View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Bilkent University Institutional Repository

PREDICTION OF İSTANBUL SECURITIES EXCHANGE COMPOSITE INDEX

A THESIS
SUBMITTED TO THE DEPARTMENT OF MANAGEMENT
AND THE GRADUATE SCHOOL OF BUSINESS
ADMINISTRATION OF BILKENT UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF BUSINESS ADMINISTRATION

HG
5706.5
.I88
T56
1993

By
Murat TİMUR
September 1993

PREDICTION OF İSTANBUL SECURITIES EXCHANGE COMPOSITE INDEX

A THESIS

SUBMITTED TO THE DEPARTMENT OF MANAGEMENT

AND THE GRADUATE SCHOOL OF BUSINESS ADMINISTRATION

OF BİLKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF BUSINESS ADMINISTRATION

Murat TİMUR
tarafından bağışlanmıştır

By

Murat TİMUR

September 1993

HG
5706.5
.I 88
T56
1983

B022919

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Business Administration.

[illegible]

Assist.Prof.. Gülnur Muradoğlu (Advisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Business Administration.

[illegible]

Assist.Prof. Can Muğan Şımga

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Business Administration.

[illegible]

Assist.Prof. Serpil Sayın

Approved for the Graduate School of Business Administration

[illegible]

Prof. Sübidey Togan

ABSTRACT

PREDICTION OF İSTANBUL SECURITIES EXCHANGE COMPOSITE INDEX

Murat Timur
Master of Business Administration
Supervisor: Assist.Prof. Gülnur Muradoğlu
September, 1993

This study presents a software developed by using Nested Generalized Exemplars, for predicting Istanbul Securities Exchange Composite Index. Information reflected in the past values of frequently used monetary variables are used to predict stock returns.

Daily returns of the composite index are predicted by using: Central Bank effective selling price of US Dollar and Deutsche Mark, İstanbul Tahtakale closing selling price of Turkish Republic gold coin and one ounce of gold, Commercial Banks (İş Bank, Akbank, Yapı Kredi Bank, and Ziraat Bank) 3-month average deposit rate and 3-month Government bond interest rates. Data prior to the dates on which the predictions are made are used to learn the forecasting power of variables on composite index and to generate the appropriate rules. The results reveal that the information reflected in the past prices of the variables have significant effects on the ISE composite index.

Keywords: Istanbul Securities Exchange (ISE), Nested Generalized Exemplars (NGE).

ÖZET

İSTANBUL MENKUL KIYMETLER BORSASI BİLEŞİK ENDEKSİNİN BELİRLENMESİ

Murat Timur
Yüksek Lisans Tezi, İşletme Enstitüsü
Tez Yöneticisi : Asist.Prof. Gülnur Muradoğlu
Eylül, 1993

Bu çalışma, İstanbul Menkul Kıymetler Borsası Bileşik Endeksini belirlemek için geliştirilen ve İçiçe Genellenen Örnekler metodunu kullanan bir yazılımı sunmaktadır. Sıkça kullanılan parasal değişkenlerin eski değerlerinin yansıttığı bilgi, hisse senetlerinin getirilerinin belirlenmesinde kullanılmıştır.

Bileşik Endeksin günlük getirisini belirlemek için Amerikan Doları ve Alman Markının Merkez Bankası efektif satış değerleri, Türkiye Cumhuriyet Altını ve bir ons altının İstanbul Tahtakale kapanış satış değeri, Ticari Bankaların (İş Bankası, Akbank, Yapı Kredi Bankası, ve T.C. Ziraat Bankası) 3 aylık ortalama faiz oranı ve 3 aylık Devlet Bonosu faiz oranları kullanılmıştır. Değişkenlerin, bileşik endeks üzerindeki tahmin gücünü öğrenmek ve uygun kuralları çıkartmak için, belirlemenin yapıldığı tarihten önceki veriler kullanılmıştır. Çıkan sonuçlar da göstermiştirki değişkenlerin eski değerlerinin yansıttığı bilgi, İstanbul Menkul Kıymetler Borsası bileşik endeksi üzerinde önemli etkilere sahiptir.

Anahtar Sözcükler : İstanbul Menkul Kıymetler Borsası, İçiçe Genellenen Örnekler.

ACKNOWLEDGMENTS

I would like to express my deep gratitude to my supervisor Assist. Prof. Gülnur Muradoğlu for her guidance, suggestions, and invaluable encouragement throughout the development of this thesis. (N.B. I gratefully acknowledge the fundamental contributions of Prof. Steven L. Salzberg to NGE without which this software would not be possible.)

I would also like to thank to Assist. Prof. Can Muğan Şımga and Assist. Prof. Serpil Sayın for reading and commenting on the thesis and I owe special thanks to Assoc. Prof. Kürşat Aydoğan for providing a pleasant environment for study.

I am grateful to the members of my family; Muzaffer, Meliha and Dilek Timur for their infinite moral support and patience that they shown not only during the completion of the thesis but also throughout my education in Bilkent University.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER I

INTRODUCTION

The high sensitivity of the capital markets to political and economical events have influenced the efficiency and trend of the markets over the last decade and brought about sensational variations of the indices as well as sudden dramatic variations of trend, thus frustrating the efforts of some authoritative followers of one method of prediction or another. For this purpose a vast amount of research has been conducted on market efficiency tests in capital markets. However these remarks should not lead the investor to an attitude of discouragement and helplessness towards the market, but only of caution, since we believe it is possible, with the help of adequate decision instruments, to reduce the risks involved to minimum [Pasquale 1991].

Systems for inducing concept descriptions from examples have been one of the valuable decision instruments for assisting in the task of knowledge acquisition for expert systems. In this study, we used the theory of learning from examples called Nested Generalized Exemplars (NGE) theory [Salzberg 1990] and applied this neural network and fuzzy computing technology to evolve the software for predicting Istanbul Securities Exchange Composite Index.

Although there has been considerable amount of studies about the presence of weak and semi-strong form efficiency at İstanbul Securities Exchange, there has been no published research about the prediction of ISE composite index using this kind of a methodology with the historical information reflected in the past prices.

The aim of this study is to predict the ISE composite index by using the information reflected in the past prices of some of the commonly used monetary variables: Central Bank effective selling price of US Dollar and DM, İstanbul Tahtakale closing selling price of Turkish Republic gold coin and one ounce of gold, Commercial Banks 3 month average deposit rate and 3 month Government bond interest rate.

In Chapter 2, a review of literature about market efficiency, empirical studies about the efficiency in ISE, and a summary of research on the predictive ability of the monetary variables in Turkish Stock Market is presented.

In Chapter 3, the data used in this study are explained. Then we introduce the theory of learning from examples called Nested Generalized Exemplar and demonstrate its importance with empirical results in several domains. Background about the method, comparison with other exemplar-based theories and econometric models, and types of generalization are explained. The steps of the Nested Generalized Exemplar Algorithm: the initialization, fetching the examples, and the matching process of the algorithm is explained in detail with pseudo codes.

In Chapter 4, modifications made to the original algorithm, its comparison with the simple model, and the findings of our study from the application of the algorithm to the Turkish Stock Market is presented.

Concluding remarks and the results of the model used is discussed and avenues for further research are presented in Chapter 5. The source code of the program is presented in Appendix A , setting file in Appendix B and the input data is available in Appendix C.

CHAPTER II

LITERATURE SURVEY

Market Efficiency and the Previous Empirical Work

The early research of market efficiency hypothesis discussed by [Fama 1970] in his well-known article, pointed out that, a market, which security prices fully reflect all available information is called efficient. Fama, in the 1970 review, divided work on market efficiency into three categories: the 'weak form', the 'semi-strong form' and the 'strong form'.

In a weakly efficient market, present prices reflect all information contained in the record of past prices, that is, investors cannot consistently earn abnormal returns by observing the past prices. In a semi-strongly efficient market, present prices reflect all available information, that is, security prices adjust rapidly and correctly to the announcement of all publicly available information. In a strongly efficient market, present prices reflect all information both privately held and insider information together with publicly available information.

The early research [Fama&Blume 1966] proved that developed markets are efficient in the weak and semi-strong sense [Fama 1965] but there is not a consensus in the strong form of market efficiency hypothesis of [Jensen 1968] and [Sharpe 1966]. Efficiency literature measures the efficiency of security markets by testing the predictability of returns by using certain information sets e.g., past prices, publicly available information, monopolistic information. The recent studies [Zarovin 1990], uses historical price series in order to test the weak form of efficient market hypothesis. Some of them used publicly available information and announcements in order to test the semi-strong form of market efficiency e.g.

[Falk&Levy 1989]. [Chan&Chen 1991] used the behaviors of investors who have private and insider information in their strong form efficiency tests.

Although research on market efficiency is assumed to be the most successful in empirical economics, with good prospects to remain so in the feature [Fama 1991], information revealed by macroeconomic variables has been ignored in the efficient market literature with few exceptions. However during the last decade, researchers have been interested in macroeconomic variables such as inflation, interest rates, foreign currency, and other term structure (monetary and fiscal) variables [Fama 1991]. In brief, the new work says that returns are predictable from past returns.

Macroeconomic variables are important information sets for developing countries, where application of financial instruments and institutions are very new and the operations in the market is not deep [Moore 1980]. These variables are proved to be efficient indicators of the market since the investors are not charged for extra costs in collecting these information sets [Mishkin 1982].

Studies [Darrat 1988] on Canadian Stock Market, verifies that a significant lagged relationship between fiscal variables and stock prices is observed. Similarly, Hancock [1989] found that the semi-strong form of efficient market hypothesis in US stock market is valid for both monetary and fiscal variables. Studies of Bulmash and Trivoli [1991] investigated the relationship between stock prices and the national economic activity as measured by a series of key economic variables like money supply, 3-month maturity Treasury bill average yield, 10-years maturity Treasury Bond, Composite Stock Market average, Industrial production rate, unemployment rate. The results of the statistical analysis of the hypothesized relationships confirm that there are time lags that transpire between economic variables and stock returns. They provide a new perspective on several studies of [Fama&French 1988], [Summers 1986], and [Poterba&Summers 1988] that stock prices contain a predictable component and can be predicted by various lagged economic factors.

[Pearce&Roley 1985] examined the daily returns of stock prices to changes in the money supply, inflation rate corresponding to percentage changes in the Consumer and

Producer Price Index (CPI & PPI), industrial production, unemployment rate, and the discount rate in order to test the efficient market hypothesis that only the unexpected part of a change in those variables effects the stock returns.

Structure of the Turkish Stock Market

The role of financial liberalization policies implemented in Turkey affected the monetary and capital market variables and these variables are being considered as the key elements of this transition process [Gültekin&Sak 1990].

The financial liberalization process in Turkey began in the early 1980s. The concept can be loosely defined as limiting the involvement of government in the financial markets and giving way to market forces [Sak&Yeldan 1993]. Prior to that time, the Turkish financial system was relatively unsophisticated; commercial banks were the sole suppliers of funds to users, and loans and deposits, the only available financial instruments. Interest rates were tightly regulated and limited the scope of bilateral negotiations. Capital movements were strictly controlled. Banks were basically acting as branches of the government for collecting and distributing surplus funds in the economy. The absence of an Interbank market to aggregate and transmit the expectations of economic agents was an additional distorting factor [Atiyas&Ersel 1992].

As for the securities markets, there was literally no activity on the ISE. Although corporations began to issue bonds in the latter half of the 1970s, there was no orderly secondary bond market. As for government bonds, some of them were not actually issued but written on the accounts as bonds when the government had to turn to the banking system in times of liquidity needs. Finally, there was a group of unorganized securities brokers who traded corporate and government bonds at huge discounts on the face value of the security [Akyüz 1988].

The reform process started with the deregulation of interest rates. Following this first step, foreign exchange regulations were liberalized to ease capital movements. Money and capital markets were introduced into the economy with the objective of creating collective markets. With deregulation, banks became relatively free to determine their deposit and loan rates. The deregulation of interest rates came together with measures on the foreign exchange regulations. Turkish citizens were left free to hold foreign exchange and open foreign exchange accounts with the domestic banks. The ultimate objective was full convertibility of the Turkish currency [Sak&Yeldan 1993].

The Interbank money market is established to reflect short term scarcity of resources which is highly volatile. For long term economic decisions, secondary capital markets, especially government and corporate bond markets, became very important variables which produce information reflecting the general expectations of the longer term trend of the general economy [Sak&Yeldan 1993].

The ISE was reorganized and began its operations in 1985. The objective was to collect all secondary market activities under the auspices of the exchange. Mutual funds were defined and established with objective of creating the institutional demand for the markets. Foreign participation in Turkish securities markets was made possible and easy with investment through collective investment funds. In 1989, foreign portfolio investments on ISE became possible. Although foreign participation gave a boost to the stock market and acted as a catalyst for domestic participation in the market, in a thin market like Turkey, it also increased the volatility of the market. It made the market very vulnerable to the impact of foreign exchange rates, especially US Dollar and DM became determinants of the market.

The allocation of the private portfolio in Turkey is interest sensitive [Ersel&Sak 1985]. The real rate of interest plays a crucial role in the portfolio decisions between the domestic and foreign currency substitution. Gold being a non-interest bearing asset, is in competition with fixed interest securities as an investment medium. So, an inverse relationship between the gold price, stock returns and the level of interest rates should be expected [Gültekin 1986]. Besides interest rates tend to rise during inflationary times and, therefore, gold, as an inflation hedge, and interest rates would move in the same direction.

However, there are some pitfalls; the number of full-proof economic indicators in Turkey are very few. To develop a forecasting method, recognized human experts are needed. But professionals in the industry that we consulted tend to agree that there is no acknowledged investment expertise and there is the persistent problem of finding ways of focusing on only the relevant facts in a large amount of data and of keeping track of the justifications for beliefs.

Empirical Tests of the Predictive Ability of the Monetary Variables in Turkey

Studies made in this context [Şengül&Önkal 1992], tests the semi-strong form of efficient market hypothesis by using certain monetary and fiscal variables as the set of publicly available information in Turkey. These variables are the monthly growth rate in total amount of money: TL in circulation plus the sight and time deposits, monthly inflation rate, change in the government budget deficit which is used as the percentage change in the short term loans of Treasury to the Central Bank of Turkey, monthly industrial growth rate as an indicator of GNP, monthly unemployment rate, monthly industry growth rate, Interbank monthly effective interest rate, and money supply. In order to differentiate the expected values and unexpected changes of the fiscal and monetary variables they used two step approach of Barro (1977-78). Their test results verify that the market is inefficient: a significant lagged relationship between fiscal and monetary policy and stock returns is observed.

Another study [Erol&Aydoğan 1992] tested Arbitrage Pricing Theory Ross [1976] as an asset pricing model to see if asset returns in the context of the Turkish stock market can be explained by a model which has been tested extensively in well-established capital markets. The pricing effects of macro-economic variables including the industrial production, inflation, real interest rate, risk premium are tested in the Turkish Stock Market. Pure judgment is used in choosing the variables and in identifying the economic factors that may explain the co-movement of stock returns e.g. [Huberman&Kandel 1985]. It is found that the portfolio returns seem to be determined to a great extent by the proposed state variables but it is seen that only the unanticipated part of those state variables: inflation, real rate and risk premium have significant effects on the return structure of Turkish Stock Market although it is a new and emerging stock market of a developing country.

Sönmez and Berik [1993] used the models developed by [Chen&Roll&Ross 1986] and [Fama&Macbeth 1973], to test the significance of monetary variables in explaining their return behavior on ISE composite index. Briefly, this method regresses the returns on the economic variables of over an initial estimation period. The resulting coefficients (Beta's) are then used as independent variables in monthly cross-sectional regressions over a later hold-out period. This generates a time series of estimates of risk premium associated with each economic variable. The time-series means of estimates are then tested by a t-test for statistical significance. The sample range of the study contains 632 examples starting from the period of January 1st, 1991 to April 4th, 1993. They used Tahtakale closing selling price of Turkish Republic gold coin and one ounce of gold, Central Bank effective selling price of US Dollars and DM, a basket which contains 50% of US Dollars and 50% of DM, 12,9,6 and 3 months of government bond interest rates, average interest rate on deposits, and overnight Interbank interest rate as the monetary variables to set up the general factor regression equation. The results of their tests proves that the Turkish stock market is inefficient in the sense that information reflected in the past prices of those variables have significant effects on ISE composite index. The results indicate that only the Commercial Banks average 3-months interest rate on deposits and overnight Interbank interest rate do not have significant effects on the composite index and should not be included in such a model.

In this study, we used pure judgment of experts (portfolio managers, dealers and brokers) and the results of the previous studies in choosing the variables and identifying the economic factors that may explain the co-movement of stock returns [Huberman&Kandel 1985]. The selected subset of these commonly used monetary variables are: Central Bank effective selling price of US Dollar and DM, İstanbul Tahtakale closing selling price of Turkish republic gold coin and one ounce of gold, Commercial Banks 3 month average deposit rate, and 3 month Government Bond interest rate.

The purpose of this study is to use these variables for predicting the ISE composite index by using Nested Generalized Exemplars which is a model different from previously employed econometric models.

CHAPTER III

DATA AND METHODOLOGY

3.1. Data

The data used in this study consists of 18 months (January 1st, 1991 to June 30th, 1992) of daily data. For each variable, percentage change in the value of the variable from the previous day value is calculated for 395 business days e.g. [Pearce&Roley 1985], i.e., 395 examples are used as an input for the algorithm.

First, a literature survey is made in order to identify monetary variables that effect the Turkish stock market and then expert opinions were taken (İş bank, Ankara head office portfolio managers, Hitit Menkul Kiymetler brokers) to identify the variables which reflect the fiscal and monetary changes, to be used at the design stage of the prediction. By taking the study of [Erol&Aydoğan 1992] and [Huberman&Kandel 1985] as a reference, we considered the pure judgment of experts in choosing the variables and in identifying the economic factors that may explain the movement of stock returns.

The following variables were selected as the commonly used monetary variables to predict stock returns: Central Bank effective selling price of US Dollar and DM, İstanbul Tahtakale closing selling price of Turkish Republic gold coin and one ounce of gold, Commercial Banks 3 month average deposit rate (İş bank, Akbank, Yapı Kredi Bank, and Ziraat Bank 3 months average deposit rate is used because these banks have the highest volume of time and sight deposits and other operations among the existing public and commercial banks in Turkey) and 3 month Government Bond interest rates. The data used in this study is taken from the Capital Market Board of Turkey monthly bulletins.

3.2. Comparison of Exemplar Based Learning Model with Econometric Model

The variables are in general complexly related and buried in noise. One advantage of the NGE algorithm is that it can learn to detect statistical relations hidden in masses of data which even a human expert would not see.

Compared to Econometric Models (EM), Exemplar Based Learning (EBL), i.e., learning from examples is relatively new, but it plays an important role since it is very effective especially when the amount of historical data on hand is not sufficient, past data are either no longer relevant or expected to change significantly, and if there is not enough time available to gather and analyze the data [Jain 1987]. Unlike other models of forecasting; Nested Generalized Exemplars algorithm (NGE) which is a type of EBL, has two steps to forecasting: first, learning the data and generating rules, second, using the rules to prepare a forecast. The broad distinction between NGE type of forecasting and EM oriented forecasting is in the generation of rules from the present data and the process of judgment. In the broad sense of the word, judgment is a necessary ingredient of all types of predictions.

In NGE, computer is left to make judgments while creating the rules depending on the variables. The selection of variables becomes very important in this type of model because after the learning step, giving personal or expertise feedback to the computer will not be possible.

Judgment in EM can enter into the forecasting process at various stages, though its proper role is to be a complement to, not a substitute for, a competent economic and statistical analysis. In NGE algorithm the judgment enters only at the design stage of the forecast. The understanding of the judgmental process can be helpful in the development and use of quantitative techniques. Informed judgment can go far in making forecasts more consistent and dependable. Compared to EM, NGE forecasting explains the significance of the variables effecting the result of the forecast but in EM it is also possible to see the cross-correlation between those variables. It should be recognized that the ability to reach a good judgment is not a well-defined, technical and transferable skill. It is rather a function of personal experience and training in EM type of forecasting.

There is enough evidence that mechanical use of Econometric Models do not produce the most accurate forecasts [Keng 1984]. All the real world forecasts reflect the information of econometric and pure judgmental inputs.

Forecasting practitioners would probably agree that the speed with which a forecast is generated is important. Econometric models meet this requirement easily. The capability producing timely simulations and of calculating dynamic multipliers are other important features of econometric models.

There is enough evidence to support the thesis that human beings are ill-suited to the task of judging in a probabilistic environment, are poor substitutes for large-scale econometric models when internally consistent forecasts of numerous variables are needed and it should be noted that human beings have only limited information processing capacity. From this sense NGE algorithm, using computer's Central Processing Unit (CPU), can handle huge amounts of processes efficiently, reliably and faster than any other human being.

At present, it is far from clear whether an econometric forecast is superior or inferior to a learning from examples type of forecast. In some cases, EBL base their forecasts largely on figures produced by econometric methods. If we assume that econometric forecasts are judgmentally adjusted (which is normally the case), then even econometric forecasts contain the forecaster's judgment. In that case, there is no point in discriminating econometric forecasts from the learning based types.

In general, whether or not a forecast will be accepted depends on the user's value. Under the presumption that all forecasters are human beings and they are neither omnipotent nor omniscient, Econometric Models, because of being used for years and presented it's forecasts to users with a fair amount of scientific justifications, have the value of user's. However, NGE is in its infancy.

Exemplar-based learning algorithm is usually developed with the help of human experts who, by solving specific problems, reveal their thought process as they proceed. After keying into the computer all known information about a particular subject, programmers interview the experts at great length to determine how they use specialized knowledge to form judgments on fuzzy, non-numerical problems. If this process of analysis is successful, the computer program based on the analysis is able to solve narrowly defined problems in specialized areas that normally only seasoned experts can handle. The program analyzes the data using the same logic as an expert develops his findings over years of study and practice.

Exemplar-based learning algorithm, when faced with situations that are complex and unstructured, are considered successful, because it can apply its experience gained in its training phase to solve the problems in an efficient manner. It can employ plausible inference and reasoning from incomplete and uncertain data. It can explain and justify what it does. It does not acquire new knowledge. It restructures and reorganizes the knowledge gained from the data. It recognizes when a problem is outside the boundaries, i.e., exceptions. In essence, it transports chunks of decision making skill from a human expert's brain to a computer's brain. But it lacks the crucial need for an extensive memory. It is driven by a knowledge-base and judgmental knowledge that is typically made up of if-then rules.

Exemplar-based learning models solve problems by symbolic inference rather than sequential mathematical calculations done in econometric models [Coats 1987]. It allows for non-numeric tradeoffs and the representation of judgment. In contrast with the econometric models, this model, when faces with a question for which no answer exists, it applies common sense, and thus does not give up. Instead, it wastes much time searching through its data and rules to come up with a solution.

3.3. Comparison of Exemplar Based Learning Model with other Machine Learning Models

Exemplar-based learning has only appeared in the literature in the past four or five years, and there are currently very few researchers taking this approach. To identify the theory of NGE, we would like to compare it with the properties of different Machine learning models.

Knowledge Representation Schemes

Many systems acquire rules during their learning process [Buchanan&Mitchell 1978]. These rules are often expressed in logical form , but also in other form such as schemata [Mooney&DeJong 1985] (i.e. Explanation Based Generalization (EBG) system learns schemata for natural language processing). Usually such systems try to generalize antecedent part of the rule so that rules will cover larger number of examples. Another way to represent what is learned, is with decision trees [Quinlan 1986]. Decision trees seem to lack of clarity as representations of knowledge; the structure of the induced trees has been shown to be confusing to human experts, especially when the effect of one variable is masked by other variables. Rules and decision trees do not exhaust the possible representations of the knowledge a learning system may acquire. The NGE learning model uses neither of these representation. Instead, it creates a memory space filled with exemplars, many of which represents generalizations and some of which represent individual examples, where the exemplars are hyper rectangles. In addition, it modifies its own distance metric, which it uses to measure distances between exemplars, based on feedback from its predictions. Instead of generalizing values by replacing symbols with more general symbols, it generalizes by expanding hyper rectangles, which corresponds to replacing ranges of numeric values with larger ones.

Learning Strategy

In general, there are two categories: incremental and non-incremental (one-shot) learning. NGE falls into incremental learning category so that, it is sensitive to the order of the examples. The non-incremental learning model offers the advantage of not being sensitive to the order of the training examples. However, it introduces the additional complexity of requiring the program to decide when it should stop its analysis.

Domain Dependency

NGE is domain independent learning algorithm. Because input is simply vector of feature values and class of the example. It does not convert examples into another representational form, it does not need a domain theory to explain what conversions are legal, or even what the representations mean.

Generalization with Exception

Perhaps the most important distinguishing feature of the NGE is its ability to capture generalizations with exceptions. This is due to the working space (Euclidean) of the NGE. NGE explicitly handles exceptions by creating hole in the hyper rectangles. These holes (exceptions) are also hyper rectangles and they can have additional exceptions inside them.

Capability of Handling Multiple concepts

Some of the models that handle multiple concepts need to be told exactly how many concepts they are learning. However, NGE will create as many concepts as it needs. It performs better with a small number of features as do all learning models. However, it degrades only very gradually, as it scales up to larger number of features.

Different Type of Feature Values

Most of the learning models handle only one type of feature and class values (i.e. binary, discrete, continuous etc.). NGE can handle mixed type of feature and class values. For continuous values user defined matching tolerance can be used. Humans have remarkable ability to learn from single example, and this ability is one that the NGE model tries to emulate.

Disjunctive Concepts

Many concept learning models have ignored disjunctive concepts [Iba 1979], because they are very difficult to learn. NGE handles disjunctive concepts very easily. It can store many distinct exemplars which carry the same prediction or they describe the same category. An example which matches any of these distinct exemplars will fall into the category they represent. Set of such exemplars represent a disjunctive concept.

Inconsistent Data

NGE does not assume that, the exemplars are consistent (i.e. two examples with same feature values but they are in different categories). NGE simply keeps track of the reliability of the exemplars (number of correct prediction and number of reference to the exemplar). If NGE faces with inconsistency, it simply reduces the reliability of the exemplar and will generate new exemplar.

Problem Domain Characteristic

As mentioned before, NGE is domain independent learning algorithm. But it is worthwhile to consider the kinds of problem domains that are more suitable to NGE. In particular, NGE is the best suited for domains in which the examples form clusters in feature space and where the behavior of the examples in a cluster is relatively constant or they fall into the same category.

3.4. Methodology

Nested Generalized Exemplar theory is a variation of a learning model called exemplar-based learning, which was originally proposed as a model of human learning by Medin and Schaffer in 1978. In the simplest form of the exemplar-based learning, every example is stored in memory, with no change in representation. Apart from being simple and fast in learning , a major advantage is that no assumptions need to be made since the model is able to extract hidden information from the historical data. Compared to the linear regression method, the neural network produce much more accurate results [Wong&Tan 1991] . However, the performance of neural networks is affected by many factors, including the network structure, the training parameters and the nature of the data series [Tang&Fishwick 1991]. NGE adds generalization on top of the simple exemplar-based learning. In NGE, generalizations take the form of hyper rectangles in Euclidean n-space, where the space is defined by the feature values for each example. This theory does not cover all types of learning, rather it is a model of a process whereby one observes a series of examples and becomes adept at understanding what those examples are examples of. The algorithm compares new examples to those it has seen before and finds the most similar example in the memory. To determine what is most similar, a similarity metric is used. The term exemplar is used specifically to denote an example stored in computer memory. Exemplars have properties such as weights, shapes and sizes which can be adjusted based on the results of the prediction. The learning itself takes place only after the algorithm receives feedback on its prediction. If the prediction is correct, the algorithm strengthens the exemplar by increasing its weight, else weakens the exemplar.

It uses numeric slots for feature values of exemplar. Generalization is a process of replacing the slot values with more general values (i.e. replacing range of values [a,b] with another range [c,d] , where $c \leq a$ and $d \geq b$). The exemplar-based paradigm performs two kinds of generalization, one implicit and the other explicit.

If an example is stored in the memory with n being the number of features the system can recognize, then the example is a point in n-dimensional feature space. These points in memory are exemplars. The algorithm uses its similarity metric to compare new examples to the stored exemplars and uses the closest exemplar to make predictions. Thus the exemplars stored in feature space, even when stored as simple points, partition the space into regions, where the region surrounding a particular exemplar contains all points closer to that exemplar than to any other. The similarity metric determines the size and shape of this region. This generalization process is implicit, since it occurs automatically whenever a new point is stored in feature space.

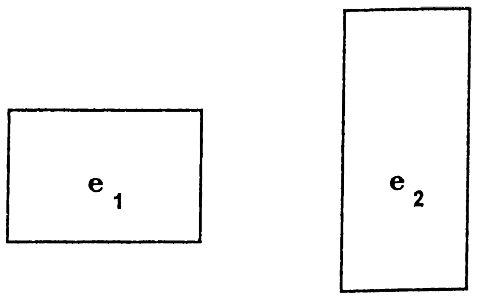

Figure 3.1: Old exemplars e_1 and e_2 .

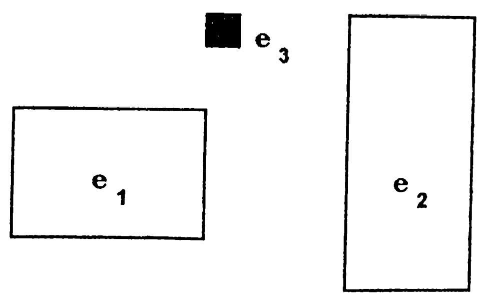

Figure 3.2: New exemplar e_3 matches e_1.

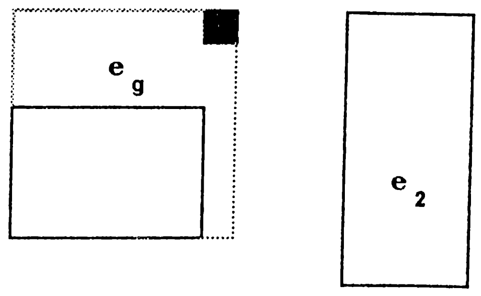

Figure 3.3: Match leads to success. Form a generalization e_g.

In NGE model, explicit generalization occurs by turning points into hyper rectangles, and by growing hyper rectangles to make them larger. Explicit generalization in the algorithm begins when a new example falls near two or more existing exemplars. This new point is combined with the closest existing point in memory to form a generalization which takes the shape of a hyper rectangle; that is, an n-dimensional solid whose faces are all rectangles. The closest existing exemplar may already be a hyper rectangle, in which case it is extended just far enough to include the new point. Figures 3.1- 3.3 provide a graphic illustration of this process. The generalization e_g includes a larger area than e_1, and replaces the former e_1 region in memory.

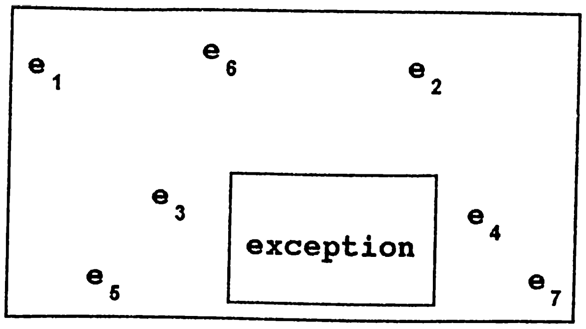

Figure 3.4: Generalization with exception .

If a new example falls inside an existing exemplar region in feature space, but the example behaves differently from others which fell in that same hyper rectangle , the learning system must recognize the new point as an exception. Once the system has recognized that the new point belongs in a different category, it simply stores the point. Prediction failures serve as a signal to the system that a new point does not fit into an existing category. If no more exceptions occur, the point remains a point. However, another exceptional point is found inside the hyper rectangle and if this second exception belongs to the same category as the first one, the two points are combined to create a small hyper rectangle inside the larger one. this new region behaves as an exception to the old generalization. Figure 3.4 illustrates, in two dimensions, what a generalization with a rectangular exception looks like in the feature space.

Once a theory moves from a symbolic space to a Euclidean space, it becomes possible to nest generalization one inside the other. This is where nested comes to the name of the theory. Its generalizations, which take the form of hyper rectangles in Euclidean n-space, can be nested to an arbitrary depth, where inner rectangles act as exception to the outer ones. This section describes the details of the nested generalized exemplar learning algorithm. The text below is organized exactly in the order the program functions. Each of the sections which follows, then, is a phase of the program's operation. Following the sections describing the individual steps in the algorithm, we give a summary and flowchart of the algorithm.

3.4.1. Initialization

In order to make predictions, the algorithm must have a history of examples on which to base its predictions. Memory is initialized by seeding it with a set of examples (the minimum size of this set is one). The seeding process simply stores each example in memory without attempting to make any predictions. An example is a vector of features, where each feature may have any number of values, ranging from two (for binary features) to infinity (for real-valued features). In addition , each exemplar has a slot containing the prediction associated with that example. The error tolerance parameter should be used in real-valued variables to indicate how close two values must be in order to be considered a match. This parameter is necessary because for real-valued variables it is usually the case that no two values ever match exactly, and yet the system needs to know if its prediction was close enough to be considered correct. The seed examples are used as the basis of memory, and memory is used to make predictions. The seeds may be scattered widely or tightly bunched, since they are chosen at random (with small or large number of seeds, the arrangement or choice of seeds has little or no effect on the program's learning performance). After initial seeding, the system begins its main processing loop.

3.4.2. Get the next example

The first step in the main loop is fetching the example. It keeps track, for every feature which has numeric values, of the maximum and minimum values experienced, to scale the features in the distance calculations.

3.4.3. The matching Process

The matching process is one of the central features of the algorithm. This process uses the distance metric to measure the distance (or similarity) between a new data point (an example) and an exemplar memory object (a hyper rectangle in n-E). We will refer to the new data point as E and the hyper rectangle as H..

The system computes a match score between E and H by measuring the Euclidean distance between the two objects. Consider the simple case where H is a point, representing an individual example. The distance is determined by usual distance function computed over every dimension in feature space :

$$D_{EH_k} = W_{H_k} \sqrt{\sum_{i=1}^{m} \left(W_{f_i} \frac{d_i f_i}{\max_i - \min_i} \right)^2}$$

$$d_i f_i = E_{f_i} - H_{k_{upper}} \quad \text{if} \quad E_{f_i} > H_{k_{upper}}$$

$$d_i f_i = H_{k_{lower}} - E_{f_i} \quad \text{if} \quad E_{f_i} < H_{k_{lower}}$$

$d_i f_i = 0$ **otherwise**

W_{H_k} **: weight of hyper rectangle$_k$ (H_k .reference / H_k .correct)**

W_{f_i} **: weight of the future$_i$**

$\max_i$, $\min_i$ **: maximum and minimum feature values respectively.**

E_{f_i} **: value of the i feature on example E.**

m : number of features recognizable on E.

The best match is the one with the smallest distance. A few special characteristics of the computation, which are not evident in the formula above , deserve mention here. First, let's suppose we measure the distance between E and H along the dimension f. Suppose for simplicity that f is a real valued feature. In order to normalize all distances, so that one dimension will not overwhelm the others, the maximum distance between E and H along any dimension is 1. To maintain this property, the system uses its statistics on the maximum and minimum values of every feature. Suppose that for E, the value of f is 10, and for H, the value of f is 30. The unnormalized distance is therefore 20. Suppose further that the minimum value of f for all exemplars is 3, and the maximum value is 53. Then the total range of f is only 50, and the normalized distance from E to H along this dimension is 20/50, or 0.4. Because the maximum and minimum values of a feature are not given a priority, the distance calculation will vary over time as these values change. This variation is a direct consequence of the incremental nature of the algorithm.

Now consider what happens when the exemplar, H, is not a point but a hyper rectangle, as is usually the case. In this case, the system finds the distance from E to the nearest face of H. The distance measured by the formula above, is equivalent to the length of a line dropped perpendicularly from the point E_{f_i} to the nearest surface, edge, or corner of H. This length is modified by the weighting factors. Notice that there are two weights on the distance metric. W_H is a simple measure of how frequently the exemplar, H, has been used to make a correct prediction. In fact, the use of this weight means that the distance metric measures more than just distance. It is a reliability measure, or the probability of making a correct prediction, of each exemplar. This weight measure says, in effect, "in this region of the feature space, the reliability of my prediction is n", and of course the algorithm wants to maximize its success rate, so it should prefer more reliable exemplars. The distance metric accomplishes this as follows: suppose in the above example, H had been used for 15 previous predictions, and that it had been correct on 12 of those occasions. The system will multiply the weight of the total match score between E and H by 15/12 or 1.25. Thus weight is a non-decreasing function of the number of times an exemplar has been used. If the exemplar always makes correct prediction than the weight will remain at 1. Larger weights will make the rectangular exemplars very distant from new examples.

The other weight measure, w_i , is the weight of the i^{th} feature. These weights are adjusted over time. Since the features do not normally have equal predictive power, they need to be weighted differently

3.4.3.1. Correct Prediction

If the algorithm above, makes the correct prediction, it records some statistics about its performance and then makes a generalization. Two objects E and H are used to form the generalization. H is replaced in memory by a larger object that is a generalization of E and H. H may have been a single point, or it may have been a hyper rectangle. (After a single generalization, an exemplar becomes a hyper rectangle.) If H was a hyper rectangle, then for every feature of E which did not lie within H, H is extended just far enough so that its boundary contains E. If H and E were both points, H is replaced by a new object which, for each feature of E and H, a range of values defined by E and H. For a simple case with just two features f_1 and f_2 if E was at (2,5) and H was a point at (3,16), then the new object would be a rectangle extending from 2 to 3 in the f_1 dimension and from 5 to 16 in the f_2 dimension. If an example falls within an area where two hyper rectangles overlap, the system matches it to the smaller exemplar, because larger exemplars may have been over-generalized.

3.4.3.2. Incorrect Prediction

If the system makes the wrong prediction, it has one more chance to make the right one. The idea is to keep down the size of the memory. Before creating a new exemplar, the algorithm first looks at the second best match in memory. Assume that H_1 was the closest exemplar to E and H_2 was second closest. If using the second best match, H_2, will give the correct prediction, then the system tries to adjust hyper rectangle shapes in order to make the second closest exemplar into the closest exemplar. It does this by first creating a generalization from H_2 and E. It then specializes the best matching exemplar, H_1, by reducing its size. It reduces the size of H_1, which must be a hyper rectangular (if not, then the system does nothing), by moving its edges away from the new exemplar just far enough so that H_2 becomes closer along that dimension. This process is basically a stretching and shrinking operation: H_2 is stretched, and H_1 is shrunken, in order to change the order of the match the next time around. The goal of this process is to improve the predictive accuracy of the system without increasing the number of exemplars stored in memory. Shrinking an existing exemplar loses information.

A very important consequence of this second chance heuristic (referring to the second closest exemplar) is that it allows the formation of hyper rectangles within other hyper rectangles. If a new point p_1 lies within an existing rectangle, its distance to that rectangle will be zero. Its distance to another point p_2 (previously stored exception) within the rectangle will be small but positive. Thus the algorithm will first assume that the new point belongs to the same category as the rectangle. If p_1 is in the same category as p_2, then the second chance heuristic will discover this fact, and form a rectangle from these two points.

If the second best match also makes wrong prediction, then the system simply stores the new example, E, as a point in memory. Thus E is made into an exemplar which can immediately be used to predict future examples, and can be generalized and specialized if necessary. This new exemplar may be inside an existing exemplar H, in which case it acts as an exception to, or hole in H.

The algorithm adjusts the weights w_i on the features f_i after discovering that it has made the wrong prediction. Weight adjustment occurs in a very simple loop : for each f_i, if E_{f_i} matches H_{f_i}, the weight w_i is increased by setting $w_i = w_i (1 + \Delta f)$, where Δf is the global feature adjustment rate. An increase in weight causes the objects to seem farther apart, and the idea here is that since the algorithm made a mistake matching E and H, it should push them apart in space. If E_{f_i} does not match H_{f_i}, then w_i is decreased by setting $w_i = w_i (1 - \Delta f)$. If the algorithm makes a correct prediction, feature weights are adjusted in exactly the opposite manner; i.e., weights are decreased for features that matched, which decreases distance, and increased for those that did not. Each weight w_i applies uniformly to the entire feature dimension f_i, so adjusting w_i will move around exemplars everywhere in feature space.

3.4.4. Take the two closest hyper rectangles to example E

```
$D_{EH_{min1}}$ and $D_{EH_{min2}}$ for $H_{min1}$ and $H_{min2}$ respectively
if $H_{min1}$.class == E.class then
  begin
      $H_{min1}$.correct ++
      $H_{min1}$.reference ++
      Generalize $H_{min1}$ with E in all feature dimensions
  end
else if $H_{min2}$.class == E.class then
  begin
      $H_{min1}$.reference ++
      $H_{min2}$.correct ++
      $H_{min2}$.reference ++
      Generalize $H_{min2}$ with E in all feature dimensions
  end
else
  begin
      $H_{min2}$.reference ++
      Store E as new exemplar
      adjust all features weights as follows
      if $E_{f_i}$ matches with $H_{min1_{f_i}}$ then
          $w_{f_i} = w_{f_i}(1 + \triangle f)$
      else
          $w_{f_i} = w_{f_i}(1 - \triangle f)$
  end
GOTO Step 2 (get new example, if no more exit)
```

$\triangle f$: is global feature adjustment rate.

CHAPTER IV

FINDINGS

4.1. Modifications to the Algorithm

As mentioned in properties of the NGE algorithm, main purpose is in learning the correct prediction of the example. There are some possible modifications to the NGE:

1. Reducing the sensitivity of the algorithm to the order of examples. To achieve this, we can do simple modification such that, for initial exemplars instead of taking first a few examples blindly, we can choose initial exemplars from training set according to average feature values. It will help us to start at a better position.

2. Second issue is reducing the number of rules which are redundant so that resultant rules can be criticized by human experts. Hence, resultant rules should be compact.

We can explain the modified NGE after pointing out the above issues. New algorithm consists of two phases: First Phase is same as original NGE. Second Phase of the algorithm tries to reduce number of exemplars (generated in first phase) by eliminating the redundant rules.

(1). Sort (descending) the exemplars according to their reliability ($1/W_H$).

(2). If H_{C_1} overlaps with H_{C_2} and $C_1 <> C_2$ then

shrink hyper rectangle which has small ($1/W_H$) value.

(3). Remove the redundant hyper rectangles (i.e. H_1 is in H_2 and both have same classes. This can happen due to the generalization).

(4). Mark the exceptions hence, no more processing is done for exceptions.

(5). List the rules and than stop.

4.2. Application of the Algorithm to the Stock Market

When an institutional investor has to make a decision on whether and how to act on the stock and bond markets on which he usually operates, he has to take into account a large number of factors which influence his choice. It has never been as difficult as over the last few years to operate on the financial markets all over the world, as well as to venture predictions on their future trend [Pasquale 1991].

It is not possible to consider all the variables that effect the stock returns, because there is no clear distinction between economical activities. There are some external effects which has to be considered in evaluation process. (insider trading, rumors, other economical activities, alternative markets, political effects, macro economical conjuncture, sector conjuncture, psychological effects etc.). Although these variables could be used for computer representation and processing, we could not find relevant data for these indicators. In this study we used a greatly simplified model of the Stock Market.

We analyzed the daily percentage in the ISE composite index of the 395 examples. 12% of the data were belonging to changes of more than 3% increase in the ISE composite index. 11% belonging to changes of more than 3% decrease in the composite index, 77% belonging to increase and decrease between 3%. The maximum decrease on the ISE composite index during this period was observed to be 10.19% and maximum increase was 10.32%. The daily changes in the composite index were commonly very close and between 3% deviations and the average change in the ISE composite index was 0.1269 which shows the increasing trend of the Turkish Stock Market. The maximum increase and decrease for the variables used during this period were: Turkish Republic Gold Coin; maximum increase 5.17%, maximum decrease 4.06%. Respectively, 9.05% and 6.70% for one Ounce of Gold, 4.02% and 3.64% for US Dollar; 4.19% and 5.72% for Deutsche Mark, 7.18% and 6.51% for Government Bond Interest Rate, finally 5.30% and 4.67% for Commercial Banks' Average Deposit Rate. Therefore, we defined three classes representing the trend of the composite index:

1. ISE Composite Index will increase more than 3% (class C_1).

2. ISE Composite Index will decrease more than 3% (class C_2).

3. ISE Composite Index (daily returns) will be between -3% to +3% (class C_3).

The package uses only historical data for training in the selected field, and no other explicit rules are needed. The program generates rules to the user to make his/her decision. The performance results of the program run are listed on Table 4.1 for twenty different values of Δf (the global feature adjustment rate). As it is seen from Table 4.1, the best global feature adjustment rate turns out to be $\Delta f = 0.030$ for this set of data since it generates minimum number of rules with the maximum first attempt correct prediction value.

The first column of the table shows different values of global future adjustment rates that are used to identify the most appropriate value. In order to find the best prediction performance of the algorithm with different values of global feature adjustment rates, we used 5 of 395 examples to seed the initialization process and the remaining 80% (312 examples) are used to train the algorithm and the rest, 78 examples, are used to test the algorithm. Second column of Table 4.1 shows the test results of making a correct prediction (the class of the example on hand is same with the class of the closest rectangle in the memory, e.g., index increase more than 3% as class c1) when algorithm tries to generalize the example on hand with the closest rectangle in the memory. The ratio is calculated as follows: the total number of correct predictions made to the closest rectangle in the memory / total number of data used during the test * 100. Third column similarly shows rate of successful prediction of the index at the second attempt (when the closest rectangle class does not match with the class of the example on hand, try to generalize with the second second closest rectangle in the memory). The fourth column of the table presents the total performance of the algorithm for predicting the stock market index (second column plus third column). Last column of Table 4.1 shows the maximum number of rules generated in the memory.

Table 4.1: Performance Results for different Global Future Adjustment Rates

Global Feature Adjustment Rate	**First attempt success (%)**	**Second attempt success (%)**	**Total attempt success (%)**	**Maximum Number of Rules Generated in the Computer Memory**
0.0050	65.64	18.21	83.85	63
0.0100	70.77	11.54	82.31	69
0.0150	71.54	10.00	81.54	69
0.0200	71.54	10.51	82.05	68
0.0250	71.28	11.28	82.56	63
0.0300	**70.00**	**12.82**	**82.82**	**61**
0.0350	71.03	11.79	82.82	70
0.0400	72.31	10.26	82.56	67
0.0450	72.82	9.23	82.05	67
0.0500	69.23	14.87	84.10	67
0.0550	69.74	13.33	83.08	64
0.0600	67.95	14.62	82.56	63
0.0650	71.03	12.31	83.33	63
0.0700	69.74	14.62	84.36	63
0.0750	69.74	14.62	84.36	65
0.0800	71.28	14.87	86.15	62
0.0850	70.82	13.59	84.41	62
0.0900	70.82	11.28	82.10	64
0.0950	70.82	13.33	84.15	68

Table 4.2 displays the set of rules that correspond to the global adjustment rate 0.030. For each feature, two values are given, labeled lower and upper, that describe the allowable percent change in that attribute.

Table 4.2 Rules Generated for Global Feature Adjustment Rate = 0.030.

Features	Turkish Republic Gold Coin		One Ounce of Gold		US Dollar		Deutsche Mark		Government Bond Interest Rate		Commercial Bank Average Deposit Rate			
Weights	2.386		1.993		1.993		1.877		5.528		4.349			
Boundary	Lower	Upper	Lower	Upper	Lower	Upper	Lower	Upper	Lower	Upper	Lower	Upper	CF	Class
Rule 1	-3.70	-2.88	-1.06	-0.93	-0.67	-0.61	-0.75	-0.41	0.00	0.00	0.00	0.00	6/11	c1
Rule 2	-3.70	-2.88	-1.53	-1.16	-0.67	-0.61	-0.75	-0.75	0.00	0.00	0.00	0.00	5/8	c1
Rule 3	-3.70	-2.88	-3.65	-1.87	-0.95	-0.95	-0.75	-0.75	0.00	0.00	0.00	0.00	3/7	c1
Rule 4	-3.70	-2.88	-0.64	-0.53	0.12	0.17	-1.26	-0.75	0.00	0.00	0.00	0.00	2/4	c1
Rule 5	0.82	0.82	1.36	1.36	0.51	0.51	1.95	1.95	0.00	0.00	0.00	0.00	2/4	c1
Rule 6	-3.70	-2.88	-0.64	-0.53	0.40	0.42	-0.75	-0.41	0.00	0.00	0.00	0.00	2/4	c1
Rule 7	-3.70	-2.88	-0.84	-0.65	-0.67	-0.61	-0.75	-0.75	-2.72	-2.04	-0.11	-0.03	2/3	c1
Rule 8	-4.06	-4.00	-0.93	-0.64	-0.67	-0.61	-0.75	-0.75	0.00	0.00	0.00	0.00	2/4	c1
Rule 9	-3.70	-2.88	-2.33	-1.87	0.00	0.11	-0.75	-0.75	1.44	6.84	-0.36	-0.11	1/3	c2
Rule 10	1.16	1.16	0.48	0.48	0.11	0.11	-0.75	-0.75	0.00	0.00	0.00	0.00	1/2	c2
Rule 11	-3.70	-2.88	-1.05	-1.05	-0.23	-0.23	-0.75	-0.41	0.00	0.00	0.00	0.00	1/2	c2
Rule 12	-3.70	-2.88	-0.64	-0.53	0.11	0.11	-0.75	-0.75	0.00	0.00	0.16	0.16	1/2	c2
Rule 13	-3.70	-2.88	-0.64	-0.53	0.56	0.56	-0.75	-0.75	0.00	0.00	0.00	0.00	1/2	c2
Rule 14	-3.70	-2.88	-1.58	-1.58	-0.94	-0.94	-0.75	-0.41	2.14	2.14	0.05	0.05	1/2	c2
Rule 15	-3.70	-2.88	-0.64	-0.56	-0.67	-0.61	-0.75	-0.41	0.00	0.00	0.00	0.00	1/3	c2
Rule 16	-3.70	-2.88	0.47	0.47	-0.67	-0.63	-0.75	-0.41	0.00	0.00	0.00	0.00	1/2	c2
Rule 17	0.80	0.80	0.80	0.80	0.21	0.21	-0.75	-0.41	0.00	0.00	0.00	0.00	1/2	c2
Rule 18	-3.70	-2.88	-0.64	-0.53	0.41	0.41	-0.75	-0.41	0.00	0.00	0.00	0.00	1/2	c2
Rule 19	-3.70	-2.88	-0.64	-0.53	-0.10	-0.10	-0.75	-0.75	0.00	0.00	0.00	0.00	1/2	c2
Rule 20	0.78	0.78	0.53	0.61	0.20	0.20	-0.75	-0.41	0.00	0.00	0.00	0.00	1/2	c2
Rule 21	-3.78	-2.88	-0.64	-0.53	0.20	0.20	-0.75	-0.41	0.00	0.00	0.00	0.00	1/2	c2
Rule 22	-3.70	-2.88	-0.64	-0.53	-0.30	-0.30	-0.75	-0.41	0.00	0.00	0.00	0.00	1/3	c2
Rule 23	2.40	2.40	1.89	1.89	-0.16	0.00	-0.75	-0.41	0.00	0.00	0.00	0.00	1/3	c2
Rule 24	1.02	1.02	0.93	1.03	0.50	0.50	-0.75	-0.41	0.00	0.00	0.00	0.00	1/2	c2
Rule 25	-3.70	-2.88	-0.64	-0.53	0.00	0.00	-0.75	-0.41	0.00	0.00	0.00	0.00	1/3	c2
Rule 26	-3.70	-2.88	-1.28	-1.28	-1.34	-1.34	-0.75	-0.41	0.00	0.00	0.00	0.00	1/2	c2
Rule 27	-3.70	-2.88	-0.64	-0.53	0.29	0.29	-0.75	-0.41	0.00	0.00	0.00	0.00	1/2	c2
Rule 28	-3.70	-2.88	-0.76	-0.76	-0.38	-0.38	-0.75	-0.41	0.00	0.00	0.00	0.00	1/2	c2
Rule 29	-3.70	-2.88	-0.65	-0.65	0.37	0.37	-0.75	-0.41	0.00	0.00	0.00	0.00	1/1	c2
Rule 30	-3.70	-2.88	-0.64	-0.53	0.27	0.27	-0.75	-0.41	0.00	0.00	0.00	0.00	1/2	c2
Rule 31	-3.70	-2.88	-1.08	-1.08	-2.22	-2.22	-0.75	-0.41	-2.72	-1.86	0.00	0.00	1/1	c2
Rule 32	-3.70	-2.88	-0.64	-0.53	0.33	0.33	-0.75	-0.41	0.00	0.00	0.00	0.00	1/2	c2
Rule 33	-3.70	-2.88	-0.64	-0.53	0.00	0.00	-0.75	-0.41	2.17	2.17	0.00	0.00	1/1	c2
Rule 34	-0.50	0.00	-0.47	-0.15	-0.67	-0.39	-0.58	-0.29	-1.15	0.00	-0.11	0.00	18/46	c3
Rule 35	2.87	5.17	0.53	0.75	0.68	0.80	-0.29	-0.23	0.00	0.00	0.00	0.00	4/6	c3
Rule 36	0.28	0.64	-0.53	0.45	-0.51	-0.45	0.55	0.80	0.00	0.00	0.00	0.00	3/5	c3
Rule 37	2.02	2.02	0.93	1.35	0.17	0.17	-0.75	0.00	0.00	0.00	0.00	0.00	2/5	c3
Rule 38	0.37	0.37	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1/2	c3
Rule 39	-0.69	-0.69	0.00	0.00	-1.91	-1.91	-1.31	-1.31	0.00	0.00	0.00	0.00	1/2	c3
Rule 40	-3.70	0.78	-0.64	0.39	0.66	0.66	-0.75	0.60	0.00	1.13	0.43	0.43	1/5	c3
Rule 41	-2.88	-2.88	0.49	0.49	0.56	0.56	-0.21	-0.21	6.84	6.84	5.30	5.30	1/1	c3
Rule 42	-0.32	-0.32	-0.11	-0.11	-1.95	-1.95	-0.41	-0.41	0.00	0.00	0.00	0.00	1/1	c3
Rule 43	-3.71	-3.71	0.69	0.69	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1/2	c3
Rule 44	0.00	0.00	0.29	0.29	0.11	0.11	-0.39	-0.39	0.00	0.00	0.00	0.00	1/2	c3
Rule 45	0.56	0.56	0.66	0.66	0.42	0.42	1.45	1.45	0.00	0.00	4.04	4.04	1/2	c3
Rule 46	-1.40	-1.40	-1.87	-1.87	0.00	0.00	-0.36	-0.36	0.00	0.00	0.00	0.00	1/1	c3
Rule 47	0.50	0.50	0.42	0.42	-0.20	-0.20	4.19	4.19	0.00	0.00	0.00	0.00	1/1	c3
Rule 48	-0.74	-0.74	-1.16	-1.16	0.29	0.29	0.00	0.00	0.00	0.00	0.00	0.00	1/2	c3
Rule 49	0.00	0.00	0.00	0.00	0.53	0.53	0.28	0.28	0.00	0.00	0.00	0.00	1/2	c3
Rule 50	0.91	0.91	1.21	1.21	1.21	1.21	-0.14	-0.14	0.00	0.00	0.36	0.36	1/2	c3

The weights row in Table 4.2 displays the weights of the features at the end of the run. The initial weights of the features are set as one at the beginning of the algorithm which means that the variables are assumed to have 'equally' significant effects on the ISE composite index. During the training process, if the information reflected in the past prices of these attributes causes an incorrect prediction then the significance of that feature's weight should be increased, i.e., to send away that example in the space, for preventing any further incorrect predictions [$w_i = w_i (1 + \Delta f)$, where Δf is the global feature adjustment rate]. Similarly, if a correct prediction is made than the weights of the features should be decreased, i.e., to make it closer [$w_i = w_i (1 - \Delta f)$]. As it is seen from the calculation of weights, the most significant feature's weight can be at least zero (if the digits used is not enough to represent very small numbers and truncates as zero, else it is seen that it gets very small but never reaches to zero) and the least significant, sent away, feature weight can be a large positive number. Therefore the closest feature weight to zero shows the significance of that attribute. Deutsche Mark variable, having the closest weight to zero means that change in its price have significance on the accuracy of the result. Then comes the US Dollars, one ounce of gold, and Turkish Republic Gold Coin.

For example, if we try to read rule number 50: If the daily percentage change in the price of Turkish Republic Gold Coin increases 0.91% and the price of one Ounce of Gold increases 1.21% and the effective selling price of a US Dollar increases 1.21% and the effective selling price of DM decreases 0.14% and there is no change in the Commercial Bank average deposit rate and the Government Bond interest rate increases 0.36% then the index will be c3; change in ISE composite index will be between +3% and -3%.

The Coverage Factor (CF) on column eight, (#correct / #references), means that the generated rule is referenced #references times and #correct of them were successful prediction. This column in other words explains the reliability of the generated rule. Larger the coverage factor value more reliable the rule is. The last column of the table explains the class of that rule.

CHAPTER V

CONCLUSION

The performance of NGE in predicting stock returns can be judged satisfactory because the test results shows that making a correct prediction of ISE composite index is about 70% (70% of the predictions were correct). As it is seen from the results of Table 4.2; the weights of Commercial Bank average deposit rate and Government Bond interest rate are about four times larger than the weights of the other attributes, which means that these features have little significance on the decision process of the system. The most significant features were Deutsche Mark, US Dollar, and One Ounce of Gold.

The complexity and the limitations of the problem mostly lie in the very high number of factors effecting the model, difficulty of interpretation, and in the availability of the relevant data. The model developed should not be interpreted as a complete production model, since a number of aspects of the problem were not represented as the available input variables because of the limited availability of the data. It is not possible to consider all the variables that effect the stock returns, because there is no clear distinction between economical activities. There are some external effects which have to be considered in evaluation process. (insider trading, rumors, political effects, psychological effects, several other currency exchange rates, import and export, national reserves, CPI, money supply, unemployment rate, trade balance, current account balance, industrial productivity index etc.). Although these variables could be used for computer representation and processing, we could not find relevant data for these indicators. In this study we used a greatly simplified model of the Stock Market.

The intention of the study was to use the previously defined monetary variables, that were assumed to have significant effects in predicting the composite index of the Turkish stock market. These variables (Central Bank effective selling price of US Dollar and DM, İstanbul Tahtakale closing selling price of Turkish republic gold coin and one ounce of gold, Commercial Bank average deposit rate, and Government Bond interest rate) are only a subset of the variables that effect the composite index of the Turkish stock market. Among these, Commercial Bank Deposit rate and Government bond interest rate contains data which does not change very frequently (daily), so their contribution in the prediction of ISE composite index were very limited.

If we further try to decrease the number of rules by making more generalization to make these rules more compact then we are left in the trade of loosing heuristic information and covering range of values that should not be included in the rule.

A prevalent problem that we have faced by the design of this model was the limited sample size. To compound this problem , the natural inclination is to use as many factors as possible, but this gives rise to the problem of overfitting. First, the computer used to run this software had limited read and access memory (RAM 640KB), which reduces the performance of the system, because available space decreases as the number of exemplars increase in the memory. Second, the personal computer rounds up the floating point calculations because of its small number of digits, which causes more rules to be generated. We recommend researchers to develop or run the package on a computer that have high processing power which will not round up the digits and hence less number of rules could be achieved. By the help of a powerful computer, many factors could be included into the model with long historical data for more reliable, efficient and faster results.

In particular the possibility of exogenous shocks makes the technical results especially misleading because of long historical series of data. So our recommendation is that the analysis of the market, through the economical and political factors, which influence it should be analyzed before any other investment decision. If the general trend of the economy is negative, the possible positive results coming from this study, should be reconsidered under the opinions of experts and compared with the results of other studies with greater caution for a profitable investment.

What we have been assuming so far obviously applies to a speculative investor who is going to operate in the short or very short term where he/she must have a very good analysis of the Turkish Market Structure (e.g., Turkish political stability, International political stability, Turkish economical situation etc.) in order to be successful while using this kind of a technical analysis method.

The feature adjustment rate, changes according to the length and type of data, so you must try different values before making your decision. The rules generated for different values of global feature adjustment rate, have some what similar rules. A good examination of them will yield limited number of rules without making further generalization. Since NGE falls into incremental learning category, it is sensitive to the order of the examples. Modifications for input order insensitivity will help the to start at a better position.

Currently, there are no systematic procedures for building neural network forecasting models in predicting stock returns. However, neural network research is still a fast growing discipline. New theoretic and algorithmic results make it possible to build automatic neural network forecasting systems. The potential of adaptive learning and the emergence of high speed parallel processing architecture's make neural networks a powerful alternative for forecasting in security markets.

APPENDIX A

SETTING FILE OF THE SOFTWARE

```
examplefile index.dat
adjustmentrate 0.030
#attributes 6
#classes 3
#examples 395
initialsetsize 2

attributes
Turkish Republic Gold Coin
One ounce of Gold
US Dollar
Deutsche Mark
Government Bond Interest Rate
Commercial Bank Average Deposit Rate

classes
index will increase more than 3%
index will decrease more than 3%
index will not change (between -3% +3%)
```

APPENDIX B

SOURCE CODE OF THE SOFTWARE

```
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <stdlib.h>
#include <alloc.h>

#define NATTRIBUTES 6    /* maximum # of attribute */
#define NTEMPLATES  500 /* maximum # of template in memory */
#define NCLASSES   3
#define INITIALWEIGHT 1.0
#define INITIALSETSIZE 10
#define NAME_SIZE     80

#define INFINITE  (float)2.0E+35
#define SUCCESS   1
#define FAIL     0
#define VALID    0            /* is valid */
#define EXCEPTION 1/* is exception (hole in other class) */
#define INVALID  3            /* is disjuncted with other template */
#define REDUNDANT 4        /* is completely in other template with same class */

struct Range {
        float x,y;              /* [x..y] */
        char type;              /* type of the attribute */
};

struct Template {
        int nref,               /* nref=# of reference to the template */
                ncor,           /* ncor=# of correct prediction */
                status;                 /* status of the template */
        struct Range a[NATTRIBUTES];            /* ranges of the attributes  */
        int dlist[NATTRIBUTES][NTEMPLATES];     /* disjunctive list */
        int dsize[NATTRIBUTES];  /* # of disjunctive element for each attribute */
        int class;                      /* number of classes          */
};

struct Example {
        float a[NATTRIBUTES];
        int class;
 };

 struct Classtemplate {
        int n;
        struct Template *t[NTEMPLATES];
 };

 struct Classtemplate ctemplate[NCLASSES];
 struct Template *templates[NTEMPLATES];
 int setsize=INITIALSETSIZE,nattr,nclass,nexample,ntemplate;
 int Specialize=1;
 float alpha=0.045;       /* attribute adjustment rate */
 float tolerance=0.0;     /* attribute match tolerace */

 float distance[NTEMPLATES];
 char efilename[100];
 char anames[NATTRIBUTES][NAME_SIZE],cnames[NCLASSES][NAME_SIZE];
 float maxof[NATTRIBUTES],minof[NATTRIBUTES],avrgof[NATTRIBUTES];
 float aw[NATTRIBUTES];   /* weights of the attributes */

 void error_exit(n,msg)
 int n;
 char *msg;
 {
        printf("Error : %d %s\n",n,msg);
        exit(n);
 }

/* read an example form fp into e with na attributes and its class */
readexample(fp,e,na)
FILE *fp;
struct Example *e;
int na;
{
int i;
   for (i=0; i<na; i++)
      if( fscanf(fp,"%f",&e->a[i])<1)
          return(1);
   if ( fscanf(fp,"%d",&e->class)<1)
      error_exit(2,"Read error in Training set");
   return(0);

}
float sqr(x)
float x;
{
 return((float)x*x);
}

findmx(fp,mx,mn,av,ne,na)
```

```
 FILE *fp;
 float *mx,*mn,*av;
 int *ne,na;
 {
 struct Example e;
 float sum[NATTRIBUTES];
 int i,j;
     for (j=0; j<na; j++) {
        mn[j]=INFINITE;
        mx[j]=-INFINITE;
        sum[j]=0.0;
     }
     i=0;
     while (1) {
       if (readexample(fp,&e,na))
         break;
       for (j=0; j<na; j++) {
           if (e.a[j] < mn[j])
             mn[j]=e.a[j];
           else if (e.a[j] > mx[j])
             mx[j]=e.a[j];
           sum[j] +=e.a[j];
       }
       i++;
     }
     if (i<setsize) {
       printf("Error 3 : Too few (%d) examples\n",i);
       exit(3);
     }
     (*ne)=i;
     for (j=0; j<na; j++)
       av[j]=(float)sum[j]/i;
     rewind(fp);
}

float dfsqr(r,w,x,mn,mx)
struct Range *r;
float x,mn,mx,w;
{
float df;
        if (x>r->y)
                df=x-r->y;
        else if (x<r->x)
                df=r->x-x;
        else return(0.0);
        return((float)sqr(w*df/(mx-mn)));
}

compute_distance(d,w,mx,mn,t,e,na)
struct Template *t;
struct Example *e;
```

```
 float *d,*w,*mx,*mn;
 int na;
 {
 int i;
 float sum=0.0;

        for (i=0; i<na; i++)
                sum +=(float)dfsqr(&t->a[i],w[i],e->a[i],mn[i],mx[i]);
        *d=(float)fabs((double)t->nref/t->ncor*sqrt(sum));
 }

 /* find the closest(m1) and second closest(m2) template indexs outof n template */
 find_two_min(d,n,m1,m2)
 float *d;
 int n,*m1,*m2;
 {int i;
 float min1=INFINITE,min2=INFINITE;
        *m1=-1; *m2=-1;

        for (i=0; i<n; i++)
                if (d[i]<min1) {
                        *m1=i;
                        min1=d[i];
                }
        if (*m1>-1) {
                d[*m1]=INFINITE;
                for (i=0; i<n; i++)
                        if (d[i]<min2) {
                          *m2=i;
                          min2=d[i];
                    }
                d[*m1]=min1;
        }
}

adjust_attr_weight(t,e,w,na,result)
struct Template *t;
struct Example *e;
float *w;
int na,result;
{
float rate;
int i;
       if (result==SUCCESS)
               rate=-alpha;
       else
               rate=alpha;
       for (i=0; i<na; i++)
               if (e->a[i] >= t->a[i].x && e->a[i]<=t->a[i].y)
                       w[i] =w[i]*(1+rate);
               else
```

```
                        w[i] =w[i]*(1-rate);
 }

 adjust_template_score(t,result)
 struct Template *t;
 int result;
 {
        t->nref++;
        if (result==SUCCESS)
                t->ncor++;
 }

 generalize(t,e,na)
 struct Template *t;
 struct Example *e;
 int na;
 {
 int i;
        for (i=0; i<na; i++) {
                if (e->a[i] < t->a[i].x)
                        t->a[i].x=e->a[i];
                else if (e->a[i] > t->a[i].y)
                        t->a[i].y=e->a[i];
        }
 }

 storenewtemplate(t,nt,na,e)
 struct Template *t[];
 struct Example *e;
 int *nt,na;
 {
 int i;
        if (*nt>=NTEMPLATES) {
                printf("Template buffer full...\n");
                return(1);
        }
        t[(*nt)]=(struct Template *)calloc(1,sizeof(struct Template));
        if (t[*nt]==NULL) {
                return(1);
        }
/*              error_exit(7,"Not enough memory");  */

        t[*nt]->nref=1;  /* 0 */
        t[*nt]->ncor=1;   /* -1; */
        t[*nt]->class=e->class;
        for (i=0; i<na; i++) {
                t[*nt]->a[i].x=e->a[i];
                t[*nt]->a[i].y=e->a[i];

        }
        (*nt)++;
        return(0);
}

/* is [x1-tolerance..y1] and [x2-tolerance..y2] overlaps */
overlaps(x1,y1,x2,y2)
float x1,y1,x2,y2;
{
        if (y1< x2*(1-tolerance))
                return(0);
        if (y2< x1*(1-tolerance))
                return(0);
        return(1);
}

/* is [x1..y1] in [x2..y2] */
inside(x1,y1,x2,y2)
float x1,y1,x2,y2;
{
        if (x1>=x2 && x1<=y2 && y1>=x2 && y1<=y2)
                return(1);
        return(0);
}

/* t[min2]=success, t[min1]=fail specialize t[min1] */
specialize(t,min1,min2,na)
struct Template *t[];
int min1,min2,na;
{
int i,n=0;
float temp;
        for (i=0; i<na; i++)
                n +=inside(t[min2]->a[i].x,t[min2]->a[i].y,t[min1]->a[i].x,t[min1]->a[i].y);
        if (n==na)  /* template[min2] is in template[min1] (if hole)*/
                return;
        n=0;
        for (i=0; i<na; i++)
                n +=inside(t[min1]->a[i].x,t[min1]->a[i].y,t[min2]->a[i].x,t[min2]->a[i].y);
        if (n==na)  /* template[min1] is in template[min2] (if hole)*/
                return;
        for (i=0; i<na; i++) {
                if (overlaps(t[min2]->a[i].x,t[min2]->a[i].y,t[min1]->a[i].x,t[min1]->a[i].y)) {
                        if (t[min2]->a[i].y> t[min1]->a[i].x )
                                t[min1]->a[i].y=t[min2]->a[i].x;
                        else
                                t[min1]->a[i].x=t[min2]->a[i].y;
                        if (t[min1]->a[i].y<t[min1]->a[i].x) {
                                temp=t[min1]->a[i].y;
                                t[min1]->a[i].y=t[min1]->a[i].x;
                                t[min1]->a[i].x=temp;
                        }
                }
        }
  }

  train(t,e,w,na,nt)
  struct Template *t[];
  struct Example *e;
  int na,*nt;
  float *w;
  {
  int i,min1,min2;

        for (i=0; i<(*nt); i++)
              compute_distance(&distance[i],w,maxof,minof,t[i],e,na);
        find_two_min(distance,*nt,&min1,&min2);
/*      printf("predictions %d %d\n",min1,min2); */
        if (min1<0)
              return;
        if (t[min1]->class==e->class) {
/*      printf("First change succes\n"); */
              adjust_template_score(t[min1],SUCCESS);
              generalize(t[min1],e,na);
        }
        else  {
              adjust_template_score(t[min1],FAIL);
              if (min2>-1) {            /*if second closest exist */
                if (t[min2]->class==e->class) {
/*                     printf("Second change succes\n"); */
                       adjust_template_score(t[min2],SUCCESS);
                       generalize(t[min2],e,na);
                  if (Specialize==1)
                       specialize(t,min1,min2,na);  /* specialize t[min1] */
                }
                else {
                       adjust_template_score(t[min2],FAIL);
                       storenewtemplate(t,nt,na,e);
                       adjust_attr_weight(t[min1],e,w,na,FAIL);
                }
              }
              else {
                     storenewtemplate(t,nt,na,e);
              adjust_attr_weight(t[min1],e,w,na,FAIL);
              }
   }
}

/* find the closest example to the avrg in n example */
find2template(fp,t,nt,n,na,avrg)
FILE *fp;
struct Template *t[];
int n,na,*nt;
float *avrg;
```

```
 {
 static struct Example exs[INITIALSETSIZE];
 static float sum[INITIALSETSIZE];
 int i,j,min1,min2;

        for (i=0; i<n; i++) {
               readexample(fp,&exs[i],na);
               sum[i]=0.0;
               for (j=0; j<na; j++)
                      sum[i] +=fabs((exs[i].a[j]-avrg[j])/(maxof[j]-minof[j]));
        }
        find_two_min(sum,n,&min1,&min2);
        if (min1>-1)
               storenewtemplate(t,nt,na,&exs[min1]);
        if (min2>-1)
               storenewtemplate(t,nt,na,&exs[min2]);
        rewind(fp);
 }

 trainall(fn,t,w,na,nt,ne)
 char *fn;
 struct Template *t[];
 float *w;
 int na,ne,*nt;
 {
 struct Example e;
 int i;
 FILE *fp;
        fp=fopen(fn,"r");
        if (fp==NULL)
               error_exit(1,"Training set not found...");
       findmx(fp,maxof,minof,avrgof,&nexample,na); /*find min max and avrg */
        ne=nexample;
        if (ne < setsize ) {
          printf("Too few (%d) examples in training set\n",ne);
          exit(3);
        }

        find2template(fp,t,nt,setsize,na,avrgof);

        for (i=0; i<ne; i++) {
               readexample(fp,&e,na);
               train(t,&e,w,na,nt);
        }
}
/* arrange the template so same class templates are in ct[class] */
rearrange(ct,t,nt,nc)
struct Classtemplate *ct;
struct Template *t[];
int nt,nc;
{
```

```
int i;
        for (i=0; i<nc; i++)
                ct[i].n=0;
        for (i=0; i<nt; i++)
                ct[t[i]->class].t[ct[t[i]->class].n++]=t[i];
}

/* sort for each class templates according to # correct prediction(ncor) */
sort(ct,nc)
struct Classtemplate *ct;
int nc;
{
int class,i,j;
struct Template *temp;
        for (class=0; class<nc; class++)
                for (i=0; i<ct[class].n-1; i++)
                        for (j=i+1; j<ct[class].n; j++)
                                if (ct[class].t[i]->ncor < ct[class].t[j]->ncor) {
                                        temp=ct[class].t[i];
                                        ct[class].t[i]=ct[class].t[j];
                                        ct[class].t[j]=temp;
                                }
}

/* specialize t1  */
interclassspecialize(t1,t2,na)
struct Template *t1,*t2;
int na;
{
int i,n=0;
float temp;
        for (i=0; i<na; i++)
                n +=inside(t2->a[i].x,t2->a[i].y,t1->a[i].x,t1->a[i].y);
        if (n==na) {  /* template2 is in template1 (if hole)*/
                t2->status=EXCEPTION;
                return;
        }
        n=0;
        for (i=0; i<na; i++)
                n +=inside(t1->a[i].x,t1->a[i].y,t2->a[i].x,t2->a[i].y);
        if (n==na) {  /* template1 is in template2 (if hole)*/
                t1->status=EXCEPTION;
                return;
        }
        for (i=0; i<na; i++) {
                if (overlaps(t2->a[i].x,t2->a[i].y,t1->a[i].x,t1->a[i].y)) {
                        if (t2->a[i].y> t1->a[i].x )
                                t1->a[i].y=t2->a[i].x;
                        else
                                t1->a[i].x=t2->a[i].y;
                        if (t1->a[i].y<t1->a[i].x) {
```

```
                                        temp=t1->a[i].y;
                                        t1->a[i].y=t1->a[i].x;
                                        t1->a[i].x=temp;
                                }
                        }
                }
        }

        /* specialze the template if overlap with the other class templates */
        interclassoverlaps(ct,nc,na)
        struct Classtemplate *ct;
        int nc,na;
        {
        int class,i,j,k,m;
        float c1,c2;      /* worng prediction ratio */
                for (class=0; class<nc; class++)
                        for (j=0; j<ct[class].n; j++)
                                for (k=0; k !=class &&k<nc; k++)
                                        for (m=0; m<ct[k].n; m++) {
                                                c1=(ct[class].t[j]->nref-ct[class].t[j]->ncor)/(ct[class].t[j]-
        >nref);
                                                c2=(ct[k].t[m]->nref-ct[k].t[m]->ncor)/(ct[k].t[m]->nref);
                                                 /* specialize t with greater wrong prediction ratio */
                                                if ( c1>c2)
                                                        interclassspecialize(ct[class].t[j],ct[k].t[m],na);
                                                else
                                                        interclassspecialize(ct[k].t[m],ct[class].t[j],na);
                                        }
        }

        /* check redundancy (inside and same class) */
        redundancy(ct,nc,na)
        struct Classtemplate *ct;
        int nc,na;
        {
        int c,i,j,k,n;
                for (c=0; c<nc; c++)
                        for (j=0; j<ct[c].n; j++) {
                         for (k=0; k<ct[c].n && k !=j; k++) {
                          if (ct[c].t[j]->status==VALID && ct[c].t[k]->status==VALID) {
                                n=0;
                                for (i=0; i<na; i++)
                                        n +=inside(ct[c].t[k]->a[i].x,ct[c].t[k]->a[i].y,ct[c].t[j]-
        >a[i].x,ct[c].t[j]->a[i].y);
                                if (n==na)  /* ct[c].t[k] in ct[c].t[j] (if redundant) */
                                        ct[c].t[k]->status=REDUNDANT;
                          }
                         }
                        }
        }
```

```
 same(x1,y1,x2,y2)
 float x1,y1,x2,y2;
 {
        if (x1==x2 && y1==y2)
                return(1);
        return(0);
 }

 /* before disjunct t[t2].a[n] with t[t1].a[n] check is allready in list */
 alreadyin(t,t1,t2,n)
 struct Template *t[];
 int t1,t2,n;
 {
 int i;
 float x,y;
 struct Template *temp;

        x=t[t2]->a[n].x;
        y=t[t2]->a[n].y;
        if (inside(x,y,t[t1]->a[n].x,t[t1]->a[n].y))
                return(1);
        else if (overlaps(t[t1]->a[n].x,t[t1]->a[n].y,x,y)) {
                if (t[t1]->a[n].x>x)
                        t[t1]->a[n].x=x;
                if (t[t1]->a[n].y<y)
                        t[t1]->a[n].y=y;
                return(1);
        }
        for (i=0; i<t[t1]->dsize[n]; i++) { /*check is already in disjucnt list*/
                temp=t[t[t1]->dlist[n][i]];
                if (inside(x,y,temp->a[n].x,temp->a[n].y))
                        return(1);
                else if (overlaps(temp->a[n].x,temp->a[n].y,x,y)) {
                        if (temp->a[n].x>x)
                                temp->a[n].x=x;
                        if (temp->a[n].y<y)
                                temp->a[n].y=y;
                        return(1);
                }
        }
        return(0);
}

/* find disjunctive generalization of t[t1] and t[t2] according to reliability ratio, generalize the
more reliable template and mark the other as invalid */
disjunctivegeneralization(t,t1,t2,na,ne)
struct Template *t[];
int t1,t2,na,ne;
{
int i,j,k;
float c1,c2;    /* reliability of the templates */
```

```
        c1=t[t1]->ncor/ne;
        c2=t[t2]->ncor/ne;
        /*j=more reliable template index, t[k] will disjunct with t[j]*/
        if (c1>c2) {
                k=t2;
                j=t1;
        }
        else {
                k=t1;
                j=t2;
        }
        for (i=0; i<na; i++) {
                if (inside(t[k]->a[i].x,t[k]->a[i].y,t[j]->a[i].x,t[j]->a[i].y))
                        continue;     /* if t[k]->a[i] in t[j]->a[i] */
                else if (inside(t[j]->a[i].x,t[j]->a[i].y,t[k]->a[i].x,t[k]->a[i].y)) {
                        t[j]->a[i].x=t[k]->a[i].x;
                        t[j]->a[i].y=t[k]->a[i].y;
                }
                else if (overlaps(t[j]->a[i].x,t[j]->a[i].y,t[k]->a[i].x,t[k]->a[i].y)) {
                        if (t[j]->a[i].x>t[k]->a[i].x)
                                t[j]->a[i].x=t[k]->a[i].x;
                        if (t[j]->a[i].y<t[k]->a[i].y)
                                t[j]->a[i].y=t[k]->a[i].y;
                }
/*              else if (! same(t[j]->a[i].x,t[j]->a[i].y,t[k]->a[i].x,t[k]->a[i].y))
                        t[j]->dlist[i][t[j]->dsize[i]++]=k;
*/
        else if (! alreadyin(t,j,k,i))
                        t[j]->dlist[i][t[j]->dsize[i]++]=k;

        }
        t[j]->nref +=t[k]->nref;
        t[j]->ncor +=t[k]->ncor;
        t[k]->status=INVALID;
}

inclassgeneralization(ct,nc,na,ne)
struct Classtemplate *ct;
int nc,na,ne;
{
int i,j,k;
        for (i=0; i<nc; i++)
                for (j=0; j<ct[i].n; j++)
                        for (k=0; k!=j && k<ct[i].n; k++)
                                if (ct[i].t[j]->status==VALID && ct[i].t[k]->status==VALID)
                                        disjunctivegeneralization(&(ct[i].t[0]),j,k,na,ne);
}

secondpass(ct,t,nt,nc,na,ne)
struct Classtemplate *ct;
struct Template *t[];
```

```
 int nt,nc,na,ne;
 {
 printf("# rules = %d\n",nt);
 printf("Second pass...\n");
        rearrange(ct,t,nt,nc);
        sort(ct,nc);
        interclassoverlaps(ct,nc,na);
        redundancy(ct,nc,na);
 /*
 tolerance=0.02;
 printf("Tolerance rate = %5.3f\n",tolerance);
        inclassgeneralization(ct,nc,na,ne);
 */
 }

 interprete(ct,w,nc,nt,na)
 struct Classtemplate *ct;
 float *w;
 int na,nc,nt;
 {
 int i,j,k=0,c,d;
 float f;
 struct Template *temp;
 FILE *fp, *fp2;  /* fp: for rules , fp2:for able like output */


      fp=fopen("rules","w");
      if (fp==NULL) error_exit(3,"Rule file creation failed");

       fp2=fopen("table","w");
       if (fp==NULL) error_exit(3,"Table file creation failed");

       fprintf(fp,"Weights\n");
       fprintf(fp2,"Weights\n");
       for (i=0; i<na; i++) {
       /*     w[i]=(INITIALWEIGHT-w[i])/INITIALWEIGHT; */ /* eksi yapmak icin bu
komenti ac */
              fprintf(fp,"%5.3f ",w[i]);
              fprintf(fp2,"%5.3f ",w[i]);
       }
       fprintf(fp,"\n");
       fprintf(fp2,"\n");
       /* produce table like output */
       fprintf(fp2,"    ");
       for (i=0; i<na; i++)
              fprintf(fp2,"%9c%d ",'F',i+1);
       fprintf(fp2,"\n");

       for (c=0,k=0; c<nc; c++) {
              for (i=0; i<ct[c].n; i++) {
                     if (ct[c].t[i]->status>EXCEPTION)/*skip redundant and invalids */
```

```
                        continue;
                    k++;
                    fprintf(fp2,"R%03d:",k);
                    f=(float)ct[c].t[i]->ncor/nexample;
                    for (j=0; j<na-1; j++)
                          fprintf(fp2,"%6.2f %6.2f",ct[c].t[i]->a[j].x,ct[c].t[i]->a[j].y);
                    fprintf(fp2," C%d CF=%6.4f(%2d/%2d)\n",ct[c].t[i]->class,f,ct[c].t[i]-
>ncor,ct[c].t[i]->nref);
                }
        }
        fclose(fp2);

        /* produce rules */
        for (c=0,k=0; c<nc; c++) {
                for (i=0; i<ct[c].n; i++) {
                    if (ct[c].t[i]->status>EXCEPTION)/*skip redundant and invalids */
                        continue;
                    k++;
                    f=(float)ct[c].t[i]->ncor/nexample;
                    fprintf(fp,"IF\n");
                    for (j=0; j<na-1; j++) {
                  /*  if (w[j] < -0.3)
                        continue;
                  */
                      if (ct[c].t[i]->a[j].x==ct[c].t[i]->a[j].y)
                              fprintf(fp,"(  %35s = %7.2f",anames[j],ct[c].t[i]->a[j].y);
                        else
                              fprintf(fp,"(  %7.2f < %25s < %7.2f",ct[c].t[i]-
>a[j].x,anames[j],ct[c].t[i]->a[j].y);
                        for (d=0; d<ct[c].t[i]->dsize[j]-1; d++) {
                              temp=ct[c].t[ct[c].t[i]->dlist[j][d]];
                              if (temp->a[j].x==temp->a[j].y)
                                    fprintf(fp,"\nOR %35s = %7.2f",anames[j],temp->a[j].y);
                              else
                                    fprintf(fp,"\nOR %7.2f < %25s < %7.2f",temp-
>a[j].x,anames[j],temp->a[j].y);
                        }
                        if (ct[c].t[i]->dsize[j]>0) {
                              temp=ct[c].t[ct[c].t[i]->dlist[j][d]];
                              if (temp->a[j].x==temp->a[j].y)
                                    fprintf(fp,"\nOR %35s = %7.2f ) &\n",anames[j],temp-
>a[j].y);
                              else
                                    fprintf(fp,"\nOR %7.2f < %25s < %7.2f ) &\n",temp-
>a[j].x,anames[j],temp->a[j].y);
                        }
                        else
                              fprintf(fp," ) & \n");

                    }
                /*  if (w[j] > -0.3) { */  /* skip negative attributes */
```

```
                    if (ct[c].t[i]->a[j].x==ct[c].t[i]->a[j].y)
                          fprintf(fp,"(  %35s = %7.2f",anames[j],ct[c].t[i]->a[j].y);
                    else
                          fprintf(fp,"(  %7.2f < %25s < %7.2f",ct[c].t[i]-
 >a[j].x,anames[j],ct[c].t[i]->a[j].y);
                    for (d=0; d<ct[c].t[i]->dsize[j]-1; d++) {
                          temp=ct[c].t[ct[c].t[i]->dlist[j][d]];
                          if (temp->a[j].x==temp->a[j].y)
                                fprintf(fp,"\nOR %35s = %7.2f",anames[j],temp->a[j].y);
                          else
                                fprintf(fp,"\nOR %7.2f < %25s < %7.2f",temp-
 >a[j].x,anames[j],temp->a[j].y);

                    }
                    if (ct[c].t[i]->dsize[j]>0) {
                          temp=ct[c].t[ct[c].t[i]->dlist[j][d]];
                          if (temp->a[j].x==temp->a[j].y)
                                fprintf(fp,"\nOR %35s = %7.2f ) \n",anames[j],temp->a[j].y);
                          else
                                fprintf(fp,"\nOR %7.2f < %25s < %7.2f ) \n",temp-
>a[j].x,anames[j],temp->a[j].y);
                    }
                    else
                          fprintf(fp," )\n");
              /* }   skip negative weights
                else fprintf(fp," )\n");
              */
                 fprintf(fp,"THEN\n");
                 fprintf(fp,"%s CF=%7.4f(%2d/%2d)\n",cnames[ct[c].t[i]->class],f,ct[c].t[i]-
>ncor,ct[c].t[i]->nref);
              }
        }
   fprintf(fp,"# rule=%d\n",k);
   fclose(fp);
}

bknowledge(fn)
char *fn;
{
FILE *fp;
int i;
char str[NAME_SIZE],*cp;
float arate;

       fp=fopen(fn,"r");
       if (fp==NULL)
             error_exit(1,"Background knowledge not found...");

       if (fscanf(fp,"%s%s",str,efilename) <2)
       error_exit(2,"Background knowledge has invalid format...");
       if (stricmp(str,"examplefile")!=0)
```

```
            error_exit(2,"Background knowledge has invalid format...");

        if (fscanf(fp,"%s%f",str,&arate) <2)
        error_exit(2,"Background knowledge has invalid format...");
        if (stricmp(str,"adjustmentrate")!=0)
            error_exit(2,"Background knowledge has invalid format...");
        if (arate>0)
            alpha=arate;
        else
            alpha=-arate;

        if (fscanf(fp,"%s%d",str,&nattr) < 2)
            error_exit(2,"Background knowledge has invalid format...");
        if (stricmp(str,"#attributes")!=0)
            error_exit(2,"Background knowledge has invalid format...");
        if (nattr>NATTRIBUTES)
            error_exit(3,"Too much attributes...");

        if (fscanf(fp,"%s%d",str,&nclass) < 2)
            error_exit(2,"Background knowledge has invalid format...");
        if (stricmp(str,"#classes")!=0)
            error_exit(2,"Background knowledge has invalid format...");
        if (nclass>NCLASSES)
            error_exit(3,"Too much classes...");

        if (fscanf(fp,"%s%d",str,&nexample) < 2)
            error_exit(2,"Background knowledge has invalid format...");
        if (stricmp(str,"#examples")!=0)
            error_exit(2,"Background knowledge has invalid format...");
        if (nexample <INITIALSETSIZE)
            error_exit(3,"Too few example size...");

        if (fscanf(fp,"%s%d",str,&i) <2)
        error_exit(2,"Background knowledge has invalid format...");
        if (stricmp(str,"initialsetsize")!=0)
            error_exit(2,"Background knowledge has invalid format...");
        if (i>1 && i<nexample)
            setsize=i;
    if (setsize>INITIALSETSIZE)
            setsize=INITIALSETSIZE;
        fscanf(fp,"%s",str);
        if (stricmp(str,"attributes")!=0)
            error_exit(2,"Background knowledge has invalid format...");

if (fgets(str,NAME_SIZE,fp)==NULL)  /* read the newline */
    error_exit(2,"Background knowledge has invalid format...");

    for (i=0; i<nattr; i++) {
      if (fgets(anames[i],NAME_SIZE,fp)==NULL)
            error_exit(2,"Background knowledge has invalid format...");
      cp=strchr(anames[i],'\n');
```

```
        if (cp !=NULL)
           *cp='\0';
       }
 /* if (fscanf(fp,"%s%s%f%f%f",anames[i],atype[i],&minof[i],&maxof[i],&avrgof[i]) < 5) */

       fscanf(fp,"%s",str);
       if (stricmp(str,"classes")!=0)
             error_exit(2,"Background knowledge has invalid format...");
  if (fgets(str,NAME_SIZE,fp)==NULL)   /* read the newline */
       error_exit(2,"Background knowledge has invalid format...");

     for (i=0; i<nclass; i++) {
         if (fgets(cnames[i],NAME_SIZE,fp)==NULL)
              error_exit(2,"Background knowledge has invalid format...");
        cp=strchr(cnames[i],'\n');
        if (cp !=NULL)
           *cp='\0';
      }
/*   if (fscanf(fp,"%s",cnames[i]) < 1) */

      fclose(fp);
}

main(argc,argv)
int argc;
char *argv[];
{
int i;
float x;
      if (argc<2)
        error_exit(4,"Usage: nge3 bknowledgefile  [adjustment rate +specialize [data file]]");
       ntemplate=0;
       bknowledge(argv[1]);

      if (argc>2) {
         x=atof(argv[2]);
         if (x>0.0)
           alpha=x;
      }
      if (argc>3) {
        if (argv[3][0]=='+' && argv[3][1]=='s')
          Specialize=1;
        else
          Specialize=0;
      }
      if (argc>4)
        strcpy(efilename,argv[4]);

printf("Training file %s\n",efilename);

      for (i=0; i<nattr; i++)
```

```
                aw[i]=INITIALWEIGHT;
tolerance=0.0;
printf("First pass...\n");
printf("Tolerance = %5.3f Feature adjustment Rate = %5.4f\n",tolerance,alpha);
printf("Specialize ");
if (Specialize==1)
  printf("ON\n");
else
  printf("OFF\n");
        trainall(efilename,templates,aw,nattr,&ntemplate,nexample);
        secondpass(ctemplate,templates,ntemplate,nclass,nattr,nexample);
        interprete(ctemplate,aw,nclass,ntemplate,nattr);
        return(0);
}
```

APPENDIX C

INPUT DATA

There are 395 examples, starting from Jan. 1, 1991 to June 30, 1992, obtained from Capital Market Board of Turkey monthly bulletins. Each row represents a combination of an example. Columns indicate the % change in value of a feature on that day relative to the previous day. There are six features used in the design of this software. The first column represents the % change in the value of the first feature which is Istanbul, Tahtakale closing selling price of Turkish Republic Gold Coin, similarly the second column is for one Once Of Gold, third is for Central Bank effective selling price of US Dollars, fourth is for Central Bank effective selling price of Deutsche Mark, fifth is for the Government Bond 3 month Interest Rate, and sixth is for the 3 month average deposit rate of Commercial Banks (Is Bank, Akbank, Yapi Kredi Bank, Ziraat Bank) in Turkey. The last column represents the class of that example:

"1" means Class Type is one (C_1) and the ISE composite index for that day increased more than 3%.

"2" means Class Type is two (C_2) and the ISE composite index for that day decreased more than 3%.

"3" means Class Type is three (C_3) and the ISE composite index for that day did not change (between -3% to +3%).

DATE	Turkish Republic Gold Coin	One Ounce of Gold	US Dollar	Deutsche Mark	Govermnt. Bond Interest Rate	Commer. Bank Deposit Rate	Class
02-Jan-91	0.816327	1.364257	0.507614	1.946721	0.000000	0.000000	1
03-Jan-91	2.024292	1.345895	0.168350	0.000000	0.000000	0.000000	2
04-Jan-91	-0.396825	0.929615	0.840336	-0.753769	0.000000	0.000000	3
07-Jan-91	1.992032	1.842105	1.500000	0.506329	0.000000	0.000000	3
08-Jan-91	0.781250	0.387597	0.656814	0.604534	1.125643	0.431034	2
09-Jan-91	0.775194	-0.643501	0.489396	1.251878	0.000000	0.000000	1
10-Jan-91	1.730769	1.489637	1.379870	0.741840	0.000000	0.000000	3
11-Jan-91	1.701323	1.467773	1.361089	0.736377	0.000000	0.000000	3
14-Jan-91	0.371747	-0.251572	-0.947867	-0.584795	0.000000	0.000000	3
15-Jan-91	0.000000	0.252207	-0.861244	-0.588235	0.000000	0.000000	1
16-Jan-91	0.000000	-3.647799	-0.257400	0.345168	0.000000	0.000000	1
17-Jan-91	-3.703704	0.000000	0.000000	0.000000	0.000000	0.000000	3
18-Jan-91	-2.692308	-2.610966	-2.096774	-0.737101	0.000000	0.000000	2
21-Jan-91	0.000000	0.402145	0.263591	1.138614	0.000000	0.000000	3
22-Jan-91	1.581028	0.133511	0.558659	0.440529	2.662457	0.000000	3
23-Jan-91	-0.389105	0.000000	0.000000	0.633528	0.000000	0.000000	1
24-Jan-91	-0.781250	-0.666667	0.980392	0.242131	0.000000	0.000000	1
25-Jan-91	0.393701	0.134228	-0.485437	0.000000	0.000000	0.000000	3
28-Jan-91	0.000000	1.206434	0.487805	0.821256	0.000000	0.000000	3
29-Jan-91	0.784314	-0.529801	0.388350	-0.431241	6.930400	0.000000	1
30-Jan-91	-0.778210	-0.932091	0.257898	0.433109	0.000000	0.000000	1
31-Jan-91	-0.784314	0.000000	0.000000	0.000000	0.000000	0.000000	1
01-Feb-91	0.000000	-0.806452	-0.128617	1.437470	0.000000	0.000000	1
04-Feb-91	-0.790514	0.542005	0.450741	0.614076	0.000000	0.000000	3
05-Feb-91	0.000000	-0.539084	0.641026	1.173709	1.795016	0.858368	3
06-Feb-91	-0.398406	0.271003	0.159236	0.464037	0.000000	0.000000	3
07-Feb-91	2.400000	1.891892	0.476948	0.461894	0.000000	0.000000	1
08-Feb-91	-2.343750	0.000000	0.000000	0.229885	0.000000	0.000000	3
11-Feb-91	2.400000	-1.061008	-0.158228	0.688073	0.000000	0.000000	3
12-Feb-91	0.000000	1.072386	0.887480	0.227790	0.000000	0.221283	3
13-Feb-91	0.000000	0.530504	0.282752	0.000000	0.000000	0.000000	3
14-Feb-91	-0.390625	0.000000	1.190476	0.227273	0.000000	0.000000	1
15-Feb-91	0.000000	0.263852	0.154799	0.000000	0.000000	0.000000	3
18-Feb-91	0.784314	1.315789	1.081917	0.317460	0.000000	0.000000	3
19-Feb-91	-0.389105	0.259740	-0.152905	-0.542495	0.000000	0.000000	3
20-Feb-91	0.390625	0.906736	1.531394	0.681818	0.000000	0.000000	3
21-Feb-91	3.112840	2.182285	3.016591	2.257336	0.000000	0.000000	3
22-Feb-91	0.377358	0.879397	2.928258	2.649007	0.000000	0.000000	3
25-Feb-91	4.511278	0.871731	0.995733	1.290323	0.000000	0.000000	2
26-Feb-91	-2.877698	0.493827	0.563380	-0.212314	6.838062	5.298912	2
27-Feb-91	1.111111	1.719902	0.140056	0.425532	0.000000	0.000000	3
28-Feb-91	0.366300	0.000000	0.000000	0.000000	0.000000	0.000000	2
01-Mar-91	-3.284672	-4.106280	-2.321678	-5.720339	0.000000	0.000000	2
04-Mar-91	0.000000	-0.125945	0.000000	0.134831	0.000000	0.000000	1

05-Mar-91	-1.132075 1.261034 -1.918671 0.000000 3.238003 0.403225 3
06-Mar-91	1.145038 -0.124533 -0.145985 -0.089767 0.000000 0.000000 3
07-Mar-91	0.000000 0.000000 -0.146199 -0.943396 0.000000 0.000000 3
08-Mar-91	1.886792 1.995013 1.317716 -0.226757 0.000000 0.000000 3
11-Mar-91	0.925926 -0.855746 -0.144509 0.000000 0.000000 0.000000 3
12-Mar-91	-1.284404 0.246609 0.434153 0.454545 -2.721504 0.000000 3
13-Mar-91	0.371747 1.107011 0.288184 0.814480 0.000000 0.000000 3
14-Mar-91	1.111111 0.000000 0.431034 -1.032316 0.000000 0.000000 3
15-Mar-91	0.366300 0.729927 1.001431 0.226757 0.000000 0.000000 3
18-Mar-91	0.000000 0.845411 2.549575 0.452489 0.000000 0.000000 3
19-Mar-91	1.459854 3.952096 2.762431 2.252252 0.000000 0.000000 3
20-Mar-91	3.597122 3.686636 3.494624 3.964758 0.000000 0.000000 3
21-Mar-91	4.166667 0.000000 -3.636364 -4.237288 0.000000 0.000000 3
22-Mar-91	-4.000000 -5.111111 0.539084 -1.769912 0.000000 0.000000 3
25-Mar-91	0.000000 2.107728 4.021448 2.702703 0.000000 0.000000 3
26-Mar-91	-1.388889 -1.146789 -2.319588 -1.315789 0.614726 0.000000 3
27-Mar-91	1.760563 9.048724 3.430079 1.777778 0.000000 0.000000 3
28-Mar-91	-0.692042 0.000000 -1.913265 -1.310044 0.000000 0.000000 2
29-Mar-91	-0.348432 0.000000 -0.520156 -0.221239 0.000000 0.000000 3
01-Apr-91	-0.349650 -6.702127 2.483660 2.882483 0.000000 0.000000 3
02-Apr-91	0.350877 -0.684151 -0.255102 -0.431034 0.000000 1.606424 3
03-Apr-91	0.699301 -0.114811 -0.767263 -0.432900 0.000000 0.000000 3
04-Apr-91	0.000000 0.000000 -0.644330 0.217391 0.000000 0.000000 3
05-Apr-91	2.430556 2.068965 0.389105 -0.650759 0.000000 0.000000 3
08-Apr-91	0.000000 2.139640 0.129199 -0.218341 0.000000 0.000000 3
09-Apr-91	1.694915 -1.433297 -1.419355 0.437637 3.354114 0.000000 3
10-Apr-91	-1.000000 -2.013423 -0.785340 -1.742919 0.000000 0.000000 3
11-Apr-91	2.693603 4.794520 1.055409 1.552106 0.000000 0.000000 3
12-Apr-91	0.000000 0.000000 0.000000 0.436681 0.000000 0.000000 3
15-Apr-91	0.000000 0.000000 0.000000 0.217391 0.000000 0.000000 3
16-Apr-91	0.000000 0.000000 0.000000 0.108460 0.000000 0.000000 3
17-Apr-91	0.000000 0.000000 0.000000 0.054171 0.000000 0.000000 3
18-Apr-91	0.000000 0.000000 0.000000 0.027282 0.000000 0.000000 3
19-Apr-91	0.000000 0.000000 3.133159 0.026852 0.000000 0.000000 3
22-Apr-91	0.983607 1.960784 1.898734 -0.432900 0.000000 0.000000 3
23-Apr-91	0.649351 -0.427350 0.000000 0.434783 1.906141 0.000000 3
24-Apr-91	-0.322581 0.214592 0.496894 0.649351 0.000000 0.000000 3
25-Apr-91	-0.323625 -0.214133 0.000000 0.000000 0.000000 0.000000 2
26-Apr-91	0.649351 0.858369 1.483313 -0.215054 0.000000 0.000000 2
29-Apr-91	0.645161 -1.276596 -1.339829 0.000000 0.000000 0.000000 1
30-Apr-91	-2.243590 -1.077586 -2.222222 0.431034 -1.858646 0.000000 1
01-May-91	0.000000 0.217865 1.010101 0.643777 0.000000 0.000000 3
02-May-91	0.819672 0.434783 0.375000 -0.213220 0.000000 0.000000 2
03-May-91	0.813008 0.432900 0.373599 -0.213675 0.000000 0.000000 3
06-May-91	0.000000 0.431034 0.198511 0.000000 0.000000 0.000000 3
07-May-91	-0.645161 0.214592 0.173353 0.856531 0.000000 2.766796 3
08-May-91	0.000000 0.642398 0.370828 -0.764331 0.000000 0.000000 3
09-May-91	0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 3
10-May-91	0.974026 -0.638298 -0.492611 0.556269 0.000000 0.000000 3

13-May-91	0.000000 0.214133 0.247525 0.425532 0.000000 0.000000 3
14-May-91	0.000000 0.106838 -0.370370 0.635593 0.000000 2.692314 3
15-May-91	0.321543 0.213447 0.371747 0.631579 0.000000 0.000000 3
16-May-91	0.000000 0.106496 0.123457 -0.627615 0.000000 0.000000 1
17-May-91	0.000000 -0.106383 0.863132 -1.263158 0.000000 0.000000 1
20-May-91	0.000000 1.064963 0.366748 1.407249 0.000000 0.000000 3
21-May-91	0.000000 -0.105374 -0.121803 0.925147 -6.513875 -3.745324 3
22-May-91	0.000000 0.105485 0.487805 -0.208333 0.000000 0.000000 3
23-May-91	0.641026 -0.526870 -0.606796 0.626305 0.000000 0.000000 2
24-May-91	-0.318471 -0.105932 -1.953602 -0.414938 0.000000 0.000000 2
27-May-91	0.319489 0.212089 1.992528 -0.208333 0.000000 0.000000 3
28-May-91	0.000000 0.317460 -0.488400 0.000000 4.503227 -4.669256 3
29-May-91	0.000000 0.421941 0.245399 -0.208768 0.000000 0.000000 3
30-May-91	0.636943 0.000000 0.489596 0.000000 0.000000 0.000000 3
31-May-91	0.632911 1.260504 0.487211 -0.418410 0.000000 0.000000 3
03-Jun-91	0.943396 0.414938 0.363636 0.000000 0.000000 0.000000 3
04-Jun-91	0.000000 0.413223 0.362319 0.000000 0.000000 -0.408163 3
05-Jun-91	0.623053 0.617284 0.361011 0.000000 0.000000 0.000000 3
06-Jun-91	0.000000 1.635992 0.239808 -0.420168 0.000000 0.000000 3
07-Jun-91	2.476780 0.301811 0.239234 0.000000 0.000000 0.000000 3
10-Jun-91	0.000000 0.000000 0.357995 0.421941 0.000000 0.000000 3
11-Jun-91	1.812689 1.604814 0.237812 -0.210084 -4.296826 0.000000 3
12-Jun-91	3.857567 -0.296150 0.948992 0.631579 0.000000 0.000000 3
13-Jun-91	0.000000 0.000000 0.470035 -0.418410 0.000000 0.000000 3
14-Jun-91	-3.714286 0.693069 0.000000 0.000000 0.000000 0.000000 2
17-Jun-91	0.000000 0.983284 0.818713 0.420168 0.000000 0.000000 2
18-Jun-91	1.780415 0.389484 -1.972158 -0.125523 0.232228 0.000000 3
19-Jun-91	0.000000 0.000000 1.775148 0.963553 0.000000 0.000000 3
20-Jun-91	-0.437318 -0.096993 0.348837 0.000000 0.000000 0.000000 3
21-Jun-91	-0.219619 -0.048544 0.173812 0.000000 0.000000 0.000000 3
24-Jun-91	-0.110051 -0.024284 0.086755 0.000000 0.000000 0.000000 3
25-Jun-91	-0.055086 -0.012145 0.043453 0.000000 0.000000 0.000000 3
26-Jun-91	-0.055117 -0.012146 0.043208 0.000000 -0.798046 0.000000 3
27-Jun-91	1.764706 1.068999 0.461894 -0.207469 0.000000 0.000000 3
28-Jun-91	-0.289017 0.192308 0.344828 0.498960 0.000000 0.000000 3
01-Jul-91	0.289855 0.287908 0.801833 -0.289615 0.000000 0.000000 3
02-Jul-91	0.000000 0.574163 0.568182 0.414938 0.000000 0.000000 3
03-Jul-91	1.156069 0.475737 0.112994 -0.413223 0.000000 0.000000 1
04-Jul-91	0.857143 -0.094697 0.000000 1.037344 0.000000 0.000000 3
05-Jul-91	0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 3
08-Jul-91	0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 3
09-Jul-91	0.000000 0.473934 0.112867 0.000000 0.583889 -0.409836 3
10-Jul-91	1.133144 -0.283019 0.000000 0.000000 0.000000 0.000000 3
11-Jul-91	-0.700280 -0.236518 -0.281849 0.410678 0.000000 0.000000 3
12-Jul-91	-0.705219 -0.237079 -0.282646 0.408998 0.000000 0.000000 3
15-Jul-91	0.000000 0.380228 0.453515 0.407332 0.000000 0.000000 3
16-Jul-91	0.284091 0.473485 0.451467 0.486815 3.044373 0.411522 3
17-Jul-91	0.566572 0.471254 0.224719 0.928543 0.000000 0.000000 3
18-Jul-91	0.281690 0.000000 -0.448430 0.800000 0.000000 0.000000 2

19-Jul-91	-0.280899 -0.281426 -0.225225 0.000000 0.000000 0.000000 2
22-Jul-91	0.281690 -0.564440 0.112867 -0.396825 0.000000 0.000000 3
23-Jul-91	0.000000 -0.473037 -0.112740 0.597610 4.844765 2.180328 3
24-Jul-91	-1.685393 -0.760456 -0.225734 0.198020 0.000000 0.000000 3
25-Jul-91	0.000000 0.000000 0.000000 0.395257 0.000000 0.000000 2
26-Jul-91	0.000000 0.287356 0.113122 -0.393701 0.000000 0.000000 2
29-Jul-91	0.000000 -1.050621 -0.225989 0.395257 0.000000 0.000000 1
30-Jul-91	0.000000 0.096525 0.113250 -0.196850 0.000000 0.160434 1
31-Jul-91	-0.857143 0.964320 0.226244 0.197239 0.000000 0.000000 3
01-Aug-91	-0.576369 0.000000 0.225734 0.000000 0.000000 0.000000 3
02-Aug-91	0.000000 -2.196753 -0.112613 0.590551 0.000000 0.000000 3
05-Aug-91	0.000000 0.097656 -0.112740 0.978474 0.000000 0.000000 3
06-Aug-91	0.000000 -0.390244 0.000000 0.193798 0.000000 -0.160177 3
07-Aug-91	0.000000 0.097943 0.000000 0.386847 0.000000 0.000000 3
08-Aug-91	0.000000 0.000000 0.564334 -0.192678 0.000000 0.000000 1
09-Aug-91	0.869565 1.565558 0.224467 0.000000 0.000000 0.000000 3
12-Aug-91	-0.287356 0.481696 1.119821 1.158301 0.000000 0.000000 3
13-Aug-91	0.288184 0.862895 0.885936 0.190840 0.000000 0.112305 3
14-Aug-91	0.000000 0.380228 -0.109769 -0.190476 0.000000 0.000000 3
15-Aug-91	2.873563 1.704545 1.978022 1.335878 0.000000 0.000000 2
16-Aug-91	0.558659 3.538175 1.939655 -1.318267 0.000000 0.000000 3
19-Aug-91	2.777778 -0.359712 1.057082 0.763359 0.000000 0.000000 3
20-Aug-91	-0.945946 -1.579422 -0.941423 0.000000 2.137319 0.048075 1
21-Aug-91	-0.954980 -1.604768 -0.844773 0.189394 0.000000 0.000000 3
22-Aug-91	-1.928375 -0.745573 -1.384452 0.945180 0.000000 0.000000 3
23-Aug-91	0.280899 0.751174 0.647948 0.000000 0.000000 0.000000 3
26-Aug-91	0.000000 -0.093197 0.214592 0.000000 0.000000 0.000000 2
27-Aug-91	0.280112 0.279851 0.428266 0.187266 2.606961 -0.048052 3
28-Aug-91	0.558659 0.465116 0.106610 0.186916 0.000000 0.000000 3
29-Aug-91	-1.111111 -1.203704 -0.532481 0.373134 0.000000 0.000000 3
30-Aug-91	-0.280899 -0.374883 0.321199 0.000000 0.000000 0.000000 3
02-Sep-91	-0.281690 -0.470367 -0.213447 -0.185874 0.000000 0.000000 3
03-Sep-91	0.282486 0.472590 0.962567 1.117318 0.000000 -0.112179 2
04-Sep-91	0.281690 0.094073 0.000000 -0.184162 0.000000 0.000000 3
05-Sep-91	0.000000 0.281955 0.529661 0.553506 0.000000 0.000000 3
06-Sep-91	0.561798 0.374883 -0.316122 0.917431 0.000000 0.000000 3
09-Sep-91	-0.558659 -0.560224 -0.422833 0.000000 0.000000 0.000000 1
10-Sep-91	0.561798 0.657277 0.424628 1.454545 0.000000 4.042998 2
11-Sep-91	-1.396648 -1.865672 0.000000 -0.358423 0.000000 0.000000 2
12-Sep-91	-0.849858 -0.665399 -0.634249 0.000000 0.000000 0.000000 3
13-Sep-91	0.285714 0.287081 -0.106383 0.359712 0.000000 0.000000 3
16-Sep-91	0.569801 0.572519 0.425985 0.716846 0.000000 0.000000 3
17-Sep-91	0.000000 -0.189753 -0.106045 0.355872 0.000000 5.011560 3
18-Sep-91	1.416431 1.140684 1.167728 0.177305 0.000000 0.000000 3
19-Sep-91	0.279330 0.469925 -0.629591 0.000000 0.000000 0.000000 1
20-Sep-91	0.278552 0.467727 0.316790 0.000000 0.000000 0.000000 3
23-Sep-91	0.000000 -0.093110 0.105263 0.707965 0.000000 0.000000 3
24-Sep-91	0.555556 0.279590 -0.315457 -0.175747 0.000000 2.422907 3
25-Sep-91	-0.276243 0.000000 -0.316456 -0.704225 0.000000 0.000000 3

26-Sep-91	0.000000 0.371747 0.105820 0.177305 0.000000 0.000000 3
27-Sep-91	-1.939058 -1.018519 -0.211416 0.176991 0.000000 0.000000 3
30-Sep-91	1.694915 0.467727 -0.317797 0.000000 0.000000 0.000000 2
01-Oct-91	0.555556 0.744879 0.743889 0.353357 0.000000 0.000000 3
02-Oct-91	0.828729 1.109057 0.210970 0.352113 0.000000 0.000000 3
03-Oct-91	0.000000 0.182815 0.210526 0.350877 0.000000 0.000000 3
04-Oct-91	1.369863 1.459854 0.735294 0.524476 0.000000 0.000000 3
07-Oct-91	0.000000 0.269784 0.312826 0.000000 0.000000 0.000000 3
08-Oct-91	0.810811 0.179372 0.831601 0.347826 -1.150729 0.000000 2
09-Oct-91	0.268097 0.268577 0.309278 -0.693241 0.000000 0.000000 3
10-Oct-91	0.267380 0.446429 0.000000 0.523560 0.000000 0.000000 3
11-Oct-91	0.800000 0.800000 0.205550 0.173611 0.000000 0.000000 1
14-Oct-91	1.058201 0.705467 0.717949 0.866551 0.000000 0.000000 3
15-Oct-91	-0.261780 0.000000 0.407332 -0.343643 -2.639470 0.286743 3
16-Oct-91	-0.262467 -0.437828 0.405680 0.000000 0.000000 0.000000 1
17-Oct-91	0.789474 0.615655 0.000000 0.172414 0.000000 0.000000 3
18-Oct-91	1.827676 1.748252 0.606061 2.065404 0.000000 0.000000 3
21-Oct-91	-1.025641 0.429553 -0.100402 -0.337268 0.000000 0.000000 1
22-Oct-91	1.036269 -0.598802 -0.301508 -0.846024 -1.479814 0.428872 3
23-Oct-91	0.000000 0.000000 0.000000 -0.511945 0.000000 0.000000 3
24-Oct-91	-0.512821 -0.172117 0.100806 0.343053 0.000000 0.000000 3
25-Oct-91	0.515464 0.258621 0.402820 0.341880 0.000000 0.000000 3
28-Oct-91	0.256410 0.257954 0.401204 0.000000 0.000000 0.000000 3
29-Oct-91	0.000000 0.000000 0.000000 0.000000 0.000000 -0.241991 3
30-Oct-91	-1.023018 -1.114923 -0.699301 0.511073 0.000000 0.000000 3
31-Oct-91	-0.775194 -0.607112 0.000000 0.338983 0.000000 0.000000 3
01-Nov-91	-1.041667 -0.698080 -0.201207 1.351351 0.000000 0.000000 3
04-Nov-91	0.526316 0.000000 0.000000 0.500000 0.000000 0.000000 3
05-Nov-91	0.000000 0.175747 0.100806 -0.165837 0.000000 -0.085614 3
06-Nov-91	-0.523560 -0.701754 -0.100705 0.498339 0.000000 0.000000 3
07-Nov-91	0.526316 0.530035 0.201613 0.165289 0.000000 0.000000 3
08-Nov-91	0.000000 -0.175747 0.201207 0.000000 0.000000 0.000000 3
11-Nov-91	0.523560 0.704225 0.401606 0.330033 0.000000 0.000000 3
12-Nov-91	-0.260417 -0.174825 -0.300000 0.328947 0.000000 0.000000 1
13-Nov-91	0.783290 0.612960 0.200602 0.163934 0.000000 0.000000 1
14-Nov-91	0.000000 0.174064 0.200200 0.654664 0.000000 0.000000 1
15-Nov-91	0.777202 0.781929 -0.099900 0.813008 0.000000 0.000000 3
18-Nov-91	0.000000 -0.172414 -0.300000 0.161290 0.000000 0.000000 1
19-Nov-91	1.028278 0.863558 0.401204 0.322061 1.802450 0.000000 3
20-Nov-91	-0.254453 0.000000 -0.299700 0.000000 0.000000 0.000000 1
21-Nov-91	1.020408 1.027397 0.501002 0.642055 0.000000 0.000000 1
22-Nov-91	-0.252525 0.169492 -0.598205 0.000000 0.000000 0.000000 1
25-Nov-91	0.000000 0.000000 -0.200602 0.318979 0.000000 0.000000 3
26-Nov-91	0.000000 0.084602 0.402010 -0.158983 0.000000 0.000000 2
27-Nov-91	0.506329 0.760778 0.800801 -0.159236 0.000000 0.000000 3
28-Nov-91	0.251889 0.167785 0.397219 0.000000 0.000000 0.000000 1
29-Nov-91	1.507538 1.088777 0.890208 0.159490 0.000000 0.000000 3
02-Dec-91	-0.990099 -0.414250 -0.882353 0.000000 0.000000 0.000000 3
03-Dec-91	0.000000 -0.166389 0.692384 0.318471 0.000000 0.000000 3

04-Dec-91	0.500000 0.250000 0.196464 0.793651 0.000000 0.000000 3
05-Dec-91	0.000000 0.083126 0.000000 -2.362205 0.000000 0.000000 3
06-Dec-91	0.497512 0.415282 -0.196078 4.193548 0.000000 0.000000 2
09-Dec-91	0.742574 0.744417 0.098232 0.309598 0.000000 0.000000 3
10-Dec-91	-0.245700 -0.246305 0.098135 0.154321 0.000000 0.000000 3
11-Dec-91	0.000000 0.082305 0.588235 0.000000 0.000000 0.000000 3
12-Dec-91	-0.738916 -0.822368 -0.389864 0.000000 0.000000 0.000000 3
13-Dec-91	-0.744417 -1.160862 0.293542 0.000000 0.000000 0.000000 2
16-Dec-91	0.000000 0.167785 0.195122 0.462250 0.000000 0.000000 3
17-Dec-91	0.000000 -0.837521 -0.097371 0.460123 -2.042015 -0.028568 1
18-Dec-91	-0.250000 -0.337838 -0.487329 -0.152672 0.000000 0.000000 3
19-Dec-91	0.250627 0.338983 0.293830 0.305810 0.000000 0.000000 1
20-Dec-91	-0.500000 -0.422297 -0.390625 1.219512 0.000000 0.000000 2
23-Dec-91	0.000000 -0.084818 -0.490196 0.753012 0.000000 0.000000 3
24-Dec-91	-0.502513 0.000000 0.000000 0.000000 0.000000 0.000000 3
25-Dec-91	0.000000 0.084890 0.000000 0.298954 0.000000 0.000000 3
26-Dec-91	-0.252525 -0.424088 0.000000 0.298063 0.000000 0.000000 3
27-Dec-91	-0.759494 -0.766610 0.098522 -0.148588 0.000000 0.000000 3
30-Dec-91	-0.510204 -0.429185 -0.196850 -0.297619 0.000000 0.000000 3
31-Dec-91	1.025641 1.034483 0.788955 0.746269 0.445832 0.000000 3
02-Jan-92	0.000000 -0.511945 0.782779 0.740741 0.000000 0.000000 3
03-Jan-92	1.522843 1.715266 1.165049 0.000000 0.000000 0.000000 3
06-Jan-92	-0.750000 -0.758853 -0.383877 0.294118 0.000000 0.000000 1
07-Jan-92	0.000000 -0.084962 0.385356 0.879765 0.000000 0.000000 3
08-Jan-92	0.503778 0.340136 0.383877 1.017442 0.000000 0.000000 3
09-Jan-92	0.250627 0.677966 0.191205 -0.431655 0.000000 0.000000 3
10-Jan-92	4.000000 4.208754 2.671756 -0.289017 0.000000 0.000000 2
13-Jan-92	-1.201923 -0.646204 0.371747 0.000000 0.000000 0.000000 1
14-Jan-92	-0.243309 -0.650406 -0.555556 -0.724638 -1.703452 0.000000 1
15-Jan-92	1.219512 2.127660 2.234637 -0.291971 0.000000 0.000000 3
16-Jan-92	0.722892 0.641026 0.364299 0.292826 0.000000 0.000000 2
17-Jan-92	0.478469 0.159236 0.000000 1.021898 0.000000 0.000000 3
20-Jan-92	0.000000 0.476948 -0.453721 -0.433526 0.000000 0.000000 3
21-Jan-92	0.000000 -0.316456 -0.182315 0.145138 0.000000 -0.742853 3
22-Jan-92	0.476190 0.317460 0.273973 0.579710 0.000000 0.000000 1
23-Jan-92	-0.236967 0.000000 0.455373 -0.144092 0.000000 0.000000 3
24-Jan-92	0.237530 0.158228 0.271986 0.432900 0.000000 0.000000 3
27-Jan-92	0.473934 0.631912 0.542495 -0.287356 0.000000 0.000000 3
28-Jan-92	0.235849 0.313972 0.179856 0.000000 1.440077 0.000000 3
29-Jan-92	-0.470588 -0.782473 -0.448833 0.288184 0.000000 0.000000 3
30-Jan-92	0.000000 0.236593 0.180343 -0.287356 0.000000 0.000000 2
31-Jan-92	0.236407 0.314713 0.405041 0.144092 0.000000 0.000000 2
03-Feb-92	0.235849 0.313726 0.403407 0.143885 0.000000 0.000000 2
04-Feb-92	0.235294 0.234558 0.000000 0.431034 4.439363 -0.359815 3
05-Feb-92	-0.234742 -0.468019 -0.446429 0.286123 0.000000 0.000000 1
06-Feb-92	0.000000 0.156740 -0.089686 0.427960 0.000000 0.000000 3
07-Feb-92	-0.235294 -0.469484 -0.359066 0.142045 0.000000 0.000000 2
10-Feb-92	0.000000 0.000000 0.180180 0.567376 0.000000 0.000000 3
11-Feb-92	0.707547 0.864780 0.809353 0.141044 -6.105284 -0.288897 3

12-Feb-92	0.702576	0.701481	0.356824	-0.281690	0.000000	0.000000	3
13-Feb-92	1.627907	1.702786	1.333333	0.282486	0.000000	0.000000	3
14-Feb-92	0.000000	0.000000	0.526316	0.281690	0.000000	0.000000	2
17-Feb-92	0.457666	0.456621	0.698080	0.280899	0.000000	0.000000	3
18-Feb-92	0.911162	1.212121	1.213172	-0.140056	0.000000	0.362161	2
19-Feb-92	0.451467	0.000000	0.342466	0.000000	0.000000	0.000000	3
20-Feb-92	5.168540	0.748503	0.682594	0.140252	0.000000	0.000000	2
21-Feb-92	-4.059829	0.148588	0.338983	0.560224	0.000000	0.000000	3
24-Feb-92	-0.445434	2.077151	0.168919	-0.139276	0.000000	0.000000	3
25-Feb-92	0.671141	-2.325581	0.168634	0.278940	2.134703	0.000000	3
26-Feb-92	0.000000	0.297619	1.010101	0.417246	0.000000	0.000000	3
27-Feb-92	0.444444	0.445104	-0.500000	0.554017	0.000000	0.000000	2
28-Feb-92	0.000000	-0.147710	-0.670017	-0.275482	0.000000	0.000000	3
02-Mar-92	-0.221239	-0.591716	0.000000	0.138122	0.000000	0.000000	1
03-Mar-92	0.886918	1.190476	1.011804	0.137931	4.276277	0.000000	3
04-Mar-92	0.219780	0.147059	0.667780	0.137741	0.000000	0.000000	3
05-Mar-92	0.877193	0.734214	1.160862	0.137552	0.000000	0.000000	3
06-Mar-92	0.434783	0.291545	-0.491803	-0.137363	0.000000	0.000000	3
09-Mar-92	-0.432900	-0.581395	0.164745	0.275103	0.000000	0.000000	3
10-Mar-92	0.000000	0.292398	0.493421	0.685871	-6.139842	0.000000	3
11-Mar-92	1.086957	1.020408	1.063830	0.408719	0.000000	0.000000	3
12-Mar-92	-0.430108	-0.432900	-0.485830	-0.271370	0.000000	0.000000	3
13-Mar-92	0.000000	-0.434783	-0.081367	0.000000	0.000000	0.000000	3
16-Mar-92	0.215983	0.145560	0.325733	0.408163	0.000000	0.000000	1
17-Mar-92	-0.862069	-1.526163	0.243506	0.948510	0.000000	0.000000	3
18-Mar-92	0.000000	0.369004	0.000000	0.536913	0.000000	0.000000	3
19-Mar-92	-0.434783	-0.441176	0.890688	-0.133511	0.000000	0.000000	3
20-Mar-92	1.091703	1.624815	0.963082	0.802139	0.000000	0.000000	3
23-Mar-92	0.431965	0.290698	0.635930	0.795756	0.000000	0.000000	3
24-Mar-92	0.215054	0.144928	0.000000	0.000000	2.172307	0.000000	1
25-Mar-92	0.000000	0.144718	-0.315956	0.263158	0.000000	0.000000	3
26-Mar-92	0.000000	0.144509	-0.158479	0.000000	0.000000	0.000000	3
27-Mar-92	-0.858369	-0.721501	-0.476190	-0.656168	0.000000	0.000000	3
30-Mar-92	0.000000	0.290698	-0.558214	0.000000	0.000000	0.000000	3
31-Mar-92	-0.216450	0.000000	0.240577	0.396301	5.813819	0.721708	3
01-Apr-92	1.518438	1.304348	0.800000	0.263158	0.000000	0.000000	3
02-Apr-92	-0.854701	-0.858369	0.079365	0.065617	0.000000	0.000000	3
03-Apr-92	0.215517	0.649351	0.039651	0.032787	0.000000	0.000000	3
04-Apr-92	0.000000	0.000000	0.019818	0.016516	0.000000	0.000000	3
05-Apr-92	0.000000	0.000000	0.019814	0.016257	0.000000	0.000000	3
07-Apr-92	0.000000	-0.645161	0.316957	1.703801	-6.345787	2.866157	3
08-Apr-92	0.000000	-0.144300	0.315956	0.773196	0.000000	0.000000	3
09-Apr-92	0.645161	0.939306	-0.236220	0.127877	0.000000	0.000000	3
10-Apr-92	0.427350	0.357910	0.473560	0.127714	0.000000	0.000000	3
11-Apr-92	0.425532	0.356633	0.000000	0.000000	0.000000	0.000000	3
12-Apr-92	0.423729	0.675195	0.314218	-0.127551	0.000000	0.000000	3
13-Apr-92	0.421941	0.670667	-0.313234	0.127714	0.000000	0.000000	3
14-Apr-92	-0.210084	-0.561010	1.256873	0.255102	-5.078785	0.000000	3
15-Apr-92	0.210526	0.705219	0.310318	0.127226	0.000000	0.000000	3

16-Apr-92	0.210084 0.000000 0.696056 0.127065 0.000000 0.000000 3
17-Apr-92	-0.419287 0.840336 1.152074 0.507614 0.000000 0.000000 3
18-Apr-92	0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 3
19-Apr-92	0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 3
20-Apr-92	1.052632 -0.138889 -0.075930 -0.252525 0.000000 0.000000 3
21-Apr-92	0.208333 0.278164 0.303951 0.379747 7.176601 0.000000 3
22-Apr-92	0.623701 0.208044 0.151515 0.000000 0.000000 0.000000 3
24-Apr-92	-1.033058 -0.899654 -0.302572 0.504414 0.000000 0.000000 3
25-Apr-92	0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 3
26-Apr-92	0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 3
27-Apr-92	0.208768 0.418994 0.606980 0.376412 0.000000 0.000000 3
28-Apr-92	0.208333 0.417246 0.150830 0.000000 5.814375 0.000000 3
29-Apr-92	-0.623701 -0.761773 -0.301205 0.000000 0.000000 0.000000 3
30-Apr-92	0.418410 0.348918 0.226586 0.125000 0.000000 0.000000 3
01-May-92	0.625000 0.834492 0.678222 1.123595 0.000000 0.000000 3
04-May-92	0.207039 0.344828 0.224551 0.123457 0.000000 0.000000 3
05-May-92	0.413223 0.412371 0.896191 1.109741 4.053592 0.000000 3
06-May-92	0.617284 0.616016 0.444115 1.097561 0.000000 0.000000 3
07-May-92	0.000000 0.136054 0.000000 0.482509 0.000000 0.000000 3
08-May-92	0.204499 0.203804 0.589536 0.600240 0.000000 0.000000 3
11-May-92	1.632653 1.423729 1.098901 0.238663 0.000000 0.000000 3
12-May-92	-0.803213 -0.534759 -0.724638 0.000000 0.000000 0.000000 3
13-May-92	-1.214575 -1.478495 -0.802920 -0.119048 0.000000 0.000000 3
14-May-92	0.819672 0.954980 0.367918 0.595948 0.000000 0.000000 3
15-May-92	1.016260 0.945946 0.366569 0.473934 0.000000 0.000000 3
18-May-92	1.207243 1.338688 0.511322 1.886792 0.000000 0.000000 3
20-May-92	-0.596421 -0.660502 0.145349 0.578704 4.068828 0.000000 3
21-May-92	0.400000 0.265957 0.870827 -0.230150 0.000000 0.000000 2
22-May-92	0.000000 0.198939 -0.143885 -0.576701 0.000000 0.000000 3
25-May-92	0.000000 0.198544 -0.288184 -0.232019 0.000000 0.000000 3
26-May-92	0.000000 -0.132100 0.000000 0.000000 1.611724 0.000000 3
27-May-92	0.398406 0.529101 0.939306 -0.232558 0.000000 0.000000 3
28-May-92	0.198413 0.263158 0.357910 0.116550 0.000000 0.000000 3
29-May-92	-0.198020 -0.524934 -0.927247 0.232829 0.000000 0.000000 3
01-Jun-92	-1.190476 -1.055409 -0.791937 -0.232288 0.000000 0.000000 3
02-Jun-92	0.803213 0.933333 0.798258 0.116414 0.000000 0.000000 3
03-Jun-92	0.398406 0.264201 -0.287977 0.000000 -4.533357 0.000000 3
04-Jun-92	-0.396825 -0.263505 -0.216606 -0.116279 0.000000 0.000000 1
05-Jun-92	-0.398406 -0.594452 -0.506512 0.349243 0.000000 0.000000 3
08-Jun-92	0.400000 0.398671 0.218182 0.348028 -1.071925 0.000000 3
09-Jun-92	0.000000 -0.198544 0.145138 0.693642 0.000000 0.000000 3
10-Jun-92	0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 3
11-Jun-92	0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 3
12-Jun-92	0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 3
13-Jun-92	0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 3
15-Jun-92	2.788845 3.116711 1.739130 2.870264 0.000000 0.000000 3
16-Jun-92	0.387597 0.321543 0.498576 0.111607 0.000000 0.000000 3
17-Jun-92	0.193050 0.256410 0.141743 0.780379 0.000000 0.000000 3
18-Jun-92	0.192678 0.000000 0.424628 0.000000 0.000000 0.000000 1

19-Jun-92	0.961538	0.895141	-0.352361	-0.110619	0.000000	0.000000	3
22-Jun-92	-0.190476	-0.506971	-0.070721	-0.221484	0.000000	0.000000	1
23-Jun-92	-0.190840	0.127389	0.000000	0.221976	0.000000	0.000000	1
24-Jun-92	0.000000	0.000000	-0.353857	0.110742	0.000000	0.000000	3
25-Jun-92	-0.764818	-1.017812	-0.568182	0.110619	0.000000	0.000000	3
26-Jun-92	-0.385356	-0.064267	-0.214286	0.000000	0.000000	0.000000	3
29-Jun-92	-1.353965	-1.736334	-1.431639	0.000000	0.000000	0.000000	3
30-Jun-92	0.392157	0.785340	0.653595	0.331492	0.000000	0.000000	3

REFERENCES

Akyüz, Y., (1988), "Financial System and Policies in Turkey in the 1980's", United Nations Conference on Trade and Development, Geneva.

Atiyas, I., and H. Ersel, (1992), "The Impact of Financial Reform: The Turkish Experience", **World Bank Publication.**

Buchanan, B., and T. Mitchell, (1978), "Model-directed learning of production rules",. In Waterman, D. and Hayes-Roth, F. (eds.), **Pattern-Directed Inference Systems**. New York: Academic Press.

Bulmash, B.S., and W.G. Trivoli, (1991), "Time-lagged interractions between stock prices and selected economic variables", **The Journal of Portfolio Management**, Summer, 61-67.

Chan, K.C., and N. Chen, (1991), "Structural and Return Characteristics of Small and Large Firms", **Journal of Finance**, 46, 1467-1484.

Chen, N., R. Roll, and S.A. Ross, (1986), "Economic Forces and the Stock Market", **Journal of Business**, 56, 383-403.

Coats, P.K., (1987), "Replacing Managerial Judgement With Computer", **Journal of Business Forecasting**, Vol. 5. No. 1, Spring 1986, pp. 12-15.

Darrat, A.F., (1988), "On Fiscal Policy and the Stock Market", **Journal of Money Credit and Banking**, 20, 355-63.

Erol, Ü., and K. Aydoğan, (1992), "Asset Pricing in an Emerging Market: The Turkish Case", Department of Management, Bilkent University, Ankara, Turkey.

Ersel, H., and G. Sak, (1985), "The Financial Structure of the Corporations Subject to CMB Supervision:1974-84", **Capital Market Board Publications**, 7, 89-140.

Falk, H., and H. Levy, (1989), "Market Reaction to Quarterly Earnings' Announcements: A Stochastic Dominance Based Test of Market Efficiency", **Management Science**, 35, 425-66.

Fama, E.F., (1965), "The Behavior of Stock Market Prices", **Journal of Business**, 38, 34-105.

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

Fama, E.F., (1991), "Efficient Capital Markets: II", **Journal of Finance**, 5, 1575-1617.

Fama, E.F., and M. Blume, (1991), "Filter Rules and Stock Market Trading Profits", **Journal of Business**, 39, 226-41.

Fama, E.F., and K.R. French, (1988), "Permanent and Temporary Components of Stock Prices", **Journal of Political Economy**, April 1988, pp. 246-273.

Fama, E.F., and J. MacBeth, (1973), "Risk, return and equilibrium: Empirical tests", **Journal of Political Economy**, 81, 607-636.

Gültekin, N.B., (1986), "Stock Market Returns Under Inflation", **Capital Market Board Publications**, 7, 303-23.

Gültekin, N.B., and G. Sak, (1990), "The Role of Securities Markets in the Financial Liberalization Process: An evaluation of the Policies in the 1980's", **Capital Markets Board Publications**.

Hancock, D.G., (1989), "Fiscal Policy, Monetary Policy and the Efficiency of the Stock Market", **Economics Letters**, 31, 65-9.

Huberman, G., and S. Kandel, (1985), "A size Based Stock Returns Model", **Center of Research in Security Prices Working Paper** 148, Graduate School of Business, University of Chicago.

Iba, G.A., (1979), "Learning Disjunctive Concepts from Examples", **Artificial Intelligence Memo,** pp. 548, MIT AI Lab.

Jain, C.L., (1987), "A Managerial Guide To Judgmental Forecasting", Flushing, NY: Graceway Publishing Company, Inc.

Jensen, M.C., (1968), "The Performance of Mutual Funds in the Period of 1954-64", **Journal of Finance**, 23, 389-416.

Keng, K.C.W., (1984), "Econometric Forecasting Modelling--A Utility Maximization Approach", Economics and Forecasts Division, Ontario Hydro, Toronto, Canada.

Mishkin, F.S., (1982), "Does Anticipated Monetary Policy Matter? An Econometric Investigation", **Journal of Political Economy**, 90, 22-51.

Mooney, R., and G. DeJong, (1985), "Learning Schemata for Natural Language Processing", **Proceedings of IJCAI-85,** 681-687, Los Angeles, CA: Morgan Kaufmann Publishers.

Moore, G.H., (1980), "Business Cycles, Inflation and Forecasting", National Beureau of Economic Research, Studies in Business Cycles No:24.

Panas, E.E., (1990), "The Behavior of Athens Stock Prices", **Applied Economics**, 22, 1715-1727.

Pasquale, S., (1991), "Integration of Heterogeneous Knowledge in a Financial Intelligent Support System", In Cercone, N. and Gardin, F. (eds.), **Computational Intelligence III**, Elsevier Science Publishers B.V. North-Holland.

Pearce, D.K., and V.V. Roley, (1985), "Stock Prices and Economic News", **Journal of Business**, vol. 8, no. 1, pp. 49-67.

Poterba, J.M., and L.H. Summers, (1988), "Mean reversion in Stock Prices: Evidence and Implications", **Journal of Financial Economics**, October, pp. 27-59.

Quinlan, J.R., (1986), "Induction of decision Trees", **Machine Learning**, Vol. 1 No 1.

Ross, S.A., (1976), "The Arbitrage Theory of Capital Asset Pricing", **Journal of Economic Theory**, 13, 341, 360.

Sak, G., and E.A. Yeldan, (1993), "Reflections on Asset Backed Securitization in Turkey", **Capital Markets Board Publications**.

Salzberg, S.L., (1990), "Learning with Nested Generalized Exemplars", Norwell, MA: Kluwer Academic Publishers.

Şengül, G.M., and D. Önkal, (1992), "Türk Hisse Senedi Piyasasında Yarı-Güçlü Etkinlik", **ODTÜ Gelişme Dergisi**, 19(2) 1992, 197-207.

Sharpe, W.F., (1966), "Mutual Fund Performance", **Journal of Business**, 39, 119-138.

Sönmez, C., and A. Berik, (1993), "Monetary Variables effecting the ISEM Index", **Capital Market Board Publication**.

Summers, L.H., (1986), "Does the Stock Market Rationally Reflect Fundamental Values?", **Journal of Finance**, July, 41, 591-601.

Tang, Z., and P.A. Fischwick, (1991), "Feed-forward Neural Nets as Models for Time Series Forecasting", **Computer Science: Feed-forward neural networks, application**.

Wong, F., and P.Y., Tang, (1991), "Neural Networks and Genetic Algorithm For Economic Forecasting", **AI in economics and business administration**.

Zarovin, P., (1990), "Size, Seasonality, and Stock Market Overreaction", **Journal of Financial and Quantitative Analysis**, 25, 113-25.