GMDH-TYPE NEURAL NETWORK ALGORITHMS FOR SHORT TERM FORECASTING

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

OSMAN DAĞ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
STATISTICS

AUGUST 2015

Approval of the thesis:

GMDH-TYPE NEURAL NETWORK ALGORITHMS FOR SHORT TERM FORECASTING

submitted by **OSMAN DAĞ** in partial fulfillment of the requirements for the degree of **Master of Science in Statistics Department, Middle East Technical University** by,

Prof. Dr. Gülbin Dural Ünver _____
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. Ayşen Akkaya _____
Head of Department, **Statistics**

Assist. Prof. Dr. Ceylan Yozgatlıgil _____
Supervisor, **Statistics Department, METU**

Examining Committee Members:

Assoc. Prof. Dr. Erdem Karabulut _____
Biostatistics Department, Hacettepe University

Assist. Prof. Dr. Ceylan Yozgatlıgil _____
Statistics Department, METU

Assoc. Prof. Dr. Pınar Özdemir _____
Biostatistics Department, Hacettepe University

Assoc. Prof. Dr. Serdal Kenan Köse _____
Biostatistics Department, Ankara University

Assist. Prof. Dr. Ceren Vardar Acar _____
Statistics Department, METU

Date: 10/08/2015

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name : Osman Dağ

Signature :

ABSTRACT

GMDH-TYPE NEURAL NETWORK ALGORITHMS FOR SHORT TERM FORECASTING

Dağ, Osman

M.Sc., Department of Statistics

Supervisor: Assist. Prof. Dr. Ceylan Yozgatlıgil

August 2015, 76 pages

Group Method of Data Handling (GMDH) - type neural network algorithms are the heuristic self-organization method for modelling the complex systems. GMDH algorithms are utilized for the variety of purposes, which are identification of physical laws, extrapolation of physical fields, pattern recognition, clustering, approximation of multidimensional processes, forecasting without models and so on. In this study, GMDH - type neural network algorithms were applied to make forecasts for time series data sets. We mainly focused on development of free software. For this purpose, we developed an R package GMDH. Moreover, we integrated different transfer functions, sigmoid, radial basis, polynomial, and tangent functions, into GMDH algorithm. We proposed an algorithm in which all transfer functions are used simultaneously or separately if desired. Also, we used regularized least square estimation for the estimation of weights to overcome multi-collinearity problem. The methods were illustrated on real life datasets having different properties to see the prediction and forecasting performance of the algorithm. We included ARIMA models and exponential smoothing methods for the comparison purpose. GMDH algorithms show the same or even better performance than the other methods.

Keywords: Time Series Analysis, Neural Network, Regularized Least Square Estimation, Transfer Function, Statistical Software

ÖZ

KISA DÖNEM ÖNGÖRÜ İÇİN GMDH TÜRÜNDE SİNİR AĞI ALGORİTMALARI

Dağ, Osman
Yüksek Lisans, İstatistik Bölümü
Tez Yöneticisi: Yrd. Doç. Dr. Ceylan Yozgatlıgil
Ağustos 2015, 76 sayfa

Veri işleme grup yöntemi (GMDH) türünde sinir ağı algoritmaları, karmaşık sistemleri modellemeye yarayan bulgusal, kendi kendini organize eden yöntemlerdir. GMDH algoritmaları fizik kanunlarını tanımlama, fiziksel alanların dış kestirimi, örüntü tanıma, kümeleme, çok boyutlu işlemlerin yaklaştırımı, modelsiz öngörü gibi çeşitli amaçlar için kullanılmaktadır. Bu çalışmada, GMDH türünde sinir ağı algoritmalarına, zaman serisi veri setleri için öngörü yapmak amacıyla başvurulmuştur. Çoğunlukla ücretsiz bir yazılım geliştirmeye odaklanıldı ve bu amaçla GMDH isimli bir R paketi geliştirildi. Ek olarak sigmoid, radyal temelli, polinomiyal ve tanjant fonksiyonları gibi aktarma fonksiyonları GMDH algoritmasına entegre edildi. İstenildiğnde eş zamanlı veya ayrı ayrı aktarma fonksiyonlarının kullanılabildiği bir algoritma önerildi. Çoklu bağlantı problemini çözmek için ağırlıkların kestiriminde, düzeltilmiş en küçük kareler kestirimi kullanıldı. Farklı özelliklere sahip gerçek hayat veri setleri üzerinde yöntemler uygulanarak algoritmanın tahminleme ve öngörü performansı incelenmiştir. Karşılaştırma amaçlı ARIMA modelleri ve üstel düzgünleştirme yöntemleri de dahil edildi. GMDH algoritmalarının diğer yöntemlerle aynı hatta daha iyi performans gösterdiği saptanmıştır.

Anahtar Kelimeler: Zaman Serisi Analizi, Sinir Ağları, Düzeltilmiş En Küçük Kareler Yöntemi, Aktarma Fonksiyonu, İstatistiksel Yazılım

ACKNOWLEDGEMENTS

I would like to put my deepest gratefulness into words to my advisor, Assist. Prof. Dr. Ceylan Yozgatlıgil, for her endless support. Her great support made me feel confident and encouraged. It is difficult for me to express my whole feelings in words, but I should point out that I am happy and lucky to have had an opportunity to work with her.

I am also thankful to Assoc. Prof. Dr. Erdem Karabulut, Assoc. Prof. Dr. Pınar Özdemir, Assoc. Prof. Dr. Serdal Kenan Köse and Assist. Prof. Dr. Ceren Vardar Acar for their relevant discussions, suggestions and comments.

I would like to thank to all instructors who lectured me and made contribution to my skills in the past at Middle East Technical University. Within a special parenthesis, I would like to thank all members of Department of Biostatistics at Hacettepe University for their all support.

My warm and sincere thanks to my best friends, Ramazan Seyhan, Özgür Saman, Atilla Eyüpoğlu, Önay Burak Doğan for being always with me.

I would like to express my appreciations to my family, Murat, Döndü, Nazmi, Nazan, Murathan and Ceren. This thesis is the product of endless support of Dağ family.

Last but not least, many thanks to my wife, Özlem, for her all support and not complaining about writing my thesis in the time that I need to spend with her. Also, many thanks to my newborn baby, Ada, for entering our life and making it colorful for us.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

AIC	Akaike Information Criterion
AICc	corrected Akaike Information Criterion
ARIMA	Autoregressive Integrated Moving Average
BIC	Bayesian Information Criterion
CRAN	Comprehensive R Archive Network
FMSE	Forecasting Mean Square Error
GMDH	Group Method of Data Handling
MAPE	Mean Absolute Percentage Error
PMSE	Prediction Mean Square Error
PSS	Prediction Sum of Squares

CHAPTER 1

INTRODUCTION

Time series data are ordered successive observations which are measured in equally or unequally spaced time. Time series data include dependency among successive observations. Hence, the order of the data is important. They are commonly appearing in various areas. In medical studies, we take measurements on blood sugar, blood pressure, electrocardiogram tracing over time. In economics, we record gross national income, gross national expenditure over months or/and years. In energy industry, frequency of electrical signals and power of devices are measured over time. In agriculture, we record total production of hazelnut and prices in each year. In meteorology, total amount of rainfall in a region is recorded hourly or/and weekly or/and monthly or/and yearly. Application fields of time series, not given here, are limitless.

Modelling time series data is the method which utilizes history of the data and makes forecasting by the help of the history of the data. Forecasting is the prediction of the future observations by processing data at hand. It is important to construct accurate model mechanism for forecasting to obtain reliable results. Many statistical tools including the independence assumption are not applicable in time series. Therefore, different tools considering the dependence among the lags of the data are required.

Autoregressive integrated moving average (ARIMA) models, which consider the dependency among the successive observations, are introduced in Box and Jenkins (1970). These models are called stationary providing that all properties are the same over time. The variation is around the mean is constant over time. Also, there is no trend in the stationary process. Apart from these properties, ARIMA models include some assumptions being necessary to be satisfied. For example, the

residuals of the process are required to come from normal distribution. However in real life it is almost impossible to satisfy this assumption. There are some methods to handle this problem. One of which is a group method of data handling (GMDH) – type neural network algorithms for the objective of forecasting time series will be introduced in the next paragraph. For the comparison purpose with respect to forecasting accuracy, we include the application of the ARIMA models and exponential smoothing methods in chapter 4.

The main objective of this thesis is to make forecasts via GMDH-type neural network algorithms. There are some difficulties to apply GMDH-type neural network, since there is no free available code for the user to reach GMDH algorithm in the literature. Available commercial software programs do not explain the steps and how the algorithm is working. Moreover, they are not convenient for the time series data set. The question arises "What kind of contributions of this thesis to the statistical literature and science is made?". The followings are the answers:

- We applied GMDH-type neural network which is used in very few studies of statistical literature for the purpose of forecasting time series and illustrated an application of the algorithm on real life data sets.
- We proposed an algorithm having an option in which all transfer functions are used simultaneously or separately.
- We developed an R package "GMDH" and we made it publicly available. We did not only make it publicly available, but also we presented all algorithm of the system step by step. Since commercial statistical programs do not explain all steps, just present how to use the program, the estimation step of algorithm is not shown. Therefore, most of the time, the user exerts the program, but does not know how it is working inside.
- We integrated regularized least square estimation which is utilized when there may be a possibility of occurring multi-collinearity problem.

- We also state data preparation of a time series shown in section 3.1. Our proposed package is making it convenient for the algorithm.

The outline of this thesis is organized as follows. In chapter 2, we present the literature review of GMDH-type neural network and usage of that for the objective of forecasting time series. In chapter 3, we discuss the GMDH-type neural network algorithms. Moreover, we present how to manipulate time series data set to make it convenient GMDH-type neural network. We discuss the transfer functions used in the algorithms. Estimation of weights in neurons via regularized linear regression and estimation of regularization parameter via cross validation are presented. In chapter 4, real life data applications and related results are stated. The details of R package "GMDH" is instructed with examples in this chapter. In chapter 5, we close the thesis by discussion, conclusion and further research parts.

CHAPTER 2

LITERATURE REVIEW

The previous studies on GMDH can be divided into four parts. In the first part, the origin of GMDH-type neural network is introduced. Second, we present some areas in which GMDH algorithm is utilized. Third, the studies with the methodology of GMDH algorithm are given. At last, we state the studies in which GMDH-type neural network is applied for the purpose of forecasting time series. We should note that chronological order is followed in each part.

2.1. The Origin of GMDH-Type Neural Network

The background of GMDH-type neural network is based on the end of the 1960s years and the beginning of 1970s years. First in first, Ivakhnenko (1966) introduced a polynomial, which is the basic algorithm of GMDH, to construct higher order polynomial. The polynomial, which is known as Ivakhnenko polynomial, is described in section 3.2. Also, Ivakhnenko (1970) introduced heuristic self-organization method which constructed the main working system of GMDH algorithm. Heuristic self-organization method defines the way that the algorithm follows by the rules such as external criteria (see section 3.5). GMDH method, convenient for the complex and unstructured system, has superiority on the high order regression (Farlow, 1981).

2.2. Application Areas of GMDH Algorithms

A variety of the problems which the GMDH algorithm solves was described in Ivakhnenko and Ivakhnenko (1995). Some of these problems are the identification of physical laws, extrapolation of physical fields, pattern recognition, clustering,

forecasting without models, approximation of multidimensional processes, and so on.

Kalavrouziotis et al. (2002) applied GMDH algorithm in environmental studies. They had cultivated trees and irrigated those trees with processed wastewater. They considered the non-linear relationship between characteristics of wood obtained from the trees irrigated with processed wastewater and characteristics of wood obtained from the trees grown up in a common way. Therefore, they utilized GMDH algorithm to capture the non-linear relation between input and output variables. Nariman-zadeh et al. (2002) used GMDH algorithm in material processing studies. GMDH was exerted to see the relation between considerable variables and depth penetration when they modeled explosive cutting process of plates.

GMDH algorithm was used in design of experiments (Astakhov and Galitsky, 2005). When they constructed their experiment, they had some difficulties such as limited number of variables, pre-setting the model. To solve these difficulties, they applied GMDH algorithm and they obtained very complex model to explain which parameters have an effect on tool life in gundrilling. It was shown that the tool life in gundrilling is the model of various regime and design parameters. In another study, GMDH was applied to make feature ranking and selection of the medical data (Abdel-Aal, 2005). Baig et al. (2013) used GMDH-type neural network algorithm for intelligent intrusion detection. In that study, they classified network traffic into two classes: normal and anomalous. Najafzadeh et al. (2014) utilized GMDH algorithm in pipeline systems studies. Depth of scour below pipelines which were exposed to waves was predicted via GMDH-type neural network. Sheikholeslami et al. (2014) applied GMDH-type neural network to investigate the impact of magnetic field on heat transfer of Cu-water nanofluid.

2.3. The Methodology of GMDH Algorithms

Muller et al. (1998) studied GMDH algorithms for the objective of modelling the complex systems. Sometime statistical/mathematical models are not sufficient to solve the problems, such as pattern recognition, forecasting, identification, etc. Extracting the information from the measurements has advantages while modelling complex systems since there is no enough prior information and/or no theory is defined to model the complex systems. Selecting model automatically is a powerful way for the users who are interested in the result and do not have sufficient statistical knowledge and sufficient time. In Muller et al., they also stated that basically prediction mean square error (PMSE) is applied for the neuron selection.

Kondo (1998) proposed GMDH-type neural network in which the algorithm works according to the heuristic self-organization method. Kondo and Ueno (2006a) proposed GMDH algorithm which has a feedback loop. According to the algorithm, the output obtained from the last layer is set as a new input variable, if threshold is not satisfied in the last layer. The system of algorithm is organized by heuristic self-organization method. In this algorithm, sigmoid function is integrated in the algorithm as an activation function. This algorithm is applied to medical image recognition of brain (Kondo and Ueno, 2006b). Kondo and Ueno (2007) proposed logistic GMDH-type neural network. The difference from conventional GMDH algorithm was that they take linear function of all inputs at last layer. They applied GMDH algorithm to identify the X-ray film characteristic curve. Kondo and Ueno (2012) included three transfer functions in feedback GMDH algorithm. These transfer functions are sigmoid function, radial basis function and polynomial function. They used this algorithm in medical image analysis of liver cancer.

2.4. The Studies In Which GMDH-Type Neural Network Is Applied For The Purpose of Forecasting

Srinivasan (2008) used GMDH-type neural network to forecast energy demand prediction. This study also included the comparison of GMDH-type algorithm with traditional time series models on real life data. It was shown that GMDH-type neural network was superior in forecasting energy demand prediction compared to traditional time series models. In another study, Xu et al. (2012) applied GMDH algorithm to forecast the daily power load. When they made forecast on the daily power load, they also included ARIMA models. According to the results, GMDH results were superior to the results of ARIMA models.

All in all, the origin of GMDH-type neural network algorithm is stated. A variety of areas which GMDH algorithm is applied in is presented. Also, we state the methodological background of GMDH algorithm and the studies using GMDH algorithm for the objective of forecasting time series. In following chapters, we discuss the methodology and its applications on real life data. Moreover, we introduce our proposed R package and its implementation on real life data sets.

CHAPTER 3

METHODOLOGY

In this chapter, we discuss GMDH - type neural network algorithms in time series forecasting. The chapter is divided into seven sections. In section 3.1, we present the data preparation before GMDH-Type neural network algorithms. In section 3.2, we continue with GMDH - type neural network algorithms in forecasting. We discuss transfer functions in section 3.3. In section 3.4, estimation of weights via regularized least square estimation and estimation of regularization parameter through cross validation are presented. We present the external criteria of accuracy in section 3.5. In section 3.6, we state the algorithm of GMDH - type neural network. This chapter is closed by methods included for comparison purpose.

3.1. Data Preparation

Data preparation has important role in GMDH - type neural network algorithms. Since we could not interfere with the algorithm while it is running, we are pre-processing the data before starting the algorithm. Scaling the data is essential before starting GMDH - type neural network algorithms. There are two main benefits of this. One is to get rid of very big numbers in calculation, the other one is to be able to use all transfer functions (see section 3.3) which are used in GMDH - type neural network algorithm for forecasting. For these two main objectives, it is necessary for whole data to be in interval of (0, 1). This necessity is guaranteed by following transformation,

$$w_t = \frac{\alpha_t + \delta_1}{\delta_2} \tag{3.1}$$

with

$$\delta_1 = \begin{cases} |\min(\alpha_t)| + 1 & , \quad \text{if} \quad \min(\alpha_t) \leq 0 \\ 0 & , \quad \text{if} \quad \min(\alpha_t) > 0 \end{cases}$$

and

$$\delta_2 = \max(\alpha_t + \delta_1) + 1$$

where α_t is actual time series dataset at hand. During the estimation and forecasting process in GMDH neural network algorithm, all calculations are done using the scaled data set, w_t. After all processes are ended; in other words, all predictions and forecasts are obtained, we apply the inverse transformation as follows,

$$\hat{\alpha}_t = \hat{w}_t \times \delta_2 - \delta_1. \tag{3.2}$$

The main purpose of this section is the pre-processing the data; that is, we make data manipulation and illustrate the structure of data. It is difficult to visualize the structure of data since we have a univariate time series dataset. As it is in autoregressive time series models, we model the time series data with time lags of the data in GMDH - type neural network algorithms.

Let's assume a time series dataset for *t* time points, and *p* inputs (see section 3.2). Since we construct the model for the data with lags, the number of subject is equal to be $t - p$ and the number of covariates is to be p. An illustration of time series data structure in GMDH algorithms is presented in Table 3.1. In this table, the variable called $Resp$ is put in the models as a response variable, and the rest of the variables are taken into models as covariates Cov_i, where i = 1, 2, ..., p. The notations in Table 3.1 are followed throughout this study.

Table 3.1: An illustration of time series data structure in GMDH algorithms

Subject	$Resp\ (z)$	$Cov_1(x_1)$	$Cov_2(x_2)$	$\cdots$	$Cov_p(x_p)$
1	w_t	w_{t-1}	w_{t-2}	$\cdots$	w_{t-p}
2	w_{t-1}	w_{t-2}	w_{t-3}	$\cdots$	w_{t-p-1}
3	w_{t-2}	w_{t-3}	w_{t-4}	$\cdots$	w_{t-p-2}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
$t-p$	w_{p+1}	w_p	w_{p-1}	$\cdots$	w_1

3.2. GMDH-Type Neural Network Algorithms

GMDH-type neural network algorithms are the modeling techniques which learn the relations among the variables. In the perspective of time series, the algorithm learns the relationship among the lags. After learning the relations, it automatically selects the way to follow in algorithm. First, GMDH was introduced by Ivakhnenko (1966) to construct high order polynomial. The following equation is known Ivakhnenko polynomial given by

$$y = a + \sum_{i=1}^{m} b_i x_i + \sum_{i=1}^{m}\sum_{j=1}^{m} c_{ij} x_i x_j + \sum_{i=1}^{m}\sum_{j=1}^{m}\sum_{k=1}^{m} d_{ijk} x_i x_j x_k + \cdots \tag{3.3}$$

where m is the number of variables and $a, b, c, d, \ldots$ are coeffients of variables in the polynomial. In general, the terms are used in calculation up to square terms in GMDH algorithm presented below,

$$y = a + \sum_{i=1}^{m} b_i x_i + \sum_{i=1}^{m}\sum_{j=1}^{m} c_{ij} x_i x_j \tag{3.4}$$

In the following subsections, GMDH-type neural network algorithms are mainly focused and the architectures of the algorithms are presented.

3.2.1. Architecture of GMDH Algorithm

The conventional GMDH algorithm considers all possible combinations of two covariates. Therefore, each combination of two input variables enters each neuron. Using these two inputs, a model is constructed. With the help of this model, we are able to estimate the desired output. The structure of the model is specified by Ivakhnenko polynomial. Therefore, the following equation is derived from the Ivakhnenko polynomial:

$$y = \beta_0 + \beta_1 x_i + \beta_2 x_j + \beta_3 {x_i}^2 + \beta_4 {x_j}^2 + \beta_5 x_i x_j. \quad (3.5)$$

Here, y is a response variable, x_i and x_j are the covariate variables to be regressed. β_l's are named as the weights where $l = 0, 1, \ldots, 5$. The response variable (y) is modelled by all possible combination of two input variables (x_i and x_j). It means that two input variables go in a neuron, one result goes out as an output.

GMDH algorithm is a system of layers in which there exist neurons. In each layer, there exist a number of neurons. The number of neurons in a layer is defined by the number of input variables. To illustrate, assume that the number of input variables equals to p, since we include two input variables, the number of neurons is to be equal to $h = \frac{p \times (p-1)}{2}$. The architecture of GMDH algorithm is illustrated in Figure 3.1 when there exist three layers and four inputs. In this architecture, the number of inputs is equal to four; therefore, the number of nodes in a layer is determined to be six. In input layer, there exist four input variables. There is no any process at this layer. This is just a starting layer to the algorithm. All plausible combination of four input variables enters to each neuron at first layer. The coefficient of Equation 3.5 is estimated in each neuron. By using estimated coefficients and input variables in each neuron, the desired output is predicted. According to external criteria, p neurons are selected and h-p neurons are eliminated from the network. In Figure 3.1, four neurons are selected while two neurons are eliminated from the network. The outputs obtained from selected

neurons become the inputs for the next layer. This process continues until the last layer. At the last layer, only one neuron is selected. The obtained output from last layer is the predicted values for the time series at hand.

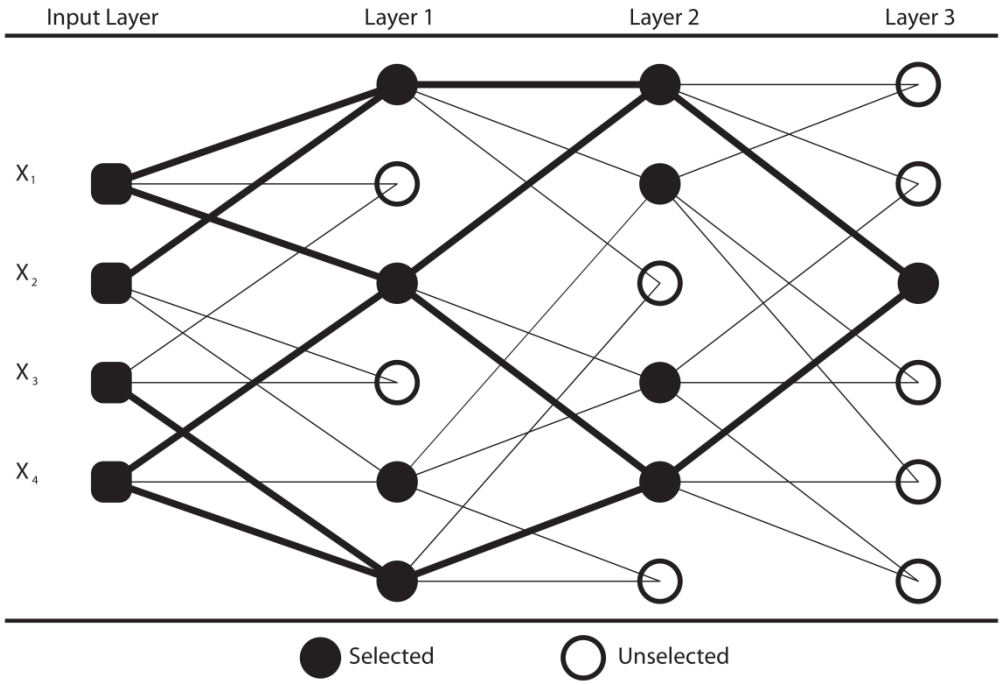

Figure 3.1: Architecture of GMDH Algorithm

3.2.2. Architecture of RGMDH Algorithm

GMDH-type neural network constructs the algorithm by investigating the relation between two inputs and the desired output. Architecture of revised GMDH (RGMDH) - type neural network does not only consider this relation, but it also considers individual effect on desired output (Kondo and Ueno, 2006a). There are two different types of neurons in RGMDH-type neural network. In first type of neuron, it is same as in GMDH-type neural network that two inputs enter the neuron, one output goes out. In second type of neuron, *r* inputs enter the neuron, one output goes out.

First type neuron:

$$y = \beta_0 + \beta_1 x_i + \beta_2 x_j + \beta_3 {x_i}^2 + \beta_4 {x_j}^2 + \beta_5 x_i x_j \tag{3.6}$$

where y is a response variable, x_i and x_j are the covariate variables defined in Table 3.1. β_l's are the weights where $l = 0, 1, \ldots, 5$.

Second type neuron:

$$y = \beta_0 + \sum_{i=1}^{r} \beta_i x_i \quad , \quad r \leq p \tag{3.7}$$

where y is a response variable, x_i the covariate variables defined in Table 3.1. β_i's are the weights, $i = 1, \ldots, r$. Here, r is the number of inputs in the corresponding second type neuron.

In GMDH-type neural network, there exist $h = \frac{p \times (p-1)}{2}$ neurons in one layer. In addition to this, with the p neuron from the second type of neuron, the number of neurons in one layer becomes $\eta = \frac{p \times (p-1)}{2} + p$ in RGMDH-type algorithm. The architecture of RGMDH algorithm is shown in Figure 3.2 when there exist three layers and three inputs. In this architecture, since the number of inputs is equal to three, the number of nodes in a layer is determined to be six. Like GMDH algorithm, there is no any process at starting layer. All plausible combination of three inputs enters to each neuron at first layer. Moreover, the individual effect of inputs starting from lag 1 is added to the rest ones as shown in Figure 3.2. In each neuron, coefficients of models are calculated by using corresponding models in equation 3.6 or 3.7. The desired output is predicted by utilizing estimated coefficients and input variables in each neuron. p neurons are selected as living cells and $\eta - p$, death cells are eliminated from the network according the external criteria. The outputs obtained from selected neurons become the inputs for

the next layer. This process goes on until the last layer. Only one neuron is selected at the last layer according to external criteria. The obtained output at the last layer is the predicted values for the data set at hand.

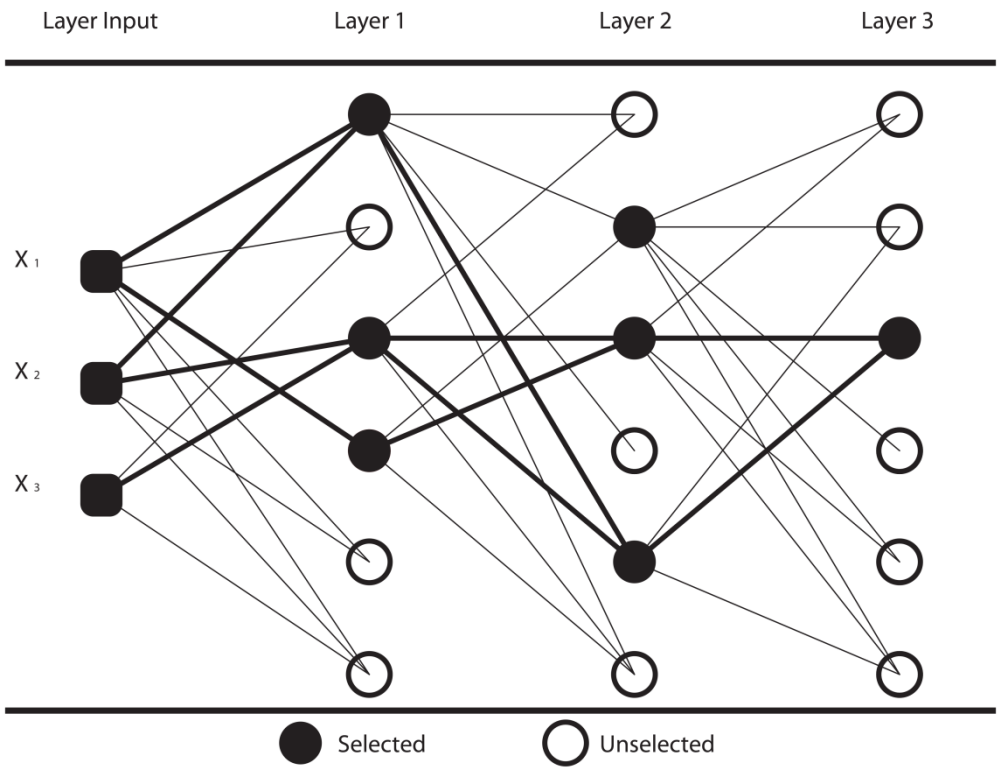

Figure 3.2: Architecture of RGMDH Algorithm

3.3. Transfer Functions

Transfer functions (also known as activation functions, utilized throughout interchangeably) are used to capture the better model which explains the relation between inputs and desired outputs. Mainly, sigmoid function, radial basis function (RBF), polynomial function were used to explain the relation between inputs and output in GMDH-type neural network (Kondo and Ueno, 2012). Also, we include tangent transfer function to consider the sinusoidal relation between covariates and response variable. Transfer functions used in study are as follows,

Sigmoid Function:

$$z = \frac{1}{1 + e^{-y}} \tag{3.8}$$

Radial Basis Function:

$$z = e^{-y^2} \tag{3.9}$$

Polynomial Function:

$$z = y \tag{3.10}$$

Tangent Function:

$$z = \tan y \tag{3.11}$$

Here, y is presented below for GMDH and RGMDH algorithms.

GMDH algorithm:

$$y = \beta_0 + \beta_1 x_i + \beta_2 x_j + \beta_3 {x_i}^2 + \beta_4 {x_j}^2 + \beta_5 x_i x_j \tag{3.12}$$

where y is a response variable, x_i and x_j are the covariate variables defined in Table 3.1. β_l's are the weights where $l = 0, 1, \ldots, 5$.

RGMDH algorithm:

First type neuron:

$$y = \beta_0 + \beta_1 x_i + \beta_2 x_j + \beta_3 {x_i}^2 + \beta_4 {x_j}^2 + \beta_5 x_i x_j \tag{3.13}$$

where y is a response variable, x_i and x_j are the covariate variables defined in Table 3.1. β_l's are the weights where $l = 0, 1, \ldots, 5$.

Second type neuron:

$$y = \beta_0 + \sum_{i=1}^{r} \beta_i x_i \quad , \qquad 1 \leq r \leq p \tag{3.14}$$

where y is a response variable, x_i's are the covariate variables defined in Table 3.1. β_i's are the weights, $i = 1, \ldots, r$. For the purpose of constructing the models defined in above, we use following transformations before constructing model in neuron,

For Sigmoid Function:

$$y = \log_e \left(\frac{z}{1 - z} \right) \tag{3.15}$$

For Radial Basis Function:

$$y = \sqrt{-\log_e z} \tag{3.16}$$

For Polynomial Function:

$$y = z \tag{3.17}$$

For Tangent Function:

$$y = arctan\ z \tag{3.18}$$

In this study, we use four activation functions. These functions are able to be used separately. For example, assume that only sigmoid function is wanted to be used, all calculations in each neuron are done according to sigmoid function. In other

words, the model using sigmoid function is constructed in each neuron. The other transfer functions are never used throughout the whole process.

We propose that all transformations mentioned above are able to be used simultaneously. Our proposed algorithm is that four models in each neuron are constructed by using transfer functions mentioned above. Within the group of these four models, one which has smallest external criteria (see section 3.5) is selected in each node. In each neuron, one transfer function is selected. After selection, the selected transfer function is responsible for that neuron.

3.4. Estimation of Weights

In this section, the estimation procedure in neurons is mentioned. In each node, the weights (coefficients) of inputs are calculated. We integrated regularized linear regression into estimation of weights in each neuron in order to overcome multi-collinearity problem. In the following two sub-sections, we present regularized least square estimation and estimation of regularization parameter via cross-validation.

3.4.1. Regularized Least Square Estimation

In each estimation step of a node, there exists the coefficients to be estimated. While we are estimating these coefficients, we use regularized least square estimation. It is stated that regularized least square estimation is utilized when there may be a possibility of occurring multi-collinearity problem. It is important to note that regularized least square estimation is equal to least square estimation when regularization parameter is zero.

3.4.1.1. Estimation of Weights in GMDH Algorithm

In GMDH algorithm, there exist six coefficients to be estimated in each model. Coefficients are able to be estimated via the equation as follows,

$$\hat{\beta}_k = (X_k{}'X_k + \lambda I_{6\times 6})^{-1}X_k{}'y \tag{3.19}$$

where

$$X_k = \begin{bmatrix} 1 & x_{k,i,1} & x_{k,j,1} & {x_{k,i,1}}^2 & {x_{k,j,1}}^2 & x_{k,i,1}x_{k,j,1} \\ 1 & x_{k,i,2} & x_{k,j,2} & {x_{k,i,2}}^2 & {x_{k,j,2}}^2 & x_{k,i,2}x_{k,j,2} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_{k,i,t-p} & x_{k,j,t-p} & {x_{k,i,t-p}}^2 & {x_{k,j,t-p}}^2 & x_{k,i,t-p}x_{k,j,t-p} \end{bmatrix}, y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{t-p} \end{bmatrix}$$

and

$$I_{6\times 6} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

with

$$\hat{\beta}_k = \begin{bmatrix} \hat{\beta}_{k,0} & \hat{\beta}_{k,1} & \hat{\beta}_{k,2} & \hat{\beta}_{k,3} & \hat{\beta}_{k,4} & \hat{\beta}_{k,5} \end{bmatrix}'.$$

Here, X_k represents the covariates matrix at neuron k $(k = 1, \dots, h)$. In the matrix X_k, $x_{k,i,1}$ is the first observation in covariate i at neuron k. y is a response variable including $t - p$ observations. Here, h is the number of neurons in each layer.

3.4.1.2. Estimation of Weights in RGMDH Algorithm

In RGMDH algorithm, there exist two types of neurons. For the first type neuron, estimation of weights is same with estimation in GMDH algorithm. Estimation of coefficients in the second type neuron is as follows,

$$\hat{\beta}_k = ({X_k}'X_k + \lambda I_{(p+1)\times(p+1)})^{-1}{X_k}'y \tag{3.20}$$

where

$$X_k = \begin{bmatrix} 1 & x_{k,1,1} & \dots & x_{k,r,1} \\ 1 & x_{k,1,2} & \dots & x_{k,r,2} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{k,1,t-p} & \dots & x_{k,r,t-p} \end{bmatrix} \text{and } y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{t-p} \end{bmatrix}$$

and

$$I_{(p+1)\times(p+1)} = \begin{bmatrix} 0 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{bmatrix}$$

with

$$\hat{\beta}_k = \begin{bmatrix} \hat{\beta}_{k,0} & \hat{\beta}_{k,1} & \dots & \hat{\beta}_{k,r} \end{bmatrix}'.$$

Here, X_k represents the covariates matrix at neuron k $(k = h+1, \dots, \eta)$. In the matrix X_k, $x_{k,2,1}$ is the first observation in second covariate at neuron k. y is a response variable including $(t-p)$ observations. Here, h is the number of neurons of which coefficients are estimated according to first type neuron. $\eta - h$ is the number of neuros whose coefficients are estimated through second type neuron in each layer.

3.4.2. Estimation of Regularization Parameter

In this part, we present the estimation of regularization parameter via cross-validation. For this purpose, we divide the data into two parts as a learning set (70%) and a testing set (30%). Since the data set is time dependent, order of data is saved in division process. In other words, first 70% of the data is used for learning set and the last 30% of the data is utilized for testing set. This whole process is applied for each model constructed in each neuron. The algorithm of regularization parameter estimation is as follows,

i) Clarify the possible regularization parameter, $\lambda = 0, 0.01, 0.02, 0.04, 0.08, \ldots, 10.24$.
ii) For each possible λ value, coefficients are estimated via the equation 3.19 (or 3.20) by using learning set.
iii) After calculation of weights, calculate the predicted values by utilizing test set and obtain MSE for each regularization parameter.
iv) Select the regularization parameter which gives minimum MSE value.

3.5. External Criteria of Accuracy

The external criterion is utilized to check the accuracy of the model. In GMDH algorithm, the external criterion is the decision measure in choosing the best way to make the most accurate forecasts. In general, mean square error is used as the external criteria to control the system of GMDH algorithm. After one output is obtained in each neuron of a layer, external criteria are calculated. According to the external criteria, some neurons continue to live, some cells, called death cells, are discarded. For each neuron, prediction mean square error (PMSE) is calculated between the obtained output (predicted values) and the desired output (observed values).

$$PMSE = \frac{1}{t-p}\sum_{i=1}^{t-p}(\hat{z}_i - z_i)^2 \tag{3.21}$$

In a layer of GMDH algorithm, all PMSE values are calculated for all neurons. According to PMSE values, some neurons are superior to the others. Better ones have the right to continue to the next layer. For this reason, after PMSE values of all neurons are calculated, *p* neurons which give smaller PMSE values compared to others are selected as living cells. The selected outputs continue with next layer as inputs. This process lasts until all layers are completed. At the last layer, the neuron which gives the smallest PMSE value is selected and the estimation procedure is ended. The details of the algorithms are discussed in following sub-sections.

3.6. Algorithm of GMDH-Type Neural Network

GMDH is the effective technique to make accurate forecasts by learning the relation between the lags of the data set. The most advantageous side of the method is that exact model is not needed. In addition to this, there is no restriction to apply the GMDH algorithm in the perspective of time series forecasting. The method learns the relation among the lags of time series in training process. Some neurons which give better performance are selected in network according to PMSE value. On the other hand, some neurons which have higher PMSE compared the other ones are eliminated from the system of network. The outputs obtained from the selected neurons become the inputs at next layer. The process continues until all layers are finished. The flowchart of the algorithm is able to be depicted in Figure 3.3.

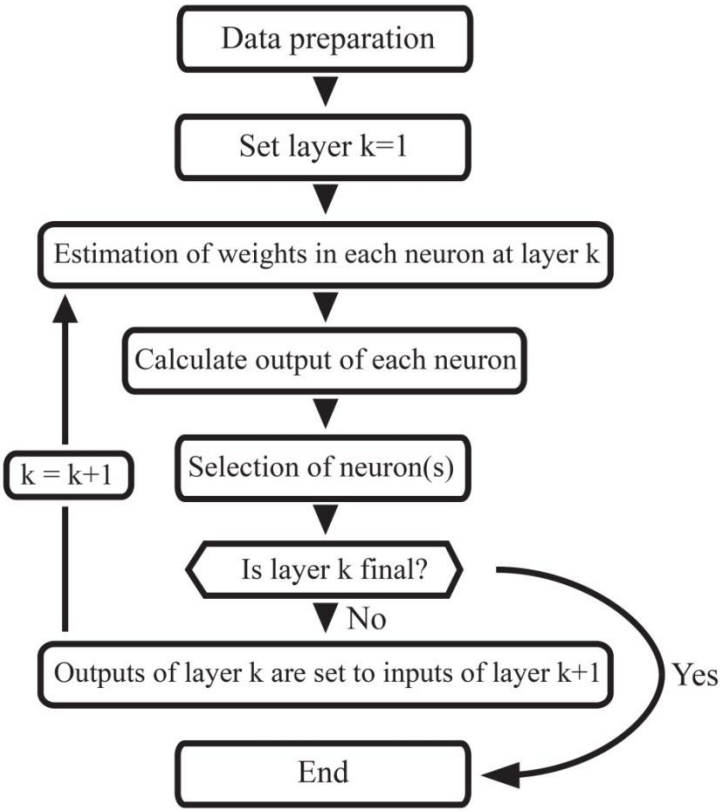

Figure 3.3: Flowchart of GMDH Algorithms

Before the algorithm starts, number of inputs and number of layers need to be clarified. There is no criterion to define the number of inputs. However, since time series has dependency among the lags, this is considered when the input number is chosen. For example, if the dependency between α_t and α_{t-12} is high, the input number are able to be chosen to be more than 12. This is not only parameter, but also the number of layers affects the system. In general, the layer number is not specified in GMDH algorithms. In that case, the system of algorithm is the decision maker of the number of layers. If the decreasing of error stops, the system is terminated (Samsudin et al., 2011). However, in this study, we clarify the number of layers in the algorithm, since it can cause over-fitting problem when the number of layers is large.

The algorithm of GMDH-type neural network algorithms in time series forecasting is provided in following steps:

i) Decide number of inputs and layers.
ii) Make data manipulation proposed in section 3.1.
iii) Make estimation of weights in neurons depending on choice of transfer function (see section 3.3).
iv) Obtain predicted values from all neurons and calculate PMSE for each neuron.
v) If the process is not at the last layer, select p outputs which have smaller PMSE values compared to the rest ones, then, take obtained p outputs to next layer as inputs and repeat steps iii) to v) again. Otherwise, select the output which has the smallest PMSE value among the outputs at the last layer.

GMDH algorithms continue until all layers are completed. Since there exist p inputs, there are h neurons for GMDH algorithm and η neurons for RGMDH algorithm in each layer. Within a group of neurons, p neurons which have smaller PMSE values compared to the rest are selected in each layer except for the last layer. At the last layer, the neuron which gives the smallest PMSE value is chosen as an output.

3.7. Methods Included for Comparison Purpose

3.7.1. ARIMA Models

ARIMA model, which is introduced in Box and Jenkins (1970), is the most dominating one in time series area. The ARIMA model represented by *ARIMA(p, d, q)* is written as follows:

$$\phi_p(B)\,(1-B)^d \alpha_t = \theta_q(B)\varepsilon_t \qquad (3.22)$$

with

$$\phi_p(B) = (1 - \phi_1 B - \phi_2 B^2 - \ldots - \phi_p B^p)$$
$$\theta_q(B) = (1 - \theta_1 B - \theta_2 B^2 - \ldots - \theta_q B^q)$$

where α_t is a time series data set; B is the backshift operator; $(1 - B)^d$ is non-seasonal differencing operator; $\mathcal{E}_t$ is a white noise random process with mean 0 and variance ${\sigma_a}^2$; $\phi_p(B)$ is autoregressive operator; $\theta_q(B)$ is moving average operator. To illustrate, let *ARIMA*(1, 0, 1) with ϕ_1 = 0.4, θ_1 = 0.3. The model becomes as follows:

$$(1 - 0.4B)\alpha_t = (1 - 0.3B)\,\mathcal{E}_t$$
$$\alpha_t = 0.4\alpha_{t-1} + \mathcal{E}_t - 0.3\mathcal{E}_{t-1}.$$

In real life, it is not always plausible to construct model by using ARIMA model because of existence of seasonality. The seasonal ARIMA model is depicted as *ARIMA(p,d,q)*$\times(P, D, Q)_s$ is stated as:

$$\phi_p(B)\Phi_P(B^s)\,(1 - B)^d(1 - B^s)^D Z_t = \theta_q(B)\Theta_Q(B^s)\,a_t \qquad (3.23)$$

with

$$\phi_p(B) = (1 - \phi_1 B - \phi_2 B^2 - \ldots - \phi_p B^p)$$
$$\Phi_P(B^s) = (1 - \Phi_1 B^s - \Phi_2 B^{2s} - \ldots - \Phi_P B^{Ps})$$
$$\theta_q(B) = (1 - \theta_1 B - \theta_2 B^2 - \ldots - \theta_q B^q)$$
$$\Theta_Q(B^s) = (1 - \Theta_1 B^s - \Theta_2 B^{2s} - \ldots - \Theta_Q B^{Qs})$$

where Z_t is a time series, B is the backshift operator, $(1 - B)^d$ and $(1 - B^s)^D$ are non-seasonal and seasonal differencing operators, respectively. a_t is a white noise random process with mean 0 and variance ${\sigma_a}^2$, $\phi_p(B)$ is autoregressive

operator, $\Phi_P(B^s)$ is seasonal autoregressive operator, $\theta_q(B)$ is moving average operator, $\Theta_Q(B^s)$ seasonal moving average operator. For example, let *ARIMA*(1, 0, 1)×$(0, 0, 1)_{12}$ with ϕ_1 = 0.5, θ_1=0.2, Θ_1=0.4. The model becomes as follows:

$$(1 - 0.5B)Z_t = (1 - 0.2B)(1 - 0.4B^{12})a_t$$
$$Z_t = 0.5Z_{t-1} + a_t - 0.2a_{t-1} - 0.4a_{t-12} + 0.08a_{t-13}$$

The main issue for the users constructing ARIMA models is to select the proper order of the ARIMA models. Hyndman and Khandakar (2008) proposed an R package "forecast" related to automatic time series forecasting. Within this package, the function "auto.arima" selects the best order for the data set according to either Akaike information criterion (AIC), corrected Akaike information criterion (AICc) or Bayesian information criterion (BIC) while constructing ARIMA model. The details of the methodology can be found in Hyndman and Khandakar (2008).

3.7.2. Exponential Smoothing

The history of exponential smoothing is based on the 1950s, development of the methods is relatively recent. Exponential smoothing is the method which smooths the fluctuation of time series by giving the exponentially decreasing weights for the observations getting older. Hyndman and Khandakar (2008) studied a total of fifteen methods with additive and multiplicative errors. These methods are given in Table 3.2. They also proposed the R package "forecast" in which there exists a function called "ets". This function selects the appropriate exponential technique for the data set according to either AIC, AICc or BIC.

Table 3.2: The fifteen exponential smoothing methods with additive and multiplicative errors

Errors		Trend Component	Seasonal Component		
			N (None)	A (Additive)	M (Multipl.)
A (Additive)	N	(None)	A, N, N	A, N, A	A, N, M
	A	(Additive)	A, A, N	A, A, A	A, A, M
	Ad	(Additive damped)	A, Ad, N	A, Ad, A	A, Ad, M
	M	(Multipl.)	A, M, N	A, M, A	A, M, M
	Md	(Multipl. damped)	A, Md, N	A, Md, A	A, Md, M
M (Multipl.)	N	(None)	M, N, N	M, N, A	M, N, M
	A	(Additive)	M, A, N	M, A, A	M, A, M
	Ad	(Additive damped)	M, Ad, N	M, Ad, A	M, Ad, M
	M	(Multipl.)	M, M, N	M, M, A	M, M, M
	Md	(Multipl. damped)	M, Md, N	M, Md, A	M, Md, M

The components of the triplet (E, T, S) are called: error, trend and seasonality. For instance, the model (A, M, N) means additive errors, multiplicative trend and no seasonality. People call some of the methods given in Table 3.2 with the well-known names. The first cell (A, N, N) is known as the simple exponential smoothing with additive errors. The cell (A, A, N) is called Holt's linear method with additive errors which is good to explain a time series with trend. The method (M, A_d, N) is known as the damped trend method with multiplicative errors. The method (A, A, A) is named as the additive Holt-Winters' method with additive errors errors which is good to explain a time series with trend and seasonality. The cell (M, A, M) is called the multiplicative Holt-Winters' method with multiplicative errors. The details of these methods can be found in Hyndman and Khandakar (2008; see also Ord et al., 1997; Hyndman et al., 2002; Hyndman et al., 2005b).

CHAPTER 4

APPLICATION OF THE ALGORITHMS ON REAL LIFE DATASETS

GMDH algorithms are illustrated on four real life data sets in this chapter. These time series are well-known data sets and permanently used in time series text books. We mainly focus on answering the following three questions on these well-known data sets. How do GMDH algorithms perform when trend exists? How do GMDH algorithms perform for the data set with small sample size? What is the performance of GMDH algorithms when seasonality is available in the data set?

The main objective of this section is to show the performance of the GMDH-type neural network algorithms on real data applications with respect to prediction and short term forecasting accuracy. For this purpose, the last five observations of each data set were separated to show forecasting ability in short term.

In this chapter, we present information regarding the data sets. We implemented GMDH algorithms on real data sets. Also, we applied ARIMA models and exponential smoothing methods on real data applications for the comparison purpose. Moreover, we introduce our proposed R package "GMDH" for the implementation of the GMDH algorithms.

4.1. Cancer Death Rate

Data used in this application are yearly cancer death rate (per 100,000 population) of Pennsylvania between 1930 and 2000 (Figure 4.1). The data were documented in Pennsylvania Vital Statistics Annual Report by the Pennsylvania Department of Health in 2000 (Wei, 2006).

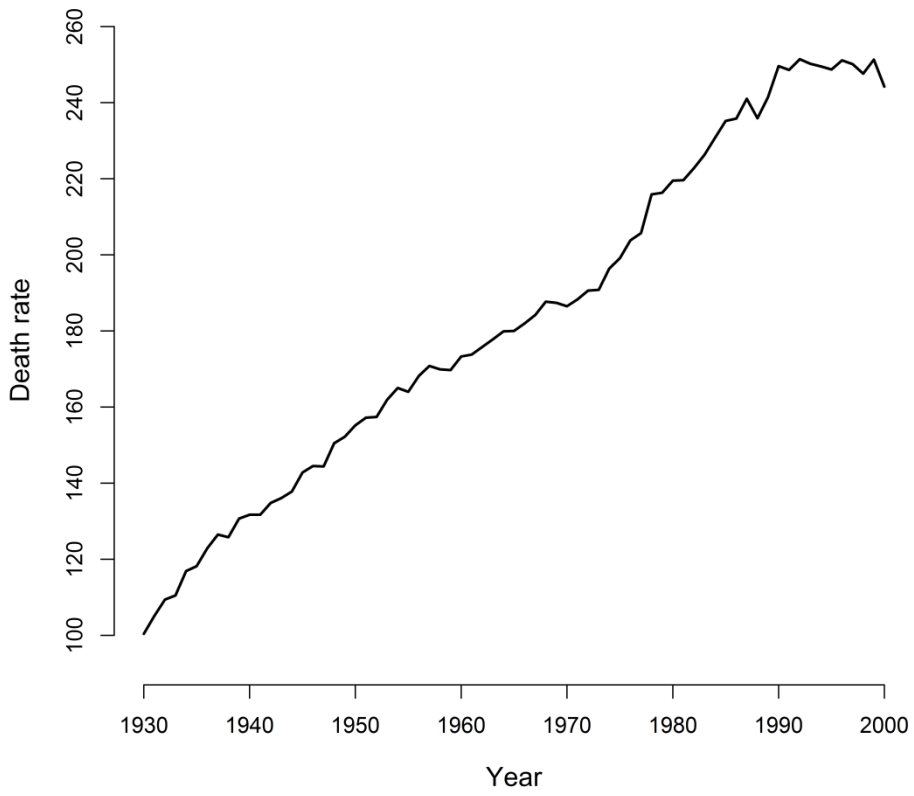

Figure 4.1: Yearly cancer death rate (per 100,000 population) in Pennsylvania between 1930 and 2000

This dataset is also available as a demo dataset in our R package GMDH. After installing package “GMDH”, it can be loaded in R workspace by

```
R> library(GMDH)
R> data(cancer)
R> cancer
```

After the cancer death rate data set is loaded, “fcast” function is utilized for forecasting via GMDH-type neural network.

```
R> out = fcast(cancer, method = "GMDH", input = 15, layer = 1, f.number = 5, tf = "all", plotit = TRUE)
R> out$fitted # displays fitted values
R> out$MSE # returns the MSE value of prediction
```

R> out$forecasts # shows forecasts

In this part, we divided the data into 2 parts for the aim of observing the ability of methods on prediction (n = 66) and forecasting (n = 5). We include ARIMA models and ES methods for the comparison purpose. For the determination of the best order of ARIMA models and the best method of ES techniques, there are two functions in R package "forecast" (Hyndman et al., 2014). These functions, auto.arima and ets, which use grid search, select the best model according to the criteria of either AIC, AICc or BIC. The functions suggested the model ARIMA (1, 1, 0) with intercept and ES method with multiplicative errors, additive damped trend and no seasonality (M, Ad, N), respectively. We also added the model ARIMA (0, 1, 0) with intercept for this data set suggested by Wei (2006). For all models, prediction mean square error (PMSE) and forecasting mean square error (FMSE) are stated in Table 4.1.

Table 4.1: Comparison of GMDH algorithms with other models on cancer death rate

	PMSE	FMSE
GMDH	4.985	4.575
RGMDH	4.287	4.102
ARIMA(1, 1, 0) with intercept	5.995	81.874
ARIMA(0, 1, 0) with intercept	6.324	73.756
ES (M, Ad, N)	6.153	17.508

The best forecasting performance belongs to RGMDH algorithm and its prediction accuracy also yields better results compared GMDH, ARIMA and ES models. Moreover, GMDH algorithm outperforms ARIMA and ES models in prediction and forecasting. To avoid visual pollution in Figure 4.2, we include one of the GMDH algorithms and one of ARIMA models or ES method which have higher performance with respect to forecasting compared to the rest of them. Figure 4.2 includes predictions and forecasts of RGMDH algorithm and ES (M, Ad, N).

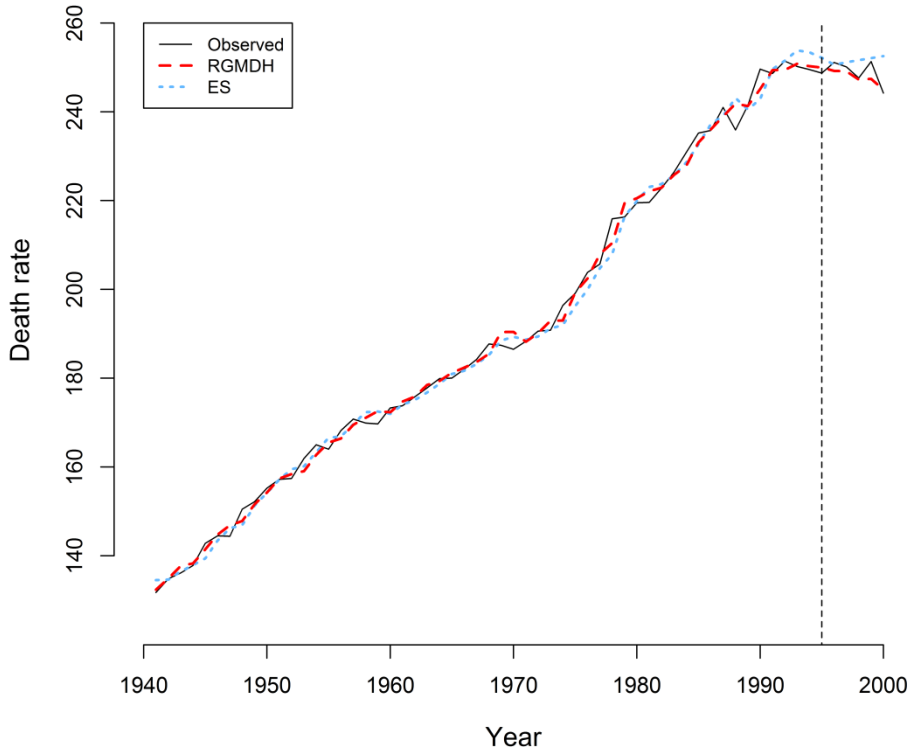

Figure 4.2: Yearly cancer death rate (per 100,000 population) in Pennsylvania between 1941 and 2000 with predictions and forecasts obtained via RGMDH and ES(M,Ad,N)

4.2. Melanoma Incidence

Data used in this part are melanoma skin cancer incidences (per 100,000 people) which represent age-adjusted numbers of melanoma skin cancer. The data are provided from Connecticut Tumor Registry in Connecticut from 1936 to 1972. The data and its description are available in R package "lattice" (Sarkar, 2014) under the name of melanoma. The illustration of melanoma skin cancer incidence data set is given in Figure 4.3. The dataset is a very short time series to model. Our aim in choosing this dataset is to see the performance of the GMDH algorithms in a short series.

Figure 4.3: Melanoma skin cancer incidence (per 100,000 people) in Connecticut between 1936 and 1972

Melanoma skin cancer incidence data set is divided into 2 parts for the purpose of observing the ability of methods on prediction (n = 31) and forecasting (n = 5). In order to find best order of ARIMA models, auto.arima suggested ARIMA(0, 1, 0) with intercept by comparing the models according to AIC, AICc and BIC. Function ets suggested ES method with multiplicative errors, additive trend and no seasonality (M, A, N), respectively. For all models, PMSE and FMSE are stated in Table 4.2.

Table 4.2: Comparison of GMDH algorithms with other models on melanoma incidence

	PMSE	FMSE
GMDH	0.034	0.095
RGMDH	0.034	0.095
ARIMA(0, 1, 0) with intercept	0.129	0.403
ES (M, A, N)	0.118	0.044

The performance of GMDH and RGMDH are same in both prediction and forecasting. It is plausible since the architecture of RGMDH is an extension of GMDH's. The best prediction performance belongs to GMDH algorithms. ES yields better results compared GMDH algorithms and ARIMA model with respect to forecasting. However, ES method does not capture the behavior of the data set whereas GMDH algorithms do (Figure 4.4). GMDH algorithms are able to be applied if the length of series is short.

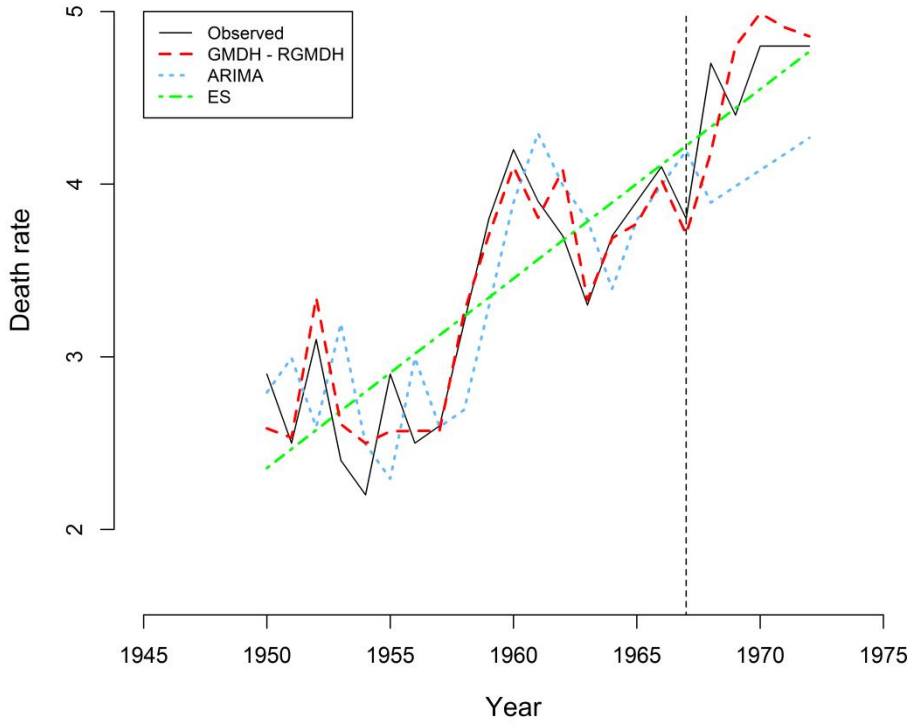

Figure 4.4: Melanoma skin cancer incidence (per 100,000 people) in Connecticut between 1950 and 1972 with predictions and forecasts obtained from GMDH-RGMDH, ARIMA(0,1,0) and ES(M,A,N)

4.3. Accidental Deaths

Data utilized in this part are monthly totals of accidental deaths in the US from 1973 to 1978. The data are able to be reached in Brockwell and Davis (1991). The data and its description are also available in R package "datasets" (R Core Team,

2014) under the name of USAccDeaths. The accidental deaths data set is illustrated in Figure 4.5. There is seasonality in the series like many time series.

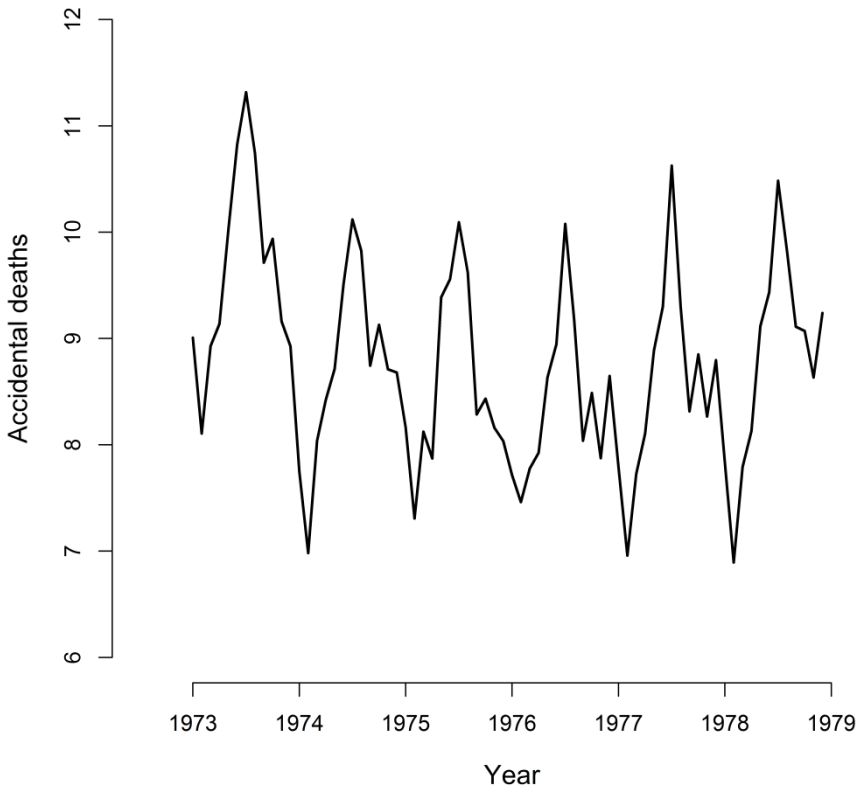

Figure 4.5: Monthly totals of accidental deaths (×1,000) in the US from 1973 to 1978

In this part, the data are divided into 2 parts for the performance of prediction (n = 67) and forecasting (n = 5). ARIMA models and ES methods are included for the comparison purpose. For the purpose of finding the best order of ARIMA models and the best method of ES techniques, the functions auto.arima and ets are utilized. These functions suggested the model ARIMA (0, 1, 1) (0, 1, 1) [12] and ES method with multiplicative errors, no trend and multiplicative seasonal component (M, N, M), respectively. For all models, PMSE and FMSE are presented in Table 4.3.

Table 4.3: Comparison of GMDH algorithms with other models on accidental deaths

	PMSE	FMSE
GMDH	0.130	0.088
RGMDH	0.068	0.239
ARIMA(0, 1, 1) (0, 1, 1) [12]	0.082	0.197
ES (M, N, M)	0.063	0.146

GMDH algorithm outperforms the rest ones in forecasting performance; however, it does not perform as well as the other methods in prediction. The best prediction performance belongs to ES technique and its forecasting performance is also fairly good. RGMDH algorithm's prediction accuracy also yields better results compared to GMDH and ARIMA, but its forecasting accuracy is not as good as the rest models. To prevent visual pollution, for only RGMDH algorithm and ES method, the predicted values and forecasts are illustrated in Figure 4.6.

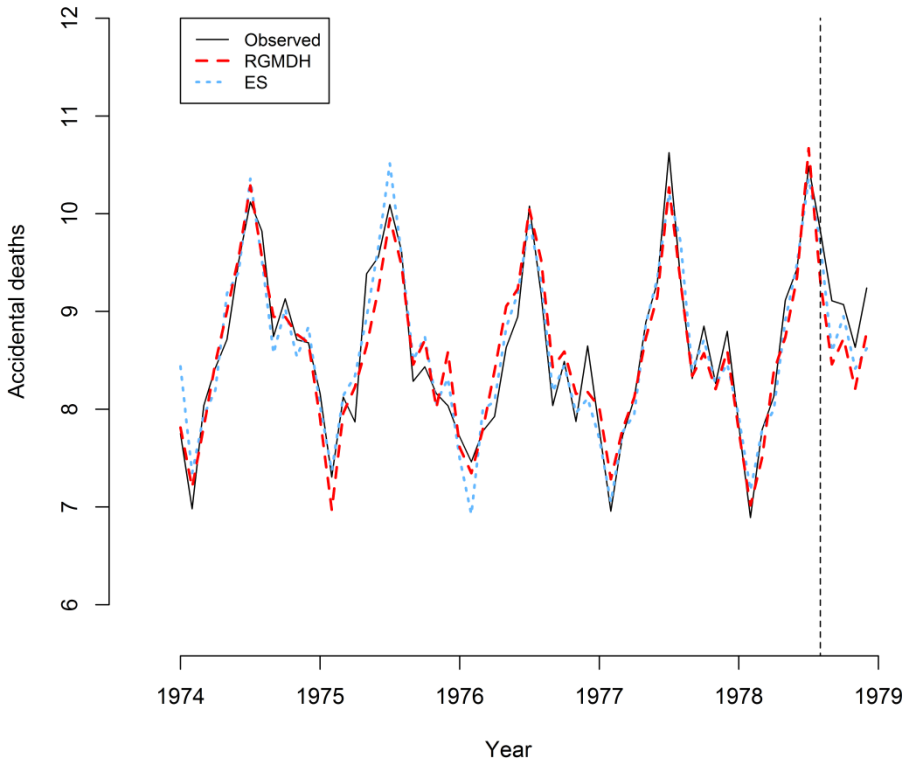

Figure 4.6: Monthly totals of accidental deaths (×1,000) in the US from 1974 to 1978 with the predictions and forecasts obtained from RGMDH and ES(M,N,M)

4.4. Airline Passenger Numbers

Data used in this part are monthly totals of international airline passengers from 1949 to 1960; therefore, it includes 144 observations. The data and its description are able to be reached in Box et al. (1976). The data are also available in R package "datasets" (R Core Team, 2014) under the name of AirPassengers. Airline passenger numbers data set is illustrated in Figure 4.7. This dataset has an increasing trend and seasonal behavior.

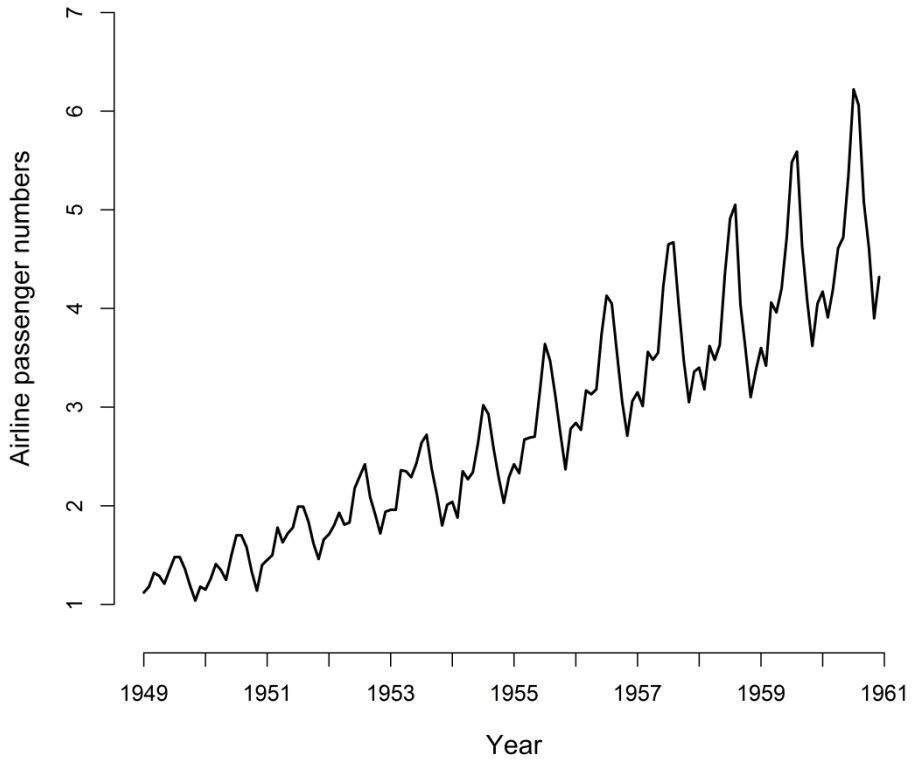

Figure 4.7: Monthly totals of international airline passengers (×100,000) from 1949 to 1960

We divided the data set into 2 sets in order to show the performance of GMDH algorithms with respect to prediction (n = 139) and forecasting (n = 5). ARIMA models and ES methods are included for the comparison purpose. For ARIMA models, the best order of fitting this data set is ARIMA(0, 1, 1) (0, 1, 0) [12]. The method which fit airline passenger numbers data set is found to be ES method with

multiplicative errors, additive trend and multiplicative seasonal component (M, A, M). For all models, PMSE and FMSE are presented in Table 4.4.

Table 4.4: Comparison of GMDH algorithms with other models on airline passenger numbers

	PMSE	FMSE
GMDH	0.020	0.015
RGMDH	0.009	0.018
ARIMA(0, 1, 1) (0, 1, 0) [12]	0.012	0.098
ES (M, A, M)	0.011	0.063

GMDH and RGMDH are superior to ARIMA and ES for the forecasting performance. The best forecasting performance belongs to GMDH algorithm, but its prediction accuracy is not as good as RGMDH, ARIMA and ES models. In order to visualize prediction and forecasting performance, RGMDH algorithm and ES method are illustrated in Figure 4.8. GMDH algorithms are able to be applicable even if both seasonality and trend are available in the data set.

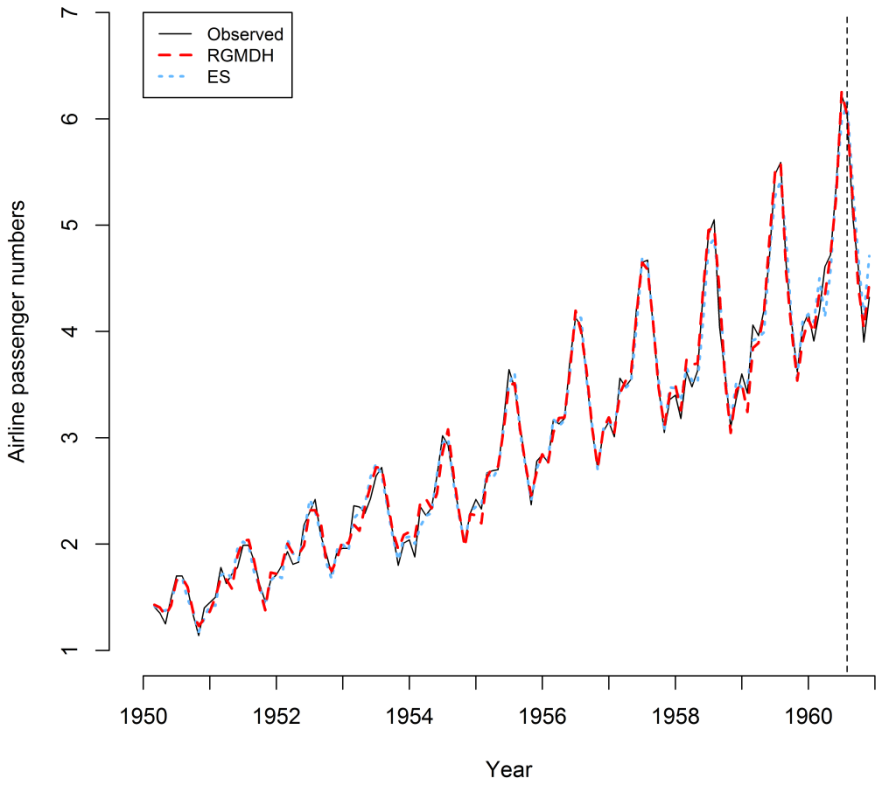

Figure 4.8: Monthly totals of international airline passengers (×100,000) from 1949 to 1960 with predictions and forecasts obtained from RGMDH and ES(M,A,M)

4.5. Discussion

In the light of whole analysis on the data sets included, GMDH – type neural network algorithms generally outperform ARIMA and ES models in prediction and forecasting performance even if the series is short or/and includes trend or/and seasonality. As an alternative to ARIMA and ES models, GMDH algorithms are applicable for prediction and forecasting. In this study, we proposed an R package GMDH for easy implementation of users. Therefore, researchers are able to reach these algorithms in an easy way.

CHAPTER 5

CONCLUSION

In this study, we used GMDH - type neural network algorithms, the heuristic self-organization method for modelling the complex systems, to make forecasts for time series data sets. We primarily concentrated to develop free software. Concretely, we developed an R package called GMDH to make forecasting in short term via GMDH - type neural network algorithms. Also, we integrated different transfer functions, sigmoid, radial basis, polynomial, and tangent functions, into GMDH algorithms. Our R package proposed that these functions are able to be exerted simultaneously or separately depending on the desire.

In estimation of coefficients, since we construct the model for the data with lags, there exists high possibility of occurring multi-collinearity problem. Therefore, we utilized regularized least square estimation to handle such a problem. It is important to note that estimation of regularization parameter is the question of interest. Cross validation was applied in order to estimate regularization term. Data were divided into two pieces as a learning set and a testing set. Order of the observations was important in time series data; therefore, division was done by taking that into consideration. Coefficients were estimated by using the learning set and MSE was calculated by utilizing test set. The regularization parameter which gave the smallest MSE in all possible regularization parameters was selected. After selection of regularization term, coefficients were estimated by the help of all observations and regularization parameter. It is important to point that regularized least square is equal to least square estimation when regularization parameter is estimated to be zero.

Application of the algorithms on real life datasets illustrated that GMDH – type neural network algorithms are as good as ARIMA and ES models or better in prediction and short term forecasting performance. GMDH algorithms are able to

be applied even if the length of the series is short. Also, they are able to be used when trend and/or seasonality exist(s) in the data set. Researchers are able to reach these algorithms since our proposed R package GMDH is available on Comprehensive R Archive Network (CRAN) at http://CRAN.R-project.org/package=GMDH.

One of the questions is "what will be the number of inputs?". The answer of this question is not that easy since increase in number of inputs causes decrease in the number of observations. Therefore, it depends on the length of the series. It means providing that there exist very long time series, it is better to use high number of inputs. This of course results in very large computational time. If the sample size is small, it is not plausible to use large number of inputs. In this case, optimum feasible input numbers are tried to be exerted, the number of input is able to be chosen according to MSE.

Future studies are planned in the direction of transfer functions. In this study, we used four different transfer functions - sigmoid, radial basis, polynomial, and tangent functions - into GMDH algorithms. We plan to integrate Box-Cox transformation into GMDH-type neural network algorithms. GMDH algorithms with four transfer functions and GMDH algorithms with Box-Cox transformation are going to be performed on real data applications to compare the prediction and short term forecasting. After well-documented, the related R function of GMDH algorithms with Box-Cox transformation are going to be released under our proposed R package GMDH.

REFERENCES

Abdel-Aal, R. E. (2005). GMDH-Based Feature Ranking and Selection for Improved Classification of Medical Data. *Journal of Biomedical Informatics*, 38, 456-468.

Astakhov, V. P., Galitsky, V. V. (2005). Tool Life Testing in Gundrilling: An Application of the Group Method of Data Handling (GMDH). *International Journal of Machine Tools & Manufacture*, 45, 509-517.

Baig, Z. A., Sait, S. M., Shaheen, A. (2013). GMDH-Based Networks for Intelligent Intrusion Detection. *Engineering Applications of Artificial Intelligence*, 26, 1731–1740.

Box, G. E. P., Jenkins, G. M. (1970). Time Series Analysis, Forecasting and Control. Oakland, CA: Holden-Day.

Box, G. E. P., Jenkins, G. M. and Reinsel, G. C. (1976). *Time Series Analysis, Forecasting and Control*. Third Edition. Holden-Day. Series G.

Brockwell, P. J. and Davis, R. A. (1991). *Time Series: Theory and Methods.* Springer, New York.

Farlow, S. J. (1981). The GMDH Algorithm of Ivakhnenko. *The American Statistician*, 35:4, 210-215.

Hyndman, R. J., Athanasopoulos, G., Razbash, S., Schmidt, D., Zhou, Z., Khan, Y., Bergmeir, C., and Wang, E. (2014). forecast: Forecasting functions for time series and linear models. R package version 5.5. http://CRAN.R-project.org/package=forecast.

Hyndman, R.. J., Khandakar, Y. (2008). Automatic Time Series Forecasting: The forecast Package for R. *Journal of Statistical Software*, 27:3, 1-22.

Hyndman, R.J., Koehler, A.B., Ord, J.K., Snyder, R.D. (2005). Prediction Intervals for Exponential Smoothing Using Two New Classes of State Space Models. *Journal of Forecasting*, 24, 17-37.

Hyndman, R.J., Koehler, A.B., Snyder, R.D., Grose, S. (2002). A State Space Framework for Automatic Forecasting Using Exponential Smoothing Methods. *International Journal of Forecasting*, 18:3, 439-454.

Ivakhnenko, A. G. (1966). Group Method of Data Handling – A Rival of the Method of Stochastic Approximation. *Soviet Automatic Control*, 13, 43-71.

Ivakhnenko, A. G. (1970). Heuristic Self-Organization in Problems of Engineering Cybernetics. *Automatica*, 6:2, 207-219.

Ivakhnenko, A. G., Ivakhnenko, G. A. (1995). The Review of Problems Solvable by Algorithms of the Group Method of Data Handling (GMDH). *Pattern Recognition and Image Analysis*, 5:4, 527-535.

Kalavrouziotis, I., Stepashko, V., Vissikirsky, V., Drakatos, P. (2002). Group Method of Data Handling (GMDH) Application for Modelling of Mechanical Properties of Trees Irrigated with Wastewater. *International Journal of Environment and Pollution*, 18:6, 589-601.

Kondo, T. (1998). GMDH Neural Network Algorithm Using the Heuristic Self-Organization Method and Its Application to the Pattern Identification Problem. *Proc. of the 37th SICE Annual Conference, 1143-1148.*

Kondo, T., Ueno, J. (2006a). Revised GMDH-Type Neural Network Algorithm with a Feedback Loop Identifying Sigmoid Function Neural Network.

International Journal of Innovative Computing, Information and Control, 2:5, 985-996.

Kondo, T., Ueno, J. (2006b). Medical Image Recognition of the Brain by Revised GMDH-Type Neural Network Algorithm with a Feedback Loop. *International Journal of Innovative Computing, Information and Control*, 2:5, 1039-1052.

Kondo, T., Ueno, J. (2007). Logistic GMDH-Type Neural Network and Its Application to Identification of X-ray Film Characteristic Curve. *Journal of Advanced Computational Intelligence and Intelligent Informatics*, 11:3, 312-318.

Kondo, T., Ueno, J. (2012). Feedback GMDH-Type Neural Network and Its Application to Medical Image Analysis of Liver Cancer. *International Journal of Innovative Computing, Information and Control*, 8:3(B), 2285-2300.

Muller, J. A., Ivachnenko, A. G., Lemke, F. (1998). GMDH Algorithms for Complex Systems Modelling. *Mathematical and Computer Modelling of Dynamical Systems: Methods, Tools and Applications in Engineering and Related Sciences*, 4:4, 275–316.

Najafzadeh, M., Barani, G., Hessami Kermani, M. (2014). Estimation of Pipeline Scour due to Waves by GMDH. *Journal of Pipeline Systems Engineering and Practice*, 5:3, 06014002.

Nariman-zadeh, N., Darvizeh, A., Darvizeh, M., Gharababaei, H. (2002). Modelling of Explosive Cutting Process of Plates Using GMDH-Type Neural Network and Singular Value Decomposition. *Journal of Materials Processing Technology*, 128, 80-87.

Ord, J. K., Koehler, A. B., Snyder, R. D. (1997). Estimation and Prediction for a Class of Dynamic Nonlinear Statistical Models. *Journal of the American Statistical Association*, 92, 1621-1629.

R Core Team (2014). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL http://www.R-project.org/.

Samsudin, R., Saad, P., Shabri, A. (2011). River Flow Time Series Using Least Squares Support Vector Machines. *Hydrology and Earth System Sciences*, 15, 1835-1852.

Sarkar, D. (2014). Lattice: Multivariate Data Visualization with R. New York: Springer. ISBN 978-0-387-75968-5.

Sheikholeslami, M., Sheykholeslami F. B., Khoshhal, S., Mola-Abasia, H., Ganji, D. D., Rokni, H. B. (2014). Effect of Magnetic Field on Cu–Water Nanofluid Heat Transfer Using GMDH-Type Neural Network, *Neural Computing and Applications*, 25, 171-178.

Srinivasan, D. (2008). Energy Demand Prediction Using GMDH Networks. *Neurocomputing*, 72, 625-629.

Wei, W. W.S. (2006). Time Series Analysis: Univariate and Multivariate Methods. Boston: Addison-Wesley.

Xu, H., Dong, Y., Wu, J., Zhao,W. (2012). Application of GMDH to Short Term Load Forecasting. *Advances in Intelligent Systems*, 138, 27–32.

APPENDIX A

MANUAL OF OUR PROPOSED R PACKAGE GMDH

Package 'GMDH'

July 20, 2015

Type Package

Title Predicting and Forecasting Time Series via GMDH-Type Neural Network Algorithms

Version 1.1

Date 2015-7-20

Depends R (>= 3.1.1)

Imports MASS, graphics, stats, utils

Author Osman Dag, Ceylan Yozgatligil

Maintainer Osman Dag <osman.dag@hacettepe.edu.tr>

Description

Group method of data handling (GMDH) - type neural network algorithm is the heuristic self- organization method for modelling the complex systems. In this package, GMDH-type neural network algorithms are applied to predict and forecast a univariate time series.

License GPL (>= 2)

NeedsCompilation no

Repository CRAN

Date/Publication 2015-07-20 18:08:41

R topics documented:

GMDH-package	Predicting and Forecasting Time Series via GMDH-Type Neural Net- work Algorithms

Description

Package GMDH includes a function for predicting and forecasting a univariate time series by using GMDH-type neural network, and a dataset for implementation.

Details

Package: GMDH
Type: Package
Version: 1.1
Date: 2015-07-20
License: GPL (>=2)

cancer	Cancer Data

Description

Yearly cancer death rate (per 100,000 population) of Pennsylvania between 1930 and 2000.

Usage

data(cancer)

Format

A time series with 71 observations on the following variable.

cancer a time series for yearly cancer death rate

References

Wei, W. W.S. (2006). Time Series Analysis: Univariate and Multivariate Methods (2nd ed.) Boston:Addison-Wesley

Examples

```
data(cancer)
plot(cancer)
out  = fcast(cancer, f.number = 2)
out$forecasts
```

---

fcast	A Function to Predict and Forecast Time Series via GMDH-Type Neural Network Algorithms

---

Description

fcast predicts and forecasts time series via GMDH-type neural network algorithms.

Usage

```
fcast(data, method = "GMDH",  input = 4, layer = 3,  f.number = 10,  tf
= "all", plotit = TRUE, weigth = 0.7, lambda=c(0,0.01,0.02,0.04,0.08,
0.16, 0.32,0.64,1.28,2.56,5.12, 10.24))
```

Arguments

data	is an univariate time series
method	expects a character string to choose the desired method to forecast time series. To utilize GMDH-type neural network in forecasting, method is set to "GMDH". One should set method to "RGMDH" for forecasting via Revised GMDH-type neural network. Default is set to "GMDH"
input	is the number of inputs. Defaults input = 4
layer	is the number of layers. Default is set to layer = 3
f.number	is the number of observations to be forecasted. Defaults f.number = 10
tf	expects a character string to choose the desired transfer function to be used in forecasting. To use polynomial

function, tf should be set to "polynomial". Similarly, tf should be set to "sigmoid", "RBF", "tangent" to utilize sigmoid function, radial basis function and tangent function, respectively. To use all functions simultaneously, default is set to "all"

plotit — is logical which controls whether historical data with forecasts should be plotted. Defaults plotit = TRUE

weigth — is the percent of the data set to be utilized as learning set to estimate regularization parameter via cross validation. Default is set to weigth = 0.70

lambda — is a vector which includes the sequence of feasible regularization parameters. Defaults lambda=c(0,0.01,0.02, 0.04,0.08,0.16,0.32,0.64, 1.28,2.56,5.12,10.24)

Value

Returns a list containing following elements:

fitted — fitted values

MSE — MSE of prediction

forecasts — forecasts

Note

This is the version 1.1 of this user documentation file.

Author(s)

Osman Dag, Ceylan Yozgatligil

References

Dag, O., Yozgatligil, C. (2015). GMDH: An R Package for Predicting and Forecasting Time Series via GMDH-Type Neural Network Algorithms. To be submitted.

Ivakhnenko, A. G. (1966). Group Method of Data Handling - A Rival of the Method of Stochastic Approximation. Soviet Automatic Control, 13, 43-71.

Kondo, T., Ueno, J. (2006). Revised GMDH-Type Neural Network Algorithm With A Feedback Loop Identifying Sigmoid Function Neural Network. International Journal of Innovative Comput- ing, Information and Control, 2:5, 985-996.

Examples

```
data = rnorm(100, 10,  1)
out  = fcast(data)
out


data = rnorm(100, 10,  1)
out  = fcast(data,  input = 6, layer = 2,  f.number = 5)
out$forecasts
out$fitted
out$MSE
```

APPENDIX B

OUR R FUNCTION FOR FORECASTING VIA GMDH ALGORITHMS

```
fcast=function(data, method="GMDH", input=4, layer=3, f.number=5, tf="all", plotit=TRUE, weigth=0.7, lambda=c(0,0.01,0.02,0.04,0.08,0.16,0.32,0.64,1.28, 2.56,5.12,10.24)){

if (tf=="all"){tf_options=c(101:104)
}else if (tf=="polynomial"){tf_options=c(101)
}else if (tf=="sigmoid"){tf_options=c(102)
}else if (tf=="RBF"){tf_options=c(103)
}else if (tf=="tangent"){tf_options=c(104)

}else {stop("Transfer function you entered is not available")}

transf=function(h,dataaa){

if (h==101) dat=dataaa
if (h==102) dat=log(dataaa/(1-dataaa))
if (h==103) dat=sqrt(-log(dataaa))
if (h==104) dat=atan(dataaa)/pi*180
dat
}


back_transf=function(h,dataaa){
if (h==101) dat=dataaa
if (h==102) dat=1/(1+exp(-dataaa))
if (h==103) dat=exp(-dataaa^2)
```

```
if (h==104) dat=tan(dataaa*pi/180)
dat
}


cross=function(X, y, lambda=lambda, weigth=weigth){
n=length(y)
n1=round(n*weigth)
n2=n-n1
p=dim(X)[2]
store=NULL
cost=NULL
Ident=diag(p)
Ident[1,1]=0

X1=X[1:n1,]
X2=X[(n1+1):n,]
y1=matrix(y[1:n1],ncol=1)
y2=y[(n1+1):n]
for (j in 1:length(lambda)){
coef=ginv(t(X1)%*%X1+lambda[j]*Ident)%*%t(X1)%*%y1
ypred= t(coef)%*%t(X2)
cost=mean((ypred-y2)^2)
store=rbind(store,c(lambda[j],cost))
cost=NULL
}
lamb=store[which.min(store[,2]),][1]

coef2=ginv(t(X)%*%X+lamb*Ident)%*%t(X)%*%y
as.numeric(coef2)
```

```
}


if (min(data)<=0){
stt1=abs(min(data))+1
}else{stt1=0}

stt2=max(data+stt1)+1
y=(data+stt1)/stt2
if (method=="GMDH"){

store_Astore<- list()
store_z=list()
ss=length(y)

threshold=c(rep(input,layer-1),1)
nnode=input*(input-1)/2
idn=c(1:input)

yt=y[-input:-1]
x=NULL
for (i in 1:(input-1)){
x=cbind(x,matrix(y[c(-1:-(input-i),-ss:-(ss+1-i))]))
}
x=cbind(x,matrix(y[c(-ss:-(ss-input+1))]))

for (k in 1:layer){
w=t(combn(order(idn), 2))

Astore=NULL
```

```
z=NULL
for (j in 1:nnode){

qq=cbind(1,x[,w[j,]],x[,w[j,]][,1]*x[,w[j,]][,2],x[,w[j,]]^2)



tfunc=NULL
tfunc_z=NULL
for (g in tf_options){

est_coef=cross(qq,transf(g,yt),lambda=lambda,weigth=weigth)
ee=as.numeric(est_coef)

est_zt=rowSums(t(ee*t(qq)))
tfunc=rbind(tfunc,c(est_coef,mean((back_transf(g,est_zt)-yt)^2),g))
tfunc_z=cbind(tfunc_z,matrix(back_transf(g,est_zt)))
}

z=cbind(z,tfunc_z[,which.min(tfunc[,7])])
Astore=rbind(Astore,tfunc[which.min(tfunc[,7]),])
}

Astore=cbind(Astore,c(1:nnode))
store_Astore[[k]]=Astore[which(Astore[,7]<=sort(Astore[,7])[threshold[k]]),]
store_z[[k]]=z[,which(Astore[,7]<=sort(Astore[,7])[threshold[k]])]

x=store_z[[k]]
if (k==layer){
store_Astore[[k]]=matrix(store_Astore[[k]],nrow=1)
store_z[[k]]=matrix(store_z[[k]],ncol=1)
```

```
}
}

for (h in 1:f.number){

yt_input=matrix(rev(tail(y,input)),nrow=1)
idn2=c(1:input)
w2=t(combn(order(idn2), 2))

for (k2 in 1:layer){

selected_coef=selected_qq2=NULL
store_qq2=NULL
for (j2 in 1:nnode){

qq2=c(1,yt_input[,w2[j2,]],yt_input[,w2[j2,]][1]*yt_input[,w2[j2,]][2],yt_input[,w
2[j2,]]^2)
store_qq2=rbind(store_qq2,qq2)
}

selected_qq2=store_qq2[store_Astore[[k2]][,9],]
selected_coef=store_Astore[[k2]][,1:6]

if (k2==layer){
selected_qq2=matrix(selected_qq2,nrow=1)
selected_coef=matrix(selected_coef,nrow=1)
}

yt_input=matrix(rowSums(selected_qq2*selected_coef),nrow=1)
```

```
for (k5 in 1:threshold[k2]){
yt_input[1,k5]=back_transf(store_Astore[[k2]][k5,8],yt_input[1,k5])
}
}
y=c(y,yt_input)
}

fitted=store_z[[layer]][,1]*stt2-stt1
}

if (method=="RGMDH"){

store_Astore<- list()
store_z=list()
store_Astore2<- list()
store_z2=list()
ss=length(y)

threshold=c(rep(input,layer-1),1)
nnode=input*(input-1)/2+input
p=input*(input-1)/2
idn=c(1:input)

yt=y[-input:-1]
x=NULL
for (i in 1:(input-1)){
x=cbind(x,matrix(y[c(-1:-(input-i),-ss:-(ss+1-i))]))
}
x=cbind(x,matrix(y[c(-ss:-(ss-input+1))]))
```

```
for (k in 1:layer){

w=t(combn(order(idn), 2))
m2 <- matrix(rep(1:input,input),input,input,byrow=T)
m2[upper.tri(m2)] <-0

Astore=NULL
z=NULL
z2=NULL
Astore2=NULL

for (j in 1:nnode){

if (j<=p){
qq=cbind(1,x[,w[j,]],x[,w[j,]][,1]*x[,w[j,]][,2],x[,w[j,]]^2)
}else{
qq=cbind(1,x[,m2[j-p,]])
}

tfunc=NULL
tfunc_z=NULL
tfunc2=NULL
tfunc_z2=NULL

if (j<=p){

for (g in tf_options){
est_coef=cross(qq,transf(g,yt),lambda=lambda,weigth=weigth)
ee=as.numeric(est_coef)
```

```
est_zt=rowSums(t(ee*t(qq)))
tfunc=rbind(tfunc,c(est_coef,mean((back_transf(g,est_zt)-yt)^2),g))
tfunc_z=cbind(tfunc_z,matrix(back_transf(g,est_zt)))
}
}else{

for (g in tf_options){
est_coef=cross(qq,transf(g,yt),lambda=lambda,weigth=weigth)
coef=c(est_coef,rep(0,input+1-length(est_coef)))
ee=as.numeric(est_coef)
est_zt=rowSums(t(ee*t(qq)))
tfunc2=rbind(tfunc2,c(coef,mean((back_transf(g,est_zt)-yt)^2),g))
tfunc_z2=cbind(tfunc_z2,matrix(back_transf(g,est_zt)))
}
}

if (j<=p){
z=cbind(z,tfunc_z[,which.min(tfunc[,7])])
Astore=rbind(Astore,tfunc[which.min(tfunc[,7]),])
}else{
z2=cbind(z2,tfunc_z2[,which.min(tfunc2[,(input+2)])])
Astore2=rbind(Astore2,tfunc2[which.min(tfunc2[,(input+2)]),])
}
}

Astore=cbind(Astore,c(1:p))
Astore2=cbind(Astore2,c((p+1):(p+input)))
checkk=rbind(Astore[,c(7,9)],Astore2[,c((input+2),(input+4))])
ord=which(checkk[,1]<=sort(checkk[,1])[threshold[k]])
ord1=ord[which(ord<=p)]
```

```
ord2=ord[which(ord>p)]

store_Astore[[k]]=Astore[ord1,]
store_Astore2[[k]]=Astore2[ord2-p,]
store_z[[k]]=z[,ord1]
store_z2[[k]]=z2[,ord2-p]

x=cbind(store_z[[k]],store_z2[[k]])

if (class(store_Astore[[k]])!="matrix"){
store_Astore[[k]]=matrix(store_Astore[[k]],nrow=1)
store_z[[k]]=matrix(store_z[[k]],ncol=1)
}

if (class(store_Astore2[[k]])!="matrix"){
store_Astore2[[k]]=matrix(store_Astore2[[k]],nrow=1)
store_z2[[k]]=matrix(store_z2[[k]],ncol=1)
}
}

for (h in 1:f.number){

yt_input=matrix(rev(tail(y,input)),nrow=1)
idn2=c(1:input)
w2=t(combn(order(idn2), 2))

for (k2 in 1:layer){

selected_coef=selected_qq2=NULL
store_qq2=NULL
```

```
selected_coef5=selected_qq5=NULL
store_qq5=NULL


for (j2 in 1:nnode){

if (j2<=p){
qq2=c(1,yt_input[,w2[j2,]],yt_input[,w2[j2,]][1]*yt_input[,w2[j2,]][2],yt_input[,w
2[j2,]]^2)
store_qq2=rbind(store_qq2,qq2)
}else{
qq2=c(1,yt_input[,m2[j2-p,]],rep(0,input-(j2-p)))
store_qq5=rbind(store_qq5,qq2)
}
}

selected_qq2=store_qq2[store_Astore[[k2]][,9],]
selected_coef=store_Astore[[k2]][,1:6]
selected_qq5=store_qq5[store_Astore2[[k2]][,(input+4)]-p,]
selected_coef5=store_Astore2[[k2]][,1:(input+1)]

if (class(selected_qq2)!="matrix"){
selected_qq2=matrix(selected_qq2,nrow=1)
selected_coef=matrix(selected_coef,nrow=1)
}

if (class(selected_qq5)!="matrix"){
selected_qq5=matrix(selected_qq5,nrow=1)
selected_coef5=matrix(selected_coef5,nrow=1)
}
```

```
uu1=matrix(rowSums(selected_qq2*selected_coef),nrow=1)
uu2=matrix(rowSums(selected_qq5*selected_coef5),nrow=1)
d1=dim(matrix(rowSums(selected_qq2*selected_coef),nrow=1))[2]
d2=dim(matrix(rowSums(selected_qq5*selected_coef5),nrow=1))[2]

if(d1!=0){
for (k5 in 1:d1){
uu1[1,k5]=back_transf(store_Astore[[k2]][k5,8],uu1[1,k5])
}
}

if(d2!=0){
for (k5 in 1:d2){
uu2[1,k5]=back_transf(store_Astore2[[k2]][k5,(input+3)],uu2[1,k5])
}
}

yt_input=cbind(uu1,uu2)
}
y=c(y,yt_input)
}
fitted=cbind(store_z[[layer]],store_z2[[layer]])[,1]*stt2-stt1
}
forecast_values=tail(y*stt2-stt1,f.number)
MSE=(mean((fitted-data[c(-1:-input)])^2))
start2=start(data)[1]
if(plotit==TRUE){
plot(ts(c(data,forecast_values),start=start2),col="black",ylab="Time Series")
abline(v=start2+ss-1,lty=2)
```

```
}

out=list()
out$fitted=ts(fitted, start=start2+input,end=start2+ss-1)
out$MSE=MSE
out$forecasts=ts(forecast_values, start=start2+ss,end=start2+ss-1+f.number)
invisible(out)
}
```

APPENDIX C

R CODES FOR REAL DATA APPLICATIONS

```
###To draw cancer data
library(GMDH)
data(cancer)
tiff("a.tiff", res = 900, width = 6200, height = 5400)
par(mar=c(4.1, 4.1, 0, 0.3))
plot(ts(cancer,start=1930,end=2000), asp=0.36,axes = FALSE, col="black", xlab="", ylab="", lwd = 2, ylim = c(95,260), xlim=c(1930,2000)) ####original data
axis(1)
axis(2, at = seq(100,260, 20))
title(xlab= "Year",ylab="Death rate",cex.lab =1.2,  font.lab= 1)
dev.off()


###To draw cancer data with predictions and forecasts obtained via RGMDH and ES(M,Ad,N)
n=length(cancer)
input=11
layer=2
f.number=5
n1=n-f.number
data=cancer[1:n1]
poli1=cancer[(n1+1):n]
out=fcast(cancer[1:n1], method="RGMDH", input=input, layer=layer, f.number=f.number, plot=FALSE, weigth=0.7)
```

```
hj=out$forecasts
ee2=out$fitted

library(forecast)
fit=auto.arima(data)
ari=forecast(fit,f.number)$mean
ari2=forecast(fit,f.number)$fitted

a=(mean((ari-poli1)^2)) ###forecast MSE of ARIMA
b=(mean((ari2-data)^2)) ###prediction MSE of ARIMA

d=(mean((hj-poli1)^2)) ###forecast MSE of GMDH
e=(mean((ee2-data[c(-1:-input)])^2))

aaa=matrix(c(d,e,a,b),2,2)
colnames(aaa)=c("GMDH","ARIMA")
rownames(aaa)=c("forecast_MSE","prediction_MSE")
aaa

tiff("a.tiff", res = 900, width = 6200, height = 5400)

par(mar=c(4.1, 4.1, 0, 0.3))
plot(ts(cancer[-1:-input], start=1930+input,end=2000), asp=0.36, axes = FALSE,
col="black", xlab="", ylab="", lwd = 1,ylim = c(135,260), xlim=c(1940,2000))
####original data
axis(1, at = seq(1940,2000, 10))
axis(2, at = seq(140,260, 20))
lines(ts(c(ari2,ari)[-1:-input],start=1930+input,end=2000), col = "steelblue1", lwd
= 2, lty = 3 )  ####ARIMA
```

```
lines(ts(c(ee2,hj),start=1930+input,end=2000),col="red",lwd = 2,lty=2)
###GMDH

title(xlab= "Year",ylab="Death rate",cex.lab =1.2,  font.lab= 1)
arrows(2000-f.number, 100, 2000-f.number, 260, length=0, angle=90,lty=2)

legend(1940, 260, c("Observed","RGMDH","ES"), col = c("black","red","steelblue1"), inset = .05, cex = 0.8, lwd = c(1,2,2), lty=c(1,2,3))
dev.off()


###To draw melanoma incidence data

library(lattice)
data(melanoma)
a=melanoma[,2]
a=ts(a,start = 1936, end =1972)

tiff("a.tiff", res = 900, width = 6200, height = 5400)

par(mar=c(4.1, 4.1, 0, 0.3), cex.axis=1.0, cex.lab = 1.2, font.lab = 1)
plot(a, xlab="", ylab="", lwd = 2, asp=7, axes = FALSE, ylim = c(0.7,5), xlim=c(1935,1975))
axis(1, at = seq(1935,1975, 10))
axis(2, at = 1:5)
title(xlab= "Year",ylab="Melanoma incidence")

dev.off()
```

```
###To draw melanoma incidence data  with predictions and forecasts obtained from GMDH-RGMDH, ARIMA(0,1,0) and ES(M,A,N)

library(lattice)
data(melanoma)

mela=melanoma[,2]
mela=ts(mela,start = 1936, end =1972)

input=14
layer=2
f.number=5

n=length(mela)
n1=n-f.number
data=mela[1:n1]
poli1=mela[(n1+1):n]

out=fcast(data,method="GMDH",input=input,layer=layer,f.number=f.number,plot
=FALSE)

hj=out$forecasts
ee2=out$fitted

library(forecast)
fit=auto.arima(data)
fit2=ets(data)

ari=forecast(fit,f.number)$mean
ari2=forecast(fit,f.number)$fitted
```

```
ari3=forecast(fit2,f.number)$mean
ari4=forecast(fit2,f.number)$fitted

a=(mean((ari-poli1)^2))###forecast MSE of ARIMA
b=(mean((ari2-data)^2))

d=(mean((hj-poli1)^2))###forecast MSE of GMDH
e=(mean((ee2-data[c(-1:-input)])^2))

f=(mean((ari3-poli1)^2))###forecast MSE of ES
g=(mean((ari4-data)^2))

aaa=matrix(c(d,e,a,b,f,g),2,3)
colnames(aaa)=c("GMDH","ARIMA","ES")
rownames(aaa)=c("forecast_MSE","prediction_MSE")
aaa

round(aaa,3)

tiff("a.tiff", res = 900, width = 6200, height = 5400)

par(mar=c(4.1, 4.1, 0, 0.3))
plot(ts(mela[-1:-input],start=input+1936,end=1972), asp=7, axes = FALSE,
col="black", xlab="", ylab="", lwd = 1,ylim = c(2,5), xlim=c(1945,1975))
####original data
axis(1, at = seq(1945,1975, 5))
axis(2, at = seq(2,5, 1))

lines(ts(c(ari2,ari)[-1:-input], start=input+1936,end=1972),col = "steelblue1", lwd
= 2, lty = 3)  ####ARIMA
```

```
lines(ts(c(ari4,ari3)[-1:-input], start=input+1936,end=1972), col="green", lwd = 2,lty=4) ####ES

lines(ts(c(ee2,hj),start=input+1936,end=1972),col="red",lwd = 2,lty=2) ###GMDH
title(xlab= "Year",ylab="Death rate",cex.lab =1.2, font.lab= 1)
arrows(1972-f.number, 0, 1972-f.number, 5, length=0, angle=90,lty=2)
legend(1945, 5, c("Observed","GMDH - RGMDH","ARIMA","ES"), col = c("black","red", "steelblue1", "green"), inset = .05, cex = 0.8,lwd = c(1,2,2,2), lty=c(1,2,3,4))

dev.off()


###To draw accidental deaths data

data2 ##accidental deaths data
tiff("a.tiff", res = 900, width = 6200, height = 5400)
par(mar=c(4.1, 4.1, 0, 0.3), cex.axis=1.0, cex.lab = 1.2, font.lab = 1)
plot(data2,xlab="",ylab="",lwd = 2,asp=1,axes = FALSE, ylim = c(6,12), xlim=c(1973,1979))
axis(1, at = seq(1973,1979, 1))
axis(2, at = 6:12)
title(xlab= "Year",ylab="Accidental deaths")

dev.off()


###To draw accidental deaths data with the predictions and forecasts obtained from RGMDH and ES(M,N,M)
```

```
input=12
layer=4
f.number=5

n=length(data2)
n1=n-f.number
data=data2[1:n1]
poli1=data2[(n1+1):n]

out=fcast(data, method="RGMDH", input=input, layer=layer, f.number=f.number,
plot=FALSE)

hj=out$forecasts
ee2=out$fitted

library(forecast)

d=(mean((hj-poli1)^2))###forecast MSE of GMDH
e=(mean((ee2-data[c(-1:-input)])^2))

xshort=ts(data,start=c(1973, 1), end=c(1978, 7),frequency=12)
poli1=ts(poli1,start=c(1978, 8), end=c(1978, 12),frequency=12)

fit=auto.arima(xshort)
ari=forecast(fit,f.number)$mean
ari2=forecast(fit,f.number)$fitted
fit2=ets(xshort)
ari3=forecast(fit2,f.number)$mean
ari4=forecast(fit2,f.number)$fitted
```

```
a=(mean((ari-poli1)^2))###forecast MSE of ARIMA
b=(mean((ari2-data)^2))

f=(mean((ari3-poli1)^2))###forecast MSE of ES
g=(mean((ari4-data)^2))

aaa=matrix(c(d,e,a,b,f,g),2,3)
colnames(aaa)=c("GMDH","ARIMA","ES")
rownames(aaa)=c("forecast_MSE","prediction_MSE")
aaa
round(aaa,3)

tiff("a.tiff", res = 900, width = 6200, height = 5400)

par(mar=c(4.1, 4.1, 0, 0.3))

plot(ts(c(data2)[-1:-
input],start=c(1973,input+1),end=c(1978,12),frequency=12),asp=0.7,axes =
FALSE,col="black",xlab="",ylab="",lwd = 1,ylim = c(6,12),
xlim=c(1973.7,1979)) ####original data
axis(1, at = seq(1974,1979, 1))
axis(2, at = 6:12)

lines(ts(c(ari4,ari3)[-1:-input], start=c(1973,input+1), end=c(1978,12),
frequency=12), col="steelblue1", lwd = 2, lty=3)  ####ES
lines(ts(c(ee2,hj), start=c(1973,input+1), end=c(1978,12), frequency=12),
col="red",lwd = 2, lty=2)   ###GMDH
```

```
title(xlab= "Year",ylab="Accidental deaths")
arrows(1978+7/12, 0, 1978+7/12, 12, length=0, angle=90,lty=2)
legend(1974, 12, c("Observed","RGMDH","ES"), col = c("black","red","steelblue1"), inset = .05, cex = 0.8, lwd = c(1,2,2), lty=c(1,2,3))

dev.off()


###To draw airline passenger numbers data

data2=AirPassengers/100

tiff("a.tiff", res = 900, width = 6200, height = 5400)

par(mar=c(4.1, 4.1, 0, 0.3), cex.axis=1.0, cex.lab = 1.2, font.lab = 1)
plot(data2, xlab="", ylab="", lwd = 2, asp=1.6, axes = FALSE, ylim = c(1,7), xlim=c(1949,1961))
axis(1, at = seq(1949,1961, 1))
axis(2, at = 1:7)
title(xlab= "Year",ylab="Airline passenger numbers")

dev.off()


###To draw airline passenger numbers data with predictions and forecasts obtained from RGMDH and ES(M,A,M)

data2=AirPassengers/100
input=14
layer=2
```

```
f.number=5

n=length(data2)
n1=n-f.number

data=data2[1:n1]
poli1=data2[(n1+1):n]

out=fcast(data,method="RGMDH",input=input,layer=layer,f.number=f.number,pl
ot=FALSE)

hj=out$forecasts
ee2=out$fitted
library(forecast)

d=(mean((hj-poli1)^2))###forecast MSE of GMDH
e=(mean((ee2-data[c(-1:-input)])^2))

xshort=ts(data,start=c(1949, 1), end=c(1960, 7),frequency=12)
poli1=ts(poli1,start=c(1960, 8), end=c(1960, 12),frequency=12)

fit=auto.arima(xshort)
ari=forecast(fit,f.number)$mean
ari2=forecast(fit,f.number)$fitted

fit2=ets(xshort)
ari3=forecast(fit2,f.number)$mean
ari4=forecast(fit2,f.number)$fitted

a=(mean((ari-poli1)^2))###forecast MSE of ARIMA
```

```
b=(mean((ari2-data)^2))
f=(mean((ari3-poli1)^2))###forecast MSE of ES
g=(mean((ari4-data)^2))

aaa=matrix(c(d,e,a,b,f,g),2,3)
colnames(aaa)=c("GMDH","ARIMA","ES")
rownames(aaa)=c("forecast_MSE","prediction_MSE")
aaa

round(aaa,3)

tiff("a.tiff", res = 900, width = 6200, height = 5400)

par(mar=c(4.1, 4.1, 0, 0.3))

plot(ts(c(data2)[-1:-input], start=c(1949,input+1), end=c(1960,12), frequency=12),
asp=1.6, axes = FALSE, col="black", xlab="", ylab="", lwd = 1, ylim = c(1,7),
xlim=c(1950,1961)) ####original data
axis(1, at = seq(1950,1961, 1))
axis(2, at = 1:7)

lines(ts(c(ari4,ari3)[-1:-
input],start=c(1949,input+1),end=c(1960,12),frequency=12), col="steelblue1",lwd
= 2,lty=3)  ####ES
lines(ts(c(ee2,hj), start=c(1949,input+1), end=c(1960,12), frequency=12),
col="red", lwd = 2,lty=2)   ###GMDH

title(xlab= "Year",ylab="Airline passenger numbers")

#abline(v=1989,lty=2)
```

```
arrows(1960+7/12, 0, 1960+7/12, 7, length=0, angle=90,lty=2)

legend(1950, 7, c("Observed", "RGMDH", "ES"), col = c("black", "red", "steelblue1"), inset = .05, cex = 0.8, lwd = c(1,2,2),lty=c(1,2,3))

dev.off()
```