is a sequence of independently and identically distributed random variables with $E(u_t) = 0$ and $\text{var}(u_t) = \sigma^2$, then q th order moving average mode, denoted by $MA(q)$ is

$$y_t = \mu + u_t + \theta_1 u_{t-1} + \theta_2 u_{t-2} + ... + \theta_q u_{t-q} \tag{4.10}$$

This can also be expressed as

$$y_t = \mu + \sum_{i=1}^{q} \theta_i u_{t-i} + u_t \tag{4.11}$$

Simply moving average process is a linear combination of white noise processes, such that y_t depends on the current and previous values of a white noise disturbance term. Using lag operator notation equation (4.10) can be written as

$$y_t = \mu + \sum_{i=1}^{q} \theta_i L^i u_t + u_t \tag{4.12}$$

The distinguishing properties of the moving average process is that it has constant mean, constant variance and autocovariances which may be non-zero to lag q and will always be zero thereafter.

4.2.2. Autoregressive Processes

In autoregressive processes, the current value of variable y depends only on the values that the variable took in the previous periods plus an error term. An autoregressive model of order p, denoted by $AR(p)$ can be expressed as equation (4.13) where u_t is the white noise disturbance term.

$$y_t = \mu + \phi_1 y_{t-1} + \phi_2 y_{t-2} + ... + \phi_p y_{t-p} + u_t \tag{4.13}$$

Equation (4.12) can be written as

$$y_t = \mu + \sum_{t=1}^{p} \phi_t y_{t-i} + u_t \tag{4.14}$$

Or it can be written using lag operator

$$y_t = \mu + \sum_{i=1}^{p} \phi_i L^i y_t + u_t \tag{4.15}$$

4.2.3. ARMA Processes

By combining the $AR(p)$ and $MA(q)$ models, $ARMA(p,q)$ models are constructed. $ARMA$ model states that the current value of some series y depends linearly on its own previous values plus a combination of current and previous values of a white noise error term. The model can be written as

$$y_t = \mu + \phi_1 y_{t-1} + \phi_2 y_{t-2} + ... + \phi_p y_{t-p} + \theta_1 u_{t-1} + \theta_2 u_{t-2} + ... + \theta_q u_{t-q} + u_t \tag{4.16}$$

The properties of $ARMA$ process are combination of AR and MA processes. If we summarize the characteristics of three models (AR , MA , $ARMA$) respectively

AR

- a geometrically decaying acf
- number of non-zero points of pacf = AR order

MA

- number of non-zero points of acf = MA order
- a geometrically decaying pacf

ARMA

- a geometrically decaying acf
- a geometrically decaying pacf

4.3. Models

4.3.1. ARMA Models

Estimating an *ARMA* model first time in a systematic manner was first accomplished by Box-Jenkins [9]. Their approach has three steps:

1. Identification
2. Estimation
3. Diagnostic checking

In step 1, identification the order of the model is made to capture the dynamic features of the data. In step 2, estimation of parameters of associated model is ensured. For the last step, the aim is determining whether the model specified and estimated is adequate. Box and Jenkins [9] suggest two methodologies; overfitting and residual diagnostics.

Overfitting is briefly designing a larger model than required to capture the dynamics of the data, then check if the model specified is adequate or not. If the model is adequate no more parameters will be added to the system. In residual diagnostics one would check the residuals for evidence of linear dependence which, if present, originally specified model is inadequate.

The main idea in designing a model for the relevant data is to form a parsimonious model. What we mean by parsimonious is simply describing all of the features of the data of interest using as few parameters as possible.

4.3.2. ARIMA Models

ARIMA modeling is different from *ARMA* that an integrated autoregressive process is used which has a characteristic equation whose root is on the unit circle. An $ARMA(p,q)$ model in the variable is differenced d times is equivalent to an $ARIMA(p,d,q)$ model on the original data.

4.3.3. Vector Autoregressive Models

Vector autoregressive models are a natural generalization of models like AR, MA and $ARMA$ [12]. A VAR's a systems regression model that can be considered a kind of hybrid between the univariate models discussed in 4.2 and 4.3 and simultaneous equations. The simplest case for VAR models is bivariate VAR, where there are only 2 variables, y_{1t} and y_{2t}, each of whose current values depend on different combinations of the previous k values of both variables, and the error terms.

$$\begin{aligned} y_{1t} &= \beta_{10} + \beta_{11} y_{1t-1} + \ldots + \beta_{1k} y_{1t-k} + \alpha_{11} y_{2t-1} + \ldots + \alpha_{1k} y_{2t-k} + u_{1t} \\ y_{2t} &= \beta_{20} + \beta_{21} y_{2t-1} + \ldots + \beta_{2k} y_{2t-k} + \alpha_{21} y_{1t-1} + \ldots + \alpha_{2k} y_{1t-k} + u_{2t} \end{aligned} \tag{4.17}$$

5. VOLATILITY MODELLING

All of the models discussed in chapter IV have been linear in the parameters. However many models in finance seem to be non-linear. Especially in our subject, financial time series analysis, the risk factor in financial asset returns is also non-linear. Linear models are unable to explain a number of important features common to much financial data, including leptokurtosis, volatility clustering and leverage effects [10]. Models in financial data analysis can be linear in mean and variance like *ARMA* models or linear in mean but non-linear in variance like GARCH models. In this chapter we will briefly look at GARCH type models.

5.1. Non-linear Models Features

5.1.1. Important Characteristics

- *Leptokurtosis:* Tendency for financial asset returns to have distributions that exhibit fat tails and excess peakedness at the mean.

- *Volatility Clustering or Volatility Pooling:* Tendency for volatility in financial markets to appear in bunches, in other words; large returns are expected to follow large returns, small returns follow small returns.

- *Leverage Effects:* Leverage effect is the tendency for volatility to rise more following a large price fall than following a price rise of the same magnitude.

5.2. Non-linear Models

Small numbers of non-linear models are useful for modeling financial data. The most popular non-linear financial models are the ARCH or GARCH models used for modeling and forecasting volatility. Volatility is on of the most important concepts in finance.

Volatility is measured by the standard deviation or variance of returns, and is used for the measure of risk of financial assets.

5.2.1 Historical Volatility

Simplest volatility model is the historical one such that it consists of calculating the variance (or standard deviation) of returns over some historical period and using this volatility measure for all the future periods. However use of volatility modeling from some more sophisticated time series models are preferred by financial economists, nevertheless historical volatility is still used as a benchmark for comparing the accuracy of different models.

5.2.2. Exponentially Weighted Moving Average Models

The exponentially weighted moving average (EWMA) is the simplest extension of historical average volatility measure. In this methodology, more recent observations have stronger effect on the forecast of volatility than the older data points. Therefore, the latest observation holds the largest weight and weights associated with previous observations decline exponentially over time. Indeed, this method has several advantages when compared to historical volatility measure such as; volatility is affected mostly from latest observations and using exponentially declining weight assigned to observations give more accurate results.

$$\sigma^2 = (1-\lambda)\sum_{j=0}^{\infty} \lambda^{j-1} \left(r_{t-j} - \bar{r}\right)^2 \tag{5.1}$$

In equation (5.1) σ^2 is the estimate of the variance for the period t and also becomes the forecast of the future volatility.

5.2.3. Autoregressive Volatility Models

A time series of observations on some volatility proxy are obtained. The standard Box-Jenkins [9] type procedures for estimating autoregressive models can then be applied to this series. If daily volatility estimate is in concern then two natural proxies are employed: squared daily returns or daily range estimators. In squared daily returns approach the squared return at each point in time, t becomes the daily volatility estimate for day t. In range estimators approach; log ratio of the highest observed price to the lowest observed price for trading day t becomes the volatility estimate for day t.

$$\sigma^2 = \log\left(\frac{high_t}{low_t}\right) \tag{5.2}$$

In either two approaches, a standard autoregressive model, with the coefficients β_i is estimated.

$$\sigma_t^2 = \beta_0 + \sum_{j=1}^{p} \beta_j \sigma_{t-j}^2 + \varepsilon_t \tag{5.3}$$

5.2.4. Autoregressive Conditionally Heteroscedastic Models (ARCH)

The assumption of the *ARMA* type's models is that the variance of the errors is constant (homoscedasticy). If the variance of the errors is not constant, it is known as heteroscedasticity. In *ARCH* modeling variance of the errors of financial time series is assumed not to be constant. In the *ARCH* model, conditional variance of the error term, σ_t^2, depends on the immediate previous value of the squared error. Therefore $ARCH(q)$ model is

$$\sigma_t^2 = \alpha_0 + \alpha_1 u_{t-1}^2 + \alpha_2 u_{t-2}^2 + \ldots + \alpha_q u_{t-q}^2 \tag{5.4}$$

Another way of expressing $ARCH(q)$ model can be given by considering $ARCH(1)$ as an example.

$$\begin{aligned} y_t &= \beta_1 + \beta_2 x_{2t} + \beta_3 x_{3t} + \beta_4 x_{4t} + u_t \\ u_t &= v_t \sigma_t \Rightarrow v_t \approx N(0,1) \\ \sigma_t^2 &= \alpha_0 + \alpha_1 u_{t-1}^2 \end{aligned} \tag{5.5}$$

$ARCH(q)$ provided a framework for the analysis and development of time series models of volatility. However it has some difficulties [10]:

- Deciding the value of q in the model is a problem and there is no clearly best approach,
- The value of q may be too large that would result in a large conditional variance model that is not parsimonious,
- The more parameters in the conditional variance, more likely there will be parameters that have negative estimated value.

5.2.5. Generalized ARCH Models

The *GARCH* model permits the conditional variance to be dependent upon previous own lags, so the conditional variance equation becomes

$$\sigma_t^2 = \alpha_0 + \alpha_1 u_{t-1}^2 + \beta \sigma_{t-1}^2 \tag{5.6}$$

Equation (5.6) is $GARCH(1,1)$ model, σ_t^2 is known as conditional variance. *GARCH* model is a better and far more widely used model since the model is more parsimonious and avoids over-fitting. Also the model is less likely to breach negative parameters. General structure for the $GARCH(p,q)$ model is

$$\sigma_t^2 = \alpha_0 + \sum_{i=1}^{q} \alpha_i u_{t-i}^2 + \sum_{j=1}^{p} \beta_j \sigma_{t-j}^2 \tag{5.7}$$

In general $GARCH(1,1)$ model is sufficient to capture the volatility clustering in the data, and rarely is any higher order model estimated. Conditional variance is variable but the unconditional variance of u_t is constant and given by

$$\text{var}(u_t) = \frac{\alpha_0}{1-(\alpha_1+\beta)} \text{ as long as } \alpha_1 + \beta < 1 \tag{5.8}$$

For both *ARCH* and *GARCH* type's models, fundamental ordinary least square (OLS) can not be used. Since OLS minimizes the residual sum of squares, thus residual sum of squares (RSS) depends only on the parameters in the conditional mean equation, not the conditional variance. In order to estimate models from these families maximum likelihood is employed.

$$L = -\frac{T}{2}\log(2\pi) - \frac{1}{2}\sum_{t=!}^{T}\log(\sigma_t^2) - \frac{1}{2}\sum_{t=!}^{T}\frac{(y_t - \mu - \phi y_{t-1})^2}{\sigma_t^2} \tag{5.9}$$

5.2.6. Extensions of Basic GARCH Model

One of the primary restrictions of *GARCH* models is that they ensure a symmetric response of volatility to positive and negative shocks. Two popular asymmetric methodologies will be discussed.

5.2.6.1. GJR Model

GJR model is a simple extension of GARCH with an additional term taken into consideration [13].

$$\sigma_t^2 = \alpha_0 + \alpha_1 u_{t-1}^2 + \beta\sigma_{t-1}^2 + \gamma u_{t-1}^2 I_{t-1} \text{ where } \begin{cases} I_{t-1} = 1, u_{t-1} < 0 \\ I_{t-1} = 0, otherwise \end{cases} \quad (5.10)$$

For leverage effect $\gamma > 0$ is ensured.

5.2.6.2. EGARCH Model

Exponential *GARCH* model was proposed by Nelson [14]. Conditional variance equation can be expressed by

$$\ln(\sigma_t^2) = \omega + \beta \ln(\sigma_{t-1}^2) + \gamma \frac{u_{t-1}}{\sqrt{\sigma_{t-1}^2}} + \left[\frac{|u_{t-1}|}{\sqrt{\sigma_{t-1}^2}} - \sqrt{\frac{2}{\pi}} \right] \quad (5.11)$$

The model in (5.11) have several advantages over the original *GARCH* model since $\log(\sigma_t^2)$ is modeled, even if the parameters are negative σ_t^2 will be positive, and also asymmetry affect is permitted by the help of the parameter γ. Essentially *GARCH* models are useful because they can be used to model the volatility of a series over time. It is possible to combine together more than one of the time series models to obtain more complex hybrid models. Indeed *GARCH* is a model to describe movements in the conditional variance of an error term, u_t, which may not appear particularly useful. Below we will describe why we use non-linear volatility models in forecasting return series.

$$\text{var}(y_t \mid y_{t-1}, y_{t-2}, ...) = \text{var}(u_t \mid u_{t-1}, u_{t-2}, ...) \quad (5.12)$$

The conditional variance of y, given its previous values, is the same as the conditional variance of u, given its previous values. Hence modeling σ_t^2 will give models and forecasts for the variance of y_t. Therefore if the dependent variable in a regression y_t is an asset return series, forecast of σ_t^2 will be forecasts of the future variance of y_t. If volatility is used as an input like our ANN model, forecasting volatility becomes important in prediction of return series.

6. NEURAL NETWORKS

A neural net is an artificial representation of the human brain that tries to simulate its learning process. The term "artificial" means that neural nets are implemented in computer programs that are able to handle the large number of necessary calculations during the learning process [15].

6.1. Analogy to the Brain

The most basic components of neural networks are modeled after the structure of the brain. Some neural network structures are not close to the brain and some does not have a biological counterpart in the brain. However, neural networks have a strong similarity to the biological brain and therefore a great deal of the terminology is borrowed from neuroscience [15].

The most basic element of the human brain is a specific type of cell, which provides us with the abilities to remember, think, and apply previous experiences to our every action. These cells are known as neurons, each of these neurons can connect with up to 200000 other neurons. The power of the brain comes from the numbers of these basic components and the multiple connections between them.

All natural neurons have four basic components, which are dendrites, soma, axon, and synapses. Basically, a biological neuron receives inputs from other sources, combines them in some way, performs a generally nonlinear operation on the result, and then output the final result. The Figure 6.1 shows a simplified biological neuron and the relationship of its four components [15].

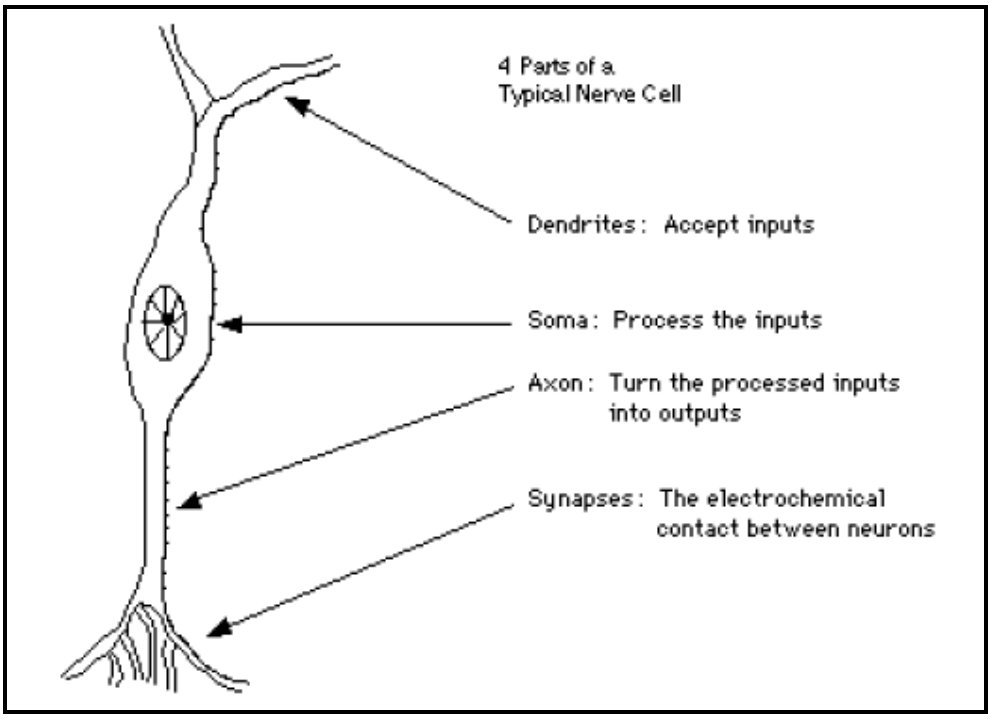

Figure 6.1. Simplified biological Neuron

Information is transported between neurons in form of electrical stimulations along the dendrites. Incoming information that reaches the neuron's dendrites is added up and then delivered along the neuron's axon to the dendrites at its end, where the information is passed to other neurons if the stimulation has exceeded a certain threshold. In this case, the neuron is said to be activated. If the incoming stimulation had been too low, the information will not be transported any further. In this case, the neuron is said to be inhibited. The connections between the neurons are adaptive, what means that the connection structure is changing dynamically. It is commonly acknowledged that the learning ability of the human brain is based on this adaptation.

6.2. The Artificial Neuron

The basic units of neural networks, artificial neurons are much simpler than the biological neuron; Figure 6.2 [15] shows the basics of an artificial neuron.

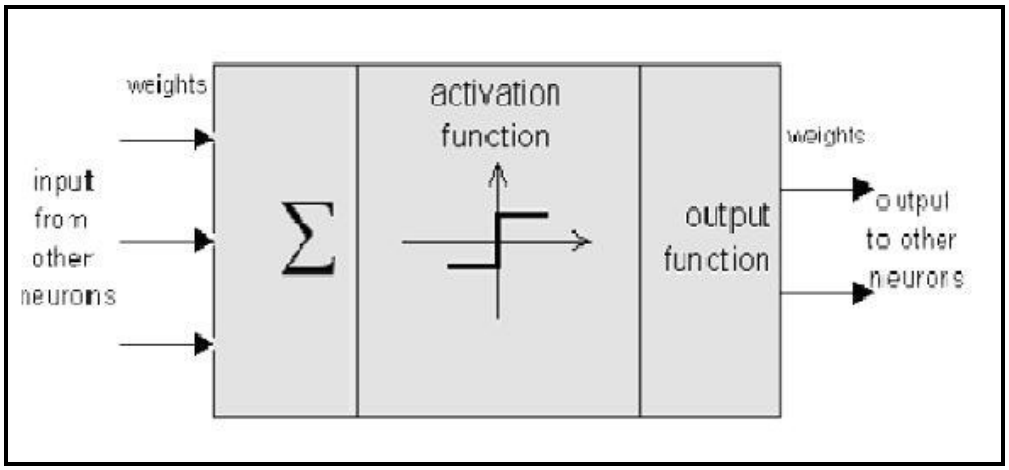

Figure 6.2. Structure of a neuron in a neural net

Information (called the input) is sent to the neuron on its incoming weights. This input is processed by a propagation function that adds up the values of all incoming weights. The resulting value is compared with a certain threshold value by the neuron's activation function. If the input exceeds the threshold value, the neuron will be activated, otherwise it will be inhibited. If activated, the neuron sends an output on its outgoing weights to all connected neurons and so on [16].

In a neural net, the neurons are grouped in layers, called neuron layers. Usually each neuron of one layer is connected to all neurons of the preceding and the following layer (except the input layer and the output layer of the net). The information given to a neural net is propagated layer-by-layer from input layer to output layer through none, one or more hidden layers. Depending on the learning algorithm, it is also possible that information is propagated backwards through the net.

6.3. Design

The developer must go through a period of trial and error in the design decisions before coming up with a satisfactory design. The design issues in neural networks are complex and are the major concerns of system developers.

- Arranging neurons in various layers.
- Deciding the type of connections among neurons for different layers, as well as among the neurons within a layer.
- Deciding the way a neuron receives input and produces output.
- Determining the strength of connection within the network by allowing the network learns the appropriate values of connection weights by using a training data set.

The process of designing a neural network is an iterative process; Figure 6.3 describes its basic steps [16].

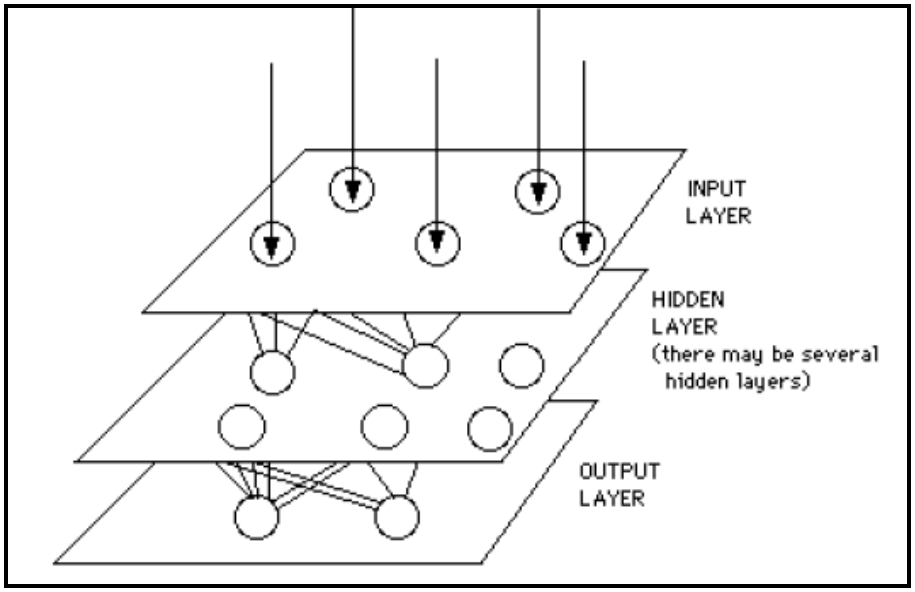

Figure 6.3. Design Layers of Neural Network

6.3.1. Layers

Biologically, neural networks are constructed in a three dimensional way from microscopic components. These neurons seem capable of nearly unrestricted interconnections. This is not true in any man-made network. Artificial neural networks are the simple clustering of the primitive artificial neurons. This clustering occurs by creating layers, which are then connected to one another. How these layers connect may also vary.

Basically, all artificial neural networks have a similar structure of topology. Some of the neurons interface the real world to receive its inputs and other neurons provide the real world with the network's outputs. All the rest of the neurons are hidden form view.

As the figure 6.3 shows, the neurons are grouped into layers. The input layer consists of neurons that receive input form the external environment. The output layer consists of neurons that communicate the output of the system to the user or external environment. There are usually a number of hidden layers between these two layers; Figure 6.3 shows a simple structure with only one hidden layer.

When the input layer receives the input its neurons produce output, which becomes input to the other layers of the system. The process continues until a certain condition is satisfied or until the output layer is invoked and fires their output to the external environment.

6.3.2. Communication and types of connections

Neurons are connected via a network of paths carrying the output of one neuron as input to another neuron. These paths is normally unidirectional, there might however be a two-way connection between two neurons, because there may be another path in reverse direction. A neuron receives input from many neurons, but produces a single output, which is communicated to other neurons.

The neuron in a layer may communicate with each other, or they may not have any connections. The neurons of one layer are always connected to the neurons of at least another layer.

6.3.2.1. Inter-Layer Connections

There are different types of connections used between layers; these connections between layers are called inter-layer connections [16].

- **Fully connected** Each neuron on the first layer is connected to every neuron on the second layer.
- **Partially connected** A neuron of the first layer does not have to be connected to all neurons on the second layer.
- **Feed forward** The neurons on the first layer send their output to the neurons on the second layer, but they do not receive any input back form the neurons on the second layer.
- **Bi-directional** There is another set of connections carrying the output of the neurons of the second layer into the neurons of the first layer.

Feed forward and bi-directional connections could be fully- or partially connected.

- **Hierarchical** If a neural network has a hierarchical structure, the neurons of a lower layer may only communicate with neurons on the next level of layer.
- **Resonance** The layers have bi-directional connections, and they can continue sending messages across the connections a number of times until a certain condition is achieved.

6.3.2.2. Intra-Layer Connections

In more complex structures the neurons communicate among themselves within a layer, this is known as intra-layer connections. There are two types of intra-layer connections.

- **Recurrent** The neurons within a layer are fully- or partially connected to one another. After these neurons receive input form another layer, they communicate their outputs with one another a number of times before they are allowed to send their outputs to another layer. Generally some conditions among the neurons of the layer should be achieved before they communicate their outputs to another layer.
- **On-center/off surround** A neuron within a layer has excitatory connections to itself and its immediate neighbors, and has inhibitory connections to other

neurons. One can imagine this type of connection as a competitive gang of neurons. Each gang excites it and its gang members and inhibits all members of other gangs. After a few rounds of signal interchange, the neurons with an active output value will win, and is allowed to update its and its gang member's weights. (There are two types of connections between two neurons, excitatory or inhibitory. In the excitatory connection, the output of one neuron increases the action potential of the neuron to which it is connected. When the connection type between two neurons is inhibitory, then the output of the neuron sending a message would reduce the activity or action potential of the receiving neuron. One causes the summing mechanism of the next neuron to add while the other causes it to subtract. One excites while the other inhibits.)

6.3.3. Learning

The brain basically learns from experience. Neural networks are sometimes called machine learning algorithms, because changing of its connection weights (training) causes the network to learn the solution to a problem. The strength of connection between the neurons is stored as a weight-value for the specific connection. The system learns new knowledge by adjusting these connection weights.

The learning ability of a neural network is determined by its architecture and by the algorithmic method chosen for training. The training method usually consists of one of three schemes:

- **Unsupervised learning** The hidden neurons must find a way to organize them without help from the outside. In this approach, no sample outputs are provided to the network against which it can measure its predictive performance for a given vector of inputs.
- **Reinforcement learning** This method works on reinforcement from the outside. The connections among the neurons in the hidden layer are randomly arranged, then reshuffled as the network is told how close it is to solving the problem. Reinforcement learning is also called supervised learning, because it

requires a teacher. The teacher may be a training set of data or an observer who grades the performance of the network results.

- **Back propagation** This method is proven highly successful in training of multilayered neural nets. The network is not just given reinforcement for how it is doing on a task. Information about errors is also filtered back through the system and is used to adjust the connections between the layers, thus improving performance.

Both unsupervised and reinforcement suffers from relative slowness and inefficiency relying on a random shuffling to find the proper connection weights.

6.3.3.1. Offline or Online

One can categorize the learning methods into yet another group, off-line or on-line. When the system uses input data to change its weights to learn the domain knowledge, the system could be in training mode or learning mode. When the system is being used as a decision aid to make recommendations, it is in the operation mode, this is also sometimes called recall.

- **Off-line** In the off-line learning methods, once the systems enters into the operation mode, its weights are fixed and do not change any more. Most of the networks are of the off-line learning type.
- **On-line** In on-line or real time learning, when the system is in operating mode (recall), it continues to learn while being used as a decision tool. This type of learning has a more complex design structure.

6.4. Basic Structures in Neural Networks

6.4.1. Perceptron

The perceptron is the basic processing element. It has inputs that may come from the environment or may be the outputs of other perceptrons. A connection weight $w_j \in \Re$ is associated with each input $x_j \in \Re$, j =1,…,d and there is output y which is simply a weighted sum of the inputs [17].

$$y = \sum_{j-1}^{d} w_j x_j + w_o \tag{6.1}$$

w_0 is the intercept value or the weight coming from extra bias unit, x_0, which is always +1. The output of the perceptron y can be represented as a dot product $y = w^T x$ of $w = [w_0, w_1, ..., w_d]^T$ and $x = [1, x_1, ..., x_d]^T$ which are vectors that include also the bias weight and input. To implement a task, it has to be learned the weights w, such that correct outputs are generated given the inputs.

6.4.1.1. Training a Perceptron

The neural network perceptron is a just a way of implementing a hyperplane. The weight values calculated offline for a data sample can be plugged in a network and then perceptron can be used to calculate the output values. In training neural networks, online learning is generally used, it is not given whole sample but instances one by one given to the network and it is preferred that parameters of the network are adapted themselves after each instance in time [17].

When online learning is used it is not written the error function over the whole sample but on individual instances. Random initial weights are assigned to the parameters of the network and at each iteration these weights are adjusted to minimize the error

without forgetting previously learned. If the error function is differentiable, gradient descent can be used.

In regression, the error on the single instance pair with index t, (x^t, r^t) is

$$E_t(w \mid x^t, r^t) = \frac{1}{2}(r^t - y^t)^2 = \frac{1}{2}\left[r^t - (w^t x^t)\right]^2 \tag{6.2}$$

The online update rule is

$$\Delta w_j^t = \eta(r^t - y^t)x_j^t \text{ st. } j = 0,\ldots,d \tag{6.3}$$

η is the learning factor, which is gradually decreased in time for converge. This methodology is known as stochastic gradient descent.

Learning factor **η** decides the magnitude of the update, if it is too large, the system has a very short memory since updates depend much on recent instances, if it is too small, much iteration are needed to converge.

6.4.2. Training Procedures

6.4.2.1. Improving Convergence

The gradient descent methodology is simple and local but has few disadvantages; it uses the presynaptic and postsynaptic units and the error. When online learning is used gradient descent converges very slowly. Therefore, simple techniques called momentum and adaptive learning rate are used to improve the performance.

6.4.2.2. Momentum

Incorporating the previous update in the current change of weight as if there is a momentum due to previous updates

$$\Delta w_i^t = -\eta \frac{\partial E^t}{\partial w_i} + \alpha \Delta w_i^{t-1} \tag{6.4}$$

Parameter α takes values between 0.5 and 1.0. This approach is especially useful when online learning is used. Thus, it uses the effect of averaging and smoothes the trajectory during convergence.

6.4.2.3. Adaptive Learning Rate

In gradient descent, the learning factor η, which determines the magnitude of change to be made in the parameter, can be made adaptive to the cases. It is kept large when learning fastens and is kept decreased when learning slows down.

$$\Delta \eta = \begin{cases} +a, E^{t+T} < E^t \\ -b\eta, otherwise \end{cases} \tag{6.5}$$

6.4.2.4. Overtraining

Complex models lead to poor generalization; when training is continued more than necessary, new parameters are added to the system, the training error decreases but validation error starts to increase beyond a certain level. Learning should be stopped before too late to alleviate the problem of overtraining [17].

6.4.2.5. Starting Weights

The initialization of the starting weights of the network is a crucial point for the performance. In the error surface initial points takes the error function to the global minima. Learning does not take place if initial weights are set to 0. Weights that are randomly chosen from a single distribution ensure uniform learning and generalization of the network. Therefore weights are chosen between -0.1 and 0.1 randomly.

6.4.2.6. Early Stopping

An alternative way of controlling the complexity of the network is using unseen data which is also called test data. The error measured with respect to test data decreases at first and then increases as the network start to over-fit. Training should be stopped at that point where smallest error occurs. This gives the network to have the best generalization performance.

6.5. Neural Network Models

6.5.1. MLP (Multilayer Perceptron)

A perceptron that has a single layer of weights can only approximate linear functions of the input. Feed-forward networks with intermediate or hidden layers between the input and output layers which are called MLP; do not have limitations like single layer perceptrons.

In case of regression, MLP can approximate nonlinear functions of the input. Input x is fed to the input layer (including the bias), the activation propagates in the forward direction, and the values of the hidden unit's z_h are calculated. Each hidden unit is also a perceptron and applies the nonlinear sigmoid function to its weighted sum.

$$z_h = sigmoid\left(w_h^T x\right) = \frac{1}{1+\exp\left[-\left(\sum_{j-1}^{d} w_{hj} x_j + w_{h0}\right)\right]} \text{ st. h = 1,…, } H \tag{6.6}$$

$$sigmoid(x) = \frac{1}{1+e^{-x}} \tag{6.7}$$

The outputs (y_i) are perceptrons in the second layer taking the hidden units as their inputs.

$$y_i = v_i^T z = \sum_{h=1}^{H} v_{ih} z_h + v_{i0} \tag{6.8}$$

There are also bias weights in hidden layer which are z_0 and v_{i0}. The input layer of x_i is not in equation (6.8) since output is calculated by the hidden layers weights.

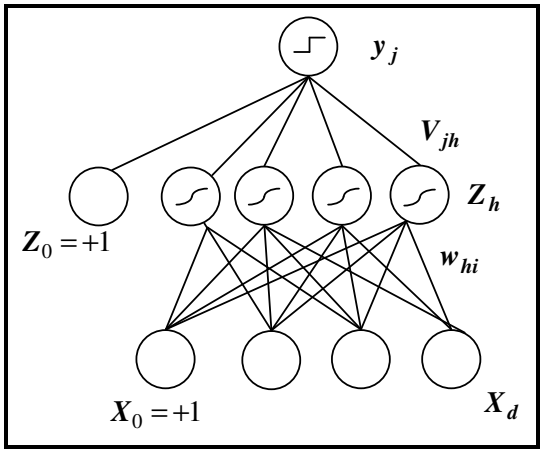

Figure 6.4. A simple regression MLP network

Any arbitrary function with continuous input and outputs can be approximated with multilayer perceptron that can be seen in Figure 6.4 [17].

6.5.1.1. Backpropagation Algorithm

The difference between training a multilayer perceptron with single layer perceptron is; the output is nonlinear function of the input in multilayer one [17]. But if the hidden units are considered as input, second layer is a perceptron and update rule is truly defined. The parameters will be updated, v_{ij} in this case, by using inputs h_j. For the first layer weights, w_{hj}, the chain rule can be used to calculate the gradient.

$$\frac{\partial E}{\partial w_{hj}} = \frac{\partial E}{\partial y_i} * \frac{\partial y_i}{\partial z_h} * \frac{\partial z_h}{\partial w_{hj}} \tag{6.9}$$

The error propagates from the output y to the input x; therefore the algorithm is called the backpropagation [18].

In case of linear regression with single output, calculation is

$$y_t = \sum_{h=1}^{H} v_h z_h^t + v_0 \text{ where } z_h \text{ is calculated by (6.6)} \tag{6.10}$$

The error function over the whole sample in this case is

$$E(W, v \mid X) = \frac{1}{2} \sum_t \left(r^t - y^t\right)^2 \tag{6.11}$$

The second layer is another perceptron which hidden units as the inputs, least squares rule is used to update the second layer weights.

$$\Delta v_h = \eta \sum_t \left(r^t - y^t\right) z_h^t \tag{6.12}$$

The first layer are also perceptrons with the hidden units as the output units but in updating the first layer weights, least squares rule can not be applied directly since we do

not have desired output specified for the hidden units. This is the situation where the chain rule is used.

$$\Delta w_{hj} = \eta \sum_t \left(r^t - y^t\right) v_h z_h^t \left(1 - z_h^t\right) x_j^t \tag{6.13}$$

Starting weights w_{hj} and v_h are initialized to small random values for not saturating sigmoids or in other words for starting learning. For the beginning normalizing inputs is preferred since they can have 0 mean and unit variance.

In batch learning the changes are accumulated over all patterns and the update is made once after a complete pass over the whole training set is made. A complete pass over all the patterns in the training set is called an epoch. On the other hand in online learning, weights are updated after each pattern, thus stochastic gradient descent is implemented. Online learning converges faster because there may be similar patterns in the dataset.

6.5.2. Recurrent Neural Networks (RNN)

In recurrent network structure, additional to the feedforward connections, units have self connections or connections to units in the previous layers. This recurrence ensures a short term memory and permits the network remember what happened in the past.

Frequently partial recurrent network is used where a limited number of recurrent connections are added to a multilayer perceptron.

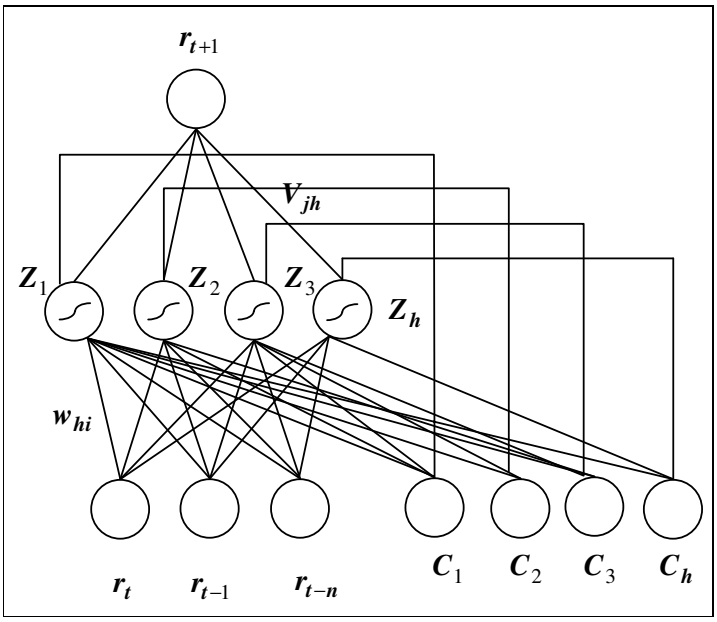

Figure 6.5. A simple recurrent network

Such a recurrent structure in Figure 6.5 has ability of nonlinear approximation and temporal representation ability of recurrence. It is also possible to have hidden units in the recurrent backward connections, these being known as context units [17].

In case of sequences have small maximum length; unfolding in time can be used to convert an arbitrary recurrent network to an equivalent network. The resulting network can be trained with backpropagation algorithm. The solution is backpropagation through time [19], summing up the different weights changes in time and changing the weight by the average. The problem with this approach is the memory requirement if the length of the sequence is large. For training recurrent networks without unfolding real time recurrent learning [20] is used and it has the advantage of arbitrary length sequences.

6.5.3. Local Models

The neural network structures where the first layer contains locally receptive units that respond to instances in a localized region of the input space are called local models. The second layer learns the regression or classifications function for these local regions.

Divide and conquer is a principle with wide applicability throughout applied mathematics. Divide and conquer algorithms attack a complex problem by dividing it into simpler problems whose solutions can be combined to yield a solution to the complex problem. These algorithms fit surfaces to data by explicitly dividing input space into a nested sequence of regions, and by fitting simple surfaces (e.g. constant functions) within these regions. They have convergence times that are often orders of magnitude faster than gradient based neural network algorithms [21].

Another way for function approximation is to divide the input space into local patches and learn separate fit in each local patch. If the fit in a local patch is constant, then the technique is called the radial basis function (RBF) networks; if it is linear function of the input, it is called the mixture of experts (MoE).

6.5.3.1. Radial Basis Functions

In a multilayer perceptron input is encoded by simultaneous activation of many hidden units, this is called distributed representation. Another possibility is to have a local representation where for a given input, only one or a few units are active. This is achieved by partitioning the input space by these locally tuned units and selection of only certain inputs. The part of the input space where a unit has nonzero response is called its receptive field. The input space is then paved with such units.

The concept of locality is ensured by a distance function to measure the similarity between the given input x and the position of unit h, m_h. This measure is frequently taken as the Euclidean distance, $\|x - m_h\|$. The response function is chosen to have a maximum where $x = m_h$ and decreasing as they get less similar. Gaussian function is commonly used.

$$p_h^t = \exp\left[-\frac{\|x^t - m_h\|^2}{2s_h^2}\right] \tag{6.14}$$

Equation (6.14) which is called radially symmetric basis function, m_j and s_j respectively denote the center and the spread of the local unit j.

The reason why it is defined such a local basis function is assigning a basis function p_h^t to each cluster of the input data and it becomes nonzero if instance x^t belongs to cluster h. Any of the online competitive methods can be used to find the centers, m_h. In this thesis we use the statistical clustering method, EM (Expectation Maximization) on gaussian mixtures.

Expectation Maximization (EM) algorithm; is used in maximum likelihood estimation where the problem involves two sets of random variables of which one, X, is observable and the other Z, is hidden. The aim of the algorithm is to find the parameter vector Φ that maximizes the likelihood of the observed values of X, $\xi(\Phi \mid X)$. But in the cases where this is not feasible, we associate the extra hidden variables Z and express the underlying model using both, to maximize the likelihood of the joint distribution of X and Z, the complete likelihood $\xi_c(\Phi \mid X, Z)$.

Since the Z values are not observed, it is not possible to work directly with the complete data likelihood ξ_c, instead it is worked with its expectations, ς, given X and the current parameter values Φ^l, where l indexes iteration. This is expectation (E) step of the algorithm. Then in the maximization (M) step, we look for new parameter values, Φ^{l+1}, which maximizes this [17].

$$\text{E-step: } \varsigma(\Phi \mid \Phi^l) = E\left[\zeta_c(\Phi \mid X, Z) \mid X, \Phi^l\right] \tag{6.15}$$

$$\text{M-step: } \Phi^{l+1} = \arg\max_{\Phi} \varsigma(\Phi, \Phi^l) \tag{6.16}$$

It is proved that an increase in ς implies an increase in the incomplete likelihood [22].

When we turn to radial basis functions, p_h^t h=1,…, H define a new H-dimensional space and form a new representation of x^t. It can also be used b_h^t to code the input but b_h^t take values of 0 or 1, on the other hand p_h^t have additional advantage that they code the distance to their center by a value in (0,1). If input space is high dimensional many more local units are needed when compared with distributed representation.

In case of supervised learning, we can then use this new local representation as the input. If we use a perceptron, we have

$$y_t = \sum_{h=1}^{H} w_h p_h^t + w_0 \tag{6.17}$$

H is the number of basis functions. This structure is called a radial basis function (RBF) network [23]. H is the complexity parameter, like the number of hidden units in distributed representation. In case of unsupervised learning it corresponds to the number of centers.

In RBF structure there is a two stage process, an unsupervised method is used for determining the centers, then a supervised layer is built on top of that. This is also called hybrid learning. Alternatively all parameters can be learned online, including m_h and s_h, in a supervised manner. The radial basis function is differentiable and we can backpropagate, just as we backpropagated in multilayer perceptron to update network parameters. The structure is similar to a multilayer perceptron with p_h are the hidden units, m_h and s_h are the first layer parameters, the gaussians are the activation functions in the hidden layer, and w_h are the second layer weights like in Figure 6.6 [17].

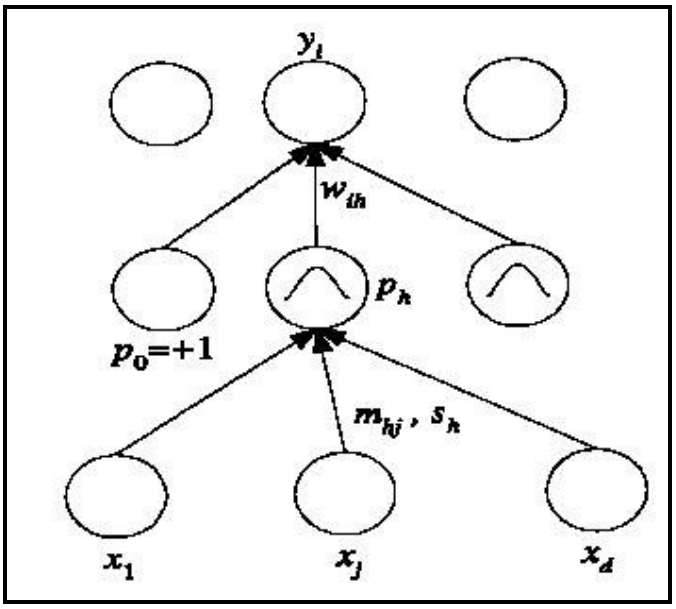

Figure 6.6. RBF network

In case of regression the batch error is

$$E\left(\{m_h, s_h, w_{ih}\}_{i,h} \mid X\right) = \frac{1}{2}\sum_t \sum_i \left(r_i^t - y_i^t\right)^2 \tag{6.18}$$

$$y_i^t = \sum_{h=1}^{H} w_{ih} p_h^t + w_{i0} \tag{6.19}$$

By applying gradient descent, the update rule for the second layer weights is composed.

$$\Delta w_{ih} = \eta \sum_t \left(r_{ii}^t - y_{ii}^t\right) p_h^t \tag{6.20}$$

Equation (6.20) is the usual perceptron rule, with p_h are the inputs. p_h do not overlap much and at each iteration only a few p_h are nonzero and only their w_h are

updated. That is why RBF networks learn very fast, and faster than multilayer perceptrons that use a distributed representation.

6.5.3.2. Normalized Basis Functions

In RBF structure, for an input, it is possible that entire p_h are 0. In some cases, it is needed to have a normalization step to make sure that the values of the local units sum up to 1, thus for any input there is at least one non zero unit.

$$g_h^t = \frac{p_h^t}{\sum_{l=1}^{H} p_l^t} \tag{6.21}$$

Taking p_h as $p(x \mid h)$, g_h correspond to $p(h \mid x)$, the posterior probability that x belongs to unit h. g_h can be thought as a classifier in itself, choosing the responsible unit for a given input.

$$y_i^t = \sum_{h=1}^{H} w_{ih} g_h^t \tag{6.22}$$

In normalized case there is no need for a bias term since there is at least one nonzero g_h for each x. Using g_h instead of p_h does not imply any more parameters; it only couples the units together; g_h depends on the centers and spreads of all the units. Update rule in case of regression is:

$$\Delta w_{ih} = \eta \sum_t \left(r_i^t - y_i^t\right) g_h^t \tag{6.23}$$

6.5.3.3. Mixture of Experts

In RBF structure, corresponding to each local patch constant fit is given. In the case where for any input, we have one g_h 1 and all others 0, we have a piecewise constant approximation where for output i, the local fit by patch h is given by w_{ih}. From the Taylor expansion, for each point, the function can be written as

$$f(x) = f(a) + (x-a)f'(a) + \ldots \tag{6.24}$$

Therefore constant approximation is good if x is close enough to a and $f'(a)$ is close to 0, that is, if $f(x)$ is flat around a. if this is not the case, we need to divide the space into large number of patches, which is particularly serious when the input dimensionality is high.

An alternative is to have a piecewise linear approximation by taking into account the next term in Taylor expansion, namely, the linear term [24].

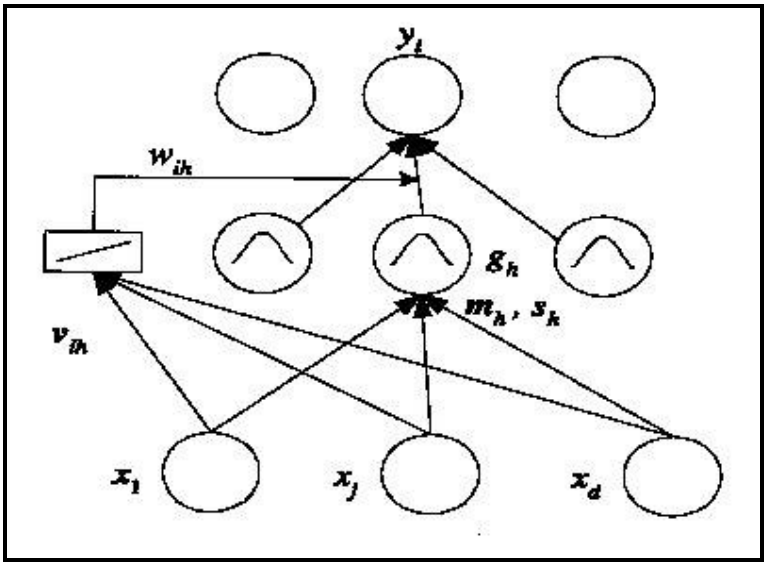

Figure 6.7. Mixture of Experts Architecture

The mixture of experts in Figure 6.7 can be seen as an RBF network where the second layer weights are output of linear models. We use equation (6.22) which is the same as equation in normalized RBF structure, but here, w_{ih}, the contribution of patch h to output i is not a constant but linear function of the input.

$$w_{ih}^t = v_{ih}^T x^t \tag{6.25}$$

In mixture of experts, v_{ih} is the parameter vector that defines the linear function and includes a bias term. The unit activations can be taken like equation (6.21) in normalized basis functions. This can be seen as an RBF network except that the second layer weights are not constants but are the outputs of linear models. In another perspective that is visualized in Figure 6.8, it is considered that w_h as linear models, each taking the input, and is called experts. g_h is considered to be the outputs of a gating network [24].

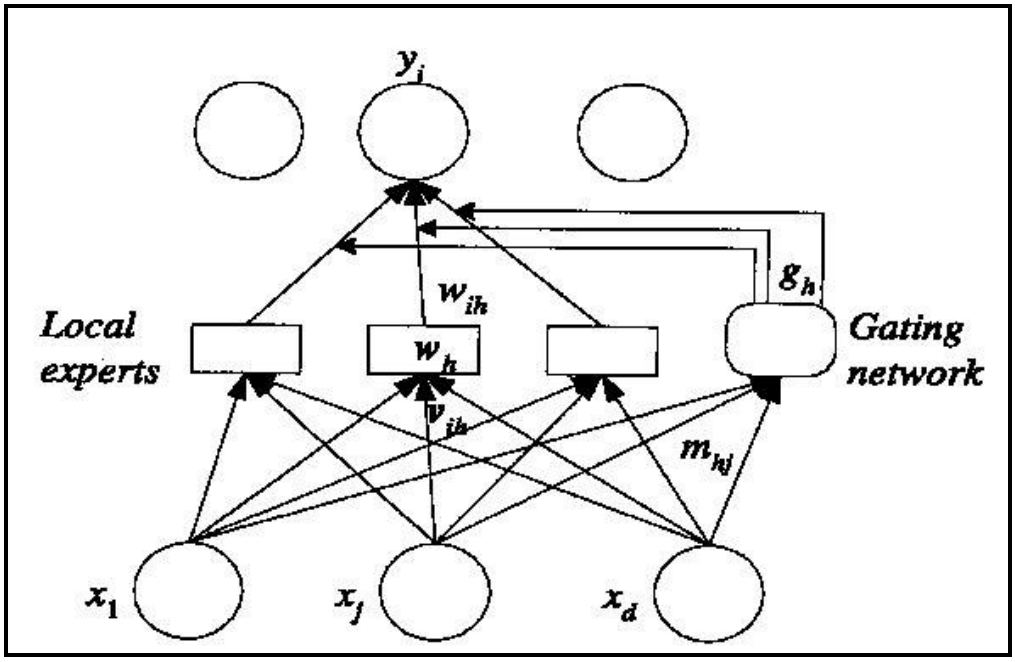

Figure 6.8. Jacobs perspective Mixture of Experts

Considering the gating network in this manner, any classifier can be used in gating. When input is high dimensional, using local gaussian unit may require a large number of experts and it is proposed to take [24]

$$g_h^t = \frac{\exp\left[m_h^t x^t\right]}{\sum_l \exp\left[m_l^T x^t\right]} \tag{6.26}$$

In this case m_h no longer represents centers but hyperplanes, and as such include bias values. This gating network is implementing a classification where it is dividing linearly the input region for which expert h is responsible from the expertise regions of the other experts [17].

The mixture of experts is a general architecture for combining multiple models; the experts and gating may be nonlinear instead of linear perceptrons.

6.5.3.3.1. Cooperative Experts

In the cooperative mixture of experts structure, y_i^t is given by equation (6.22), and the goal is to make y_i^t close as possible to the required output, r_i^t. Therefore in regression the error function becomes

$$E\left(\{m_h, s_h, w_{ih}\}_{i,h} \mid X\right) = \frac{1}{2}\sum_t \sum_i \left(r_i^t - y_i^t\right)^2 \tag{6.27}$$

Using gradient descent, second layer (expert) weight parameters are updated as

$$\Delta v_{ih} = \eta \sum_t \left(r_i^t - y_i^t\right) g_h^t x^t \tag{6.28}$$

Compared with equation (6.23), the only difference is that this new update is a function of the input.

6.5.3.3.2. Competitive Experts

In the competitive experts case, $y_{ih}^t = w_{ih}^t = v_{ih} x^t$, again using gradient descent we get

$$\Delta v_{ih} = \eta \sum_t \left(r_i^t - y_{ih}^t \right) f_h^t x^t \tag{6.29}$$

$$\Delta m_h = \eta \sum_t \left(f_h^t - g_h^t \right) x^t \tag{6.30}$$

When cooperative and competitive models are compared for classification tasks; the cooperative model is generally more accurate but the competitive version learns faster [25]. The reason behind that; the cooperative case, models overlap more and implement a smoother approximation, and thus it is preferable in regression problems. The competitive model makes a harder split; generally only one expert is active for an input and therefore learning is faster.

6.5.3.4. Hierarchical Mixture of Experts

In hierarchical mixture of experts, each expert is replaced with a complete system of mixture of experts in a recursive manner [21]. This architecture may be seen as a decision tree where gating networks can be seen as decision nodes. When the gating network is linear, the structure seems like the linear multivariate decision tree. The difference is that the gating network does not make a hard decision but takes a weighted sum of contributions coming from children. Leaf nodes are linear models, and their predictions are averaged and propagated up the tree. The root gives the final output, which is a weighted average of all of the leaves [17].

7. PERFORMANCE METRICS

7.1. Basic Performance Metrics

Clearly, one can not determine whether a forecasting model is good or not based upon only one forecast and one realization. Thus in practice, forecasts would usually be produced for the whole of the out of sample period, which would then be compared with the actual values, and the difference between them aggregated in some way. The forecast error for observation i is defined as the difference between actual values for observation i and forecast made for it. The forecast error, defined in this way, will be positive (negative) if the forecast was too low (high). Therefore, it is not possible simply to sum the forecasts errors, since the positive and negative errors will cancel one another out. Thus, before forecast errors are aggregated, they are usually squared or the absolute value taken, which renders them all positive. In this case the mean squared error, MSE, and mean absolute error, MAE, are used as basic performance metrics [10].

$$mse = \frac{1}{N}\sum_{t=1}^{N}(p_t - \hat{p}_t)^2 \tag{7.1}$$

$$mae = \frac{1}{N}\sum_{t=1}^{N}\left|p_t - \hat{p}_t\right| \tag{7.2}$$

MSE provides a quadratic loss function, and so may be particularly useful in situations where large forecast errors are disproportionately more serious than smaller errors. This may, however, also be viewed as a disadvantage if large errors are not disproportionately more serious, also the same critique could also, of course, be applied to the whole least squares methodology. Indeed when there are outliers present, least absolute values should be used to determine model parameters rather than least squares [26]. It is argued that MAPE is ‘a relative measure that incorporates the best characteristics among the various accuracy criteria’ [27]. Correlation ($corr$) and R^2 are other two measures that are used to help to ensure the fit of the prediction over the actual data.

$$mape = \frac{1}{N}\sum_{t=1}^{N}\left|\frac{p_t - \hat{p}_t}{p_t}\right| \tag{7.3}$$

$$\boldsymbol{corr} = \frac{\sum_{t=1}^{N}(\boldsymbol{r_t} - \bar{\boldsymbol{r}}_t)(\hat{\boldsymbol{r}}_t - \hat{\bar{\boldsymbol{r}}}_t)}{\sqrt{\sum_{t=1}^{N}(\boldsymbol{r_t} - \bar{\boldsymbol{r}}_t)^2}\sqrt{\sum_{t=1}^{N}(\hat{\boldsymbol{r}}_t - \hat{\bar{\boldsymbol{r}}}_t)^2}} \tag{7.4}$$

$$ess = \sum_{t=1}^{N}(r_t - \hat{r}_t)^2, rss = \sum_{t=1}^{N}(\hat{r}_t - \bar{r})^2, tss = \sum_{t=1}^{N}(r_t - \bar{r})^2, R^2 = 1 - \frac{ess}{tss} \tag{7.5}$$

Unfortunately, it is not possible to use MAPE if the series and the forecasts can take on opposite signs. This is due to fact that the prediction and the actual value may, purely by coincidence, take on values that are almost equal and opposite, thus almost canceling each other out in the denominator. This leads to extremely large and erratic error values of MAPE.

MAPE has the advantage that for a random walk in the levels (i.e. a zero forecast), the criterion will take the value one, so that if a forecasting model gives a MAPE smaller than one, it is superior to the random walk model. In fact the criterion is also not reliable if the series can take on absolute values less than one [10].

7.2. Benchmarks

The need for good benchmarks when evaluating stock prediction algorithms can not be overestimated. What is good and bad performance depends on the alternatives. Even a prediction algorithm that loses money for some period of time may turn out to be successful when compared to other alternatives. We will discuss three benchmarks commonly used for evaluating prediction algorithms.

7.2.1. Naïve Prediction of Stock Prices

The naïve predictor asserts today's price as the best estimate for tomorrow's price. This is direct consequence of Random Walk Hypothesis (RWH). This way of measuring the goodness of a predictor should always be considered [28].

7.2.2. Naïve Prediction of Stock Returns

The naïve predictor asserts today's return as the best estimate for tomorrow's return. This naïve prediction is formed from the observation of a one-step memory in the price generation process. Autocorrelation of stock returns exhibits a significant first lag component that indicates a correlation between adjacent returns. It is a good idea to measure the goodness of a predictor with this naïve predictor [28].

7.2.3. Buy-And-Hold

The Buy-and-hold test finds out whether the net profit result using a prediction algorithm or strategy is due to prediction accuracy or merely to a general market trend [28]. The buy and hold return r_b for a time period $\{t,,,t+n\}$of a stock is defined as

$$r_b = \frac{(y_{t+n} - y_t)}{y_t} \tag{7.6}$$

7.3. Metrics for Testing

The predictions of stock prices for time t are expressed by the time series $\{\hat{y}_t, t = 1,...,N\}$. The actual prices are denoted by the time series $\{y(t), t = -d+1, -d+2,...,0,1,...,N\}$. It is assumed that the d first points in $y(t)$ are used in the initial upstart phase.

7.3.1. Predicting Stock Prices

The Theil coefficient of inequality provides a measure of the model performance relative to the naive price predictor.

$$T_p = \frac{\sqrt{\sum_{t=!}^{N} \left(y(t) - \hat{y}(t)\right)^2}}{\sqrt{\sum_{t=1}^{N} \left(y(t) - y(t-1)\right)^2}} \tag{7.7}$$

For $T_p > 1$ the predictor is worse than the naive price predictor while $T_p < 1$ implies that the predictor is making better predictions. The naive predictor asserts that the best estimate of tomorrow's price $y(t+1)$ is today's price $y(t)$. The Theil coefficient is often referred to as the information coefficient or the t-test.

7.3.2. Predicting Returns

The Theil coefficient compares the performance to that of the naive price predictor. The predictions of returns at time t are denoted by the time series $\{\hat{\Re}(t), t = 1,..., N\}$. The actual returns are denoted by the time series $\{\Re(t), t = 1,..., N\}$. The Theil coefficient for returns provides a measure of the prediction performance relative to the naive return predictor.

$$T_r = \frac{\sqrt{\sum_{t=1}^{N} \left(\Re(t) - \hat{\Re}(t)\right)^2}}{\sqrt{\sum_{t=1}^{N} \left(\Re(t) - \Re(t-1)\right)^2}} \tag{7.8}$$

For $T_r > 1$ the predictor is worse than the naïve return predictor while $T_r < 1$ implies that the predictor is making better predictions.

7.3.3. Hit Rate

The hit rate of a predictor indicates how often the sign of the return, is correctly predicted. It is computed as the ratio between the numbers of correct non-zero predictions $\hat{\Re}(t)$ and the total number of non zero moves in the stock time series.

$$H = \frac{\left|\left\{t \mid \Re(t)\hat{\Re}(t) > 0, t = 1,...,N\right\}\right|}{\left|\left\{t \mid \Re(t)\hat{\Re}(t) \neq 0, t = 1,...,N\right\}\right|} \tag{7.9}$$

$\hat{\Re}(t)$ is the prediction of $\Re(t)$ computed at time $t-1$. The norm of the set in the definition is simply the number of elements in the set.

In analogy with the Theil coefficient the corresponding measure for the naïve return predictor is proposed:

$$H_N = \frac{\left|\left\{t \mid \Re(t)\Re(t-1) > 0, t = 1,...,N\right\}\right|}{\left|\left\{t \mid \Re(t)\Re(t-1) \neq 0, t = 1,...,N\right\}\right|} \tag{7.10}$$

Finally, the ratio between the two hit rates provides the predictor's hit rate performance relative to the naïve return predictor:

$$H_0 = \frac{H}{H_N} \tag{7.11}$$

This entity H_0 which we call "Relative Hit Rate" compares the hit rate of the predictor to that of the naïve return predictor. For $H_0 < 1$ the predictor is worse than the naive return predictor, while $H_0 > 1$ implies that the predictor is making better predictions.

7.3.4. Mean Profit per Trade $P_{\Re}$

The ultimate measure of success for a prediction algorithm is its ability to make profit when applied to real trading. Mean profit per trade is calculated as simulating a trading environment. This measure is calculated by trading at each time interval according to the prediction using a trading rule. Buys and sells are executed according to the trading rule and the profit is accumulated for the whole time period. Mean profit per trade is calculated as

$$P_{\Re} = 100\frac{1}{h}\frac{1}{N-h}\sum_{t=h+1}^{N} sign(p_t - p_{t-h})\frac{p_t - p_{t-h}}{p_{t-h}} \tag{7.12}$$

8. EXPERIMENTS AND DISCUSSION

In this chapter, experiments regarding neural networks for financial time series prediction have been studied. Firstly the time series data sets we have used are described. The statistical and market related properties of the series have been investigated. After presenting the data sets, the models that are proposed for the prediction problem of stock markets are studied.

8.1. Financial Time Series Data Sets

In this research we have used daily weighted indices from Turkish and Brazilian Stock Exchanges. In experiments, Istanbul Stock Exchange (ISE) National-100, National-30 and Brazilian National BVSP indices are used. The whole data used in this study is collected from company Finnet[1] in Turkey. Turkish stock market index National-100 and Brazilian National index BVSP data are between January 2001 and February 2006. Turkish stock market index National-30 data is between January 1998 and February 2006.

In Table 8.1, the beginning and the end of the return series have been summarized.

Table 8.1. Return series

Series	Start Date	End Date	Observations
ISE National – 100	02/01/2001	22/02/2006	1262
ISE National – 30	05/01/1998	22/02/2006	1987
BVSP	02/01/2001	22/02/2006	1265

The reason behind the selection of the same period for both ISE 100 and BVSP indices is investigating interaction of these two emergent markets. ISE National – 30 indexes is mainly used to show the prediction power of the neural network architecture in financial time series.

[1] Finnet is located in Istanbul Turkey.

The similarity of Turkish and Brazilian market depends on historical regulations of Turkish government. During financial liberalization of the 1980-1990s, Turkey imported many constitutional laws from Latin American economies especially from Brazil. Therefore these two emergent markets show highly similar reactions to economical regulations. Both markets are highly volatile and have non-linearity with high noise because of the political and economical instability.

8.2. Models

In this research, we have investigated the use of neural networks in financial time series prediction concentrating on the stock market data of Turkey and Brazil. Stock market series can be studied in three different ways; the price or returns that are calculated from underlying prices can be predicted. Also risk of the market can be predicted based on volatility models. All these prediction problems can be studied separately with respect to regression perspective. We mainly concern the return prediction and interaction of different markets to each other. Obviously since studied markets are highly volatile, volatility factor is taken into consideration in simulations. What we have seen from the past studies [29], Turkish market is chaotic and highly volatile. It includes patterns that can be extracted and used for prediction purposes by the use of ANNs. The success and accuracy of the experiments are measured by the metrics that we have defined in the performance metrics section. In defining the interaction between two international markets we also used ANN and previously defined metrics. In experiments closing index values are used to calculate the log returns. The reason to use log returns in application is to keep the series in constant range by applying normalization.

$$R_t = 100\left[\log\left(\frac{P_t}{P_{t-1}}\right)\right] \tag{8.1}$$

8.3. Methodology

In all of the simulations ANN's architecture Mixture of Experts is used. For prediction purpose sliding window techniques is applied and windows size is chosen to be 20. Same performance metrics are used for all simulations for comparison purpose.

In simulations we will not give detailed explanation for mixture of experts since necessary information is given in neural network section, but we will explain how we used it for time series prediction. We have used cooperative Mixture of Experts structure for modeling and prediction. All of the series used are divided into three parts in order to have good generalization over the data including train, validation and test datasets. Train dataset is used to train the model and learn the system parameters, and then validation data set is used to calibrate the system wide parameters such as the number of hidden units. At the end test set is used to calculate our predefined measures. There are several important factors that affect the performance of the neural network architecture besides the relevance of the data. These are number of hidden units, learning rate and application of cross validation. For the time series, also window size is important, which is the measure we have used to determine how many observations to use for predicting of $t+1$ at time t. All of these network parameters will be given in each simulation in a tabular form.

8.4. Experiments

8.4.1. Case I – ISE National-100 Index

ISE National-100 index data is used between 02 January 2001 and 22 February 2006 for this simulation. From total of 1262 observations 70 percent is used for train set, 20 percent for validation set and 10 percent for test set. Epoch length was decided by training the overall network and stopping at the point where minimum error occurred. At each iteration number of hidden units is increased for improving converges. Mixture of experts' architecture is applied and networks parameters are given in Table 8.2.

Table 8.2. [Case I] Network parameters

Neural Network Parameters	
Start Date	02/01/2001
End Date	22/02/2006
Observations	1262
Train Set	909
Validation Set	227
Test Set	126
Hidden Units	10
Window size	20
Learning Rate	0.0001
Epochs	50

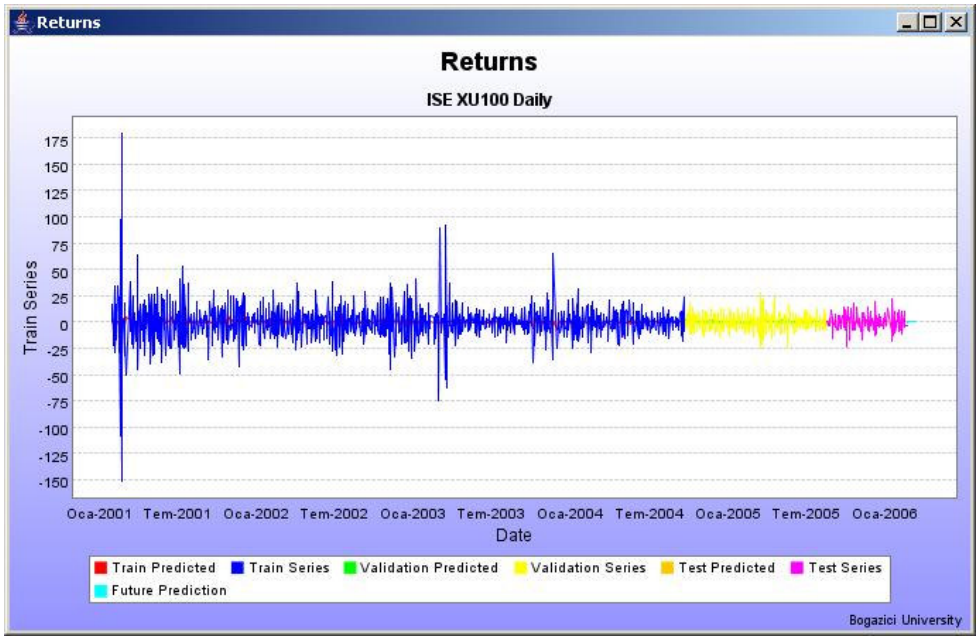

Figure 8.1. ISE National-100

Simulation results are shown in Table 8.3:

Table 8.3. [Case I] Simulation results

	H_R	H_R^+	H_R^-	*MSE*	*MAE*	*RMSE*	*MAPE*	*TIC*	$P_{\Re}$
MoE	*0.9921*	*1.0*	*0.5*	*0.0292*	*0.1454*	*0.1711*	*2.065*	*0.099*	*1.548*

In this simulation only the past returns of ISE National-100 are used to predict the future returns.

8.4.2. Case II – ISE National-100 Index & Volatility

ISE National-100 index data is used between 02 January 2001 and 22 February 2006 for this simulation. From total of 1262 observations 70 percent is used for train set, 20 percent for validation set and 10 percent for test set. Epoch length was decided by training the overall network and stopping at the point where minimum error occurred. At each iteration number of hidden units is increased for improving converges. Mixture of experts' architecture is applied and networks parameters are given in Table 8.4.

Table 8.4. [Case II] Network parameters

Neural Network Parameters	
Start Date	02/01/2001
End Date	22/02/2006
Observations	1262
Train Set	909
Validation Set	227
Test Set	126
Hidden Units	10
Window size	20
Learning Rate	0.0001
Epochs	50

In this simulation not only the past returns of ISE National-100 are used but also the volatility factor is taken into consideration to predict the future returns. Volatility calculated by RiskMetrics™ [30] is used as another input to the model to better identify the pattern and to get more precise future prediction.

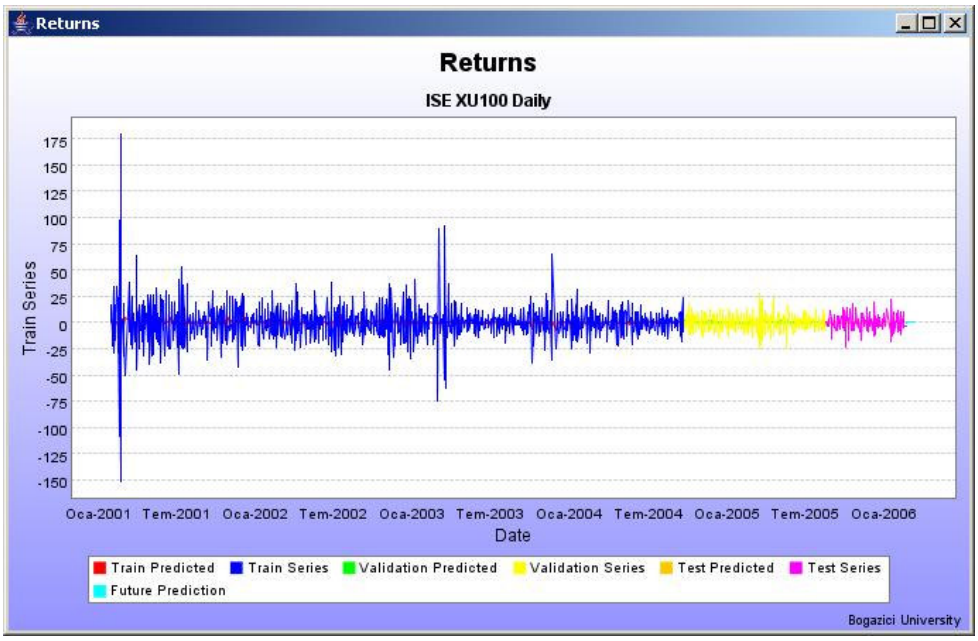

Figure 8.2. ISE National-100 & Volatility

Simulation results are shown in Table 8.5:

Table 8.5. [Case II] Simulation results

	H_R	H_R^+	H_R^-	*MSE*	*MAE*	*RMSE*	*MAPE*	*TIC*	$P_{\Re}$
MoE	*0.9927*	*1.0*	*0.512*	*0.01329*	*0.0915*	*0.1152*	*1.4512*	*0.084*	*1.548*

Thus better performance results are gained with the volatility factor when predicting future returns. Hit rates increase and error measures associated with regression statistics improve significantly.

8.4.3. Case III – ISE National-30 Index

ISE National-30 index data is used between 05 January 1998 and 22 February 2006 for this simulation. From total of 1987 observations 70 percent is used for train set, 20 percent for validation set and 10 percent for test set. Epoch length was decided by training

the overall network and stopping at the point where minimum error occurred. At each iteration number of hidden units is increased for improving converges. Mixture of experts' architecture is applied and networks parameters are given in Table 8.6.

Table 8.6. [Case III] Network parameters

Neural Network Parameters	
Start Date	05/01/1998
End Date	22/02/2006
Observations	1987
Train Set	1432
Validation Set	357
Test Set	198
Hidden Units	10
Window size	20
Learning Rate	0.0001
Epochs	50

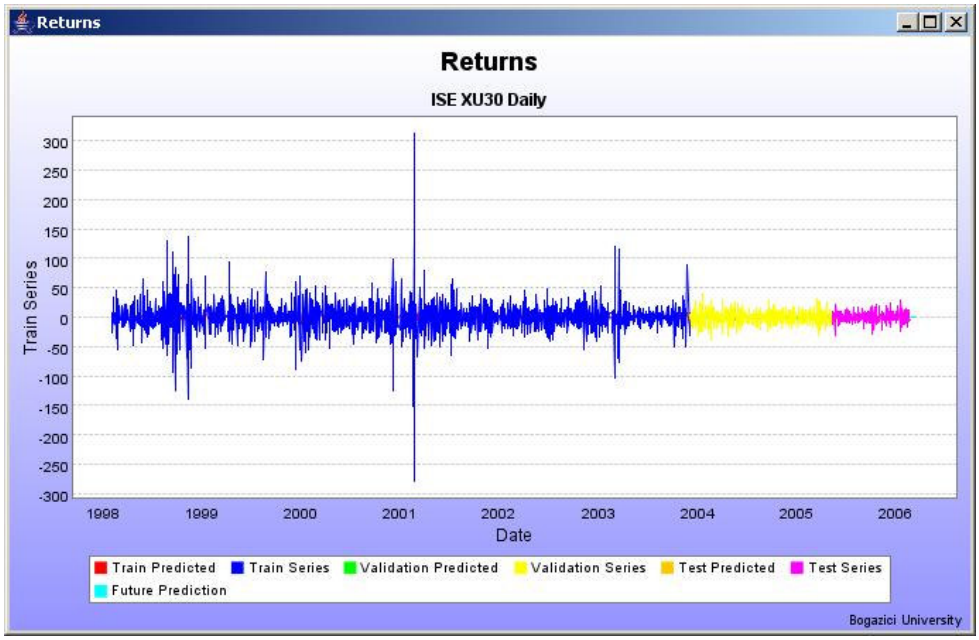

Figure 8.3. ISE National-30

Simulation results are shown in Table 8.7:

Table 8.7. [Case III] Simulation results

	H_R	H_R^+	H_R^-	*MSE*	*MAE*	*RMSE*	*MAPE*	*TIC*	$P_{\Re}$
MoE	*0.8542*	*1.0*	*0.5*	*0.007*	*0.065*	*0.084*	*3.390*	*0.6547*	*14.148*

In this simulation only the past returns of ISE National-30 are used to predict the future returns.

8.4.4. Case IV – ISE National-30 Index & Volatility

ISE National-30 index data is used between 05 January 1998 and 22 February 2006 for this simulation. From total of 1987 observations 70 percent is used for train set, 20 percent for validation set and 10 percent for test set. Epoch length was decided by training the overall network and stopping at the point where minimum error occurred. At each iteration number of hidden units is increased for improving converges. Mixture of experts' architecture is applied and networks parameters are given in Table 8.8.

Table 8.8. [Case IV] Network parameters

Neural Network Parameters	
Start Date	05/01/1998
End Date	22/02/2006
Observations	1987
Train Set	1432
Validation Set	357
Test Set	198
Hidden Units	10
Window size	20
Learning Rate	0.0001
Epochs	50

In this simulation again not only the past returns of ISE National-30 are used but also the volatility factor is taken into consideration to predict the future returns. Volatility calculated by RiskMetrics™ [30] is used as another input to the model to better identify the pattern and to get more precise future prediction.

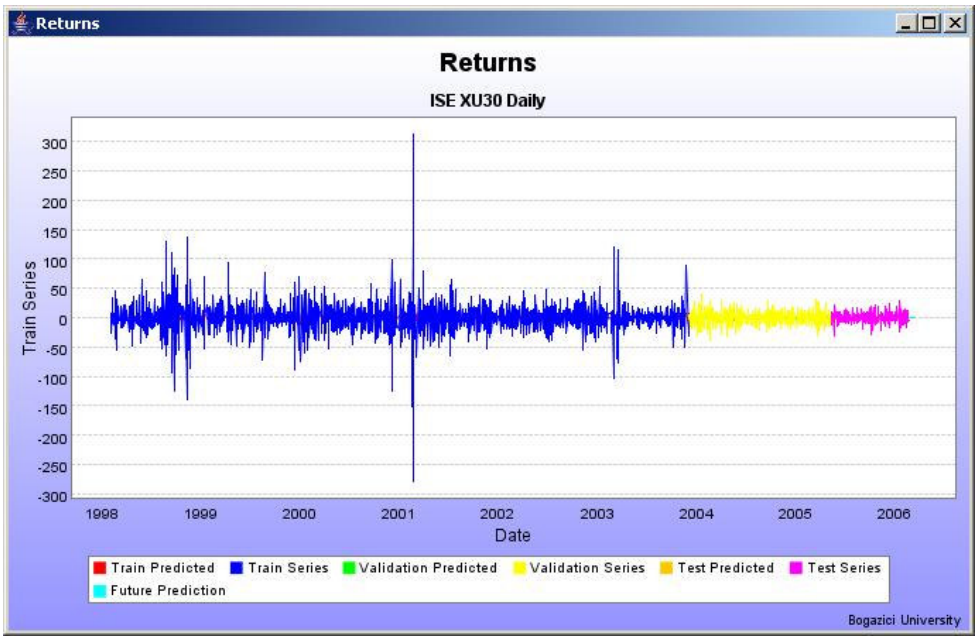

Figure 8.4. ISE National-30 & Volatility

Simulation results are shown in Table 8.9:

Table 8.9. [Case IV] Simulation results

	H_R	H_R^+	H_R^-	*MSE*	*MAE*	*RMSE*	*MAPE*	*TIC*	$P_{\Re}$
MoE	*0.8657*	*1.0*	*0.6*	*0.006*	*0.059*	*0.094*	*2.440*	*0.534*	*17.178*

Thus better performance results are gained with the volatility factor when predicting future returns. Hit rates increase and error measures associated with regression statistics improve significantly.

8.4.5. Case V – BVSP National Index

Brazilian National Index BVSP data is used between 02 January 2001 and 22 February 2006 for this simulation. From total of 1265 observations 70 percent is used for train set, 20 percent for validation set and 10 percent for test set. Epoch length was decided

by training the overall network and stopping at the point where minimum error occurred. At each iteration number of hidden units is increased for improving converges. Mixture of experts' architecture is applied and networks parameters are given in Table 8.10.

Table 8.10. [Case V] Network parameters

Neural Network Parameters	
Start Date	02/01/2001
End Date	22/02/2006
Observations	1265
Train Set	912
Validation Set	227
Test Set	126
Hidden Units	10
Window size	20
Learning Rate	0.0001
Epochs	50

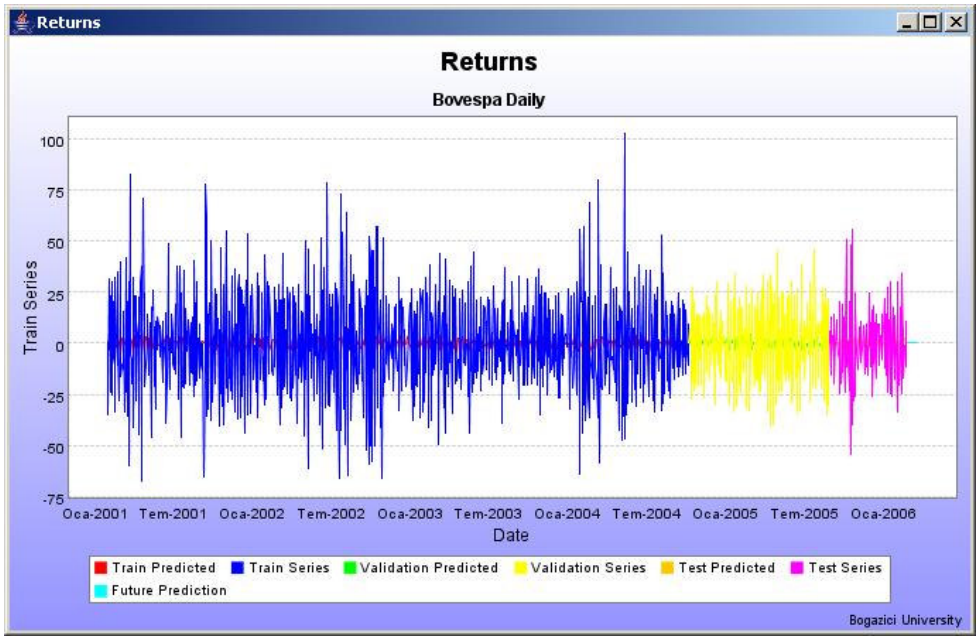

Figure 8.5. BVSP National

Simulation results are shown in Table 8.11:

Table 8.11. [Case V] Simulation results

	H_R	H_R^+	H_R^-	*MSE*	*MAE*	*RMSE*	*MAPE*	*TIC*	$P_{\Re}$
MoE	*0.6614*	*1.0*	*0.0227*	*0.0353*	*0.1441*	*0.1881*	*4.5399*	*0.6358*	*25.779*

In this simulation only the past returns of BVSP are used to predict the future returns.

8.4.6. Case VI – BVSP National Index & Volatility

BVSP index data is used between 02 January 2001 and 22 February 2006 for this simulation. From total of 1265 observations 70 percent is used for train set, 20 percent for validation set and 10 percent for test set. Epoch length was decided by training the overall network and stopping at the point where minimum error occurred. At each iteration number of hidden units is increased for improving converges. Mixture of experts' architecture is applied and networks parameters are given in Table 8.12.

Table 8.12. [Case VI] Network parameters

Neural Network Parameters	
Start Date	02/01/2001
End Date	22/02/2006
Observations	1265
Train Set	912
Validation Set	227
Test Set	126
Hidden Units	10
Window size	20
Learning Rate	0.0001
Epochs	50

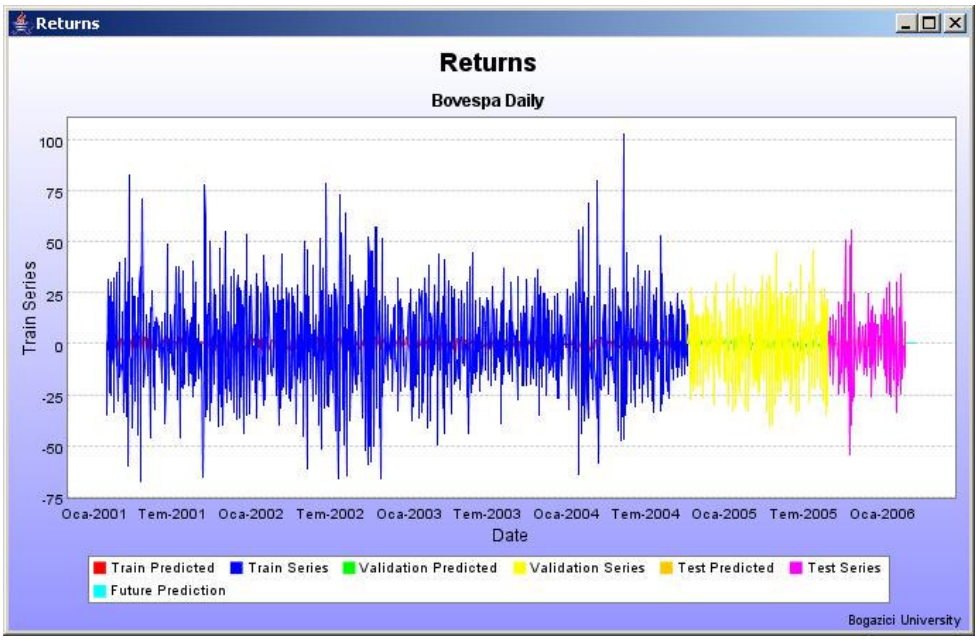

Figure 8.6. BVSP National & Volatility

In this simulation not only the past returns of BVSP National are used but also the volatility factor is taken into consideration to predict the future returns. Volatility calculated by RiskMetrics™ [30] is used as another input to the model to better identify the pattern and to get more precise future prediction.

Simulation results are shown in Table 8.13:

Table 8.13. [Case VI] Simulation results

	H_R	H_R^+	H_R^-	*MSE*	*MAE*	*RMSE*	*MAPE*	*TIC*	$P_{\Re}$
MoE	*0.6715*	*1.0*	*0.0367*	*0.0346*	*0.1430*	*0.1862*	*3.556*	*0.1480*	*1.143*

Thus better performance results are gained with the volatility factor when predicting future returns. Hit rates increase and error measures associated with regression statistics improve significantly.

8.4.7. Case VII – ISE National-100 Index & BVSP National Index

In this simulation different from formers we have used a secondary financial series for the input which is BVSP index. We tried to identify the interaction between Turkish market and Brazilian market by expressing Turkish stock exchange data by the help of Brazilian stock exchange data. For consistency the common trading days are taken into consideration. Calculated log returns are used in the simulation but in this case different time zones play an important role in the simulation. Different time zone concept means that Turkish Stock Exchange is opened when Brazilian Stock Exchange is closed; in this case today's Turkish market index close price is affected by the yesterday's Brazilian index close price. Thus we have used one day lag log returns while simulating. From total of 1230 observations 70 percent is used for train set, 20 percent for validation set and 10 percent for test set. Epoch length was decided by training the overall network and stopping at the point where minimum error occurred. At each iteration number of hidden units is increased for improving converges. Neural network parameters are given in Table 8.14.

Table 8.14. [Case VII] Network parameters

Neural Network Parameters	
Start Date	02/01/2001
End Date	22/02/2006
Observations	1230
Train Set	886
Validation Set	221
Test Set	123
Hidden Units	10
Window size	20
Learning Rate	0.0001
Epochs	50

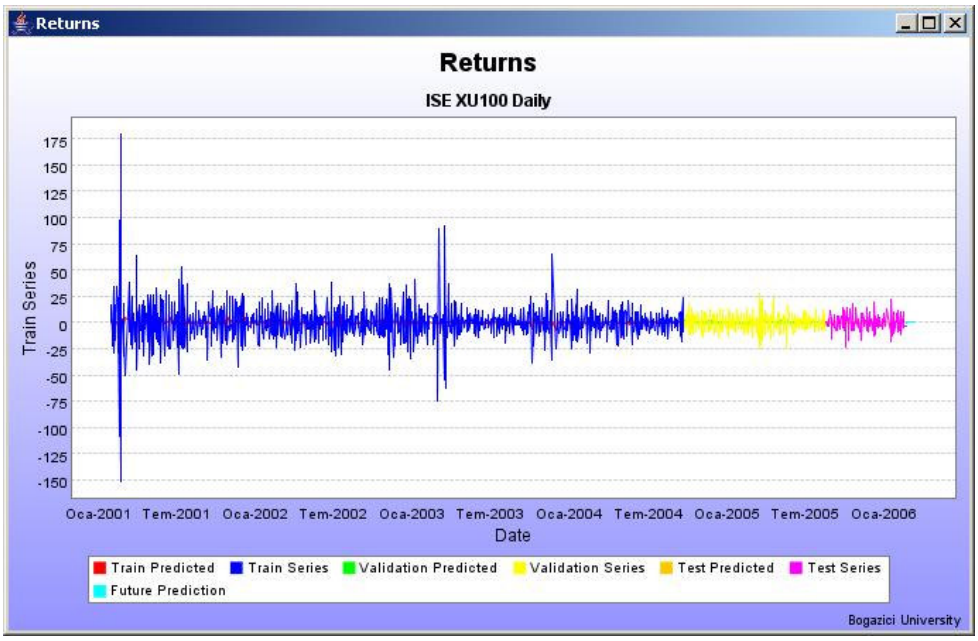

Figure 8.7. ISE National-100 & BVSP National

Simulation results are shown in Table 8.15:

Table 8.15. [Case VII] Simulation results

	H_R	H_R^+	H_R^-	*MSE*	*MAE*	*RMSE*	*MAPE*	*TIC*	$P_{\Re}$
MoE	*0.9917*	*1.0*	*0.0476*	*0.097*	*0.0788*	*0.098*	*1.254*	*0.2194*	*0.3087*

8.4.8. Case VIII – BVSP National Index & ISE National-100 Index

For cross validation, we have simulated BVSP returns using ISE National-100 as a secondary input series. In this case again different time zone concept plays an important role in the simulation: when Brazilian stock exchange is opened, Turkish stock exchange is closed at the same day. Thus Turkish stock exchange index closing price directly affects the same day closing price of the Brazilian index. From total of 1230 observations 70 percent is used for train set, 20 percent for validation set and 10 percent for test set. Epoch length was decided by training the overall network and stopping at the point where

minimum error occurred. At each iteration number of hidden units is increased for improving converges. Neural network parameters are given in Table 8.16.

Table 8.16. [Case VIII] Network parameters

Neural Network Parameters	
Start Date	02/01/2001
End Date	22/02/2006
Observations	1230
Train Set	886
Validation Set	221
Test Set	123
Hidden Units	10
Window size	20
Learning Rate	0.0001
Epochs	50

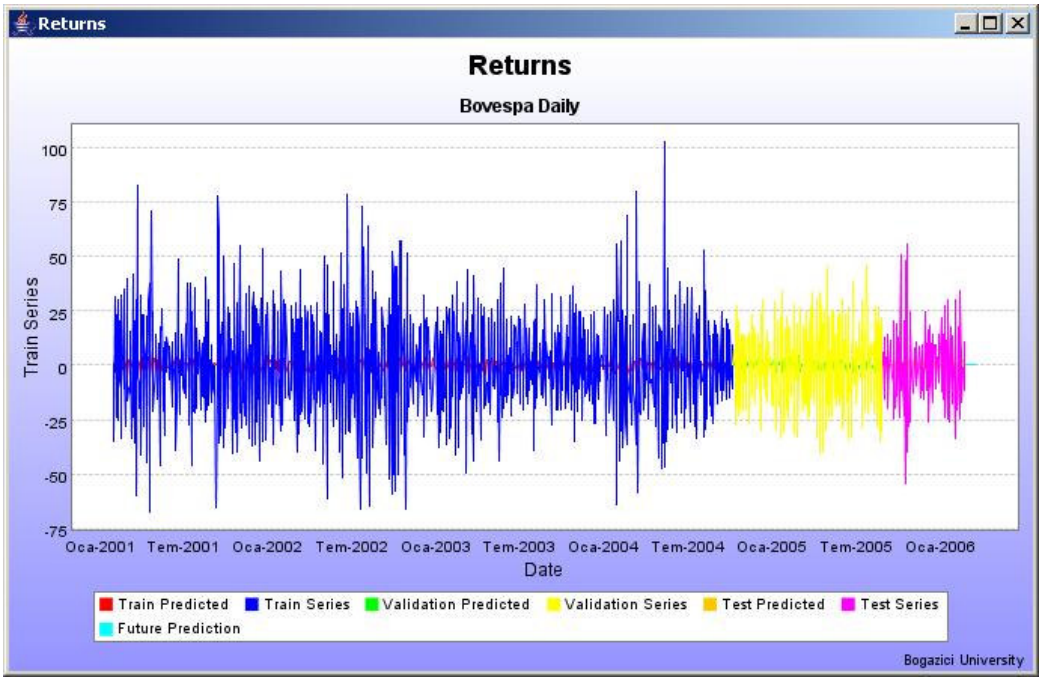

Figure 8.8. BVSP National & ISE National-100

Simulation results are shown in Table 8.17:

Table 8.17. [Case VIII] Simulation results

	H_R	H_R^+	H_R^-	*MSE*	*MAE*	*RMSE*	*MAPE*	*TIC*	$P_{\Re}$
MoE	*0.6715*	*1.0*	*0.0367*	*0.0346*	*0.1430*	*0.1862*	*3.556*	*0.2639*	*0.2554*

8.5. Comparison of Simulations

All the networks are compared to each other by means of their performance metrics in tabular form in Table 8.18.

Table 8.18. Comparison of simulation results

	H_R	H_R^+	H_R^-	*MSE*	*MAE*	*RMSE*	*MAPE*	*TIC*	$P_{\Re}$
XU100	0.9921	1.00	0.500	0.0292	0.1454	0.1711	2.0650	0.099	1.548
XU100*	0.9927	1.00	0.512	0.01329	0.0915	0.1152	1.4512	0.084	1.548
XU30	0.8542	1.00	0.500	0.007	0.065	0.0840	3.3900	0.6547	14.148
XU30*	0.8657	1.00	0.600	0.006	0.059	0.0940	2.4400	0.534	17.178
BVSP	0.6614	1.00	0.0227	0.0353	0.1441	0.1881	4.5399	0.6358	25.779
BVSP*	0.6715	1.00	0.0367	0.0346	0.1430	0.1862	3.5560	0.1480	1.143
BVSP-ISE	0.9917	1.00	0.0476	0.097	0.0788	0.0980	1.2540	0.2194	0.3087
ISE-BVSP	0.6715	1.00	0.0367	0.0346	0.1430	0.1862	3.5560	0.2639	0.2554

The results denoted with * means, volatility had been taken into consideration.

Regression metric *mse* checks whether predicted series is close to the target or not. Prediction error *mape* shows fluctuated results when the complexity in the model increases. The correct number of the sign of the predictions is very high for almost all simulations.

By looking at the performance metrics over all, mixture of experts' architecture seems successful and suitable method for predicting returns when compared to the benchmarks. And also consequences of simulations show that there is a great interaction in two different macroeconomic structured emergent markets.

Simulation results clearly points out that Turkish Stock Exchange (ISE) index (National-100) is affected from Brazilian index (BVSP) as much as its native historical data. Opposite is again true that Brazilian index (BVSP) is affected by Turkish Stock Exchange (ISE) index (National-100).

9. CONCLUSION AND FUTURE WORK

In this thesis, we investigated the use of ANN's structure Mixture of Experts in return estimation of Turkish and Brazilian National indices. A financial time series prediction without considering volatility will not be a complete predicting. Financial time series exhibit time dependent heteroskedastic variance known as the conditional variance (volatility): an indirectly observable feature of the stock market. Therefore volatility calculated from RiskMetrics™ [30] has been taken into consideration in simulations as an input. The ANN model in this research targets to model the return series and predict future returns.

Well-chosen inputs such as a window of time-delayed inputs, which contains all the data that the network can usefully relate to the output, or intelligently preprocessed inputs have great impact on the generalization performance. Not only traditional methods, but also neural networks generalize poorly if the inputs are not chosen good and the initial conditions for the network are not suitable. We have found that regarding the well-chosen inputs and preprocessing, the proposed ANNs have the ability to represent the time series data.

Total of eight case studies are made with artificial neural network structure cooperative Mixture of Experts. Istanbul Stock Exchange National 100 and National 30 indices are used in four case studies. Continuously compounding log returns are used in all case studies. Since Turkish market is highly volatile and has time dependent heteroscedastic variance, to compare with the former simulations two more simulations are made with the volatility as another input to financial series. The results show that the model with volatility has more capability to capture the dynamics of the data.

For the international market, Brazilian national index is used in two simulations, again one with volatility the other without volatility in it. Again the same results as Turkish market are gained since Brazilian stock market is also highly volatile.

For the last two simulations Brazilian and Turkish National 100 indices are used. In first case study BVSP is used as a secondary series input to model the Turkish return series. Since there is a time gap between two countries, today's Turkish national index close price is affected from yesterday's Brazilian close price. The results show that using Brazilian data as secondary input to the model is as good as the model with only past returns used for Turkish market. For cross validation, ISE National 100 index is also used as an input to model and predict Brazilian National index. Again the time gap between two countries is taken into consideration as if today's Turkish stock market index close price affects the tomorrow's close price of Brazilian market. In this situation again model with Turkish data is used as an input is as good as the model where only Brazilian data is used.

Last two simulations show that, two different emergent markets have great interaction with each other. This is not a surprising result since during the financial liberalization of the 1980-1990s, Turkey imported many constitutional laws from Latin American economies especially from Brazil.

Performance metrics defined in chapter seven are used to compare the results of the simulations. Mse, mae and $mape$ are used as basic performance measures in all simulations. In addition to that H_R, H_R^+, H_R^-, TIC, $P_{\Re}$ are used as other performance metrics to measure how precise our model is able to define and predict the financial time series data. In simulations where Turkish stock market is modeled H_R of almost ninety percent results are gained. And for TIC measure which value of 1 represents the naïve approach as a benchmark, we have results where TIC measure's value is below 1 that means our model performs better than naïve approach.

9.1. Future Work

This study is neither the first nor the last study of stock market time series prediction, but the results and the architectural throughput may lead possible studies in showing the trends and the pinpoints of the problem. Future studies may concern improving the architectures at the time where MoE may fail. In financial perspective $ARMA$ with $GARCH$ models highly satisfies the needs in financial time series prediction, but with

using ANNs alternatively, self adaptive systems can be constructed. Gathering high frequency data is also important for the future studies, we have investigated daily observations but for the online return prediction high frequency intra-day data is needed. Generally two important future improvements can be made in this area; a combined structure of return and risk estimation can be performed and this architecture can be made in real time.

REFERENCES

1. Yumlu, M. S., F. S. Gurgen and N. Okay, "A comparison of global, recurrent and smoothed-piecewise neural models for Istanbul stock exchange (ISE) prediction" Pattern Recognition Letters, 2005.

2. Cheng, B. and D. M. Titerington, "Neural Networks: A Review from a Statistical Perspective", *Statistical Science*, Vol. 9, No. 1, pp. 2-54, 1994.

3. ISE Stock Market Indices, IMKB official site, www.ise.org/indices/stckindx.htm, 2006.

4. Tompkins, R. Options Explained, Macmillan Press, 1994.

5. Burton G. M., "the Efficient Market Hypothesis and its Critics.", Princeton University CEPS Working Paper No: 91, April 2003.

6. Tesfatsion, L. Rational Expectations, "The Efficient Market Hypothesis, and the Santa Fe Artificial Stock Market Model.", Department of Economics Iowa State University, http://www.econ.iastate.edu/tesfatsi/, 2004.

7. Fama, E., "Efficient Capital Markets: A Review of Theory and Empirical Work", *Journal of Finance*, Vol. 25, pp. 383-417, 1970.

8. Murphy, J. J., "Technical Analysis of Futures Markets", New York Institute of Finance, 1986.

9. Box, G. E. P. and G. M. Jenkins, Time Series Analysis: Forecasting and Control, revised edition, Holden-Day, San Francisco, CA, 1976.

10. Brooks C. Introductory Econometrics for Finance, Cambridge University Press, 2002.

11. Tong, H. Nonlinear time Series: A Dynamical System Approach, Oxford University Press, Oxford, 1990.

12. Sims, C. A. Macroeconomics and Reality, Econometrica 48, 1-48, 1980.

13. Glosten, L. R., R. Jagannathan, and D. E. Runkle, "On the Relation Between the Expected Value and the Volatility of the Nominal Excess Return on Stocks", The Journal of Finance 48(5), 1779-1801, 1993.

14. Nelson, D. B. Conditional Heteroskedasticity in Asset Returns: A New Approach, Econometrica 59(2), 347-70, 1991.

15. Jochen Fröchlich, Neural Networks Introduction, www.jochen-froehlich.de, November 09, 1999.

16. Daniel Klerfors, "Artificial Neural Networks," St. Louis, Mo.:St. Louis University, http://www.hj.se/~de96klda/NeuralNetworks.htm, November 1998.

17. Alpaydın E., Introduction to Machine Learning, The MIT Press Cambridge, Massachusetts London, England, 2004.

18. Rumelhart, D. E., G. E. Hinton, and R. J. Williams. "Learning Representations by Backpropagating Errors." Nature 323: 533-536, 1986.

19. Rumelhart, D. E., G. E. Hinton, and R. J. Williams. "Learning Internal Representations by Error Propagation." In Parallel Distributed Processing, ed. D. E. Rumelhart, J. L. McClelland, and the PDP Research Group, 318-362. Cambridge, MA: The MIT Press, 1986.

20. Williams, R. J., and D. Zipser. “A learning Algorithm for Continually Running Fully Recurrent Neural Networks.” Neural Computation 1: 270-280, 1989.

21. Jordan, M. I., and R. A. Jacobs. “Hierarchical Mixtures of Experts and the EM Algorithm.” Neural Computation 6: 181-214, 1994.

22. Dempster, A. P., N. M. Laird, and D. B. Rubin. “Maximum Likelihood from Incomplete Data via the EM Algorithm.” Journal of Royal Statistical Society B39: 1-38, 1977.

23. Broomhead, D. S., and D. Lowe. “Multivariate Functional Interpolation and Adaptive Networks.” Complex Systems 2: 321-355, Moody, J., and C. Darken. “Fast Learning in Networks of Locally Tuned Processing Units.” Neural Computation 1: 281-294, 1989.

24. Jacobs, R. A., M. I. Jordan, S. J. Nowlan, and G. E. Hinton. “Adaptive Mixtures of Local Experts.” Neural Computation 3: 79-87, 1991.

25. Alpaydın, E., and M. I. Jordan. “Local Linear Perceptrons for Classification.” IEEE Transactions on Neural Networks 7: 788-792, 1996.

26. Dielman, T. E. A Comparison of Forecasts from Least Absolute Value and Least Squares Regression, Journal of Forecasting 5, 189-95, 1986.

27. Makridakis, S. Accuracy Measures: Theoretical and Practical Concerns, International Journal of Forecasting 9, 527-9, 1993.

28. Hellström, T. and K. Hellström, Central of Mathematical Modeling (CMM). Department of Mathematics and Physics Mälardalen University, Published as Opuscula ISRN, August 12, 1998.

29. Yumlu M. S., F. S. Gurgen and N. Okay, "Financial Time Series Prediction Using Mixture of Experts", *Proceedings of the Eighteenth International Symposium on Computer and Information Sciences, ISCIS 2003*, Antalya, pp. 553-560, LNCS 2869, November 2003.

30. Morgan, J. P., *Introduction to RiskMetrics*™, 4th Edition, November 1995.

31. Bovespa Indices, Brazilian stock exchange official site, www.bovespa.com.br/ Market/MarketIndexes, 2006.