FORECASTING STOCK MARKET VOLATILITY USING ARTIFICIAL NEURAL NETWORKS

by

Mustafa Serdar Yümlü

B.S. in C.S.E., Yıldız Technical University, 2000

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 Computer Engineering

Boğaziçi University

2004

FORECASTING STOCK MARKET VOLATILITY
USING ARTIFICIAL NEURAL NETWORKS

APPROVED BY:

Prof. Fikret S. Gürgen (Thesis Supervisor)	…….
Assoc. Prof. Nesrin Okay (Thesis Co-Supervisor)	…….
Assoc. Prof. Levent H. Akın	…….
Prof. Ethem Alpaydın	…….
Prof. Cem Say	…….

DATE OF APPROVAL …….

ACKNOWLEDGEMENTS

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

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. Ethem Alpaydın, Prof. Cem Say and Assoc. Prof. Levent H. Akın for their participation to my thesis jury.

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

This thesis is dedicated to the lady of my life, my love and my family.

ABSTRACT

FORECASTING STOCK MARKET VOLATILITY USING ARTIFICIAL NEURAL NETWORKS

Stock markets play a very significant role in the economy of capitalist countries. Millions of people trade everyday in order to have more and more profit. At this point, being able to know the future of the market gets more importance. Future prediction in a stock market is the prediction of the probability of the future losses. This probability forms the future risk profile for the interested market participant. The important thing is the estimation, measurement and the definition of the risk in mathematical norms. In this context, the variance in the time series is measured and the predictions are based over this variance. Financial time series exhibit time dependent heteroskedastic variance known as the conditional variance (volatility) that is not a directly observable feature.

In this thesis study, we focus on the volatility modeling using artificial neural networks (ANNs) and future predictions with these volatility models. Specifically, we are investigating the use of ANNs in risk estimation of asset returns. On the contrary to traditional methods, we have used ANNs to model the relationship, the dependence in time in volatility.

We have divided the space into clusters with the help of the Mixture of Experts (MoE)'s divide and conquer technique and we have assigned local experts to each cluster. Having localized experts learn their region of interest and having combined the outputs of these local experts via a gating expert we have been able to model the relationship in time and we have used this technique for the future prediction of Istanbul Stock Exchange (ISE) National-100 index. Also we modeled the relation between the input and output space by the help of the hybrid recurrent neural networks (RNN)'s multiple feedback mechanism. As a result, we have determined that MoE and hybrid RNN are very promising in modeling the volatility of ISE National-100 index.

ÖZET

HİSSE SENEDİ DEĞİŞEBİLİRLİĞİNİN YAPAY SİNİR AĞLARI İLE TAHMİN EDİLMESİ

Hisse senetleri piyasası, kapitalist ülkelerin ekonomilerinde çok önemli bir rol oynamaktadır. Milyonlarca insan hergün daha fazla kar elde edebilmek için alış/satış yapmaktadırlar. Bu noktada piyasanın geleceğini bilebilmek daha fazla önem kazanıyor. Borsada gelecek tahmini, olası kayıpların tahmin edilmesidir. Bu olasılık, ilgili piyasa katılımcısı için gelecek risk profilini oluşturur. Önemli olan bu riskin tahmini, ölçülebilirliği ve matematik normlarıyla ifade edilebilmesidir. Bu bağlamda zaman serisi içindeki varyans ölçülür ve tahminler bunun üzerine gerçekleştirilir. Finans zaman serileri, direkt olarak ölçülemeyen şartlı varyans (volatilite) olarak bilinen zamana bağlı varyans sergiler.

Biz bu çalışmada, yapay sinir ağları kullanarak volatilite modellemesi ve bu modellerle gelecek tahminleri üzerine odaklandık. Özellikle, yapay sinir ağlarının hisse getirilerinin risk tahminlerinde kullanımını araştırıyoruz. Geleneksel yöntemlerin aksine, volatilite içerisinde zamana bağımlı ilişkiyi yapay sinir ağları ile modelliyoruz.

Karışık Uzmanlar içerisindeki böl ve yönet tekniği ile once uzayı gruplara ayırdık ve her gruba bir yerel uzman atadık. Yerel uzmanlara kendi ilgi alanlarını öğreterek ve bu yerel uzmanların çıktılarını kapı uzmanı aracılığıyla birleştirerek zaman içindeki ilişkiyi modelleyebildik ve bu tekniği İstanbul Menkul Kıymetler Borsası (İMKB) Ulusal 100 endeksinin gelecek tahmininde kullandık. Ayrıca girdi ve çıktı uzayı arasındaki ilişkiyi Tekerrür Eden Yapay Sinir Ağları' nın çoklu geribesleme mekanizması ile modelledik. Geleneksel ve sinirsel yöntemleri birleştirerek yeni modeller ortaya koyduk ve bütün bu modeller tanımlanmış kriterler ile test edilip karşılaştırıldılar. Sonuç olarak, Karışık Uzmanlar ve Tekerrür Eden Yapay Sinir Ağları' nın İMKB Ulusal 100 endeksinin günlük zaman serileri modellemesinde geleceği parlak volatilite modelleri olduğuna karar verdik.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS iii
ABSTRACT iv
ÖZET v
LIST OF FIGURES ix
LIST OF TABLES xi
LIST OF SYMBOLS/ABBREVIATIONS xiii
1. INTRODUCTION 1
1.1. Aims and Objectives 2
1.2. Organization of the Thesis 2
2. THE STOCK MARKET 3
2.1. ISE Stock Market Indices 4
2.1.1. Calculation of ISE Stock Market Indices 4
2.2. Data Related to the Stock Market 5
2.2.1. Technical Data 5
2.2.2. Fundamental Data 5
2.2.3. Derived Data 6
2.2.3.1. Prices and Returns 6
3. PREDICTION OF THE MARKET 8
3.1. Why Stock Price Prediction? 8
3.2. Is the Market Predictable? 9
3.3. Prediction Methods 10
3.3.1. Technical Analysis 10
3.3.2. Fundamental Analysis 12
3.3.3. Traditional Time Series Prediction 13
3.3.4. Machine Learning Methods 13
4. VOLATILITY AND RISK 15
4.1. Financial Time Series Characteristics 16
4.2. Volatility in Turkish Stock Market 17
4.3. Risk Management 18

4.4. Value-at-Risk (VaR) …….. 18
5. Financial Time Series Modeling …… 20
5.1. Autoregressive Conditional Heteroskedasticity (ARCH) …….. 20
5.2. Generalized Autoregressive Conditional Heteroskedasticity (GARCH) …….. 22
5.3. Exponential Generalized Autoregressive Conditional Heteroskedasticity (EGARCH) …….. 23
6. NEURAL NETWORKS …… 25
6.1. Analogy to the Brain ……. 25
6.2. How do Artificial Neurons Work ……. 26
6.3. Building Neural Networks …… 28
6.4. Learning in Neural Networks …… 29
6.4.1. Input Data Selection …… 32
6.4.2. Preprocessing ……... 33
6.4.3. Cross-Validation …….. 33
6.4.4. Number of Hidden Units ……. 34
6.4.5. Initializing Weights ……. 34
6.4.6. Learning and Momentum Rate …… 35
6.4.7. Weight Decay ……..... 35
6.4.8. Early Stopping ……. 36
6.5. Neural Network Models …… 36
6.5.1. Multilayer Perceptrons (MLP) ……. 38
6.5.2. Radial Basis Functions (RBF) ……. 37
6.5.3. Recurrent Neural Networks (RNN) ……. 38
6.5.4. RNN Architectures …….. 39
6.5.4.1. Jordan RNN ……... 39
6.5.4.2. Elman RNN ……... 40
6.5.5. Real-Time Recurrent Learning (RTRL) …….. 42
6.5.6. Mixture of Experts (MoE) …….. 46
7. PERFORMANCE METRICS ……... 51
7.1. Benchmarks ……... 51
7.1.1. Naïve Prediction of Stock Prices ……... 52
7.1.2. Naïve Prediction of Stock Returns …….. 52
7.2. Metrics ……... 52

7.2.1. Regression Statistics …… 52
7.2.2. Theil Inequality Coefficient ……. 53
7.2.3. Hit Rate …… 55
7.2.4. Mean Profit per Trade …….. 56
7.2.5. Net Profit ……. 57
8. EXPERIMENTS AND DISCUSSION ……..... 58
8.1. Financial Time Series Data Sets …… 58
8.2. Models ……... 60
8.2.1. ISE National-100 ……. 62
8.2.2. Methods and Experiments ……... 64
8.2.3. Case I - Risk Metrics ……... 65
8.2.4. Case II - Traditional GARCH – EGARCH ……. 65
8.2.5. Case III - GARCH after ANN ……. 68
8.2.6. Case IV - ANN Volatility Models ……... 69
8.2.6.1. Multilayer Perceptron (MLP) …… 70
8.2.6.2. Radial Basis Functions (RBF) ……... 71
8.2.6.3. Mixture of Experts (MoE) ……. 72
8.2.6.4. Recurrent Neural Networks (RNN) …….. 74
8.2.6.5. Elman Recurrent Neural Network ……..... 74
8.2.6.6. Jordan Recurrent Neural Network ……. 77
8.2.6.7. Hybrid Recurrent Neural Network …… 79
8.2.7. Case V - Data Set is Split Into Years …….. 85
8.2.8. Case VI - Forecasting Intraday Volatility …… 90
8.2.9. Individual Stocks ……. 92
9. CONCLUSIONS AND FUTURE WORK …… 95
9.1. Future Work …….. 97
REFERENCES …….. 98

LIST OF FIGURES

Figure 6.1. A simple biological neuron 26

Figure 6.2. The structure of the artificial neuron 27

Figure 6.3. Sigmoid and tanh transfer functions 28

Figure 6.4. The structure of a basic multi-layer artificial neural network 29

Figure 6.5. Error surface 34

Figure 6.6. A simple regression MLP network 37

Figure 6.7. Topology of Jordan network 40

Figure 6.8. A simple recurrent network in which activations are copied from hidden layer to context layer on a one-for-one basis, with fixed weight of one 41

Figure 6.9. Capacitive Elman network topology 42

Figure 6.10. A system of expert and gating networks 48

Figure 8.1. Price series for four series that have been studied on 60

Figure 8.2. Log return series for four series that have been studied on 61

Figure 8.3. Return distributions of stock market data sets 63

Figure 8.4. RiskMetrics™ volatility 66

Figure 8.5. Volatility predictions for MLP network 74

Figure 8.6. Volatility predictions for MoE 75

Figure 8.7. Volatility predictions of Elman RNN 78

Figure 8.8. Volatility predictions for Jordan RNN network 80

Figure 8.9. Volatility predictions for hybrid RNN networks 83

Figure 8.10. Future predictions for ISE National-30 94

LIST OF TABLES

Table 8.1. Summary of the series 58

Table 8.2. Summary statistics for the daily returns of four series 62

Table 8.3. ISE National-100 data set neural network model parameters 64

Table 8.4. GARCH (1,1) parameters for ISE National-100 65

Table 8.5. Metrics for residual GARCH (1,1) volatility model 67

Table 8.6. EGARCH (1,1) parameters for ISE National-100 67

Table 8.7. Metrics for residual EGARCH (1,1) volatility model 68

Table 8.8. Elman network parameters 68

Table 8.9. GARCH (1,1) parameters for ANN predicted residuals 69

Table 8.10. Metrics for GARCH (1,1) applied after ANN 69

Table 8.11. EGARCH (1,1) parameters for ANN predicted residuals 69

Table 8.12. Metrics for EGARCH (1,1) applied after ANN 69

Table 8.13. MLP network parameters 70

Table 8.14. Out-of-sample performance statistics for MLP network 71

Table 8.15. Out-of-sample performance statistics for RBF network 72

Table 8.16. Out-of-sample performance statistics for MoE Network 74

Table 8.17. Elman recurrent neural network parameters 74

Table 8.18. Out-of-sample performance statistics for capacitive Elman RNN 76

Table 8.19. Out-of-sample performance statistics for non-capacitive Elman RNN 77

Table 8.20. Jordan recurrent neural network parameters 77

Table 8.21. Metrics calculated in the test data set for Jordan RNN 79

Table 8.22. Hybrid recurrent neural network parameters 79

Table 8.23. Metrics calculated in the test data set for hybrid RNN 81

Table 8.24. GARCH based neural networks out-of-sample results 82

Table 8.25. RiskMetrics based neural network out-of-sample results 84

Table 8.26. Elman network parameters for year-by-year prediction 85

Table 8.27. Out-of-sample results regarding each year of concern using Elman 88

Table 8.28. Out-of-sample results for two-year data periods 89

Table 8.29. Out-of-sample results for three-year data periods 89

Table 8.30. Summary statistics for ISE National-30 intraday data 91

Table 8.31. Intraday prediction results for ISE National-30 91

Table 8.32. Prediction results for ISCTR, TUPRS and NETAS 93

LIST OF SYMBOLS/ABBREVIATIONS

$\boldsymbol{C_h}$	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
$\boldsymbol{H\varepsilon}$	Hit rate for ε – *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
$\boldsymbol{p_{high}}$	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
$\boldsymbol{p_t}$	Closing price at the end of the day
Q (5)	Ljung-Box statistics of the autocorrelation of residuals
$\boldsymbol{r_s}$	i.i.d. variable

r_t	One-step return (continuously compounded returns)
$\boldsymbol{r_t}$	Return at time t
$\bar{\boldsymbol{r}}_{\boldsymbol{t}}$	Mean of the returns
$\hat{\boldsymbol{r}}_{\boldsymbol{t}}$	Predicted return at time t
$\boldsymbol{sigmoid}(\boldsymbol{x})$	Sigmoid function
U_m	Bias proportion in *TIC*
U_s	Variance proportion in *TIC*
U_c	Covariance proportion in *TIC*
$\boldsymbol{Vol_t}$	Total number of traded stocks during the day
$\boldsymbol{V_t}$	Volatility
$\boldsymbol{V_{jh}}$	Weight between output unit j and hidden unit h
$\boldsymbol{w}$	A weight in the network
$\boldsymbol{w_{hi}}$	Weight between hidden unit h and input unit i
$\mathrm{w_{ij}}$	Weight to unit $\boldsymbol{i}$ from unit $\boldsymbol{j}$
$\boldsymbol{X_d}$	Input layer units in the network
$\boldsymbol{y_j}$	Output of the network
$\boldsymbol{y_k}$	Artificial neural network output
$\boldsymbol{y_t}$	Value of the market index on time $\boldsymbol{t}$
$\boldsymbol{Z_h}$	Hidden units in the network
ω , α , β and γ	Constant parameters
$\partial_{\boldsymbol{ik}}$	Kronecker delta
φ_{t-1}	The information set available at time t-1
$\phi(\boldsymbol{x})$	Gaussian basis function
$\Delta \boldsymbol{w}$	Change in weight
$\varepsilon - Increase$	Increasing trend benchmark
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
ARCH (5)	Engle's ARCH Test
ARMA	Autoregressive Moving Averages
BPTT	Back-Propagation Through Time
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
ISCTR	Is Bankasi C
ISE	Istanbul Stock Exchange
ISE National-100	Istanbul Stock Exchange National 100 Index
J-B	Jarque-Bera statistic for normality test
mape	Mean absolute percentage error
MLP	Multilayer Perceptron
MoE	Mixture of Experts
mse	Mean square error
NETAS	Nortel Networks Netas Telecommunications
PACF	Partial Autocorrelation Function
pdf	Probability Density 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
TUPRS	Tupras
U.S.	United States
VaR	Value-at-Risk
XU100	Istanbul Stock Exchange National 100 Index

1. INTRODUCTION

Financial markets are incredible systems. Millions of people try to make more money trading over thousands of instruments all around the world every day. National economies are strongly linked and heavily influenced of the performance of their stock markets. Moreover, recently the markets have become a more accessible investment tool, not only for strategic investors but for other investors as well. Consequently they are not only related to macroeconomic parameters, but they influence everyday life in a more direct way. Is there a relationship that can help people to understand where the market will go? Is it possible to predict the stock market? In this thesis study we are trying to answer these questions by investigating the use of Artificial Neural Networks (ANNs) in risk estimation of asset returns [1, 2].

In recent years, ANNs have provided an alternative tool for both forecasting researchers and practitioners. Werbos reported that ANNs trained with the backpropagation algorithm outperform the traditional statistical methods such as the regression and Box-Jenkins approaches [3, 4].

Unlike the traditional model-based methods, ANNs are data-driven and self-adaptive and they make very few assumptions about the models for problems under study. ANNs learn from examples and attempt to capture the subtle functional relationship among the data. Thus, ANNs are well suited for practical problems where it is easier to have data than knowledge governing the underlying system being studied. Generally, they can be viewed as one of many multivariate nonlinear and nonparametric statistical methods [5].

Financial Time Series Prediction (FTSP) aims to find underlying patterns, trends, cycles and aims to forecast future using historical and currently observable data [1]. We have used ANNs in FTSP, in volatility modeling of Turkish Stock Market index, Istanbul Stock Exchange National-100 because of the fact that asset prices vary with changes in volatilities of the underlying asset and compare our results with traditional economic time series modeling techniques.

1.1. Aims and Objectives

This thesis study aims to predict short-term future of the stock market. ISE stock market index National-100 is studied using ANNs including multilayer perceptrons, radial basis functions, mixture of experts and recurrent neural networks. For an accurate forecast of price, we need a good forecast of volatility measure. Volatility is also very important for risk management. For this purpose, besides neural networks approaches mentioned in this study for the prediction of prices and returns, we have applied neural networks to model the relation of Turkish stock market returns. The aim here is to predict future volatility of ISE National-100.

1.2. Organization of the Thesis

In the Chapters 2 to 6, we cover the background needed to develop the proposed architecture. In Chapter 2, Turkish stock market and the market data is explained. The third chapter is related to the prediction methods applied to stock market time series prediction. Chapter 4 gives aspects of volatility and risk. Fifth and sixth chapters cover economic time series modeling and artificial neural networks in details respectively. The Chapter 7 summarizes the performance metrics that we have used to compare our models. In Chapter 8, the used ISE National-100 index data and the performed experiments are described and the test results are reported explaining the neural network models studied. Finally, in the Chapter 9 the conclusions from this work are drawn and possible directions for future work are shown.

2. THE STOCK MARKET

As companies grow, they sell their shares for a pre-determined price. A share is simply a part-ownership of a company. As a part owner of a company, you are investing in the management of the company.

Buying and selling of these shares must be carried on rules. The stock market (or stock exchange) is a market, bringing people together who want to buy shares in a company, and those who want to sell their shares. The laws of supply and demand determine the prices buyers and sellers settle on. Only the companies that obey the rules of this market may be able to use the service provided by the stock market and have their shares bought and sold.

In Turkey, the Istanbul Stock Exchange (ISE) was established in early 1986. ISE is the only securities exchange in Turkey 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. Approximately 300 companies have their shares traded every business day. Securities companies and banks are the contact points of people with the stock market. The orders are passed through these securities companies to ISE and transactions occur in that central place. A safe place, where the shares are kept, was originally set up as a department of the Istanbul Stock Exchange in 1988, named Takasbank. Takasbank is the "Central Securities Depository of Turkey"**,** authorized by the Capital Markets Board (CMB). According to the CMB regulations, no institution other than Takasbank is permitted to safekeeping of physical certificates of securities in Turkey. Takasbank also provides corporate actions services for the holders of the securities under its safekeeping and custody. In this thesis study, we are concentrating on the prediction of the ISE market index (ISE National-100).

2.1. ISE Stock Market Indices

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-100 (XU100) 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.

2.1.1. Calculation of ISE Stock Market Indices

The market capitalization is weighted by the publicly held portion of each constituent stock kept in custody at Takasbank (except those kept in non-fungible accounts).

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

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

where P_{it} is the closing price of the stock i at period t, N_{it} represents total number of shares outstanding of the stock i at period t (Paid-in capital / 1,000), FW_{it} is the floatation 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 is the value of divisor at period t (Adjusted base market value) and n represents the 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.

2.2. Data Related to the Stock Market

The information about the market comes from the study of relevant data. The data can be grouped into three categories:

2.2.1. Technical Data

The daily available data for each stock is known as the *technical data*. Technical data includes:

- Close price; the closing price at the end of the day (p_t)
- The highest traded price during the day (p_{high})
- The lowest traded price during the day (p_{low})
- Volume; the total number of traded stocks during the day (Vol_t)

2.2.2. Fundamental Data

There is a lot of information concerning the activities and financial situation of companies. Professional market analysts analyze most companies quoted at the stock market on a regular basis. The analyses are often presented as numerical results, which are supposed to show the true value of the stock. This data is related to the intrinsic value of a company or category of companies as well as data related to the general economy.

A fundamental analysis of a company typically focuses on,

- Inflation
- Interest Rates
- Trade Balance
- Indexes of industries (e.g. heavy industry)
- Prices of related commodities (e.g. oil, metals, currencies)

- Net profit margin of a firm.
- Prognoses of future profits of a firm

Most of these data are not publicly available.

2.2.3. Derived Data

Transforming and combining technical and/or fundamental data can produce this type of data. Some commonly used examples are:

- Returns: Continuously compounded return r_t is defined as the relative increase in price since the previous point in the time series.
- Volatility: Describes the variability of a stock and is used as a way to measure the risk of an investment.

2.2.3.1. Prices and Returns. There are several reasons for focusing on returns rather than on prices.

First, for the average investor, financial markets may be considered close to perfectly competitive, so that the size of the investment does not affect price changes. The return is scale free. The prices vary greatly and make it difficult to create a valid model for a longer period of time and prices for different stocks may easily differ over several decades and therefore cannot be used as the same type of input in a model. Statistically, returns brings stationarity into the model.

Consider p_t, the price of an asset without dividends. The return of this asset between times t-1 and t , r_t, is defined as

$$r_t = \frac{p_t}{p_{t-1}} - 1 \tag{2.2}$$

The difficulty of manipulating geometric average operations over the whole time period brings another approach, the compounded returns which have also important implications on modeling of asset returns. The *continuously compounded return* or *log return* r_t of an asset is defined as

$$r_t = \ln(p_t / p_{t-1}) \tag{2.3}$$

Here *ln* represents the natural logarithm. In this thesis study, r_t is taken as continuously compounded return of the price. Continuously compounded returns are chosen in order to achieve comparability and stationary series in the same domain.

3. PREDICTION OF THE MARKET

We give a precise definition of the prediction before getting deeper into details:

"Given a sample of N examples, $\{(x_i, y_i),\ i = 1,...,N\}$, where $f(x_i) = y_i$ and for all i we try to return a function g that approximates f in the sense that the norm of the error vector $E = (e_1,...,e_N)$ is minimized. Each e_i is defined as $e_i = e(g(x_i), y_i)$ where e is an arbitrary error function" [6].

In other words, the definition above indicates that in order to predict the market one should search historic data and find relationships between data and the value of the market. Next step is to exploit these relationships found on future situations. Above definition is based on the assumption that such relationships do exist. But do they? or do the markets fluctuate in a totally random way leaving us no space for prediction?

3.1. Why Stock Price Prediction?

The wish to find methods to predict asset returns has occupied the minds of investors and also academics since the foundation of financial markets. In general, financial assets are influenced by the real economy. The liberalization and globalization of world asset markets have caused interest rates, exchange rates, and also other asset markets to be intimately linked, and the need for tools to monitor as well as control risk levels has become obvious both for industrial companies and financial institutions. The question of predictability in the stock markets is therefore important even outside the trading rooms.

3.2. Is the Market Predictable?

The predictability of the market has been and is being predominantly studied by researchers and academics. As a result, an hypothesis formulated as the *Efficient Market Hypothesis* (EMH) implies that there is no way to make profit by predicting the market.

The EMH states that the current market price reflects assimilation of all the information available. This means that given the information, no prediction of future changes in the price can be made. As new information enters the system, the unbalanced state is immediately discovered and it is quickly eliminated by a correct change in market price.

More specifically the EMH has three forms:

- The weak form: Only past price data is considered. This kind of EMH rules out any form of predictions based on the price data only.
- The semi-strong form: This EMH type states that you cannot even utilize published information to predict future prices.
- The strong form: This claims that you cannot predict the market no matter what information you have available.

According to Weak Form Efficiency Hypothesis, stock prices follow a *'Random Walk'* model.

Which more formally stated is:

$$y_t = y_{t-1} + r_s \tag{3.1}$$

where y_t is the value of the market on time t and r_s is an independent and identically distributed (i.i.d.) variable. If this model is valid, the best prediction about tomorrow's value is today's value.

3.3. Prediction Methods

The prediction of the market is without doubt an interesting task. In the literature, there are a number of methods applied to accomplish this task [4, 7 - 10]. They use various approaches, ranging from highly informal ways (e.g. the study of a chart with the fluctuation of the market) to more formal ways (e.g. linear or non-linear regressions).

These techniques are as follows:

- Technical Analysis Methods [7, 8],
- Fundamental Analysis Methods [9],
- Traditional Time Series Prediction Methods [4],
- Machine Learning Methods [10]

The criterion to this categorization is the type of tools and the type of data that each method is using in order to predict the market. What is common to these techniques is that they are used to predict and thus benefit from the market's future behavior. None of them has proved to be the consistently correct prediction tool an investor would like to have. Furthermore many researchers question the usefulness of many of these prediction techniques.

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 [7]. 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. It is probably the most widely used decision making tool for traders who make multi-million dollar trading decisions.

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.

Murphy summarizes the basis for technical analysis into the following three premises [7]:

- *Market action discounts everything.*

The assumption here is that the price action reflects the shifts in demand and supply, which is the basis for all economic and fundamental analysis and everything that affects the market price, is ultimately reflected in the market price itself. 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.

- *Prices move in trends.*

This assumption is the foundation of almost all technical systems that try to identify trends and trading in the direction of the trend. The underlying premise is that a trend in motion is more likely to continue than to reverse.

- *History repeats itself.*

This premise is derived from the study of human psychology, which tends not to change over time. This view of behavior leads to the identification of chart patterns that are observed to recur over time, revealing traits of a bullish or a bearish market psychology**.**

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. Another factor to consider in fundamental analysis is the reliability of the economic data. Due to the complexity of today's global economy, economic data are often revised in subsequent periods therefore posing a threat to the accuracy of a fundamental economic forecast that bases its model on the data. The frequency of the data also poses a limitation to the predictive horizon of the model.

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: *Univariate* (simple regression) and *multivariate* (multivariate regression). Univariate models, like Box-Jenkins [4], contain only one variable in the recurrence equation. Box-Jenkins is a complicated process of fitting data to appropriate model parameters. The equations used in the model contain past values of moving averages and prices. Box-Jenkins is good for short-term forecasting but requires a lot of data, and it is a complicated process to determine the appropriate model equations and parameters. M*ultivariate* models contain more than one variable in their equations.

These regression models are the most common tools used in econometrics to predict time series. The way they are applied in practice is that first a set of factors that influence the series under prediction is formed. These factors are *the explanatory variables* x_i of the prediction model. Then a mapping between their values x_{it} and the values of the time series y_t *(y is the explained variable)* is done, so that pairs $\{x_{it}, y_t\}$ are formed. These pairs are used to define the importance of each *explanatory variable* in the formulation of the *explained variable*.

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. ANNs can be used to perform *classification* and *regression* tasks. More specifically it has been proved by Cybenko that any function can be approximated to an arbitrary accuracy by a neural network [10].

ANNs consist of *neurons* (or *nodes*) distributed across *layers*. The way these neurons are distributed and the way they are linked with each other defines the *structure of the network*. Each of the links between the neurons is characterized by a *weight* value. A *neuron* is a processing unit that takes a number of inputs and gives a distinct output.

Recent research in financial time series analysis has put a lot of emphasis on modeling and forecasting asset return volatilities. This is because of the fact that asset prices vary with the changes in volatilities of the underlying asset. Therefore, we need to forecast the volatility to have accurate predictions of future prices. Another important aspect is the market risk management. In order to calculate the market risk we need a measure known as *value-at-risk* (VaR), which requires a forecast of volatilities of the risk factors such as market returns, interest rates etc. There is no doubt that in the literature the most prominent volatility model that estimates conditional variances of asset returns on the basis of historical observations is the Generalized Autoregressive Conditional Heteroskedasticity (GARCH) model [11, 12]. It is able to capture several important stylized facts of asset returns, namely heteroskedasticity, volatility clustering and excess kurtosis. Both GARCH and exponential GARCH (EGARCH) are analyzed in Chapter 5.

4. VOLATILITY AND RISK

Stock prices vary with changes in volatilities of the underlying risk factor and as a consequence, accurate prediction of future stock prices requires a forecast of asset return's volatility.

Financial time series exhibit time dependent heteroskedastic variance known as the conditional variance (volatility) that is not a directly observable feature.

There are basically two notions of volatility in the literature: historical volatility and implied volatility. Historical measure is the popular approach where exponentially declining weights are given to past volatilities, approximated by squared returns. Historical volatility is an exponentially weighted moving average of squared returns. The returns are calculated from the closing values of the index. Returns are governed from $\boldsymbol{r_t} = \log(\boldsymbol{p_t} / \boldsymbol{p_{t-1}})$ where p_t represents the closing index price and r_t represents the returns. Historical volatility (V_t) is then measured as

$$\boldsymbol{V_t} = (1-\alpha)\sum_{t}^{T} \alpha^{T-t} \boldsymbol{r_t^2} \tag{4.1}$$

The weighing factor $\alpha \in (0,1)$ determines the impact of past returns on the actual volatility. V_t can also be represented as

$$V_t = \alpha V_{t-1} + (1-\alpha) r_t^2 \tag{4.2}$$

Historical volatility measure is similar to the basic volatility measure applied in RiskMetrics™. The implied volatility measure (IV-measure) is estimated from the extracted options' volatilities implied by the Black-Scholes model [13]. This is closely related to option prices and requires an option-pricing model in order to calculate the market driven volatilities.

A volatility model like the one we propose in this thesis study is used to forecast the risk, volatility of asset returns. These forecasts are used in market risk management, portfolio selection, market timing etc. and are used by other financial decision makers. For this purpose, predictability of the volatility is important. A portfolio manager may want to sell a stock or a portfolio before the market becomes too volatile or a trader may want to know the expected volatility to give the right decision while buying or selling a stock. All these financial decisions are based on market risk management. A risk manager has the right to know that his portfolio may likely decline in near future. Market risk management plays a crucial role in giving the financial decision. None of the players want a volatile market. Value at Risk, VaR, a standard in risk management, is based on the forecasting of the volatility of market risk factors. So, estimating the volatility of asset returns, which is a basic risk factor component of the stock market, gives valuable information for the future risk in the market and this will make the players consider the expected high or low volatility in the market.

4.1. Financial Time Series Characteristics

Several characteristics for the volatility of asset returns have emerged throughout the years and these have been confirmed by numerous studies on the field of volatility modeling for the prediction of financial time series. A volatility model is expected to capture and reflect these properties of the series. These are,

- *Volatility Clustering* suggests that large changes in the price of an asset are often followed by other large changes and small changes are often followed by small changes.
- According to *volatility persistence* volatility comes and goes. *Volatility persistence* states an existence of normal level of volatility. Volatility will go up and down, but in a reasonable time, it will converge to the normal level of volatility. *Persistence* in volatility implies that current information has no effect on the long run forecast.
- Innovations may have an *asymmetric effect* on the volatility. The sign of the innovation is very important for the volatility model. Standard GARCH based models lacks this effect because it allows the variance to be effected only by the square of the

lagged innovation. So, GARCH disregards the sign of the innovation. But, for stock returns, it is unlikely that positive and negative shocks have the same impact on the volatility. Negative shocks have been shown to be more volatility prone. This is known as the *"asymmetric effect"* first noted in [14], refers to the fact that changes in stock prices tend to be negatively correlated with changes in volatility. In general, negative return shocks raise more volatility than positive return shocks of same magnitude.

Financial Time Series, distribution of returns, is fat-tailed and *exhibits leptokurtosis* [15].

4.2. Volatility in Turkish Stock Market

Research points to the fact that managers will take market volatility into account when they make investment decisions. In general, there is evidence for the negative relationship between the stock market volatility and fixed investment. Large stock market price fluctuations are related to low growth in real fixed investment. In Turkey, stock market volatility may have led to a reduction in the capital stock and hence long-run productivity and income growth. A more stable stock market will better serve as the forecasting mechanism for the economy as well as fulfill its role in challenging savings into capital investment [16].

Turkey has an emerging market economy with a high and variable inflation. Information about private firms is harder to collect than in industrialized economies and not surprisingly, securities markets therefore play a small role. The greater difficulty of acquiring information on private firms makes banks even more important in the financial system. The barriers of information collection are so great that the dominance of banks will continue for the foreseeable future. Furthermore, it is not clear that higher volatility of asset prices is the most important factor promising financial instability. There are many episodes of high asset price volatility in which there are no manifestations of financial stability [17].

A historical analysis of the financial crises of the Turkish economic history indicates that serious examples of financial instability are always associated with substantial deterioration in the balance sheets of firms, households and banks. Thus, increased financial volatility that is not linked to the deterioration in balance sheets is unlikely to produce financial instability, which has harmful effects on the economy. The large fluctuations cannot be fully explained by the macroeconomic parameters such as inflationary money growth and excess consumption. They may be generated by trading activity as well as by political risk. Stock market engineering makes volatility desirable in a rising market to the extent that volatility is not perfectly correlated with loss [16].

4.3. Risk Management

Two important developments, one in academia and one on Wall Street have facilitated the advancement in knowledge about risk management. First, the development of volatility models for measuring and forecasting volatility dynamics was proposed by Engle [12]. Hundreds of papers following Engle's original work - many of them finding applications to financial data - have had important implications for modern risk management techniques. Second, the introduction of RiskMetrics™ by JP Morgan has enabled companies with just a minimum of computational power and technical ability to compute simple measures of market risk for a given portfolio of assets [18]. RiskMetrics™ has also aroused the interest of academics as it offers a benchmark methodology upon which improvements can be made, and against which alternatives can be tested.

In RiskMetrics™, the VaR measure has only a few unknown parameters, which are simply calibrated to values found to work quite well in common situations.

4.4. Value-at-Risk (VaR)

Measuring the risk on specific assets has become increasingly important during the last decades. In a broad sense, companies want to have good control of their risk profile. The definition of risk and its translation into mathematics is of practical importance. Risk is generally defined in terms of the probability of returns and particularly of the returns

variance. In the financial market this is of even greater importance. There has been rapid development of techniques for measuring and managing financial risk, partially motivated by recent financial disasters involving derivative securities over the last decade. One of the most popular approaches to risk measurement is by calculating what is known as an institution's 'Value at Risk' (VaR). Broadly speaking, Value at Risk is an estimation of likely losses, which could arise from changes in market prices. More precisely, it is defined as the money-loss in a portfolio that is expected to occur over a pre-determined horizon and with a pre-determined degree of confidence. The advantage of *VaR* is that it provides a single number which encapsulates the portfolio risk and which can be applied easily by non-technically minded financial risk managers. Explicitly, *VaR* expresses the expected loss resulting from potential adverse market movements with a specified probability over a period of time.

The roots of VaR' s popularity stem from the simplicity of its calculation, its ease of interpretation, and from the fact that VaR can be suitably aggregated across an entire firm to produce a single number which broadly encompasses the risk of the positions of the firm as a whole. Its origin can be traced to the "4:15 report" of Dennis Weatherstone, chairman of JP Morgan who demanded that a one page report be delivered to him every day summarizing the company's market exposure and providing an estimate of the potential loss over the next trading day. Dowd provides thorough introductions to VaR, and Brooks and Persand present recent discussions of VaR model estimation issues [19 - 22].

5. FINANCIAL TIME SERIES MODELLING

Many economic time series do not have a constant mean and most show periods of tranquility followed by high volatility. Financial time series data do not have a constant mean and variance. These processes may contain both stochastic and deterministic components. A stochastic variable with a constant variance is called homoskedastic. Heteroskedasticity is a time-varying variance. For series exhibiting volatility, the unconditional variance may be constant even though the variance during some periods is unusually large. We have to decompose the series into its stochastic and deterministic components. Time series are not stationary; the sample means do not appear to be constant and/or there is the strong appearance of heteroskedasticity and they usually have a trend. For some series, increasing trend is interrupted by a decline and it is hard to maintain a time-invariant mean in these series. Any shock to these series displays a high degree of persistence causing the change over time. These series are called conditionally heteroskedastic.

5.1. Autoregressive Conditional Heteroskedasticity (ARCH)

The ARCH-model was first presented by Engle (1982) and since then received a lot of attention [12]. Professor R.F. Engle shared The Bank of Sweden Prize in Economic Sciences in Memory of Alfred Nobel 2003 with Professor Clive W.J. Granger for methods of analyzing economic time series with time varying volatility (ARCH). Engle's work explains how random fluctuations in the value of financial markets can be smoothed out, allowing the risk of holding shares to be calculated.

Conditionality of the variance implies a dependence on the observations of the past and autoregressive conditionality describes a feedback mechanism that incorporates past observations into the model. Consider an ordinary AR(q) model of the stochastic process y_t.

$$y_t = c + \alpha_1 y_{t-1} + ... + \alpha_p y_{t-p} + \varepsilon_t \tag{5.1}$$

and its functional form is defined as

$$y_t = E[y_t \mid \varphi_{t-1}]\varepsilon_t \tag{5.2}$$

where ε_t is a white noise disturbance term, φ_{t-1} is the information set available at time t-1 and $E[.|.]$ denotes the conditional expectation operator. This expectation is conditional to all past information available at time t-1. ε_t is the innovation process. The Autoregressive Conditional Heteroskedasticity (ARCH) process of Engle is any ε_t of the form

$$\varepsilon_t = \sigma_t z_t \tag{5.3}$$

where ε_t is an independently and identically distributed (i.i.d.) process with zero mean and a variance equal to one [12]. By definition, ε_t is serially uncorrelated with mean zero, but its conditional variance equals σ_t^2 and, therefore, may change over time. The ARCH(q) process is then,

$$\boldsymbol{h_t} = \alpha_0 + \sum_{\boldsymbol{i}=1}^{\boldsymbol{q}} \alpha_{\boldsymbol{i}} \varepsilon_{\boldsymbol{t-i}}^2 \tag{5.4}$$

where $\alpha_0 > 0$, $\alpha_i > 0$, $i = 1,...,q$ and h_t is the conditional variance represented also by σ_t^2.

5.2. Generalized Autoregressive Conditional Heteroskedasticity (GARCH)

Although ARCH models provide a good description of many return series, it suffers from a practical problem. This means that a large lag q is needed for the ARCH(q) process to provide an adequate description of the returns, but as an ARCH(q) process has q+1 parameters, the immediate consequence of this is that a large number of parameters have to be estimated, all subject to parameter restrictions. This imposes a serious computational burden [11]. To circumvent this problem, Bollerslev proposed the GARCH process in [11]. GARCH is a mechanism that includes past variances in the explanation of future variances. More specifically, GARCH is a time series modeling technique that uses past variances and past variance forecasts to forecast future variances. Formally, the GARCH (p, q) process only differs from the ARCH (q) process in the way that the function $\boldsymbol{h_t}$, conditional variance, is specified:

$$\varepsilon_t \mid \varphi_{t-1} \sim N(0, h_t) \tag{5.5}$$

$$h_t = \alpha_0 + \sum_{i=1}^{q} \alpha_i \varepsilon_{t-i}^2 + \sum_{i=1}^{p} \beta_i h_{t-i} \tag{5.6}$$

where p integer, q integer

$$\begin{aligned} &\alpha_0 > 0, \quad \alpha_i > 0, i = 1,..., q \\ &\beta_i \geq 0, \qquad i = 1,..., p \end{aligned} \tag{5.7}$$

thus the additional feature is that the process now also includes lagged h_{t-i} values. For $p = 0$ the process is an ARCH (q). For $p = q = 0$ (an extension allowing $q = 0$ if $p = 0$), ε_t is white noise.

GARCH has gained a lot of interest and is widely accepted. It takes into account excess kurtosis (i.e. fat tail behavior) and volatility clustering, two important characteristics of financial time series. It provides accurate forecasts of variances and covariance of asset

returns through its ability to model time varying conditional variances. As a consequence, you can apply GARCH models to such diverse fields as risk management, portfolio management and asset allocation, option pricing, foreign exchange, and the term structure of interest rates.

In this thesis study, we have used GARCH and EGARCH as a comparison model for the neural network models we have used [15].

Although GARCH models are useful across a wide range of applications, they do have limitations:

- GARCH models are only part of a solution. Although GARCH models are usually applied to return series, financial decisions are rarely based solely on expected returns and volatilities.
- GARCH models are parametric specifications that operate best under relatively stable market conditions. It is not able to capture irregular market movements.
- GARCH models often fail to fully capture the fat tails observed in asset return series. Heteroskedasticity explains some of the fat tail behavior, but typically not all of it.

5.3. Exponential Generalized Autoregressive Conditional Heteroskedasticity (EGARCH)

Asymmetric or leverage volatility models, in which good and bad news have different predictability for future volatility have gained a lot interest in the last decade. Despite their successful applications ARCH and GARCH models cannot capture some important facts in the data. The most important fact is the leverage or asymmetric effect discovered by Black and confirmed by the findings of French et. al., Nelson, Pagan and Engle [14, 23 – 26]. This effect claims that a bad news, a drop in price increases volatility more than an unexpected increase in price of similar magnitude. Because we take square of the innovations in ARCH and GARCH, we are unable to seize this effect and understand if there will be a difference from the point of view of volatility. This effect suggests that a symmetry constraint over ε_t is not appropriate. Nelson proposed a model to capture such

asymmetric effects called Exponential Generalized Autoregressive Conditional Heteroskedasticity (EGARCH) [14]. This model is defined as (5.9):

$$\log(h_t) = \omega + \beta.\log(h_{t-1}) + \gamma \frac{\varepsilon_{t-1}}{\sqrt{h_{t-1}}} + \alpha \left[\frac{|\varepsilon_{t-1}|}{\sqrt{h_{t-1}}} - \sqrt{\frac{2}{\pi}} \right] \tag{5.8}$$

where ω, α, β and γ are constant parameters. This model is asymmetric and it is able to cover the effect of the sign of the returns because $\frac{\varepsilon_{t-1}}{\sqrt{h_{t-1}}}$ has a coefficient of γ. When this coefficient is negative, it will generate more volatility than positive return shocks because of the sign effect. Two important acquisitions are

- The EGARCH model allows good news and bad news to have a different impact on volatility on the other hand GARCH does not. Bad news has a greater impact than good news.
- The EGARCH model allows big news to have a greater impact on volatility than GARCH model.

In this thesis study, we have used both GARCH and its EGARCH as comparison models to each neural network based volatility forecasting models.

6. NEURAL NETWORKS

Artificial Neural Networks are relatively crude electronic models based on the neural structure of the brain. The brain basically learns from experience. Even simple animal brains are capable of functions that are currently impossible for computers. Cognitive and perceptual powers of humans are much more powerful than today's computers. Humans can effortlessly recognize images in many inappropriate conditions, whereas machines fail to compete with human in these areas. But computers have trouble recognizing even simple patterns much less generalizing those patterns of the past into actions of the future. Now, advances in biological research promise an initial understanding of the natural thinking mechanism. This research shows that brains store information as patterns. This process of storing information as patterns, utilizing those patterns, and then solving problems encompasses a new field in computing. Artificial neural networks do not utilize traditional programming but involves the creation of massively parallel networks and the training of those networks to solve specific problems. It also utilizes words very different from traditional computing, words like behave, react, self-organize, learn, generalize, and forget.

6.1. Analogy to the Brain

The exact workings of the human brain are still a mystery. Yet, some aspects of this amazing processor are known. In particular, the most basic element of the human brain is a specific type of cell, which, unlike the rest of the body, doesn't appear to regenerate. It is assumed that these cells are what provide us with our 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 200,000 other neurons. The power of the human mind comes from the sheer numbers of these basic components and the multiple connections between them. Artificial neural networks try to replicate only the most basic elements of this complicated, versatile, and powerful organism. They do it in a primitive way. For the software engineer who is trying to solve problems, neural computing is not about replicating human brains. It is about machines and a new way to solve problems.

6.2. How Do Artificial Neurons Work

The fundamental processing element of a neural network is a neuron. This building block of human awareness encompasses a few general capabilities. Basically, a biological neuron receives inputs from other sources, combines them in some way, performs a generally nonlinear operation on the result, and then outputs the final result**.** This relationship is shown in Figure 6.1.

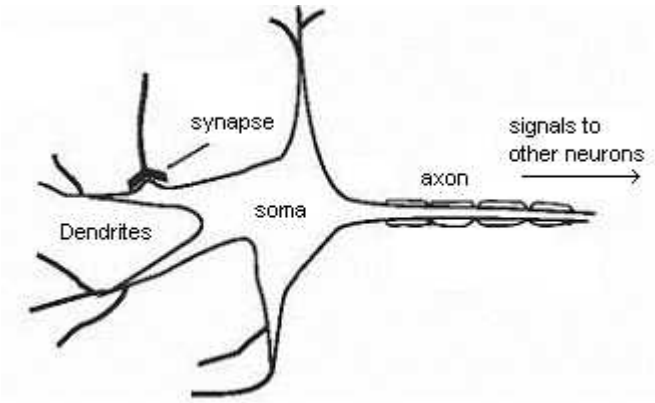


Figure 6.1. A simple biological neuron [27]

All natural neurons have the same four basic components. These components are known by their biological names - dendrites, soma, axon, and synapses. Dendrites are hair-like extensions of the soma, which act like input channels. These input channels receive their input through the synapses of other neurons. The soma processes these incoming signals over time. The soma then turns that processed value into an output, which is sent out to other neurons through the axon and the synapses.

Biological neurons are significantly more complex than the existing artificial neurons that are built into today's artificial neural networks. Neural networks do not try to recreate a brain. Researchers seek to find an understanding of the nature of the capabilities of these neurons and the brain and use this to solve engineering problems, which are not solved by traditional computing methods yet.

To do this, artificial neural networks simulate the functions of the biological neuron structures. The representation of an artificial neuron is given in Figure 6.2.

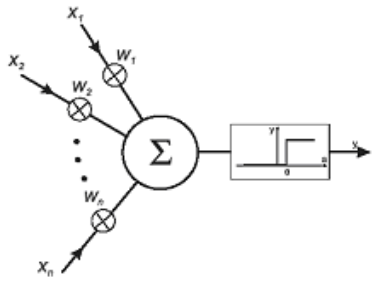


Figure 6.2. The structure of the artificial neuron

Each of the inputs is multiplied by a connection weight represented by w_i. The products are summed and passed through a transfer function. This function then turns this number into a real output via some algorithm. It is this algorithm that takes the input and turns it into a zero or a one, a minus one or a one, or some other number. The transfer functions those are commonly supported are sigmoid, sine, hyperbolic tangent, etc. This transfer function also can scale the output or control its value via thresholds. The result of the transfer function is usually the direct output of the processing element.

Sigmoid and Tanh functions are shown in Figure 6.2 and they are defined as

$$\boldsymbol{sigmoid}(\boldsymbol{x}) = \frac{1}{1+\boldsymbol{e}^{-\boldsymbol{x}}} \tag{6.1}$$

$$\boldsymbol{g}(\boldsymbol{x}) = \tanh(\boldsymbol{x}) \tag{6.2}$$

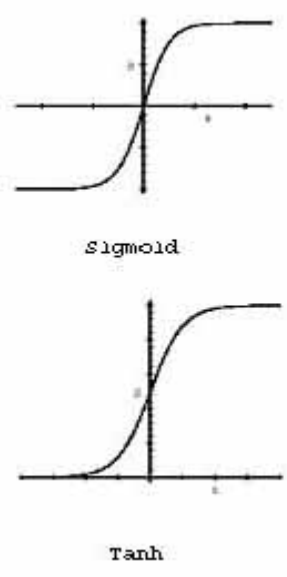


Figure 6.3. Sigmoid and tanh transfer functions

This output is then input into other processing elements, or to an outside connection. All artificial neural networks are constructed from this basic building block - the processing element or the artificial neuron.

6.3. Building Neural Networks

Biologically, neural networks are constructed in a three-dimensional world from microscopic components. These neurons seem capable of nearly unrestricted interconnections. That is not true of any proposed, or existing, man-made network. Integrated circuits, using current technology, are two-dimensional devices with a limited number of layers for interconnection. This physical reality restrains the types, and scope, of artificial neural networks that can be implemented in silicon. Clustering these neurons occurs by creating layers that are connected to one another.

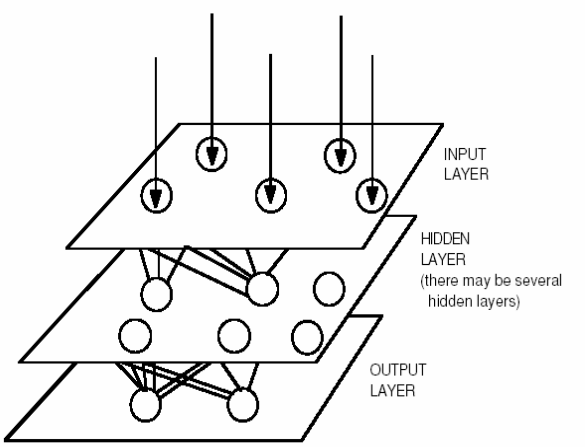


Figure 6.4. The structure of a basic multi-layer artificial neural network [28]

Today, most of the artificial neural networks have a similar structure that is given in Figure 6.4. In this structure some of the neurons contact with the outside as input, some contact as output and the rest of the neurons are hidden. A neural network has a structure. Neurons do not connect to each other randomly. They are grouped into layers and these layers are connected each other to form the neural network.

Basic neural networks are single layer networks that do not have a hidden layer in their structure. The limitations with single layer networks are to some extent overcome by the introduction of multi-layer networks. The output layer often has linear activation functions if the network is used for function approximation, and sigmoid if the network is used for classification purposes.

6.4. Learning in Neural Networks

In the human brain, information is passed between the neurons in the form of electrical stimulation along the dendrites. If a certain amount of stimulation is received by a neuron, it generates an output to all other connected neurons and so information takes its way to its destination where some reaction will occur. Explaining how the human brain learns certain things is quite difficult and nobody knows it exactly.

It is supposed that during the learning process the connection structure among the neurons is changed, so that certain stimulations are only accepted by certain neurons. This means, there exists firm connections between the neural cells that once have learned a specific fact, enabling the fast recall of this information. The artificial neural net is also made of neurons and dendrites. Unlike the biological model, a neural net has an unchangeable structure, built of a specified number of neurons and a specified number of connections between them called *weights*, which have certain values. Only the weights are changed during the network processing. Compared to the original this means: Incoming information "stimulates" (exceeds a specified threshold value of) certain neurons that pass the information to connected neurons or prevent further transportation along the weighted connections. The value of a weight will be increased if information should be transported and decreased if not.

While learning different inputs, the weight values are changed dynamically until their values are *balanced*, so each input will lead to the desired output. Very often there is a certain error left after the learning process, so the generated output is only a good approximation to the perfect output in most cases.

The basic approach in learning is to start with an untrained network, present a training pattern to the input layer, pass the signals through the net and determine the output at the output layer. These outputs are compared to the target output values. Any difference in this situation corresponds to an error. This error function is some scalar function of the weights and is minimized when the network outputs match the desired outputs. The weights are adjusted to reduce this measure of error. Training error on the n^{th} pattern E^n is the sum over output units of the squared difference between the desired output d_k and the network output y_k.

$$E^n = \frac{1}{2}\sum_{k=1}^{C}(d_k - y_k)^2 \tag{6.3}$$

C is the number of the output units in the output layer of the network that we use one for the networks in this study. Total error of the network through all patterns is then defined as

$$\boldsymbol{E} = \sum_{\boldsymbol{n}} \boldsymbol{E}^{\boldsymbol{n}} \tag{6.4}$$

The *backpropagation learning rule*, which is based on gradient descent, is used to minimize the total error. It corresponds to a propagation of errors backwards through the network. The technique of back-propagation was popularized in a paper by [29]. However, similar ideas had been developed earlier by a number of researchers [30, 31].

In backpropagation the weights are initialized with random values and these weights are changed in a direction that will reduce the error as

$$\Delta \boldsymbol{w} = -\eta \frac{\partial \boldsymbol{E}}{\partial \boldsymbol{w}} \tag{6.5}$$

where η is the learning rate that indicates the relative size of the change in the weights.

These weights are then updated each iteration as

$$\boldsymbol{w}' = \boldsymbol{w} + \Delta \boldsymbol{w} \tag{6.6}$$

The derivations for $\frac{\partial \boldsymbol{E}}{\partial \boldsymbol{w}}$ will not be shown, as it is not in the scope of this study. For multilayer networks, these derivations are applied from the output layer to the input layer using chain rule.

There are two approaches to training: supervised and unsupervised. Supervised training involves a mechanism of providing the network with the desired output either by manually "grading" the network's performance or by providing the desired outputs with the inputs.

Unsupervised training is where the network has to make sense of the inputs without outside help. The vast bulk of networks utilize supervised training. Here in this thesis, we are trying to make our network learn and able to predict the market future by considering past occurrences. Therefore this study is concentrated over supervised learning.

Also there are two major categories of network training, the *incremental* and the *batch* training. During the *incremental* training the weights of the network are adjusted each time that each one of the input samples are presented to the network, while in *batch* mode training the weights are adjusted only when all the training samples have been presented to the network [10]. The number of times that the training set will be fed to the network is called *number of epochs*.

In neural network processing, you have to consider several affecting factors to have a greater performance and a good generalization over the data. If these factors are not considered during the learning process, it is not possible to expect good generalization [32]. These factors are

- Input Data Selection
- Preprocessing
- Cross-Validation
- Number of hidden units
- Initializing weights
- Learning and Momentum Rate
- Weight Decay
- Early Stopping

6.4.1. Input Data Selection

We have to make some decisions that have to be done while choosing input data. A neural network's performance is dependent on the quality and relevance of its data. Selecting wrong data will cause poor generalization. The questions that we should ask about the problem domain are the market theory to implement the prediction on and the candidate input sources.

We should take the implications of the markets theory and decide candidate inputs for our system. The candidate input data can be technical, fundamental or inter market data.

It is a hard problem to predict time series only by looking at the historical data because financial data are not stationary and very sensitive to economic shocks. Financial markets in Turkey are extremely chaotic showing non-linearity with high noise because of the political instability structure and instability and economical problems. Any news, any rumor will cause sudden rises or falls at the stock market.

6.4.2. Preprocessing

You cannot use raw data as an input for the neural network unless you want a bad predictor. We have to preprocess the raw data. The preprocessing period is composed of two main tasks, transformation and normalization. As described in section 2.2, transforming the prices into continuously compounded returns is helpful in supplying stationary series and removing trends in the series. For normalization, scaling is a useful technique having all the data scaled in a range from -1 to 1 or 0 to 1.

6.4.3. Cross-Validation

The data is split into two parts. The models are trained in the training data set and they are tested in the production (test) data set. By applying cross-validation, the over fitting problem is avoided and a good generalization is achieved.

In this thesis study, data set consists of three parts, training, validation and test sets. During the training, the data in the training set will be examined and the network will learn the relationships between the inputs and outputs. During the training process, the cross validation set is used for performance benchmarking over the learning process and several results are compared for the selection of best weight set. The production test set is the period where the model is used for trading in the market. This study follows the common approach in considering the size of these data sets and selects 70 per cent of the data as training, 20 per cent of the data as validation and the remaining 10 per cent of the data as the production data sets.

6.4.4. Number of Hidden Units

The number of hidden units governs the expressive power and complexity of the network. Increasing the number of hidden units does not mean better performance when considering neural learning. When you increase the number of hidden units, you will see a decrease in the error during the training phase, but the test set error will be very high because of the over fitting of the data. Finding the appropriate number of hidden units is an ad-hoc process that is not exactly solved in neural processing. When increasing the number of hidden units, there comes a time that you start to memorize the data. After that time, you will see an increase in the test set error and that is the time to stop before over fitting the data. But, stopping the learning in an early stage will result in under fitting that is also not desired.

6.4.5. Initializing Weights

The starting point of the network is also one of the most important pre-conditions in the learning process of the neural network that is not also yet solved. In the error surface, you have to start in a point that will take you to the global minima.

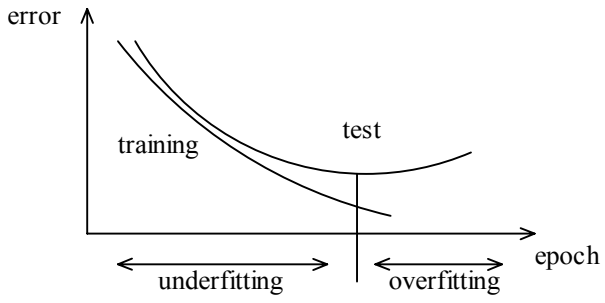


Figure 6.5. Error surface

There may be several local minima that will make the network not be able to generalize well or as desired. For this purpose, the initialization of the weights plays a crucial role in the network performance. You cannot initialize the weights to 0 otherwise learning cannot take place. In setting the weights, we choose weights randomly from a

single distribution to help ensure uniform learning and have the network to generalize well. Usually, weights are chosen between -0.1 and 0.1 randomly.

6.4.6. Learning and Momentum Rate

Learning rate value determines only the speed at which the network attains a minimum in the error function. Choosing a small value for learning rate will make the learning slow and it is not desired. On the other hand, choosing a larger value is not appropriate and will affect the performance of the network. The error may fluctuate and may degrade the performance while having speedy network learning. Momentum rate adds inertia to the motion through weight space and smoothes out the oscillation in the error surface. Momentum is defined as

$$\Delta \boldsymbol{w}^{\boldsymbol{t}} = -\eta \frac{\partial \boldsymbol{E}^{\boldsymbol{t}}}{\partial \boldsymbol{w}} + \alpha \Delta \boldsymbol{w}^{\boldsymbol{t}-1} \tag{6.7}$$

where α is usually chosen as 0.5.

6.4.7. Weight Decay

Weight decay is a *regularization* technique. Regularization refers to a set of techniques which helps to ensure that the function computed by the network is no more curved than necessary. This is achieved by adding a penalty to the error functions, giving:

$$\boldsymbol{E}' = \boldsymbol{E} + \frac{\lambda}{2} \sum_{\boldsymbol{i}} \boldsymbol{w}_{\boldsymbol{i}}^2 \tag{6.8}$$

where $\boldsymbol{E}$ shows the total error, $\boldsymbol{w}_{\boldsymbol{i}}$'s are individual weights $(\boldsymbol{i} = 1,...)$.

6.4.8. Early Stopping

This is an alternative to regularization as a way of controlling the complexity of the network. The error measured with respect to the unseen data, called test data, shows a decrease at first, followed by an increase as the network starts to over-fit. Training can therefore be stopped at the point of the smallest error. This gives the network to have the best generalization performance.

6.5. Neural Network Models

There are two types of neural networks studied in this study.

- Feed-forward neural networks
- Recurrent neural networks

Feed forward neural network have their connection in a single way from bottom-to-top i.e. There are

- Multilayer Perceptron (MLP)
- Radial Basis Functions (RBF)
- Mixture of Experts (MoE)

Recurrent neural networks studied in this study are Jordan, Elman and capacitive hybrid neural networks that will be discussed later.

6.5.1. Multi-layer Perceptron (MLP)

MLP is a useful network that is able learn the non-linearity in data and is very useful in both regression and classification problems. As a subset of regression problem, function approximation in time series has been studied before and research at this area has not been come across with results that satisfy any participant in the field. The *Universal*

Approximation Property mentions that an MLP can approximate any nonlinear function to an arbitrary degree of accuracy with a suitable number of hidden layers [33]. Most of the studies are Auto-Regressive (AR) model's simulations having assumed that the data depends on *p* previous elements in the time series. For this purpose, a window is defined that is including the previous *p* samples in the data set and having their weighted sums. This technique is known as the sliding windows technique. In Figure 6.6, a sample multilayer perceptron neural network is shown.

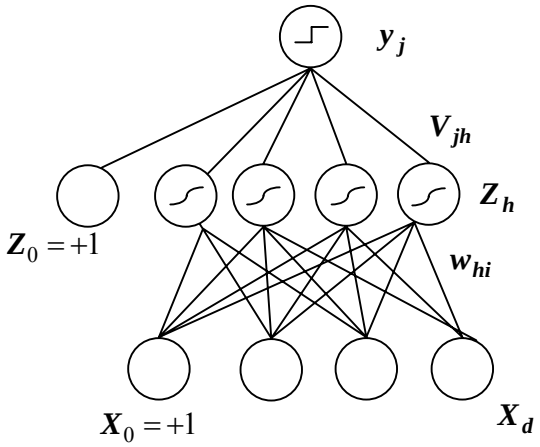


Figure 6.6. A simple regression MLP network

6.5.2. Radial Basis Functions (RBF)

The class of radial basis functions was first used to solve interpolation problems -- fitting a curve exactly through a set of points. RBF methods have their origins in techniques for performing exact interpolation of a set of data points in a multi-dimensional space [34]**.**

Consider a mapping from a d-dimensional input space x to a one-dimensional target space t. The data set consists of N input vectors $\boldsymbol{x}^{\boldsymbol{n}}$, together with corresponding targets $\boldsymbol{t}^{\boldsymbol{n}}$. The goal is to find a function $\boldsymbol{h}(\boldsymbol{x})$ such that

$$\boldsymbol{h}(\boldsymbol{x}) = \boldsymbol{t}^{\boldsymbol{n}}, \ \boldsymbol{n} = 1,..., \boldsymbol{N} \tag{6.9}$$

The RBF approach introduces a set of N basis functions, one for each data point, which take the form $\phi(\|x - x^n\|)$ where $\phi(.)$ is some non-linear function. The n^{th} function depends on the distance $\|x - x^n\|$, usually taken to be Euclidean, between x and x^n. The output of the mapping is then taken to be a linear combination of the basis functions.

$$h(x) = \sum_n w_n \phi(\|x - x^n\|) \tag{6.10}$$

For $\phi(.)$, several forms of basis functions have been considered, the most common being the Gaussian is

$$\phi(x) = \exp(-\frac{x^2}{2\sigma^2}) \tag{6.11}$$

where σ is a parameter whose value controls the smoothness properties of the interpolating function.

Girosi and Poggio have shown that radial basis functions possess the property of best approximation. An approximation scheme has this property if, in the set approximating functions there is one function which has minimum approximating error for any given function to be approximated. They have also shown that this property was not shared by multilayer perceptrons [35].

6.5.3. Recurrent Neural Networks (RNN)

Time can be introduced into ANN models in different ways. First possibility is to leave the time outside the ANN model and preprocess time to use it in standard ANN models. This method is used in Time Delay Neural Networks (TDNN) or in ANNs with a sliding window as it was applied in MLP [36]. Here the last n values of a time-dependent variable are given as an input to the network. Another possibility is to encode the temporal information into numerical values first and use these values in the ANN. Introduction of

time into the model as an index of the state of the network is another way of handling time in ANNs. In this type of ANN, states of neurons are kept to be used in the following time steps. Order is the most important property for time in these models. Therefore these models are very suitable for data where order relation between observations is important. Stock market time series data is such a data type. These models are known as RNNs.

We need memory to store the state information in ANN. This memory should have certain characteristics. These are

- Form of the memory
- Content of the memory
- Adaptability

The dimension of the memory form characterizes how the memory stores information with respect to time. The simplest form is a buffer containing only the last n inputs also called tapped delay line. Another memory form is exponential trace that stores information over an infinite time but the importance; the weight of an input decreases exponentially. The more recent inputs have a greater impact on the process. Gamma memory generalizes tapped delay line and exponential trace memory [37]. In memory, information about the inputs, outputs and the state of the network can be stored. Adaptability of a memory is also an important issue. In a static memory the parameters are fixed and the memory state is a predefined function of its inputs. Time Delay Neural Networks uses static memory. But as in recurrent ANNs parameters can adaptively change during the learning in adaptive memories.

6.5.4. RNN Architectures

There are several RNNs possible to be proposed. But in the literature two of them have gathered most of the importance that are Elman and Jordan networks [38, 39].

6.5.4.1. Jordan's RNN. This model is an early RNN proposed by Jordan [39]. This model uses connections from the output units back to the input of the network in order to

achieve the learning of sequential tasks in language processing. The units that store the last output are called state units and are connected to the hidden units in the same way as the input units. Additionally the state units are connected with each other and with themselves. This allows them to calculate their next state as a function of the current network output, their current state and the current state of other state units.

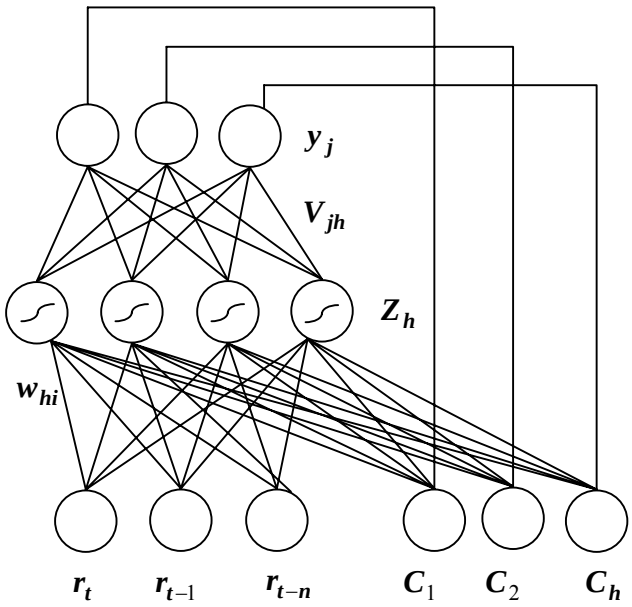


Figure 6.7. Topology of Jordan network

6.5.4.2. Elman's RNN. Instead of using recurrent connections from output units to the network input, Elman's recurrent ANN stores the activations of hidden units as representation of the network's internal state and feeds them back as part of the hidden layer's input at the next time step [38]. The activation of hidden units at time step t are directly copied into the so-called context units and kept there as input to the hidden units at the next time step t+1. There is a connection with a weight value of one and a time delay of one step. Compared to the use of Jordan, the use of hidden activations has an advantage because the stored state value has the same dimension with the hidden layer units. Elman suggests a form of back-propagation as learning algorithm, which adjusts the weights only with respect to the current input and the last internal state. Back-propagation through time (BPTT) is also a possible learning method, which makes it necessary to store a complete history of unit activation values [40]. In this study, Real-time Recurrent Learning (RTRL)

algorithm is used to learn the temporal effects of the time series for recurrent neural networks.

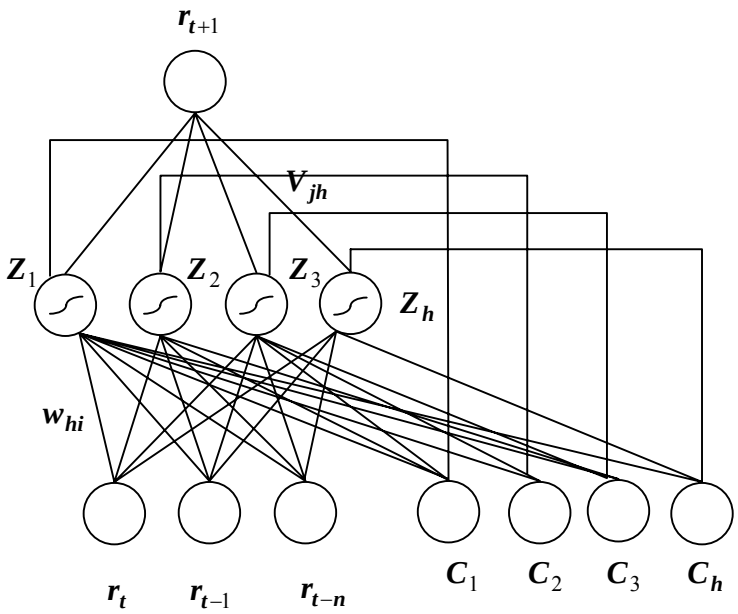


Figure 6.8. A simple recurrent network in which activations are copied from hidden layer to context layer on a one-for-one basis, with fixed weight of one

Also capacitive RNNs are considered, which has a capacitive layer for the context layer to store the previous state of the context state in addition to the hidden layer's state storage.

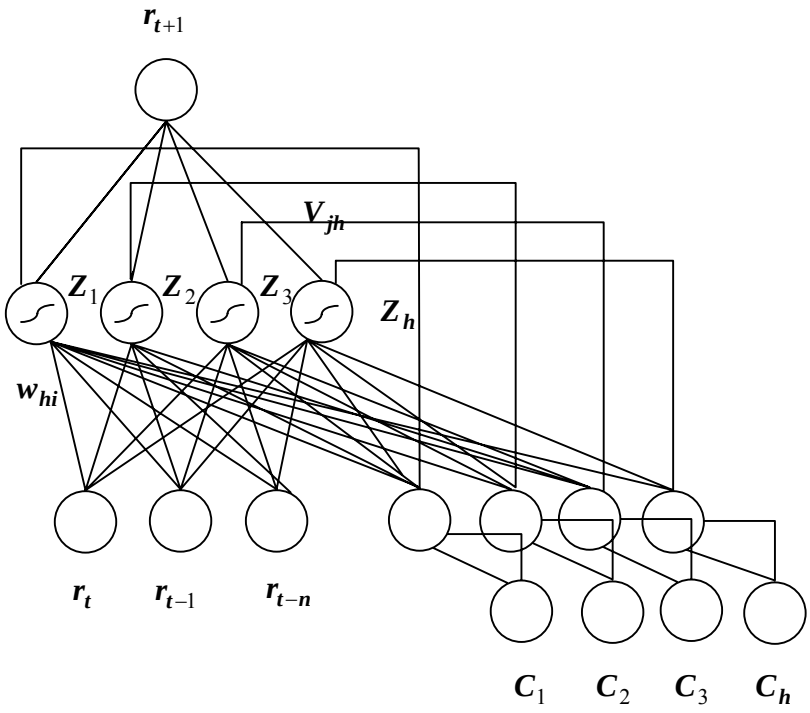


Figure 6.9. Capacitive Elman network topology

6.5.5. Real-Time Recurrent Learning (RTRL)

Suppose we have a set of inputs, $\mathrm{I} = \{\mathrm{x_k(t)}, 0 < \mathrm{k} < \mathrm{m}\}$, and a set of other units, $\mathrm{U} = \{\mathrm{y_k(t)}, 0 < \mathrm{k} < \mathrm{n}\}$, which can be hidden or output units. In order to index these units in the network,

$$z_k^t = \begin{cases} x_k^t, k \in I \\ y_k^t, k \in U \end{cases} \tag{6.12}$$

Let **W** be the weight matrix with *n* rows and $\mathrm{n+m}$ columns, where $\mathrm{w_{ij}}$ is the weight to unit i (which is in *U*) from unit j (which is in *I* or *U*). The weighted sum of the inputs is then,

$$net_k(t) = \sum_{l \in U \cup I} w_{kl} z_l(t) \tag{6.13}$$

The temporal index t represents the time. The result is then passed through a nonlinear transfer function

$$y_k(t+1) = f_k(net_k(t)) \tag{6.14}$$

Usually, both hidden and output units will have non-linear activation functions. Note that external input at time *t* does not influence the output of any unit until time *t+1*. The network is thus a discrete dynamical system. Some of the units in *U* are output units, for which a target is defined. A target may not be defined for every single input however. Let $T(t)$ be the set of indices *k* in *U* for which there exists a target value $d_k(t)$ at time *t*. We are forced to use the notation d_k instead of *t* here, as *t* now refers to time. Let the error at the output units be

$$e_k^t = \begin{cases} d_k^t - y_k^t & , k \in T(t) \\ 0 & , otherwise \end{cases} \tag{6.15}$$

and define our error function for a single time step as

$$E(\tau) = \frac{1}{2}\sum_{k \in U}\left[e_k(\tau)\right]^2 \tag{6.16}$$

The error function we wish to minimize is the sum of this error over all past steps of the network

$$E_{total}(t_0, t_1) = \sum_{\tau=t_0+1}^{t_1} E(\tau) \tag{6.17}$$

because the total error is the sum of all previous errors and the error at this time step, so also, the gradient of the total error is the sum of the gradient for this time step and the gradient for previous steps

$$\nabla \boldsymbol{wE}_{\boldsymbol{total}}(\boldsymbol{t}_0, \boldsymbol{t}+1) = \nabla \boldsymbol{wE}_{\boldsymbol{total}}(\boldsymbol{t}_0, \boldsymbol{t}) + \nabla \boldsymbol{wE}(\boldsymbol{t}+1) \tag{6.18}$$

As a time series is presented to the network, we can accumulate the values of the gradient, or equivalently, of the weight changes.

$$\Delta \boldsymbol{w}_{\boldsymbol{ij}}(\boldsymbol{t}) = -\eta \frac{\partial \boldsymbol{E}(\boldsymbol{t})}{\partial \boldsymbol{w}_{\boldsymbol{ij}}} \tag{6.19}$$

After the network has been presented with the whole series, we alter each weight $\boldsymbol{w}_{\boldsymbol{ij}}$ by

$$\sum_{\boldsymbol{t}=\boldsymbol{t}_0+1}^{\boldsymbol{t}_1} \Delta \boldsymbol{w}_{\boldsymbol{ij}}(\boldsymbol{t}) \tag{6.20}$$

We therefore need an algorithm that computes

$$-\frac{\partial \boldsymbol{E}(\boldsymbol{t})}{\partial \boldsymbol{w}_{\boldsymbol{ij}}} = -\sum_{\boldsymbol{k}\in\boldsymbol{U}} \frac{\partial \boldsymbol{E}(\boldsymbol{t})}{\partial \boldsymbol{y}_{\boldsymbol{k}}(\boldsymbol{t})} \frac{\partial \boldsymbol{y}_{\boldsymbol{k}}(\boldsymbol{t})}{\partial \boldsymbol{w}_{\boldsymbol{ij}}} = \sum_{\boldsymbol{k}\in\boldsymbol{U}} \boldsymbol{e}_{\boldsymbol{k}}(\boldsymbol{t}) \frac{\partial \boldsymbol{y}_{\boldsymbol{k}}(\boldsymbol{t})}{\partial \boldsymbol{w}_{\boldsymbol{ij}}} \tag{6.21}$$

at each time step *t*. Since we know $\boldsymbol{e}_{\boldsymbol{k}}(\boldsymbol{t})$ at all times (the difference between our targets and outputs), we only need to find a way to compute the second factor $\frac{\partial \boldsymbol{y}_{\boldsymbol{k}}(\boldsymbol{t})}{\partial \boldsymbol{w}_{\boldsymbol{ij}}}$.

This measure is a measure of the sensitivity of the value of the output of unit *k* at time *t* to a small change in the value of $\boldsymbol{w}_{\boldsymbol{ij}}$, taking into account the effect of such a change in the weight over the entire network trajectory from t_0 to *t*. $\boldsymbol{w}_{\boldsymbol{ij}}$ does not have to be connected to unit *k*. Thus this algorithm is non-local, in that we need to consider the effect of a change at one place in the network on the values computed at an entirely different place.

From (6.13) and (6.14), we get

$$\frac{\partial y_k(t+1)}{\partial w_{ij}} = f_k'(net_k(t))\left[\sum_{l \in U \cup I} w_{kl} \frac{\partial z_l(t)}{\partial w_{ij}} + \delta_{ik} z_j(t)\right] \tag{6.22}$$

where δ_{ik} is the Kronecker delta

$$\delta_{ik} = \begin{cases} 1, i = k \\ 0, otherwise \end{cases} \tag{6.23}$$

because input signals do not depend on the weights in the network,

$$\frac{\partial z_l(t)}{\partial w_{ij}} = 0 \text{ , for } l \in I \tag{6.24}$$

Equation (6.22) becomes:

$$\frac{\partial y_k(t+1)}{\partial w_{ij}} = f_k'(net_k(t))\left[\sum_{l \in U} w_{kl} \frac{\partial y_l(t)}{\partial w_{ij}} + \delta_{ik} z_j(t)\right] \tag{6.25}$$

we assume that our starting state ($t = 0$) is independent of the weights, we have

$$\frac{\partial y_k(t_0)}{\partial w_{ij}} = 0 \tag{6.26}$$

These equations hold for all $k \in U, i \in U$ and $j \in U \cup I$.

We therefore need to define the values

$$p_{ij}^k(t) = \frac{\partial y_k(t)}{\partial w_{ij}} \tag{6.27}$$

for every time step t and all appropriate i, j and k. We start with the initial condition

$$p_{ij}^{k}(0) = 0 \tag{6.28}$$

and compute at each time step

$$p_{ij}^{k}(t+1) = f_k'(net_k(t))\left[\sum_{l \in U} w_{kl}\, p_{ij}^{l}(t) + \delta_{ik}\, z_j(t)\right] \tag{6.29}$$

The algorithm then consists of computing, at each time step t, the quantities $p_{ij}^{k}(t)$ using (6.27) and (6.28), and then using the differences between targets and actual outputs to compute weight changes

$$\Delta w_{ij}(t) = \eta \sum_{k \in U} e_k(t) p_{ij}^{k}(t) \tag{6.30}$$

and the overall correction to be applied to w_{ij} is given by

$$\Delta w_{ij} = \sum_{t=t_0+1}^{t_1} \Delta w_{ij}(t) \tag{6.31}$$

6.5.6. Mixture of Experts (MoE)

The basic idea behind mixture models is to divide the input space into several parts, to build a model for each part, and then to combine the local models to get a better global model. This approach is based on the principle of divide-and-conquer and is widespread in different fields of computer science and applied mathematics. Divide-and-conquer is used, when it is not possible, too difficult, or not appropriate to solve a complex problem at once. The problem is divided into smaller problems and each solution is combined to form the solution of the complex problem. A divide-and-conquer algorithm typically consists of two

classes of modules: one class of module solves the simpler problem and the other class of module assigns which part of the problem the other class is responsible and combines the results of other class of modules that solve the simpler problems. This represents a hierarchical architecture.

There are two problems in this scenario that is considered. The first one is the splitting of the input space into a set of regions and the other one is to fit each region with a simpler function compared with a global fitting function.

Against the problems, the divide-and-conquer approach in mixture of experts applied to forecasting has several advantages. Each local model can specialize on a different part of the input space. The combination of these specialized models can result in a better model than possible with a single global model.

Jacobs and Jordan proposed a learning procedure for systems composed of many separate networks, each concentrating on its region of interest in the training data [41, 42]. If back-propagation is used to train a single multilayer network to perform different subtasks on different occasions, there will be interference and this will lead to poor generalization besides slow learning. If we divide the problem and attach experts to each of them only responsible for its region and combine the results of these experts in a gating network, then we will be able to reduce the interference and have a better generalization. The weight changes of the experts are localized to itself. There is no interference with the weight changes and each expert will be allocated to only a small local region of the space of possible input vectors.

Each expert is a feed-forward network and all experts receive the same input and have the same number of outputs. The gating network is also feed forward, and typically receives the same input as the expert networks. It has normalized outputs $\boldsymbol{p_j} = \frac{\exp(\boldsymbol{x_j})}{\sum_{\boldsymbol{i}} \exp(\boldsymbol{x_i})}$, where x_j is the total weighted input received by output unit j of the gating network. The selector acts like a multiple input single output stochastic switch. The probability that the switch will select the output from expert j is p_j [41].

The Mixture of Experts (MoE) [41] has emerged as a powerful divide and conquer algorithm. It consists of a set of competing experts moderated by a gate. When the experts are predictors and the data set is a time series [43], the algorithm is capable of segmenting and identifying the time series into stationary regions, in a completely unsupervised fashion, by observing the output of the gate over time.

Given a data set $\{\boldsymbol{X}, \boldsymbol{D}\}$, where $\boldsymbol{X}$ is the input, $\boldsymbol{D}$ is the desired signal, and a function $\boldsymbol{Y} = \boldsymbol{f}(\boldsymbol{X}, \boldsymbol{W})$, where $\boldsymbol{W}$ is a set of free parameters, the maximum likelihood solution for W under a Gaussian assumption on the error is equivalent to minimizing the mean square error. However, many data sets do not exhibit single Gaussian modes. The Mixture of Experts directly attacks this problem with a set of experts moderated by a gate, and models the overall probability density function (***pdf***) of the system as a weighted sum of the experts' ***pdf*** 's,

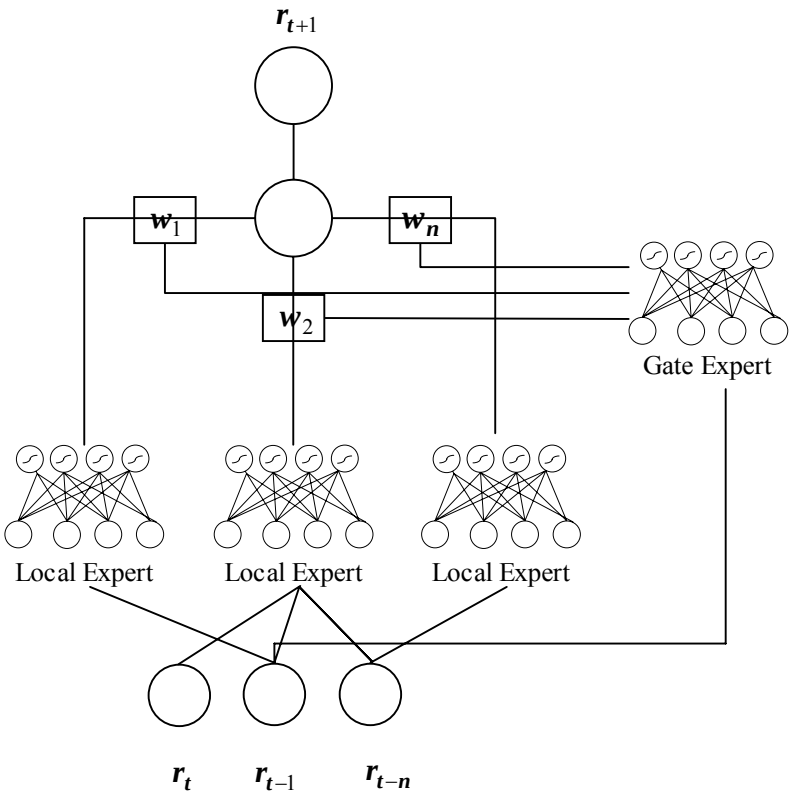


Figure 6.10. A system of expert and gating networks

$$p(d \mid x) = \sum_{k=1}^{K} P(k \mid x) p(d \mid x, k) = \sum_{k=1}^{K} g_k(x) p(d \mid x, k) \tag{6.32}$$

where $\boldsymbol{x}$ is the input, $\boldsymbol{d}$ is the desired output, and $\boldsymbol{k}$ is the expert index, and $\boldsymbol{g_k}$ is the output of the gate. Mixture of Experts differs from simple Gaussian mixture modeling in that the mixture coefficients are a function of the input. For the special case of prediction of a one-dimensional time series given by *x(t)*, $\boldsymbol{x}$ is the output of a tapped delay line , and the desired signal is the next point in the time series. For regression, the ***pdf*** 's are chosen to be a Gaussian function of the error,

$$p(d \mid x, k) = \frac{e^{\left[-\frac{1}{2}(y_k - d)^T \sum_k^{-1} (y_k - d)\right]}}{(2\pi)^{a/2} \left| \sum_k \right|^{1/2}} \tag{6.33}$$

where $\boldsymbol{y_k}$ is the output of the $\boldsymbol{k^{th}}$ expert and *a* is the dimensionality. For a one-dimensional time series, *a = 1*. In order to reduce the number of free parameters, it is common to make the assumption that the cross correlation of the error of the $\boldsymbol{k^{th}}$ expert can be approximated by

$$\sum_k = E\left[(y_k - d)(y_k - d)^T\right] \approx I\sigma_k^2 \tag{6.34}$$

Multiplying (6.32) by $\boldsymbol{d}$ and integrating (over the desired response) results in:

$$y = E[d \mid x] = \sum_{k=1}^{K} g_k(x) y_k(x) \tag{6.35}$$

Thus, the output of the network is a weighted sum of the expert's outputs, with the weighting coefficients provided by the gate. The experts and the gate can be chosen to be either generalized linear models (GLIM's) or non-linear neural networks. Weigend used a single non-linear gate and non-linear experts, and called the model non-linear gated experts [43]. The advantage of using a non-linear gate is that a single hierarchical layer can

theoretically solve any problem, whereas the linear model requires a hierarchical arrangement of gates, known as the Hierarchical Mixture of Experts (HME) [42].

7. PERFORMANCE METRICS

For every problem in science, you have to show that your model has better aspects than others. The comparisons are based on pre-defined measures. For forecasting problems also, there must be such measures that we depend our models on. Even a prediction algorithm that loses money in a period of time may turn out to be successful when compared to other alternatives. In this thesis, we have to measure the learning in network and compare the results from the statistics and finance point of views. But how shall we measure the performance of the learning networks for our forecasting problem? Firstly, we measure how well our network's output matches the true outputs by using standard regression statistics. For this purpose, we have used similar techniques of error or loss functions including mean square error (mse), root mean square error ($rmse$), mean absolute percentage error ($mape$), correlation ($corr$) and $\boldsymbol{R}^2$. But, these are not enough for a financial time series prediction problem. There is a point that should never be forgot while trying to forecast future in a time series problem; measuring the accuracy of the prediction can only be understood when the future occurs as the system states reach. But as helper measures for standard regression statistics, we have used Theil Inequality Coefficient (TIC) and its sub-components that are mean, variance and co-variance components in TIC, hit rate and derivatives for the accuracy of the forecast direction and for testing the profit assuming a real trading environment we have used mean profit per trade. The estimates are then tested through several benchmarks that are defined in Section 7.1. We can split our studies into three groups including price predictions, return predictions and volatility predictions. We can use all these measure for all three kinds of problems.

7.1. Benchmarks

During the evaluation of the stock prediction we need good benchmarks. What is good and bad performance should be separated clearly. For this purpose, in this study we have used several benchmarks stated below.

7.1.1. 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.

7.1.2. 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.

7.2. Metrics

The metrics defined in Section 7.2.1 are used to compare the predicted values to the actual outcome.

7.2.1. Regression Statistics

For testing the performance of a model proposed, the basic regression statistics used is mean square error (***mse***) that seizes the matching of the two data sets. For understanding the learning in neural network also we use *mse* as the error measure especially in multi-layer perceptron. ***Mse*** squares the error and effects the performance badly if the error gets big. Root mean square error (*rmse*) is also a very useful and highly applicable measure for testing the performance from the regression point of view. Another *mse* derived error measure is mean absolute percentage error (*mape*). 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.

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

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

$$ess = \sum_{t=1}^{N} (r_t - \hat{r}_t)^2, rss = \sum_{t=1}^{N} (\hat{r}_t - \bar{r})^2, tss = \sum_{t=1}^{N} (r_t - \bar{r})^2, R^2 = 1 - \frac{ess}{tss} \tag{7.3}$$

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

7.2.2. Theil Inequality Coefficient

The Theil Inequality Coefficient (TIC) of inequality provides a measure of the model performance relative to the naïve predictor [44]. TIC takes the naïve predictor for returns as a benchmark and compares with the given predictor model. For $TIC > 1$, the predictor is worse than the naïve predictor. TIC is expected to be as close as zero. Theil coefficient is often referred to as the information coefficient or t-test. When predicting returns, return series is used instead of price series and naïve predictor of returns is selected as the benchmark.

$$TIC = \frac{\sqrt{\sum_{t=1}^{N} (p_t - \hat{p}_t)^2}}{\sqrt{\sum_{t=1}^{N} (p_t - p_{t-1})^2}} \tag{7.5}$$

p_t represents the price value at time t in the series and $\hat{p}_t$ represents the prediction of the price at time t.

TIC includes three explanatory proportions that are bias, variance and covariance. The total of these proportions have to be equal to one.

$$U_m + U_s + U_c = 1 \tag{7.6}$$

U_m, bias proportion is an indication of a systematic error. It measures the average values of predicted and actual series deviate from each other. U_m should be close to zero for a model to be known as good. A large value of U_m such as 0.1 or 0.2 is not acceptable as it means an occurrence of a systematic bias and this means that the model has to be revised. $\bar{\hat{p}}$ is the mean of the predictions and $\bar{p}$ represents the mean of the actual data.

$$U_m = \frac{\left(\bar{\hat{p}} - \bar{p}\right)^2}{\frac{1}{N}\sum_{t=1}^{N}(p_t - \hat{p}_t)^2} \tag{7.7}$$

U_s, the variance proportion indicates the ability of the model to replicate the degree of variability in the variable of interest. If U_s is large, the predicted series shows fluctuation while the actual series does not, or vice versa. Also, this is not so acceptable for a good model. This measure is the standard deviation component in Theil inequality coefficient and it has to be a small value close to zero for a good model. $\sigma_{\hat{p}}$ and σ_p represents the standard deviations of the prediction and the actual data respectively.

$$U_s = \frac{\left(\sigma_{\hat{p}} - \sigma_p\right)^2}{\frac{1}{N}\sum_{t=1}^{N}(p_t - \hat{p}_t)^2} \tag{7.8}$$

U_c, the covariance proportion represents the remaining error after deviations, from average values and average variability has been accounted for. This should be close to 1 for an ideal model. The covariance component should be as close as possible to one for a good model. *corr* is the correlation between the actual data and the prediction.

$$U_c = \frac{2(1-corr)\sigma_{\hat{p}}\sigma_p}{\frac{1}{N}\sum_{t=1}^{N}(p_t - \hat{p}_t)^2} \tag{7.9}$$

7.2.3. Hit Rate (H_R)

The hit rate of a predictor indicates how often the sign of the return is correctly predicted. This is the ratio between the number of correct non-zero predictions and the total number of non-zero movements in the stock time series.

$$\boldsymbol{H_R} = \frac{\left\| \boldsymbol{r_t \hat{r}_t} > 0 \right|_1^{\boldsymbol{N}}\|}{\left\| \boldsymbol{r_t \hat{r}_t} \neq 0 \right|_1^{\boldsymbol{N}}\|} \tag{7.10}$$

The symbol r_t represents the return value at time t in the series and $\hat{r}_t$ represents the prediction of the return at time t. The number of correct predictions is shown as $\|.\|$ in the equations related to hit rate performance. There are several derivatives of hit rate such as positive hit rate (H_R^+), negative hit rate (H_R^-), hit rate for $\varepsilon - Increase$ benchmark ($\boldsymbol{H\varepsilon}$) and directional hit rate (H_{RD}). H_R^+ and H_R^- measures the accuracy of the direction of the positive predictions and negative predictions respectively. The $\varepsilon - Increase$ benchmark is a benchmark of increasing trend. $\varepsilon - Increase$ always assumes all the time an increasing trend. $p_{t+1} = p_t + \varepsilon$. Hit rate for $\varepsilon - Increase$, ($\boldsymbol{H\varepsilon}$) tests the accuracy of the predictions against an increasing trend.

$$H_R^+ = \frac{\left\| r_t > 0 \,\&\&\, \hat{r}_t > 0 \right|_1^N\|}{\left\| r_t > 0 \right|_1^N\|} \tag{7.11}$$

$$H_R^- = \frac{\left\| r_t < 0 \,\&\&\, \hat{r}_t < 0 \right|_1^N\|}{\left\| r_t < 0 \right|_1^N\|} \tag{7.12}$$

In analogy with the Theil Inequality Coefficient the corresponding measure for the naïve predictor is proposed:

$$H_N = \frac{\left\| r_t r_{t-1} > 0 \right|_1^N \right\|}{\left\| r_t r_{t-1} \neq 0 \right|_1^N \right\|} \tag{7.13}$$

The predictor's hit rate performance is measures as

$$H_0 = \frac{H}{H_N}, H_0 > 1 \tag{7.14}$$

This H_0 entity, called Relative Hit Rate, compares the hit rate of the predictor to that of the naïve predictor. For H_0 < 1, the predictor is worse than the naïve return predictor. Directional hit rate (H_{RD}) is a measure of the hit rate over the hit rate against naïve return predictor, which is $H_{RD} = H_R / H_N$. This should be greater than one for a real predictive power.

7.2.4. 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

$$P_{\Re} = 100 \frac{1}{h} \frac{1}{N-h} \sum_{t=h+1}^{N} sign(p_t - p_{t-h}) \frac{p_t - p_{t-h}}{p_{t-h}} \tag{7.15}$$

In a realistic environment, also trading costs should have to be included into this formula.

7.2.5. Net profit (N_p)

Net profit is a measure that is calculated at trading at each time step at the direction of the change.

For volatility modeling the actual purpose of a volatility model is to predict the future volatility, the performance of all models was compared to the performance of a naïve volatility predictor. The squared returns r_t^2 were considered the "true" volatility at time t. In this study, we have accepted the RiskMetrics™ measure as the true volatility rather than using r_t^2 and compare our models according to this assumption. All the metrics mentioned here are also used for volatility performance measurement altering the price p_t and returns r_t with conditional volatilities calculated.

8. EXPERIMENTS AND DISCUSSION

In this chapter, experiments regarding neural networks for financial time series prediction have been studied. First 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 sub-section we have described our data of interest that we have used while applying neural network techniques for financial time series prediction. The data used in this thesis study are collected from the Statistics Database of The Central Bank of the Republic of Turkey and Plato Data[1]. Turkish stock market index ISE National-100 is used as the main series in the study for a 14-year period from 03 January 1989 to 18 September 2003. Also three individual stocks from the Turkish stock market, listed as well at ISE National-100, are studied. These are Is Bankasi C (ISCTR) from banking industry, Tupras (TUPRS) from refinery industry and Nortel Networks Netas Telecommunications (NETAS) from telecommunication industry. The former two are one of the most important two equities in ISE market. Daily continuously compounded series are being used in calculation.

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

Table 8.1. Summary of the return series

Series	Start Date	End Date	Observations
ISE National-100	03/01/1989	18/09/2003	3650
ISCTR	31/05/1990	18/09/2003	3299
TUPRS	31/05/1991	18/09/2003	3042
NETAS	16/03/1993	18/09/2003	2605

[1] Plato Data is located in Istanbul Turkey.

Financial markets in Turkey are chaotic and show non-linearity with high noise because of the political and economical instability. In Figure 8.1, price series for these four target data are shown. Turkish economy has passed important milestones that can easily be extracted from these figures at a first look.

Figure 8.2 shows the continuously compounded return series for the four time series that have been used. All of these series have fluctuations not more than approximately 20 percent. This is because of the 10-percentage limit defined for a single session in a two-session market like Turkey does have.

Table 8.2 reports the statistics for ISE National-100 index; ISCTR, TUPRS and NETAS return series. The results show that ISE National-100 index series are negatively skewed with a high kurtosis, which is a measure of the thickness of the tails, is very high. For equities, skewness is positive but very close to zero. Kurtosis is also very high that shows the fat tails. Jarque-Bera test statistic rejects the null hypothesis of normal distribution (Bera) [45]. ARCH (5) tests the ARCH effects using Engle's [12] test.

In Table 8.2, $Q(5)$ and $Q^2(5)$ are the Ljung-Box statistics of the autocorrelation of residuals and squared residuals respectively; J-B is the Jarque-Bera statistic for normality test. μ and σ are mean and standard deviations respectively. The direction of the returns is very interesting that none of the equities have a dominant direction. Both positive and negative return counts are very close. But the index seems to have a tending to positive direction. The distributional effects in time series studies that are shown in Figure 8.3 are very important especially when studying over volatility predictions. ISE National-100 and other three time series have fat tailed distributions. Equities have an interesting property that smaller returns do not appear so much in the life cycle of these series.



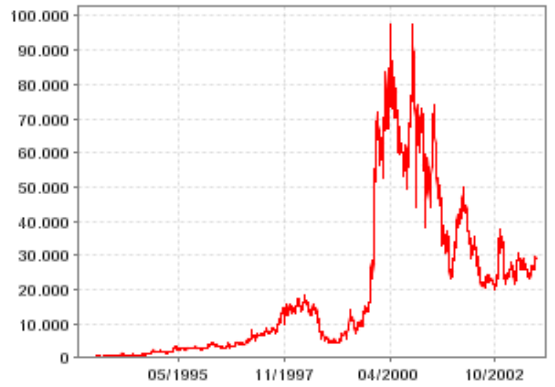
National-100 (03/01/1989-18/09/2003)



Is Bankası (31/05/1990-18/09/2003)



Tupras (31/05/1991-18/09/2003)



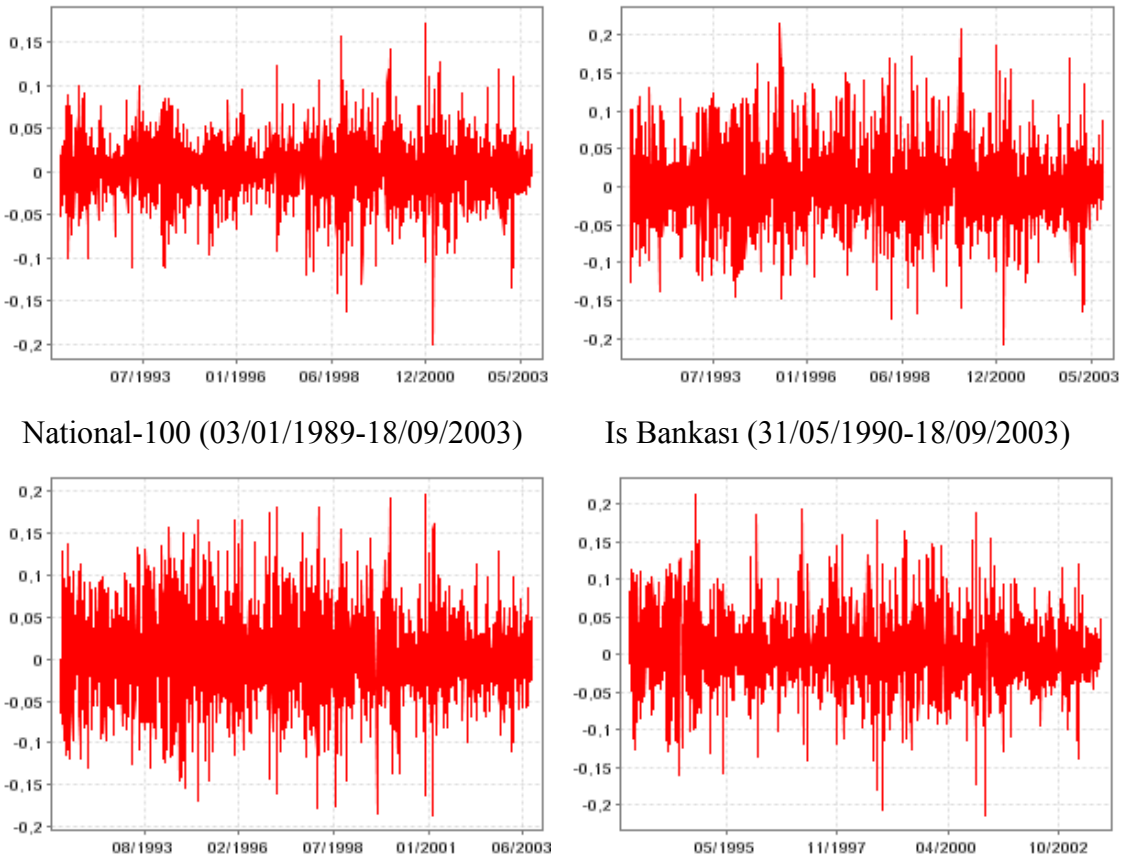
Netas (16/03/1993-18/09/2003)

Figure 8.1. Price series for four series that have been studied on

These time series data sets have been studied using the neural networks approach. The models are described in Section 8.2.

8.2. Models

In this thesis study, we have investigated the use of neural networks in financial time series prediction concentrating on the stock market data of Turkey. Stock market series can be studied in three different ways. You can predict the price, you can predict returns or you can believe in predicting the risk of the market, which is based on the sole volatility models. All these three classes of problems can be studied separately from the point of regression problems. But, in this study we mostly try to estimate the volatility because of the reasons that have been mentioned before in the volatility section and its effect over VaR, which is one of the most accepted criteria for risk management in portfolio selection.



National-100 (03/01/1989-18/09/2003) Is Bankası (31/05/1990-18/09/2003)

Tupras (31/05/1991-18/09/2003) Netas (16/03/1993-18/09/2003)

Figure 8.2. Log return series for four series that have been studied on

We have studied over ISE National-100 index that is governing the most important part of the ISE stock market and three individual equities that are also included in these 100 companies listed in the ISE National-100 index. The companies are chosen from different sectors because of the effects of these sectors over the index and the market. We separately analyzed all these four series and have volatility forecasts for them in order to calculate the VaR and their risk for later use. What we have seen from these studies is that Turkish stock market is chaotic and is very volatile. It includes patterns that can be extracted and used for prediction purposes by the use of ANNs. We have measured the success and accuracy of the predictions by the metrics that we have defined in the performance metrics section and compared these results to traditional and accepted methods of GARCH and EGARCH.

Table 8.2. Summary statistics for the daily returns of four series

	ISE National-100	ISCTR	TUPRS	NETAS
Observations	3650	3299	3042	2605
μ	0.0022	0.0023	0.0026	0.0021
σ	0.0318	0.0464	0.0493	0.0462
Skewness	-0.1187	0.2711	0.1515	0.0021
Kurtosis	5.4918	4.4828	4.0025	4.8693
Minimum	-0.1998	-0.2076	-0.01863	-0.02136
Maximum	0.1709	0.2144	0.1963	0.2126
$Q(5)$	50.369 (0.1165)	37.5719 (0.00000459)	20.5167 (0.0010)	27.1881 (0.000524)
$Q(10)$	70.075 (0.0043)	53.9646 (0.000005)	29.6872 (0.0010)	32.9617 (0.002764)
$Q^2(5)$	579.262 (0)	500.1650 (0)	442.4413 (0)	568.7617 (0)
$Q^2(10)$	746.101 (0)	629.0896 (0)	597.6610 (0)	677.7481 (0)
ARCH(5)	380.7132 (0)	340.4129 (0)	299.2863 (0)	322.9578 (0)
Jarque-Bera (p-value)	950.5516 (0)	341.4994 (0)	138.3501 (0)	402.9407 (0)
$r_t > 0$ (%)	52.6301	40.9642	42.0256	40.5914
$r_t < 0$ (%)	47.0411	40.7823	42.0256	41.2442
$r_t = 0$ (%)	0.3288	18.2535	15.9478	18.1644

8.2.1. ISE National-100

To capture volatility at the Turkish stock market, we study ISE National-100 for a 14-year period from 03 January 1989 to 18 September 2003. There are 3650 daily observations of ISE National-100 series. The series is divided into three parts in order to have good generalization over the data including train, validation and test data sets. Train, validation and test data sets contains 2268, 972 and 359 observations respectively. We used the train data set to train the model and learn the system parameters, then tested the parameters in the validation set during learning to calibrate the system wide parameters such as the number of hidden units in the network and at the end test the parameters with the production (test) data set and calculate our pre-defined measures over the test data set. There are several important factors that affect the performance of the neural network besides the relevance of the data. These are the number of hidden units, learning rate and the application of cross-validation. For time series, also window size is very important, which is the measure we have used to determine how many observations to use for

predicting of $t+1$ at time t. These are given in Table 8.3. All the tests are run on ten times on each model and the average results are taken into consideration. The mean values are reported first and the standard deviations are given in parenthesis in the tables from this point forward[2].

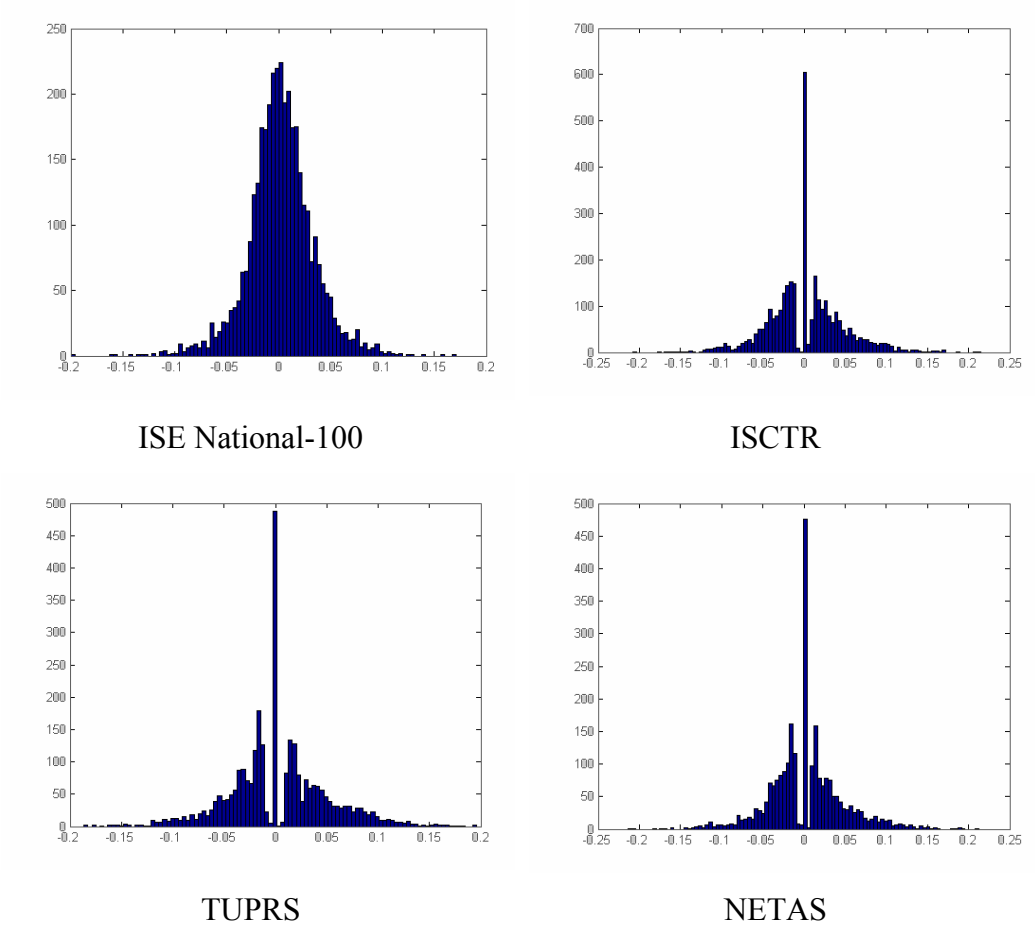


Figure 8.3. Return distributions of stock market data sets

[2] In statistics tables, the values in parenthesis are standard deviation values.

Table 8.3. ISE National-100 data set neural network model parameters

Neural Network Parameters	
Start Date	03/01/1989
End Date	18/09/2003
Observations	3650
Train Set	2268
Validation Set	972
Test Set	359
Hidden Units	20
Window size	20
Learning Rate	0.9
Epochs	50

8.2.2. Methods and Experiments

This study proposes two new methods of volatility modeling that incorporates the traditional methods and a modern approach; neural networks. The volatility measures have been extracted by the methods listed:

- Risk Metrics™
- Traditional GARCH style methods
- Predicting the residuals from ANNs and applying GARCH
- Totally predicting the volatility measure from ANNs
- Year-by-year Analysis
- Intraday Volatility Prediction

This study is mostly concerned with the use of ANNs. We have proposed several types of neural networks for financial time series prediction including MLP, RBF, MoE, and RNN. For RNN, we have studied with Elman, Jordan and recurrence from both hidden and output unit networks including their capacitive versions that are explained in neural networks section in details. In this part we will not give detailed explanations for these networks, but we will explain how we used them for time series prediction. We have analyzed and experienced these volatility models and reported the results with comparisons.

8.2.3. Case I – Risk Metrics™

Risk Metrics™ has a difference from other techniques because it did not come from academia. J.P. Morgan has proposed a method for trading environments [18]. The volatility calculated by RiskMetrics™ is given in Figure 8.4. and the formula of RiskMetrics™ is given in (4.2).

8.2.4. Case II – Traditional GARCH-EGARCH

As we have stated in Chapter 5, traditional approaches are very common and seems like irresistible for the academicians. For a comparison item to our ANN approaches we have also calculated the volatility using both GARCH and EGARCH.

The basic and the most effective GARCH method with one GARCH and one ARCH parameters, GARCH(1,1), and also EGARCH(1,1) has been studied using cross-validation technique for out-of-sample testing. For this purpose, the data set is split into two parts, first including the training part which we calculate the model parameters and the remaining part is called as the test set for testing purposes to compare with other models.

ISE National-100 return data has 3650 elements containing 3000 training and 650 test elements. GARCH(1,1) is applied to this data set and the results are given Table 8.4.

Table 8.4. GARCH (1,1) parameters for ISE National-100

α_0	α_1	β_1	Likelihood
0.000051378	0.16301	0.79363	6350.1

The results are embedded into the equation as shown in (8.1).

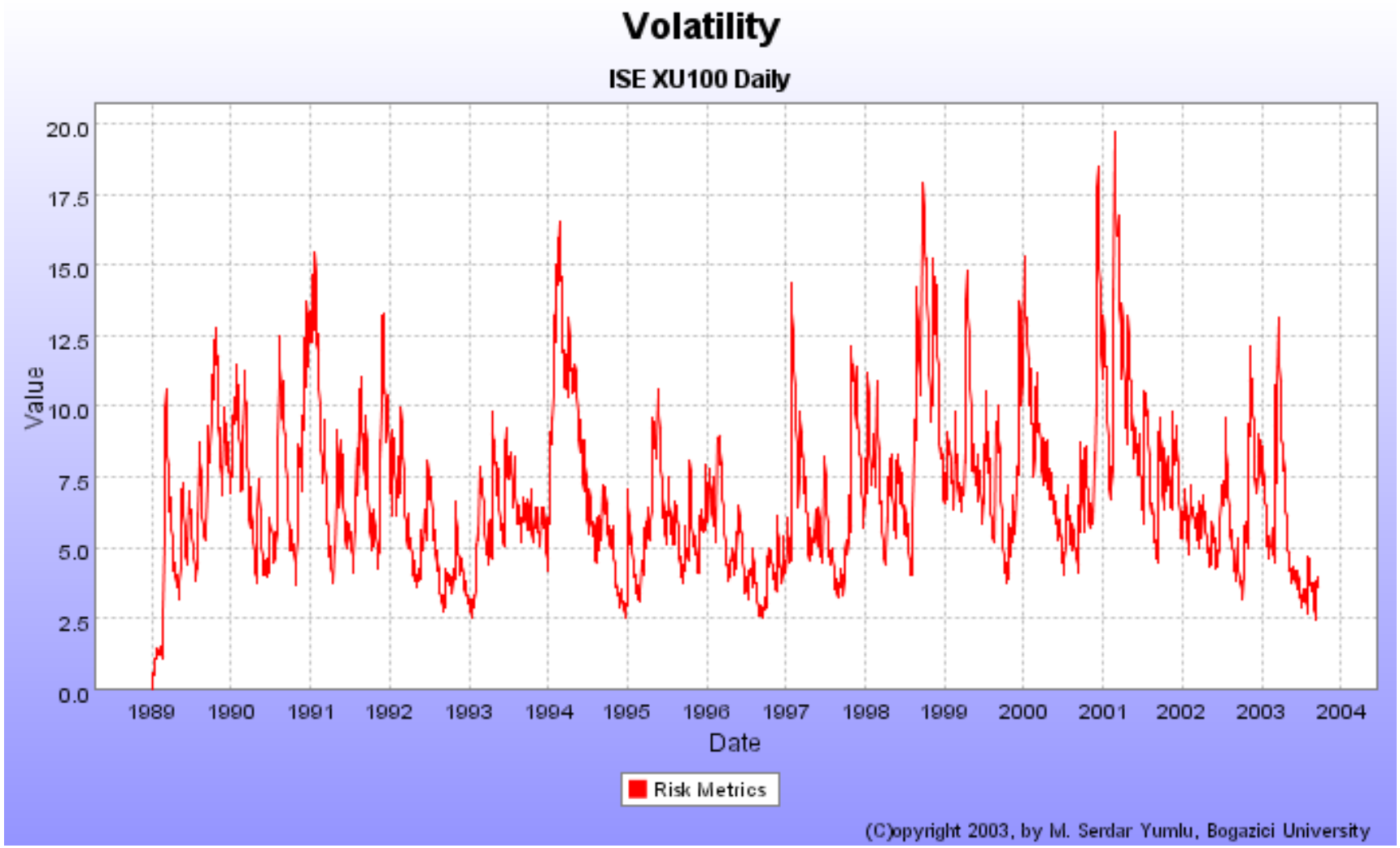


Figure 8.4. RiskMetrics™ volatility

$$h_t = 0.000051378 + 0.16301 * \varepsilon_{t-1}^2 + 0.79363 * h_{t-1} \quad (8.1)$$

The metrics calculated from GARCH is given in Table 8.5.

Table 8.5. Metrics for residual GARCH (1,1) volatility model

Residual	*mse*	*rmse*	*mape*	*corr*	*TIC*	U_m	U_s	U_c
GARCH	0.136	0.368	1.215	0.926	0.331	0.917	0.007	0.076
	$\boldsymbol{H_R}$	H_R^+	H_R^-	$\boldsymbol{H}\varepsilon$				
	0.889	1	0.886	33.45				

The results in Table 8.5 means that GARCH (1,1) is not a good model although it shows some power in handling the volatility if we only look at its hit rate or correlation performance. It is obvious that not all the metrics support this dissertation. In GARCH (1,1), we are trying to model the risk using only the previous volatility and the previous residual' s square linearly. Regression ability is not as suitable as a prediction method. Both *mse* and *mape* are very high, although correlation seems to be good as showing 92 percentage. *TIC* , a measure relative to the naïve predictor is 0.331, that is not close enough to zero and the bias component which is also expected to be close to zero is 0.517 that signals an occurrence of a systematic error. Directional performance, hit rate measures of 88 percent success has been achieved for volatility modeling.

Besides GARCH, we have also considered the asymmetric effect and applied EGARCH (1,1) of Nelson. The results are tabulated in Table 8.6 and formulated in (8.2).

Table 8.6. EGARCH (1,1) parameters for ISE National-100

ω	α	γ	β	Likelihood
-0.7387	0.3388	-0.0083	0.9320	6358.3

$$\log(h_t) = -0.7387 + 0.9320 * \log(h_{t-1}) + (-0.0083)\frac{\varepsilon_{t-1}}{\sqrt{h_{t-1}}} + 0.3388\left[\frac{|\varepsilon_{t-1}|}{\sqrt{h_{t-1}}} - \sqrt{\frac{2}{\pi}}\right] \quad (8.2)$$

The metrics calculated from EGARCH are given in Table 8.7.

Table 8.7. Metrics for residual EGARCH (1,1) volatility model

Residual	*mse*	*rmse*	*mape*	*corr*	*TIC*	U_m	U_s	U_c	$\boldsymbol{H_R}$	H_R^+	H_R^-	$\boldsymbol{H\varepsilon}$
EGARCH	0.189	0.434	1.048	0.887	0.374	0.907	0.017	0.076	0.876	1	0.874	51.49

The calculated metrics for residual EGARCH is expected to be powerful than GARCH because of the asymmetric effect that is handled in EGARCH, but the results do not seem like as we expect. Only *mape* and $\boldsymbol{H\varepsilon}$ shows better results than a traditional GARCH approach. The other measures are similar for both techniques; but as we have mentioned in GARCH results, for a good volatility model we have to see all the measures handling different aspects of prediction.

8.2.5. Case III – GARCH after ANN

Time series processing have been studied before using traditional GARCH style volatility models and ANNs also. But, there is no study that combines these two competing approaches in the way that we do. GARCH models have two phases. The first phase models the residuals and the latter models the volatility. We divided this approach into a combination of ANN and GARCH. We calculated the residuals using regression ability of an Elman style recurrent neural network and then apply GARCH volatility technique.

In the first part of this model we have applied Elman network for predicting the returns and calculated the residuals between the actual returns. Then, we have used residuals as an input to the GARCH model and calculated the volatility both using GARCH (1,1) and EGARCH (1,1). The results are compared to other cases at the end of this section. Elman network's properties are given Table 8.8.

Table 8.8. Elman network parameters

Hidden Units	Window size	Learning Rate	Momentum Rate	Epochs
20	8	0.9	0.4	30

Again in this case, we divided the data set into training and test sets. The parameters of GARCH (1,1) and EGARCH (1,1) are given in Table 8.9, Table 8.10, Table 8.11 and Table 8.12.

Table 8.9. GARCH (1,1) parameters for ANN predicted residuals

α_0	α_1	β_1	Likelihood
0.54556	0.16194	0.79205	9116.73

Table 8.10. Metrics for GARCH (1,1) applied after ANN

ANN	*mse*	*rmse*	*mape*	*corr*	*TIC*	U_m	U_s	U_c	$\boldsymbol{H_R}$	H_R^+	H_R^-	$\boldsymbol{H\varepsilon}$
GARCH	0.214	0.463	0.915	0.576	0.404	0.734	0.016	0.25	0.862	0.51	0.867	62.803

Table 8.11. EGARCH (1,1) parameters for ANN predicted residuals

ω	α	γ	β	Likelihood
-0.2504	0.3142	0.0528	0.9331	3501.8

Table 8.12. Metrics for EGARCH (1,1) applied after ANN

ANN	*mse*	*rmse*	*mape*	*corr*	*TIC*	U_m	U_s	U_c	$\boldsymbol{H_R}$	H_R^+	H_R^-	$\boldsymbol{H\varepsilon}$
EGARCH	0.37	0.608	0.713	0.603	0.471	0.863	0.045	0.092	0.863	0.842	0.863	143.785

By applying Elman RNN for mean modeling in GARCH and EGARCH, we expect the model to handle volatility in a better way. *mse*, *corr*, *TIC* and hit rate measures do not show predictive power and they are even worse than GARCH (1,1) modeling. Also if we consider ANN + EGARCH, we also have seen the same results as it was seen from ANN + GARCH.

8.2.6. Case IV – ANN Volatility Model

In this chapter, we are trying to predict the future volatility using ANNs. For ANN point of view we can accept the problem as a regression problem. You should have a comparable and common measure. It may be a traditional method such as GARCH or it may be a neural network, you need a true volatility measure for comparison. Also in neural

networks we need this measure to use as the target value. In the literature, there are several studies that accept the true volatility measure as the square of returns (r_t^2) [12, 46]. This is not true in real trading environments. Volatility is not simple and directly observable statistic and squared returns alone cannot represent the volatility. But in this study, we will accept the true volatility as RiskMetrics™ that is proposed by J.P.Morgan [18]. This is a more realistic measure of volatility and will help the neural models to handle the targeted model in a more accurate way. Also we have applied GARCH as a true volatility measure and applied ANNs to predict the future volatility. Both of these results are given together to prepare a comparable basis in the notation of the results.

For volatility estimation, first we have calculated the continuously compounded returns using (2.3), then true volatility measure is calculated using both RiskMetrics™ and GARCH separately. Each model is given with its model parameters and the results are gathered in the results table giving the details of each model against each metric.

8.2.6.1. Multilayer Perceptron (MLP). MLP regards the problem as it is a regression problem. It takes the sized window as an input and tries to predict the value at $t+1$. There is no special enhancement on this network. Learning occurs by the help of the backpropagation method. The best this network can do is what RiskMetrics™ does.

Table 8.13. MLP network parameters

Hidden Units	Window size	Learning Rate	Epochs
5	20	0.009	60

As a TDNN network, MLP regards the problem as a curve-fitting problem. But, time series data is not a simple curve-fitting problem. The capturing process is shown in Figure 8.5. If we accept the true volatility as RiskMetrics™ volatility, the basic regression statistics metrics *mse* and *mape* are very high and only 57 percent correlation is achieved. These results show that this model does not handle the underlying structure well enough from the point of regression. *TIC* is also high and occupies bias proportion instead of high covariance proportion. Hit rate shows a minimum of 57 percent for positive hit rate.

Table 8.14. Out-of-sample performance statistics for MLP network

	mse	*mape*	*corr*	*TIC*	U_m	U_s	U_c
RiskMetrics	0.1923	1.2952	0.573	0.3855	0.5922	0.0125	0.3952
MLP	(0.0581)	(0.1584)	(0.1463)	(0.0430)	(0.0643)	(0.0179)	(0.0482)
GARCH	0.0455	2.4888	0.8446	0.0156	0.0531	0.0081	0.941
MLP	(0.0151)	(3.1392)	(0.0507)	(0.0309)	(0.0345)	(0.0052)	(0.032)
	$\boldsymbol{H_R}$	H_R^+	H_R^-	$\boldsymbol{H\varepsilon}$	$\boldsymbol{P_{\Re}}$	$\boldsymbol{N_p}$	R^2
RiskMetrics	0.8728	0.5793	0.8892	16.9205	-	-	-
MLP	(0.0159)	(0.1350)	(0.0152)	(4.5112)			
GARCH	0.9388	0.6236	0.9738	10.6701	2.867	1040.626	0.6855
MLP	(0.03)	(0.1982)	(0.0087)	(2.2866)	(1.8159)	(659.2414)	(0.0847)

When GARCH volatility is the target measure, *mape* is very high which is significant for prediction. The covariance proportion in *TIC* represents the predictive content and also hit rate is high except positive hit rate. *corr* and R^2 are also acceptable although they are not superior.

The reason why MLP is not good in most of the experiments is its inability in describing and representing the complex relationship in volatility modeling well enough.

8.2.6.2. Radial Basis Functions (RBF). RBF does not handle the problem as MLP does. RBF first divides the space into parts and handles the problem trying to fit Gaussian model over these parts. This type of clustering will be also seen in Mixture of Experts (MoE) model.

RBF is first presented as a way of solution for curve-fitting problems [34]. RBF tries to fit Gaussian models over the region of the data of interest. This approach seems better than MLP when we are studying over RiskMetrics™ volatility. The results for RBF are summarized in Table 8.15. These results are gathered from the production data set like all the remaining reported results. *mse* is not below the psychological limit of 0.1 and only 75 percent correlation is achieved. Also *TIC* has variance proportion in it rather than MLP approach. Hit rate performance of 88 percent and 54 percent positive hit rate with a great *mape* error shows the inability easily and makes us move to more complex models. GARCH based volatility as we have seen in MLP again shows better results than RiskMetrics™. But this time *TIC* again rejects the model because of the bias and variance proportion in itself.

Table 8.15. Out-of-sample performance statistics for RBF network

	mse	*mape*	*corr*	*TIC*	U_m	U_s	U_c
RiskMetrics	0.1005	3.5112	0.7669	0.3176	0.007	0.4011	0.5919
RBF	(0.0286)	(4.9199)	(0.1052)	(0.0450)	(0.0052)	(0.1125)	(0.1128)
GARCH	0.0346	0.871	0.986	0.128	0.5326	0.1628	0.306
RBF	(0.0257)	(0.1920)	(0.0064)	(0.0640)	(0.1818)	(0.0060)	(0.1884)
	$\boldsymbol{H_R}$	H_R^+	H_R^-	$\boldsymbol{H\varepsilon}$	$\boldsymbol{P}_{\Re}$	$\boldsymbol{N_p}$	R^2
RiskMetrics	0.8804	0.544	0.9675	4.2817	-	-	0.5541
RBF	(0.0359)	(0.0879)	(0.0234)	(0.2067)			(0.1263)
GARCH	0.8946	0.5804	0.97636	3.943	-0.9706	-352.363	0.8806
RBF	(0.0476)	(0.125)	(0.0178)	(1.079)	(0.7676)	(278.73)	(0.0845)

8.2.6.3. Mixture of Experts (MoE). MoE is also a clustering technique that divides the problem into sub-problems and handles each part separately then combining the results in a gating network. MoE as stated in the neural networks section has two modules. One assigns the divided problems (spaces) to local experts. This assigning expert is known as the gating expert or gating network. Local experts specialize in the local spaces and forward the results to the gating expert, which then combines these outputs regarding the input of the network to each local expert. MoE as a combining technique is expected to perform better than former network models applied.

This new approach is both a clustering and a combination technique. It handles different regions of the data set in a different way. The results are reported in Table 8.16 and shown in Figure 8.6.

MoE, unlike MLP and RBF is presenting a promising approach with its two parted modules architecture. Regression statistics such as *mse*, *mape* and *corr* are very encouraging, hit rate performance is very good and *TIC* is close enough to zero, but again there are problems in distributing the properties in *TIC*. 40 percent bias proportion is very high and opens a question mark in minds for RiskMetrics™ targeted volatility. But GARCH volatility, this time is not as successful as the methods discussed. *TIC* is high and contains 53 percent bias proportion.

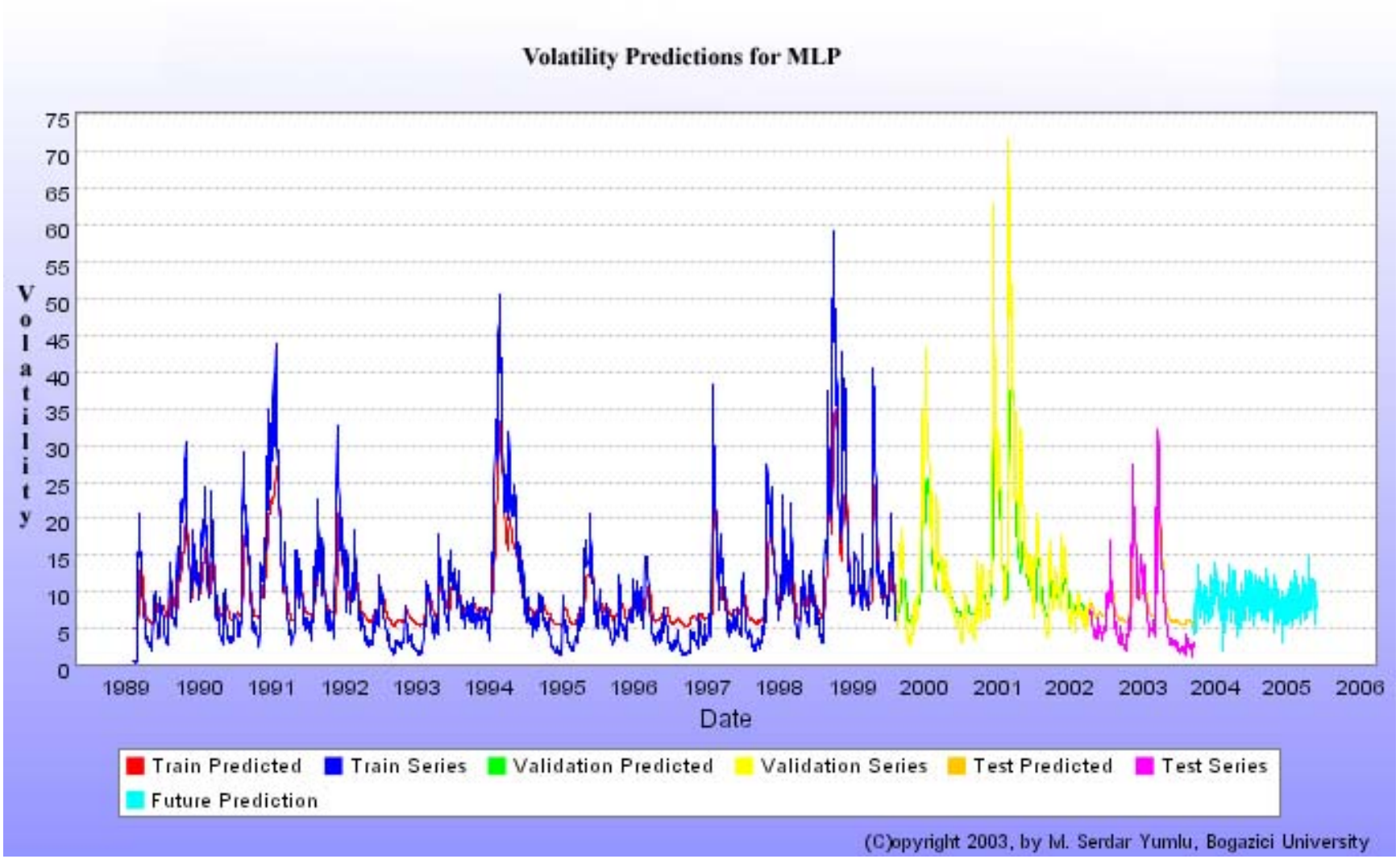


Figure 8.5. Volatility predictions for MLP network

Table 8.16. Out-of-sample performance statistics for MoE Network

	mse	*mape*	*corr*	*TIC*	U_m	U_s	U_c
RiskMetrics	0.0265	0.1071	0.9468	0.1064	0.4028	0.0218	0.5767
MoE	(0.0084)	(0.0273)	(0.0059)	(0.0142)	(0.0950)	(0.0098)	(0.1047)
GARCH	0.1291	1.136	0.7085	0.223	0.532	0.0506	0.419
MoE	(0.1434)	(1.3742)	(0.4678)	(0.1378)	(0.309)	(0.0284)	(0.312)
	$\boldsymbol{H_R}$	H_R^+	H_R^-	$\boldsymbol{H\varepsilon}$	$\boldsymbol{P_{\Re}}$	$\boldsymbol{N_p}$	R^2
RiskMetrics	0.9811	1	0.9792	15.598	-	-	0.7902
MoE	(0.0049)	(0.0036)	(0.0047)	(1.1869)			(0.0794)
GARCH	0.9275	0.9848	0.9243	18.2901	-0.0903	-32.8187	0.4856
MoE	(0.0250)	(0.0371)	(0.0236)	(9.1186)	(0.1511)	(54.9167)	(0.3753)

8.2.6.4. Recurrent Neural Networks (RNN). In this study we investigated the use of Elman, Jordan, a hybrid RNN and capacitive RNNs as recurrent neural networks. Elman network has its feedback from the hidden layers. The context layers then are put through the hidden layer with the actual network input. Jordan has its recurrence from the output layer to context layer. Hybrid RNN has both feedback connections from both the output and the hidden network layer and the capacitive one has a capacitive one-time memory for the context layer to store the previous context layer values and process them with the new comer recurrent connections. All the recurrent neural networks mentioned are capacitive unless it is specified. All the details of these networks and their learning mechanisms are explained in neural networks section in details.

8.2.6.5. Elman Recurrent Neural Network. Elman recurrent neural network is the most popular and applicable recurrent neural network proposed up to now. The recurrence is from the hidden layer to the context layer. This gives temporal processing power and a capability of handling the inner process while learning the patterns in the time series data. This inner predictive power has an important role on the performance of the network. The parameters applied to the Elman network are given in Table 8.17.

Table 8.17. Elman recurrent neural network parameters

Hidden Units	Window size	Learning Rate	Epochs
5	20	0.9	30

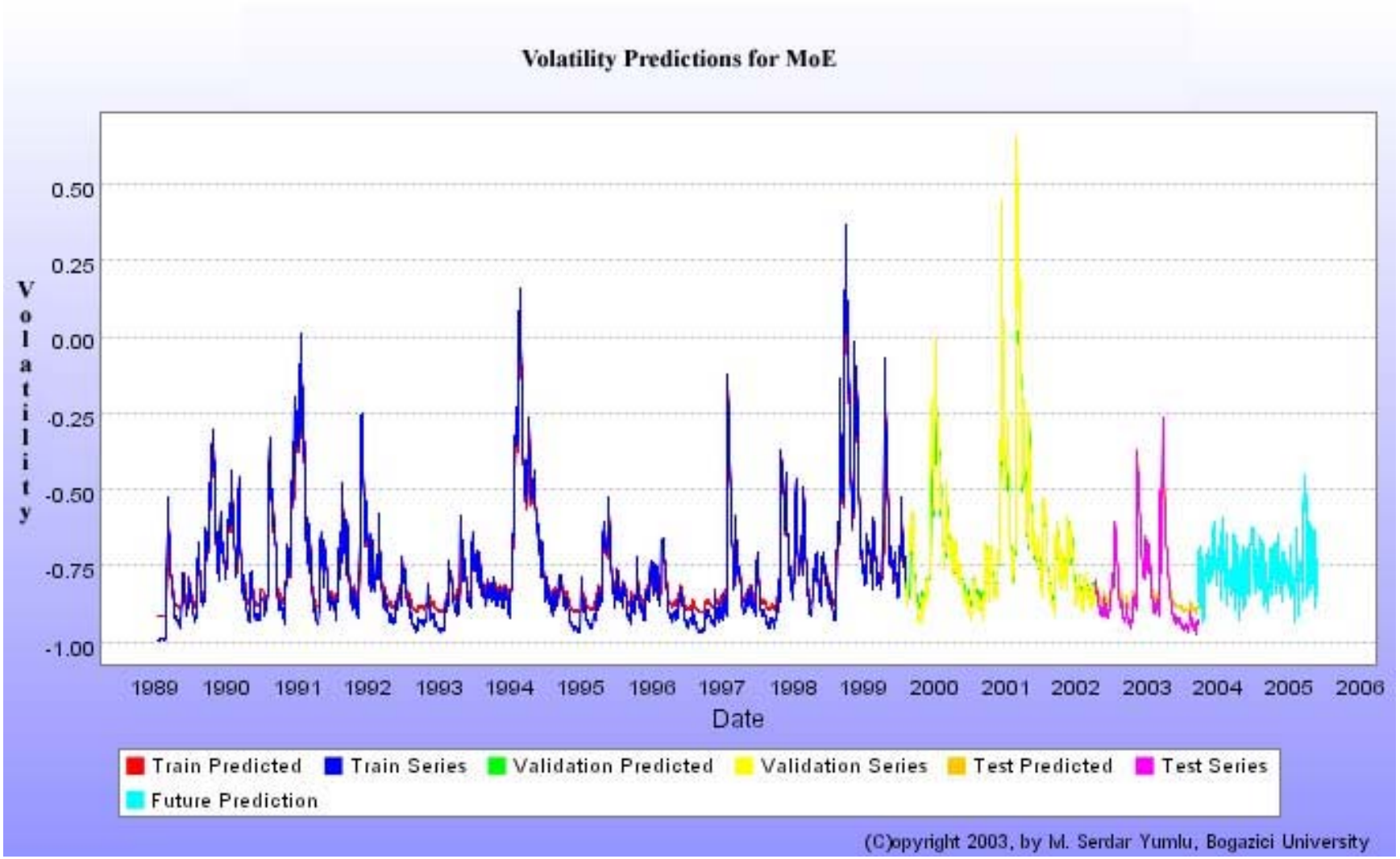


Figure 8.6. Volatility predictions for MoE

Future prediction gained by applying Elman RNN suggests that the stock market in the near future will not be very volatile, but not all of the future forecast may be accurate because we are having a forecast using another forecast as the lag gets larger. Forecast using forecast is achieved by applying sliding windows. The out-of-sample performances for both capacitive and non-capacitive Elman networks are given in Table 8.18 and Table 8.19.

Table 8.18. Out-of-sample performance statistics for capacitive Elman RNN

	mse	*mape*	*corr*	*TIC*	U_m	U_s	U_c
RiskMetrics	0.0245	0.4114	0.9283	0.1042	0.0895	0.0028	0.9102
Elman	(0.0084)	(0.3846)	(0.0130)	(0.0138)	(0.040)	(0.0033)	(0.0405)
GARCH	0.069	0.4314	0.8248	0.172	0.2784	0.1103	0.6128
Elman	(0.0457)	(0.1345)	(0.1354)	(0.0522)	(0.2128)	(0.0855)	(0.2873)
	$\boldsymbol{H_R}$	H_R^+	H_R^-	$\boldsymbol{H\varepsilon}$	$\boldsymbol{P}_{\Re}$	$\boldsymbol{N_p}$	R^2
RiskMetrics	0.9707	0.8802	0.9783	12.9943	5.4788	1950.832	0.8385
Elman	(0.0058)	(0.0295)	(0.0061)	(0.7701)	(1.010)	(447.0571)	(0.0423)
GARCH	0.9523	0.8834	0.963	31.699	3.3372	1211.423	0.548
Elman	(0.0180)	(0.1764)	(0.0227)	(24.2479)	(1.2251)	(444.761)	(0.2427)

Capacitive Elman RNN has feedback connections from the hidden layer to the context layer besides its capacitive layer which stores the previous values of the context layer. By applying Elman RNN, we expect the model to handle the recurrent relationship in volatility with past volatilities. The results reported in Table 8.18 have shown that, by involving feedback connections we are now more able to capture the time series. Both *mse* and *mape* are below the limit 0.1 that we defined. Both *mse* and *corr* shows how the model captured the series and how good learning is achieved. Inequality metric *TIC* represents the success against the naïve predictor, proving its ability with a high covariance proportion.

Sign prediction has also a very significant role in trading. When you are only able to predict the sign, although not enough, you will have more chance than others to win. Reaching a 97 percent success for volatility sign prediction brings superiority. Elman also shows power against $\varepsilon - Increase$ benchmark. GARCH volatility based Elman modeling has also shown the same success in RiskMetrics™. Both *mse* and *mape* are below the limits and correlation is high but not as high as reported before. The metric *TIC* in GARCH is not close enough to zero and approximately 30 percent of bias has

experimented; but this does not mean a lack in modeling. Also hit rate measures are very promising.

Table 8.19. Out-of-sample performance statistics for non-capacitive Elman RNN

	mse	*mape*	*corr*	*TIC*	U_m	U_s	U_c
Elman	0.0228 (0.0020)	0.2682 (0.0188)	0.9298 (0.0188)	0.102 (0.0038)	0.0888 (0.0472)	0.0014 (0.0020)	0.9124 (0.0487)
	$\boldsymbol{H_R}$	H_R^+	H_R^-	$\boldsymbol{H\varepsilon}$	$\boldsymbol{P}_{\Re}$	$\boldsymbol{N_p}$	R^2
Elman	0.9708 (0.0011)	0.8644 (0.0139)	0.9796 (0.0013)	13.023 (0.3436)	5.7254 (0.1010)	2066.783 (36.5059)	0.8434 (0.0177)

The results considered are calculated using capacitive Elman RNN. But does the capacity have a significant impact over the model performance? By examining the non-capacitive results in Table 8.19, we saw that the answer is no. Most of the metrics reported close results. Only *mape*, which is important for prediction is better than the capacitive network. We can easily report that the capacitive layer does not add more value to the model for modeling time series of ISE National-100.

Elman recurrent network's predictions are very promising and successful for three data sets that we use for cross-validation. The learning process is supported by the measures of test set that are reported in Table 8.18 and Table 8.19.

8.2.6.6. Jordan Recurrent Neural Network. Jordan style recurrent neural network gets its temporal ability from the recurrence of the outputs units. In this study, we are trying to predict the following value, so the recurrence will occur only on the predicted value. The results are close but better than an Elman recurrent network.

Table 8.20. Jordan recurrent neural network parameters

Hidden Units	Window size	Learning Rate	Epochs
7	20	0.09	30

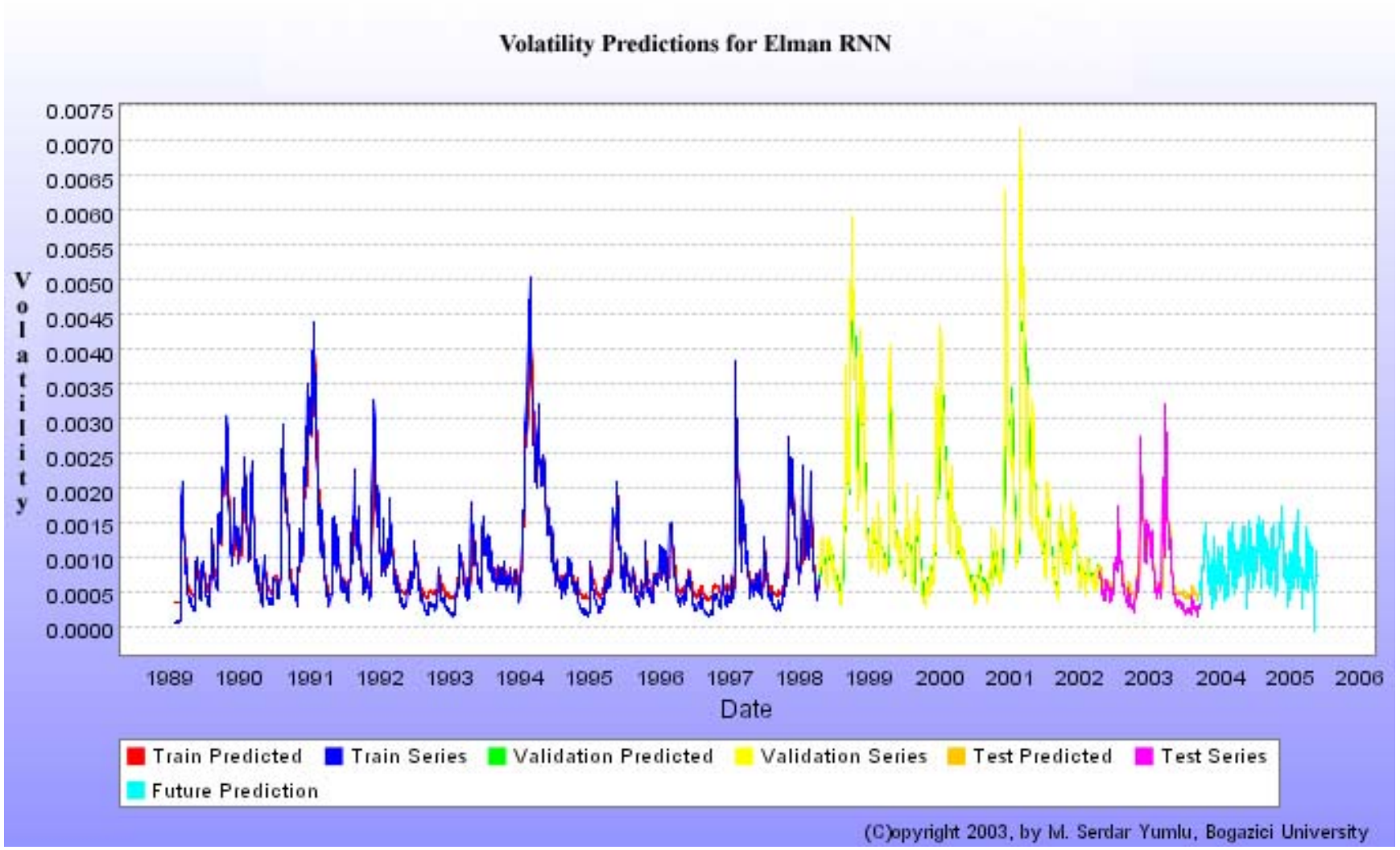


Figure 8.7. Volatility predictions of Elman RNN

Jordan network also handles the relationship in ISE National-100 series in a good way and suggests more volatility than Elman network for its future forecasts. This may mean an increase in the index price value that may be accepted as good news for investors.

Table 8.21. Metrics calculated in the test data set for Jordan RNN

	mse	*mape*	*corr*	*TIC*	U_m	U_s	U_c
RiskMetrics	0.0225	0.3795	0.9290	0.10183	0.0738	0.0018	0.9272
Jordan	(0.0008)	(0.2368)	(0.0022)	(0.0018)	(0.0213)	(0.0008)	(0.0212)
GARCH	0.0518	1.0788	0.900	0.1678	0.1966	0.0851	0.7201
Jordan	(0.0374)	(1.1918)	(0.0031)	(0.0606)	(0.2117)	(0.0413)	(0.2526)
	$\boldsymbol{H_R}$	H_R^+	H_R^-	$\boldsymbol{H\varepsilon}$	$\boldsymbol{P_{\Re}}$	$\boldsymbol{N_p}$	R^2
RiskMetrics	0.9705	0.8636	0.9795	13.0283	6.5345	2358.842	0.8457
Jordan	(0.0021)	(0.0217)	(0.0013)	(0.4166)	(0.9951)	(359.182)	(0.0069)
GARCH	0.9125	0.4792	0.986	6.9693	4.6471	2246.674	0.7123
Jordan	(0.0354)	(0.1140)	(0.0044)	(1.4562)	(0.3882)	(1510.242)	(0.1824)

Jordan RNN presents feedback connections from the output layer. We incorporate the previous calculated volatility again into the model at the next time step. In a first look, this model seems to handle better than the rest of the models investigated up to now. Jordan RNN has shown better results than Elman RNN although it does not increase the success much. Metrics like *mse*, *mape*, *corr* and *TIC* reports close but better results. This takes us to use a hybrid approach in RNN networks.

8.2.6.7. Hybrid Recurrent Neural Network. Hybrid RNN gets its name from its hybrid feedback connections both from the output and the hidden layer. This architecture is different from both Elman and Jordan neural networks. Again, for learning process RTRL has been chosen.

Table 8.22. Hybrid recurrent neural network parameters

Hidden Units	Window size	Learning Rate	Epochs
7	20	0.09	30

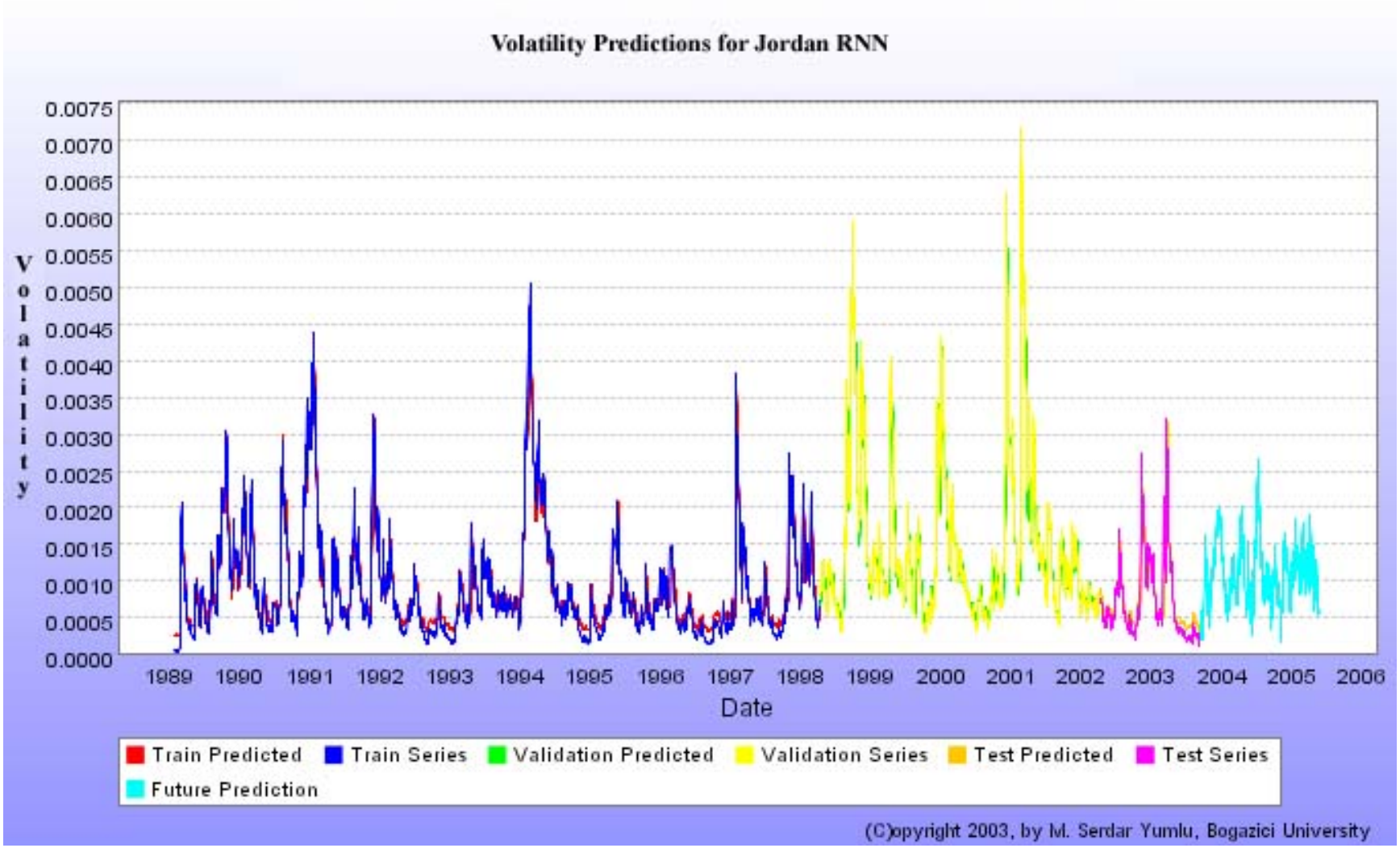


Figure 8.8. Volatility predictions for Jordan RNN network

Table 8.23. Metrics calculated in the test data set for hybrid RNN

	mse	*mape*	*corr*	*TIC*	U_m	U_s	U_c
RiskMetrics	0.0223	0.3713	0.9295	0.1011	0.0677	0.00014	0.9346
Hybrid	(0.0015)	(0.1493)	(0.0032)	(0.0025)	(0.0133)	(0.0004)	(0.0132)
GARCH	0.0266	0.284	0.9005	0.119	0.0173	0.0055	0.9798
Hybrid	(0.0010)	(0.0533)	(0.0012)	(0.002)	(0.0137)	(0.0034)	(0.0169)
	$\boldsymbol{H_R}$	H_R^+	H_R^-	$\boldsymbol{H\varepsilon}$	$\boldsymbol{P_{\Re}}$	$\boldsymbol{N_p}$	R^2
RiskMetrics	0.9701	0.8620	0.979	13.0835	6.1147	2207.3720	0.8495
Hybrid	(0.0017)	(0.0213)	(0.0)	(0.3613)	(0.5317)	(192.0196)	(0.0084)
GARCH	0.9758	0.8633	0.9845	13.2526	4.8968	1777.531	0.7926
Hybrid	(0.0025)	(0.0304)	(0.0017)	(0.5164)	(0.3420)	(124.135)	(0.0079)

Hybrid RNN approach integrates Elman and Jordan RNN and produces a two-way feedback mechanism. The results are reported in Table 8.23. This method as a combination of two successful predecessors, models the risk in time series in a more attractable way. By combining two techniques we were not able to increase the performance much but 0.0223 *mse* and 0.3713 *mape* errors with the significant correlation, we have seen this model as the best among others from the regression point of view. *TIC* and its covariance proportion proves the handling with hit rate performances. According to the results in Table 8.23, GARCH based volatility metrics are also better as the RiskMetrics™ volatility. The results are again outperforming other models and they are the best among its rivals. Figure 8.9 shows the future predictions for volatility in ISE National-100 index using Hybrid RNN approach.

All the networks are compared to each other by means of their performance metrics and their performance against the pre-defined benchmarks. Also, these results are compared to highly applied GARCH and EGARCH methods of volatility prediction. At the end, the volatility measure does not mean anything if you don't know where and how to use it. What we are trying to predict is actually the risk. The measure for risk management is VaR and we use the predicted future volatilities as a road to reach VaR and calculate our future risk.

All the results are collected in Table 8.24 for GARCH based volatility and Table 8.25 for RiskMetrics™ volatility.

Table 8.24. GARCH based neural networks out-of-sample results

	MLP	RBF	MoE	Elman	Jordan	Hybrid	GARCH	EGARCH
mse	0.0455	0.0346	0.1291	0.069	0.0518	0.0266	0.136	0.189
mape	2.4888	0.871	1.136	0.4314	1.0788	0.284	1.215	1.048
corr	0.8446	0.986	0.7085	0.8248	0.900	0.9005	0.926	0.887
R^2	0.6855	0.8806	0.4856	0.548	0.7123	0.7926	-	-
TIC	0.156	0.128	0.223	0.172	0.1678	0.119	0.331	0.374
U_m	0.0531	0.5326	0.532	0.2784	0.1966	0.0173	0.917	0.907
U_s	0.0081	0.1628	0.0506	0.1103	0.0851	0.0055	0.007	0.017
U_c	0.941	0.306	0.419	0.6128	0.7201	0.9798	0.076	0.076
$\boldsymbol{H_R}$	0.9388	0.8946	0.9275	0.9523	0.9125	0.9758	0.889	0.876
H_R^+	0.6236	0.5804	0.9848	0.8834	0.4792	0.8633	1	1
H_R^-	0.9738	0.97636	0.9243	0.963	0.986	0.9845	0.886	0.886
$\boldsymbol{H}\varepsilon$	10.670	3.943	18.2901	31.699	6.9693	13.2526	33.45	33.45
$\boldsymbol{P}_{\Re}$	2.867	-0.9706	-0.0903	3.3372	4.6471	4.8968	-	-
$\boldsymbol{N_p}$	1040.6	-352.36	-32.818	1211.42	2246.674	1777.531	-	-

Regression metric *mse* checks whether predicted series is close to the target or not. Hybrid RNN has overwhelmed its rivals in this metric because of its regression abilities as a combination of two other RNN techniques. MoE is not successful enough in *mse*. Prediction error *mape* shows fluctuated results when the complexity in the model increases but again hybrid RNN has shown its superiority also in this metric. RBF shows correlated results in GARCH based ANN volatility estimation. *TIC* inequality and its proportions highlight the valuable approach of hybrid RNNs. The correct number of the sign of the predictions is very high for almost all volatility targeted ANN approaches; but if the target is taken as directly the price or continuously compounded return series, these sign metrics does not present more than 60 percent success. All the methods rejects $\varepsilon - Increase$, increasing benchmark for time series.

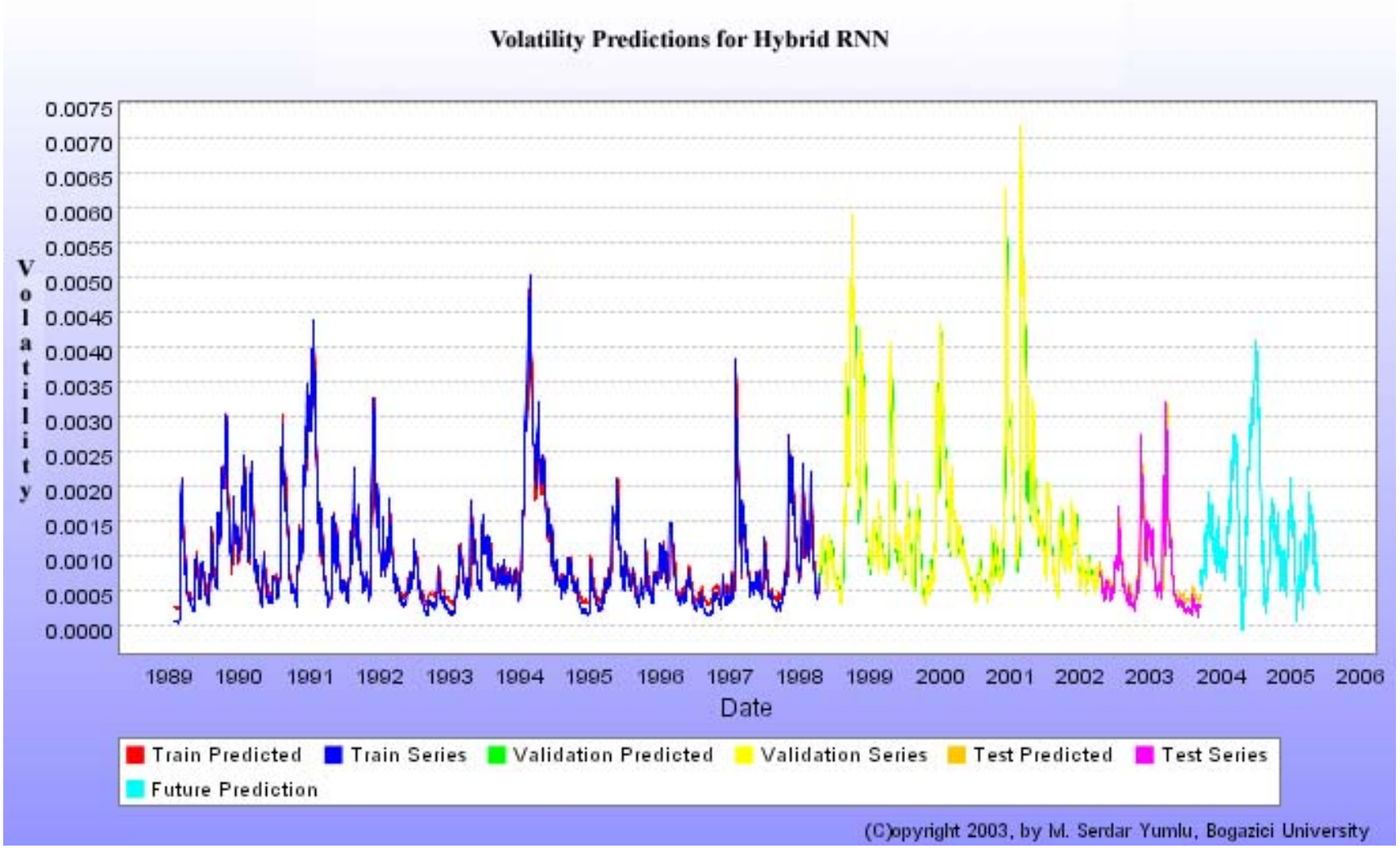


Figure 8.9. Volatility predictions for hybrid RNN networks

Table 8.25. RiskMetrics based neural network out-of-sample results

	MLP	RBF	MoE	Elman	Jordan	Hybrid	NN+GARCH	NN+EGARCH
mse	0.1923	0.1005	0.0265	0.0245	0.0225	0.0223	0.214	0.37
mape	1.2952	3.5112	0.2182	0.4114	0.3795	0.3713	0.915	0.713
corr	0.573	0.7669	0.9468	0.9283	0.9290	0.9295	0.576	0.603
R^2	-	0.5541	0.7902	0.8385	0.8457	0.8495	-	-
TIC	0.3855	0.3176	0.1064	0.1042	0.10183	0.1011	0.404	0.471
U_m	0.5922	0.007	0.4028	0.0895	0.0738	0.0677	0.734	0.863
U_s	0.0127	0.4011	0.0218	0.0028	0.0018	0.00014	0.016	0.045
U_c	0.3952	0.5919	0.5767	0.9102	0.9272	0.9346	0.25	0.092
$\boldsymbol{H_R}$	0.8728	0.8803	0.9811	0.9707	0.9705	0.9701	0.862	0.863
H_R^+	0.5793	0.544	1	0.8802	0.8636	0.8620	0.51	0.842
H_R^-	0.8892	0.9675	0.9792	0.9783	0.9795	0.979	0.867	0.863
$\boldsymbol{H\varepsilon}$	16.9205	4.2817	15.5988	12.9943	13.0283	13.0835	62.803	143.785
$\boldsymbol{P}_{\Re}$	-	-	-	5.4788	6.5345	6.1147	-	-
$\boldsymbol{N_p}$	-	-	-	1950.832	2358.842	2207.3720	-	-

When we are considering *mse* performance for both studies, we easily extract a 0.1-point limit and when you analyze the results MLP, RBF, NN+GARCH and NN+EGARCH results are above this limit and these techniques cannot be accepted as good enough. Again in *mape* MLP and RBF is over 1, so far from their rivals. MoE and RNN networks have shown close and successful results both in *mse* and *mape*. Again hybrid RNN has outperformed other techniques same as in *corr* and R^2 except the MoE peak in correlation. *TIC* should be close to zero for rejecting the naïve predictor approach and the proportions in *TIC* supports this idea. Bias and variance existence brings systematic errors for this metric. High results of MLP and RBF are experimented in *TIC*. On the other hand, MoE and hybrid RNN outperformed their rivals. Although hit rate and related sign metrics are very high for most of the techniques, we can easily separate out MoE and hybrid RNN among others.

MoE and hybrid RNN approaches are two most successful and suitable methods with Elman and Jordan RNN for time series prediction of financial data sets.

8.2.7. Case V – Data Set is Split into Years

Up to now, we have studied ISE National-100 index time series using the whole data set. We divided the dataset into three parts including train, validation and test sets. We have trained our neural networks in train data set, validate the network parameters on the validation data set and at the end gathered the out-of-sample results according to the performance metrics that we have defined in Chapter 7 in the test (production) data set. The results are very encoring for the use of ANNs in the field of stock market volatility prediction problems.

We then analyze the performance of Elman RNN by dividing the data set into years and analyze each year as a separate data set. At the end we have reported the mean and standard deviations for our metrics in Table 27. As we have mentioned before in the data section, the whole data set starts at 3 January 1989 and ends at 18 September 2003. We divided the data set into years starting from January 1989 to January 1990 and etc. Each set now has approximately 250 observations and the effects of the global economy have been indulged into only that year of concern. For example, we do not consider the effect of the crisis in 1994 for the data in 1995. The results have shown that Elman recurrent neural network have the prediction power for understanding the future derivations of volatility even though we split the whole dataset into one year periods. The network parameters for this RNN are given in Table 8.26.

Table 8.26. Elman network parameters for year-by-year prediction

Hidden Units	Window size	Learning Rate	Epochs
15	5	0.9	50

Although the average performance of analyzing the data set year by year is also acceptable, some of the periods are not successful enough that also decreases the whole performance. Analyzing these data sets in that year of perspective did not give good results only in periods of 1989-1990, 1997-1998 and after 2003. Correlation is not good; *TIC* coefficient is higher than other year's *TIC* values. In Table 8.27 the first year period from 1989 up to 1990 is not stated. Not all of the periods are as successful as using the whole data set. From *mse*, *mape* and *corr* perspective 1992-1993 and 1998-1999 periods

are the most performing periods. By avoiding the effects of more than one-year events we now easily see how the model handles each year only regarding the events on that year. This degrades the average performance. *TIC* which tests the performance against the naïve predictors is not close to zero in all of the periods and the covariance proportion in *TIC* is high in average surprisingly. Because *TIC* and its proportions are moving together. Hit rate performances are over 80 percent but not as high as previous studies. Hit rate for $\varepsilon - Increase$ is also not as high as expected in average when we study the data set splitting it into years. This brings the effect of the far past behaviors in the time series of ISE National-100.

We also studied the data set by dividing them into two-year and three-year periods. Studying over a two-year period of data gives some power for handling the data because of the number of observations included in the training phase. Having split the data into years and two-years periods bring us to see how a past year affects the performance of the model. In most of the metrics, the increase in the size of the data set powers up the performance in the training phase. Although *mse* is over the limit that we have defined, it geared up from the one-year experience. *mape* has a 15 percent increase in performance and *corr* closes up to 80 percent of success. The problems in periods 1989-1990 and 1997-1998 have been solved because we use one more year of data. The *TIC* measure is now close to 0.3 and covariance proportion is as high as in the one-year study. Approximately 90 percent hit rate, high net profit and mean profit per trade takes us to try the experiments using more data in the training and validation phase. In Table 8.28 and Table 8.29, the performance of the network over predicting the volatility using two year and three-year periods are analyzed respectively.

The performance of the model at the end between January 2003 and September 2003 was not good enough. This problem also has been solved in the analysis of three-year data periods. Considering the performance upgrade from one-year period to three-year period data sets, we are now able to see how the data is affected from past years. *mse* metric is as we have observed from previous studies except the period of 1995 and 1998 than can also be seen from other metrics. These years are the years that the effects of the 1994 economical crisis are handled. Economical improvement studies made the market a more stable and predictable market after these fluctuated years. The most successful period when

considering *rmse* is 1992 and 1995 period. When *mape* prediction error is considered, the limit of 1 is reached and successful periods have been observed. *corr* and R^2 have are supporting the improvement in the model performance using three-year data set. *TIC* is now more close to zero although it is expected to be much more close and now the proportions in *TIC* are more distributed, showing a 20 percent bias and 10 percent variance proportion signaling an occurrence of error. Hit rate performances also now move to 90 percent stages and increasing benchmark is now rejected in a more stable way.

All these studies have shown that using daily observations we need as much as data that we can use. When the train data set involves only one-year period of data, it is not possible to understand the underlying complex relationship even though we use ANNs. The probability of extracting the structure of the time series increased when we have applied more items using two and three-year periods. But, using daily observations we are not able to seize the market in a session. Being able to know how the market will behave in a single session or in a trading day is gaining much more importance than monthly, weekly or even daily forecasts. Because most of the active market players are seeking for seconds of order times and they are now trying to make profit using their intraday movements. In today's environment, not only the big boys but also every investor has real-time visualization of the market. They can see any movement, any order that are passed in seconds to the ISE's ordering system. When you have such technology, you are more enabled to real playing and beating the market. For this purpose, we have left the sixth case study for intraday forecast ability of ISE.

Table 8.27. Out-of-sample results regarding each year of concern using Elman network

	90-91	**91-92**	**92-93**	**93-94**	**94-95**	**95-96**	**96-97**	**97-98**	**98-99**	**99-00**	**00-01**	**01-02**	**02-03**	**03-**	μ	σ
mse	0.137	0.098	0.073	0.177	0.123	0.164	0.158	0.411	0.075	0.184	0.374	0.104	0.299	0.583	0.218	0.1476
mape	1.136	1.553	0.883	1.99	0.745	0.656	1.048	2.468	0.564	1.163	1.031	1.343	1.244	4.225	0.4451	0.1473
corr	0.874	0.868	0.868	0.766	0.804	0.735	0.755	0.473	0.965	0.795	0.627	0.869	0.633	0.348	0.723	0.1783
TIC	0.261	0.252	0.206	0.325	0.318	0.285	0.338	0.51	0.195	0.313	0.415	0.263	0.453	0.563	0.347	0.1159
U_m	0.099	0.101	0.297	0.001	0.114	0	0.07	0.331	0.71	0.017	0.03	0.078	0.318	0	0.1448	0.1952
U_s	0.001	0.002	0.007	0.001	0	0.068	0.004	0.089	0.002	0.01	0.012	0.028	0	0.001	0.016	0.0268
U_c	0.9	0.897	0.696	0.998	0.886	0.932	0.926	0.58	0.288	0.973	0.958	0.894	0.682	0.999	0.8387	0.1973
$\boldsymbol{H_R}$	0.8	0.88	0.899	0.88	0.92	0.92	0.84	0.72	0.92	0.75	0.8	0.84	0.68	0.7	0.8186	0.0853
H_R^+	0.846	0.7	0.786	0.882	1	1	0.857	1	1	0.8	0.8	1	1	0.667	0.8744	0.1187
H_R^-	0.769	1	0.921	0.889	0.889	0.909	0.842	0.708	0.905	0.733	0.812	0.733	0.619	0.8	0.8167	0.1029
$\boldsymbol{H\varepsilon}$	1.6	2.347	5.93	1.32	3.154	7.36	3.36	17.28	5.52	1.917	2.133	2.016	4.08	1.26	4.104	4.0755
$\boldsymbol{P}_{\text{Я}}$	9.387	1.097	5.785	6.065	0.29	4.163	7.157	-2.278	-3.85	10.53	14.68	3.732	4.651	-0.975	4.2058	5.0204
$\boldsymbol{N_p}$	215.9	25.24	1024	139.5	6.7	95.7	164.6	-52.4	-88.5	231.7	337.7	85.8	106.9	-7.80	156.57	265.22

Table 8.28. Out-of-sample results for two-year data periods

	89-91	**91-93**	**93-95**	**95-97**	**97-98**	**99-01**	**01-03**	**03-**	μ	σ
mse	0.13	0.112	0.054	0.083	0.079	0.169	0.091	0.573	0.161375	0.170011
rmse	0.36	0.335	0.233	0.288	0.281	0.412	0.302	0.757	0.371	0.165135
mape	1.837	1.074	1.128	0.397	0.643	0.644	2.611	2.592	1.36575	0.878296
corr	0.831	0.837	0.86	0.84	0.887	0.786	0.896	0.325	0.78275	0.188132
R^2	0.6	0.603	0.524	0.628	0.78	0.577	0.603	-0.491	0.478	0.398339
TIC	0.256	0.287	0.214	0.234	0.23	0.244	0.243	0.574	0.28525	0.11861
U_m	0.061	0.115	0.285	0.144	0.007	0.021	0.405	0	0.12975	0.14577
U_s	0.02	0.011	0.042	0	0.008	0.003	0.022	0.007	0.01412	0.013601
U_c	0.939	0.893	0.681	0.873	1.006	0.998	0.585	1.118	0.88662	0.175832
$\boldsymbol{H_R}$	0.94	0.84	0.911	0.92	0.96	0.875	0.88	0.7	0.87825	0.081558
H_R^+	1	0.867	1	0.875	0.947	0.727	1	0.667	0.885375	0.128968
H_R^-	0.921	0.833	0.904	0.93	0.969	0.921	0.857	0.8	0.891875	0.056529
$\boldsymbol{H\varepsilon}$	3.838	2.94	11.584	6.44	2.613	4.112	5.39	1.26	4.772125	3.189875
$\boldsymbol{P_{\Re}}$	10.19	0.585	4.599	6.535	7.573	5.841	5.044	-0.975	4.924	3.623915
$\boldsymbol{N_p}$	489.113	28.09	404.72	313.68	363.53	268.67	242.131	-7.804	262.768	174.383

Table 8.29. Out-of-sample results for three-year data periods

	89-92	**92-95**	**95-98**	**98-01**	**01-**	μ	σ
mse	0.065	0.029	0.106	0.05	0.09	0.068	0.030749
rmse	0.256	0.169	0.325	0.223	0.3	0.2546	0.061938
mape	0.511	1.327	1.63	0.559	1.192	1.0438	0.491096
corr	0.923	0.954	0.891	0.911	0.946	0.925	0.025681
R^2	0.7	0.909	0.577	0.821	0.404	0.6822	0.199676
TIC	0.179	0.131	0.26	0.131	0.187	0.1776	0.05297
U_m	0.238	0.005	0.357	0.012	0.434	0.2092	0.196083
U_s	0.143	0.068	0.051	0	0.311	0.1146	0.121171
U_c	0.629	0.94	0.6	1.001	0.265	0.687	0.296489
$\boldsymbol{H_R}$	0.907	0.88	0.893	0.945	0.902	0.9054	0.024399
H_R^+	1	0.8	1	0.818	1	0.9236	0.104808
H_R^-	0.894	0.911	0.857	0.968	0.895	0.905	0.040404
$\boldsymbol{H\varepsilon}$	7.455	3.427	3.479	6.805	13.525	6.9382	4.122055
$\boldsymbol{P_{\Re}}$	5.557	-1.399	5.182	8.916	-1.327	3.3858	4.572427
$\boldsymbol{N_p}$	405.68	-102.141	378.304	633.029	-78.321	247.3102	323.7172

8.2.8. Case VI - Forecasting Intraday Volatility

Turkish stock market is a rapidly changing and highly volatile market. Investors usually try to make money in short time periods in a market. Prediction methods such as the ones we study and propose in this study works on the prediction of short time periods. Future prediction period could be a month; a week or it could be measured in seconds. Usually financial time series prediction over stock markets is applied to daily time series of indexes, mostly concerned on Dow Jones index. Being able to predict the following trading day's volatility is a very important aspect; but using high frequency data in a prediction problem may lead people to invest intraday having a predictive power in hand and also it may provide more and applicable data for daily time series prediction. Even seconds are very important in today's trading. Real-time trading is not a utopia nowadays. Like most of the businesses, trading and traders are also following the technology closely. ISE provided an Ex-API for most of the securities companies to make them send their orders more properly and in seconds. Traders did not skip this idea and they are now more powerful in giving orders and in following them. Even they have seconds based changing portfolio watcher programs. For this reason, we have adapted our studies for high frequency data approaches.

GARCH based intraday predictability has been studied by Rahman and Ang [47]. Kryzanowski and Liu also have studied the short-run predictability of price changes using technical trading rules [48]. In this thesis study, we approach the short-run predictability problem using intraday data of ISE National-30 index that includes the top 30 equities in the Turkish stock market. Turkish stock market has two sessions that both continue two and a half hours starting at 9.30 am and 14.00 pm respectively.

Table 8.30 gives the summary statistics of ISE National-30. Q (5), and Q2 (5) are the Ljung-Box statistics of the autocorrelation of residuals and squared residuals respectively; J-B is the Jarque-Bera statistic for normality test [45]. μ and σ are mean and standard deviations respectively. High frequency data is captured in normal stock market conditions for Turkey. This can be also extracted from statistical features of this series. Minimum and maximum are so close to zero, kurtosis is very high and skewness is not negative.

We have chosen Elman RNN for intraday prediction of ISE National-30 time series by using a window length of five. The results of the neural prediction approach for high frequency data is given in Figure 8.10 and Table 8.31.

Table 8.30. Summary statistics for ISE National-30 intraday data

ISE National-30	
Start Date	23.07.2003 09:30
End Date	29.07.2003 16:30
Observations	424
μ	0.0000034582
σ	0.0014
Skewness	0.6653
Kurtosis	10.3080
Minimum	-0.0060
Maximum	0.0078
Q(5)	16.3282 (0.0060)
Q(10)	32.8492 (0.0003)
$Q^2(5)$	6.5792 (0.2539)
$Q^2(10)$	10.2060 (0.4226)
ARCH(5)	6.3413 (0.7858)
Jarque-Bera (p-value)	962.0724 (0)

In Figure 8.10, the intraday prediction of ISE National-30 is shown. The series started from the beginning of 23 July 2003 1. Session and ends at the end of the 2. Session at 29 July 2003. The market is forecasted to go down for a period, have a peak and then again go down for the test data set in the figure. This has occurred as expected from the high frequency training data set.

Table 8.31. Intraday prediction results for ISE National-30

	mse	*mape*	*corr*	*TIC*	U_m	U_s	U_c
ISE National-30	0.1184 (0.0174)	1.4938 (0.9203)	0.8386 (0.0157)	0.2758 (0.0228)	0.0394 (0.0399)	0.0058 (0.0046)	0.982 (0.0431)
	$\boldsymbol{H_R}$	H_R^+	H_R^-	$\boldsymbol{H\varepsilon}$	$\boldsymbol{P_{\Re}}$	$\boldsymbol{N_p}$	R^2
ISE National-30	0.8704 (0.0483)	0.752 (0.0910)	0.9568 (0.0034)	2.3146 (0.4279)	43.47 (2.6345)	1521.449 (92.2117)	0.664 (0.0492)

The results are encouraging; but not as good as we have experimented from the daily observation studies when we look at the performance metrics. *mse* is above the 0.1 limit that we have defined for daily observations, *mape* is high but not so much. Correlation and R^2 are performing good enough and hit rate performances are not good as previous studies. *TIC*, Naïve predictor metric is 0.2758 and contains covariance proportion of 98 percent.

8.2.9. Individual Stocks

After studying over the index series of we have studied three individual stock series using hybrid RNN networks. We have seen the similarity between these series and the ISE National-100 index series not only in their distributions and their statistics and also in the results that we have achieved. Is Bankasi is one of the largest banks in Turkey and ISCTR is the main equity of Is Bankasi. Because of this ISCTR forms a great and information part for the index. TUPRS is also another building block of the ISE National-100 index. NETAS is also very important and valuable company with its equity in Turkey. We have seen that not only on index series, the uses of ANNs are also very promising in individual stock series.

In these three stocks the metrics showed similar results except the extraordinary *mape*. The results are summarized in Table 8.32. *mse* is as good as the measured index metrics. ISCTR is left behind amongst TUPRS and NETAS. TUPRS's *mape* is 2.019 and is highly over other two stocks and the index results. Correlation of 94 percent is very high and the TIC metrics are very close to each other and they are supported with their covariance proportions. Mean and variance proportions are close to zero as they should be. In the entire three stocks hit rate measures are high as expected, however TUPRS is left behind in these metrics of hit rate. Net profit and mean profit per trade is also very promising.

Table 8.32. Prediction results for ISCTR, TUPRS and NETAS

	mse	*mape*	*corr*	*TIC*	U_m	U_s	U_c
ISCTR	0.065	0.4356	0.9426	0.0826	0.0086	0.0216	0.973
TUPRS	0.022	2.019	0.9455	0.117	0.0005	0.039	0.964
NETAS	0.019	0.2895	0.9435	0.1075	0.0285	0.0065	0.969
	$\boldsymbol{H_R}$	H_R^+	H_R^-	$\boldsymbol{H\varepsilon}$	$\boldsymbol{P_{\Re}}$	$\boldsymbol{N_p}$	R^2
ISCTR	0.979	0.8546	0.9876	16.0793	5.9603	1871.475	0.887
TUPRS	0.943	0.798	0.967	6.794	5.174	1517.903	0.894
NETAS	0.976	0.9145	0.984	8.9045	7.565	1845.76	0.885

Although this study is mostly concerned on the risk estimation of ISE National-100 using ANNs, we have seen that ANNs can be useful in risk estimation of individual stocks.

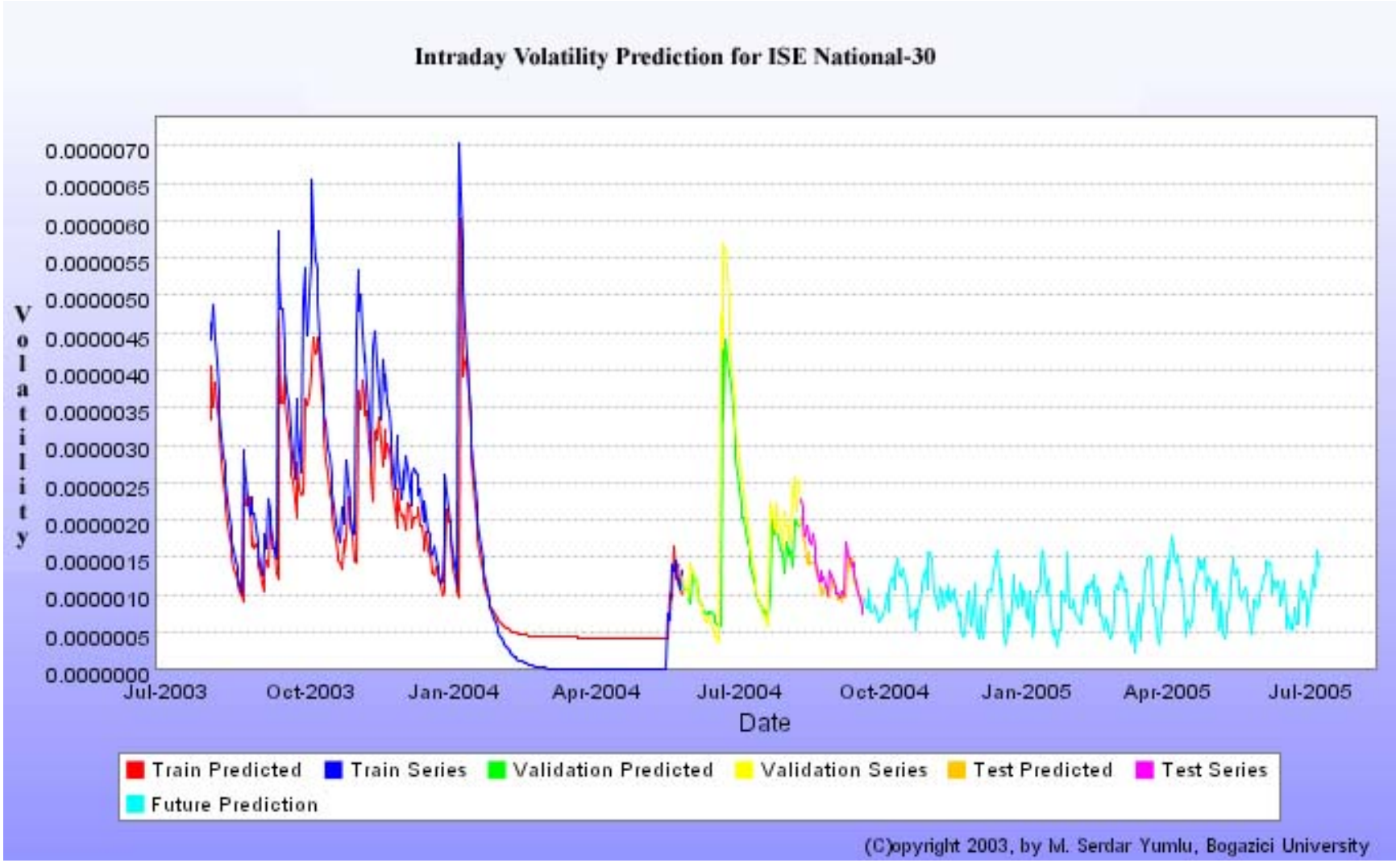


Figure 8.10. Future predictions for ISE National-30

9. CONCLUSIONS AND FUTURE WORK

In this thesis study we investigated the use of ANNs in risk estimation of asset returns. We try to estimate volatility. A financial time series prediction without considering volatility will not be a complete predicting. The risk in the stock market is its variance and the variance in a stock market is not constant over time. Financial time series exhibit time dependent heteroskedastic variance known as the conditional variance (volatility): an indirectly observable feature of the stock market. The models proposed in this study are trying to model this conditional variance. Because the issue is about risk, a risk manager or a portfolio manager or every participant in the stock market has the right to know his or her possible future loss. When you are able to model the volatility, you may predict the future volatility, which takes us to estimate our future risk profile. This is what we are trying to answer by investigating the use of ANNs in risk estimation. We have used ANNs as volatility models against the traditional GARCH models.

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.

We have studied volatility models including RiskMetrics™ and GARCH models. Four other case studies have been studied besides RiskMetrics™ and traditional GARCH. Conditional volatility process has two important sections in traditional approaches. First the mean modeling is applied and then the volatility is modeled afterwards. As far as we know, we are the first in distributing these issues in ANNs and GARCH at the same time. We predicted the residuals from ANNs and then applied GARCH in one of our case studies; but this approach did not perform as well as the traditional GARCH approach. We have used ANNs as volatility models. For our inputs to the neural networks we have used

RiskMetrics™ and GARCH based volatility due to the unobservability of volatility. After applying MLP, RBF, MoE and RNN to this problem we have seen that MoE and hybrid RNN architectures have significant generalization power over other models. Mixture of Experts (MoE) and hybrid recurrent neural networks (RNN) architectures were capable of finding a significantly better representation for the ISE National-100 index time series, than both traditional approaches of GARCH and EGARCH and simpler neural network architectures such as MLP and RBF. The experiments over daily observations suggest the use of MoE and/or hybrid RNN for the prediction of ISE National-100 index. We tested this generalization ability and compared our models by defining several performance metrics. The most basic and highly applied metric is mean square error (mse). We have also tested mean absolute prediction error ($mape$), TIC against the naïve predictor and its proportions, correlation and R^2 for testing the regression ability and hit rate metrics for the ratio of the correct sign predictions and also mean profit per trade and net profit. During the study we have extracted limits for such as mse, TIC and hit rate from the experiments. In all of the models we seek for a consortium of all the accepting metrics. Only in MoE and hybrid RNN have achieved this consortium. We have the network learning the system parameters in the training data set, validate these parameters in the validation data set and then calculate the test results in the production data set. We have also studied the data in a year-by-year basis. We have applied the volatility model individually in each one-year, two-year and three-year period and have seen the dependence in the volatility throughout the years. When the period is extended to two or three years, also the performance and the generalization ability are increased with the number of affecting items in the train and test set series. The poor generalization in some periods of the one-year study is solved in two-year an three-year periods.

Besides daily observations, we have also studied high-frequency intraday data of ISE National-30 index by following the trend in stock markets nowadays. High frequency studies are the futures of time series prediction problems because in today's markets, participants are seeking for seconds based ordering systems. So the market moves in seconds and the future risk may also change in seconds. For this purpose, we have also studied with intraday data using Elman RNN. By applying RNN architecture we tried to capture the model dependencies. Although the results are not as successful as the hybrid

RNN using daily observations, the results are encouraging for future studies of ANNs in the field of high frequency financial time series prediction.

As a consequence, the experiments suggest the use of MoE and hybrid RNN architectures for the prediction of future of the ISE market. Volatility rather than price and returns have more concern on prediction problems. The possibility of a future loss and the calculations of that loss is the important concept in stock market forecasts. We have shown the successful uses of ANNs in this volatility estimation problem in this thesis study.

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 or hybrid RNN may fail. GARCH is very important and prepares a core framework for volatility modeling from the economics perspective. The optimization process in GARCH may be projected into a neural model generating a Neuro-GARCH architecture. Gathering high frequency data is a very important part of intraday studies. Having a 5-minute data, we can have the prediction power for the future periods in a session in real-time. Future studies should cover studies of highly private intraday data including the number of people trading at the same time, the quantities of their orders, prices and their returns in a session. Generalized future directions in three categories are of equal importance and are the following:

- High frequency Volatility Estimation
- A Neuro-GARCH architecture
- A Real-time Risk Profile Software

REFERENCES

1. 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.

2. Yumlu, M. S., F. S. Gurgen and N. Okay, "Turkish Stock Market Analysis Using Mixture of Experts", *to appear in Proceedings of Engineering of Intelligent Systems (EIS)*, Madeira, March 2004.

3. Werbos, P. J., "Generalization of Backpropagation with Application to Recurrent Gas Market Model", *Neural Networks*, Vol. 1, pp. 339-356, 1988.

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

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

6. Hellstrom, T. and K. Holmstrom, "Predicting the Stock Market", *Opuscula ISRN HEV-BIB-OP-26-SE*, 1998.

7. Murphy, J. J., *Technical Analysis of Futures Markets*, New York Institute of Finance, 1986.

8. Davidson, C., *Development in FX Markets [Online]*, Olsen and Associates: Professional Library, http://www.olsen.ch/library/prof/dev_fx.html, June 1995.

9. Freisleben, B., "Stock Market Prediction with Backpropagation Networks", *Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*

5th International Conference IEA/AEI-92, Paderborn Germany, pp. 18-20, April 1991.

10. Mitchell, M. T., *Machine Learning*, The McGraw-Hill Companies, New York, 1997.

11. Bollerslev, T., "Generalized Autoregressive Conditional Heteroskedasticity", *Journal of Econometrics*, Vol. 31, pp. 307-327, 1986.

12. Engle, R., "Autoregressive Conditional Heteroskedasticity with Estimates of the Variance of United Kingdom Inflation", *Econometrica*, Vol. 50, pp. 987-1007, 1982.

13. Black, F. and M. Scholes, "The Pricing of Options and Corporate Liabilities", *Journal of Political Economy*, Vol. 81, May-June 1973.

14. Nelson, D., "Conditional Heteroskedasticity in Asset Returns: A New Approach", *Econometrica*, Vol. 59, pp. 349-370, 1991.

15. Okay, N., "Asymmetric Volatility Dynamics: Evidence from the Istanbul Stock Exchange", *Business & Economics for the 21th Century*, Vol. 2, B&ESI, pp. 207-216, 1998.

16. Okay, N., "Evidence for Conditional Heteroskedasticity in Turkish Stock Returns", *Readings Book of the 1998 ABA National Conference*, pp. 241-247, 1998.

17. Okay, N., "Dynamic Predictability of the Istanbul Stock Exchange Market Using a Vector Autoregressive Model", *Boğaziçi Journal*, Vol. 12, pp. 31-56.

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

19. Dowd, K., *Beyond Value at Risk: The New Science of Risk Management*, John Wiley, London, 1998.

20. Brooks, C., "Linear and Non-linear (Non)-Forecastibility of High-Frequency Exchange Rates", *Journal of Forecasting*, Vol. 16, pp. 125-145, 1997.

21. Brooks, C. and G. Persand, "Value at Risk and Market Crashes", *Journal of Risk*, Vol. 2(4), pp. 5-26, 2000a.

22. Brooks, C. and G. Persand, "Lies, Damned Lies and Value at Risk Estimates", *Journal of Risk*, Vol. 13(5), pp. 63-66, May 2000b.

23. Black, F., "Studies of Stock Price Volatility Changes", *Proceedings of the 1976 Meetings of the Business and Economic Statistics Section*, American Statistical Association, pp. 177-181, 1976.

24. French, K., G. Schwert and R.Stambaugh, "Expected Stock Returns and Volatility", *Journal of Financial Economics*, Vol. 19, pp. 3-30, 1987.

25. Pagan, A. R. and G. W. Schwert, "Alternative Models for Conditional Stock Volatilies", *Journal of Econometrics*, Vol. 45, pp. 267-290, 1990.

26. Engle, R. F. and V. K. Ng, "Measuring and Testing the Impact of News on Volatility", *The Journal of Finance*, Vol. 48, Issue 5, pp. 1749-1778, December 1993.

27. Gurney, K., *Computers and Symbols versus Nets and Neurons*, Dept. Human Sciences, Brunel University, Uxbridge, Middx.

28. Anderson, D. and G. McNeill, "Artificial Neural Networks Technology", *A DACS State of the Art Report*, Rome Laboratory, 1992.

29. Rumelhart, D. E, G. E. Hinton and R. J. Williams, "Learning Internal Representations by Error Propagation", in D. E. Rumelhart and J. L. McClelland (Eds.), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, pp. 318-362, MA: MIT Press, Cambridge, 1986.

30. Werbos, P. J., *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*, Ph.D. Thesis, Harvard University, 1974.

31. Parker, D., "Learning Logic: Casting the Cortex of the Human Brain in Silicon", *Technical Report TR-47*, Center for Computational Research in Economics and Management, MIT, Cambridge, MA, 1985.

32. Bishop, M. C., *Neural Networks for Pattern Recognition*, Oxford University Press, New York, 1995.

33. White, H., "Parametric Statistical Estimation With Artificial Neural Networks", *Technical Report*, University of California, San Diego, Discussion Paper, pp. 92-113, March 1992.

34. Powell, M. J. D., "Radial Basis Functions for Multivariable Interpolation: A Review." in J. C. Mason and M.G. Cox (Eds.), *Algorithms for Approximation*, pp. 143-167, Oxford: Clarendon Press, 1987.

35. Girosi, F. and T. Poggio, "Regularization Algorithms for Learning that are Equivalent to Multilayer Networks", *Science*, Vol. 247, pp. 978-982, 1990.

36. Weigend, A. S., B. A. Huberman and D. E. Rumelhart, "Predicting the Future: A Connectionist Approach", *International Journal of Neural Systems*, Vol. 1(3), pp. 193-209, 1990.

37. Mozer, M. C., "Neural Net Architectures for Temporal Sequence Processing", in A. S. Weigend and N. A. Gershenfeld, editors, *Time Series Prediction: Forecasting the Future and Understanding the Past*, pp. 243-264, 1993.

38. Elman, J. L., "Finding Structure in Time", *Cognitive Science*, Vol. 14, pp. 179-211, 1990.

39. Jordan, M. I., "Attractor Dynamics and Parallelism in a Connectionist Sequential Machine", *Proceedings of Ninth Annual Conference of the Cognitive Science Society*, pp. 531-546, Lawrence Erlbaum Associates, 1986.

40. Williams, R. J. and D. Zipser, "Gradient-Based Learning Algorithms for Recurrent Networks and Their Computational Complexity", in Chauvin and D. E. Rumelhart (Eds.), *Back-propagation: Theory, Architectures and Applications*, Hillsdale, N.J.: Erlbaum, pp. 433-486, 1995.

41. Jacobs, R. A., M. I. Jordan, S. J. Nowlan and G. E. Hinton, "Adaptive Mixture of Local Experts", *Neural Computation*, Vol. 3, pp. 79-87, 1991.

42. Jordan, M. I, and R. A. Jacobs, "Hierarchical Mixtures of Experts and the EM Algorithm", *Neural Computation*, Vol. 6, pp. 181-214, 1994.

43. Weigend, A. S, M. Mangeas, and A. N. Srivastava, "Nonlinear Gated Experts for Time Series: Discovering Regimes and Avoiding Overfitting", *International Journal of Neural Systems*, Vol. 6, pp. 373-399, 1995.

44. Theil, H., *Applied Economic Forecasting*, North-Holland Publishing Company, Amsterdam, 1966.

45. Bera, A. K. and M. L. Higgins, "Arch Models. Properties, Estimation and Testing", *Journal of Economic Surveys*, Vol. 7, pp. 305-362, 1993.

46. Schittenkopf, C. and G. Dorffner, "Risk Neutral Density Extraction from Option Prices: Improved Pricing with Mixture Density Networks", *University of Vienna*, 1999.

47. Rahman, S. and K.P. Ang, "Intraday Return Volatility Process: Evidence from NASDAQ Stocks", *Review of Quantitative Finance and Accounting*, Vol. 19, No. 2, 2002.

48. Kryzanowski, L. and F. Liu, “Intraday Predictability of Market Microstructure Statistics and Technical Trading Rules”, *Center for Financial Services*, pp. 97-104, 1997.