AN ALGORITHM FOR THE FORWARD STEP OF ADAPTIVE REGRESSION SPLINES VIA MAPPING APPROACH

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

BY

ELÇİN KARTAL KOÇ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF DOCTOR OF PHILOSOPHY
IN
STATISTICS

SEPTEMBER 2012

Approval of the thesis:

AN ALGORITHM FOR THE FORWARD STEP OF ADAPTIVE REGRESSION SPLINES VIA MAPPING APPROACH

Submitted by **ELÇİN KARTAL KOÇ** in partial fulfilment of the requirements for the degree of **Doctor of Philosophy in Department of Statistics, Middle East Technical University** by,

Prof. Dr. Canan Özgen ________________
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. Öztaş Ayhan ________________
Head of Department, **Statistics**

Assoc. Prof. Dr. İnci Batmaz
Supervisor, **Statistics Department, METU** ________________

Assist. Prof. Dr. Cem İyigün
Co-Supervisor, **Industrial Engineering Dept., METU** ________________

Examining Committee Members:

Prof. Dr. Gerhard Wilhelm Weber ________________
Institute of Applied Mathematics, METU

Assoc. Prof. Dr. İnci Batmaz ________________
Supervisor, Statistics Department, METU

Prof. Dr. Gülser Köksal ________________
Industrial Engineering Department, METU

Assist. Prof. Dr. Özlem İlk ________________
Statistics Department, METU

Assist. Prof. Dr. Özlem Çavuş ________________
Industrial Engineering Department, Bilkent University

Date: 14.09.2012

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: Elçin KARTAL KOÇ

Signature:

ABSTRACT

AN ALGORITHM FOR THE FORWARD STEP OF ADAPTIVE REGRESSION SPLINES VIA MAPPING APPROACH

Kartal Koç, Elçin
Ph.D., Department of Statistics
Supervisor: Assoc.Prof. Dr. İnci Batmaz
Co-Supervisor: Assist. Prof. Dr. Cem İyigün

September 2012. 129 pages

In high dimensional data modeling, Multivariate Adaptive Regression Splines (MARS) is a well-known nonparametric regression technique to approximate the nonlinear relationship between a response variable and the predictors with the help of splines. MARS uses piecewise linear basis functions which are separated from each other with breaking points (knots) for function estimation. The model estimating function is generated in two stepwise procedures: forward selection and backward elimination. In the first step, a general model including too many basis functions so the knot points are generated; and in the second one, the least contributing basis functions to the overall fit are eliminated. In the conventional adaptive spline procedure, knots are selected from a set of distinct data points that makes the forward selection procedure computationally expensive and leads to high local variance. To avoid these drawbacks, it is possible to select the knot points from a subset of data points, which leads to data reduction. In this study, a new method (called S-FMARS) is proposed to select the knot points by using a self organizing map-based approach which transforms the original data points to a lower dimensional space. Thus, less number of knot points is enabled to be evaluated for model building in the forward selection of MARS algorithm. The results obtained from simulated datasets and of six real-world datasets show that the proposed method is time efficient in model construction without degrading the model accuracy and prediction performance. In this study, the proposed approach is implemented to MARS and CMARS methods as an alternative to their forward step to improve them by decreasing their computing time.

Keywords: Multiple Adaptive Regression Splines (MARS), Model selection, Computational Efficiency, Mapping Algorithm, Self-Organizing Maps

ÖZ

UYARLANABİLİR REGRESYON EĞRİLERİNİN İLERİYE DOĞRU SEÇME AŞAMASI İÇİN GÖNDERİM YAKLAŞIMI İLE YENİ BİR ALGORİTMA

Kartal Koç, Elçin
Doktora, İstatistik Bölümü
Tez Yöneticisi: Doç. Dr. İnci Batmaz
Ortak Tez Yöneticisi: Yard. Doç. Dr. Cem İyigün

Eylül 2012, 129 sayfa

Çok değişkenli uyarlanabilir regresyon eğrileri (MARS), çok boyutlu veri modellemesinde çıktı değişkeni ile girdi değişkenleri arasındaki doğrusal olmayan ilişkiyi eğriler yardımıyla tahminlemede iyi bilinen bir doğrusal olmayan regresyon yöntemidir. Fonksiyon tahminlemesinde MARS, kırılma noktalarıyla birbirinden ayrılan parçalı doğrusal fonksiyonlar kullanır. Fonksiyon tahminlemesinde kullanılan model iki aşamalı bir yöntemle oluşturulur: İleriye doğru seçme ve geriye doğru eleme. İlk aşamada çok fazla temel fonksiyonun yani kırılma noktasının bulunduğu genel bir model oluşturulur ve ikincide genel uyuma az katkıda bulunan temel fonksiyonlar elenir. Klasik uyarlanabilir eğri yöntemlerinde kırılma noktaları, ileriye doğru seçme yöntemini sayısal olarak pahalı yapan ve bölgesel yüksek yayılıma neden olan farklı veri noktalar kümesinden seçilirler. Bu zorluklardan kaçınmak için kırılma noktalarını verinin küçültülmesine yol açan veri noktalarının altkümesinden seçmek mümkün olabilir. Bu çalışmada orijinal veriyi daha az boyutlu uzaya dönüştüren, kendini örgütleyen eşleştirmeye dayalı bir yaklaşımı kullanılarak kırılma noktalarının seçilmesi için yeni bir yöntem önerilmiştir. Böylece MARS algoritmasının ileriye doğru seçme yönteminde model oluşturmak için daha az sayıdaki kırılma noktasının kullanımına olanak tanınmaktadır. Benzetim yöntemiyle edilen ve altı gerçek hayat verisinden elde edilen sonuçlar, önerilen yöntemin model doğruluğunu ve tahminleme performansını düşürmeden model kurmada zaman açısından etkili bir yöntem olduğunu göstermektedir. Bu çalışmada önerilen yaklaşım hesaplama zamanlarını azaltarak MARS ve CMARS yöntemlerini iyileştirmek için yöntemlerin ileriye doğru aşamalarına alternatif olarak uyarlanmıştır.

Anahtar Kelimeler: Çok değişkenli uyarlanabilir regresyon eğrileri (MARS), Model seçimi, Hesap etkinliği, Eşleme Algoritması, Öz Düzenleyici Haritalar (SOM)

To My Family

ACKNOWLEDGEMENTS

I would like to express my great gratitude to my advisor Assoc. Prof. Dr. İnci Batmaz. Her great experience, guidance, support, and feedbacks have turned this study to an invaluable learning experience for me. As well as her contributions to my academic career, she has always been guiding me throughout my life. She has been not only my advisor, but also companion.

I would like to offer my special thanks to Assist. Prof. Dr. Cem İyigün for giving me the opportunity of studying with him. It would not be possible to complete this study without his exceptional effort and invaluable contribution. He always encouraged me and helped me to rely on myself and on my study.

I would like to thank to Prof. Dr. Gerhard-Wilhelm Weber for initiating this study. I really appreciate his valuable ideas and contributions for this study. The support by Prof. Dr. Gülser Köksal is also acknowledged for contributing to building a profound content for this thesis. I am also thankful to Prof. Dr. Öztaş Ayhan for his relevant discussions, suggestions and comments. I also would like to thank to Assist. Prof. Dr. Özlem İlk and Assist. Prof. Dr. Özlem Çavuş for their acceptance of reviewing my thesis study by spending their invaluable time.

I wish also, to thank to my husband, Gencer Koç, for his professional and personal support. His analytical thinking helped me to handle some problems that I encountered during my study in a more creative way. He always encouraged me and let me feel that he would always stand by me.

I owe my special thanks to my friend Ceyda Yazıcı for her contribution to the current thesis and for her friendship, kindness, and patience. I also want to express my thanks to Fatma Yerlikaya Özkurt for proof reading certain parts of the thesis. She always tried to answer my questions and contributed to this study. I also want to express my thanks to them for the amusing time we spent. I would like to give special thanks to Könül Bayramoğlu for her full support and help during my study.

This work was supported by The Scientific and Technical Research Council of Turkey PhD Scholarship Program (TUBITAK Yurt İçi Yüksek Lisans Burs Program). In this respect, I wish to thank to TUBITAK for financial support.

None of my studies would have been possible without the support of my family. Thanks each of them for their continuous support, endless love and patience. They always believed and encouraged me.

TABLE OF CONTENTS

ABSTRACT iv

ÖZ vi

TABLE OF CONTENTS xi

NOMENCLATURE xiii

ABBREVIATIONS xiv

LIST OF TABLES xv

LIST OF FIGURES xvii

CHAPTERS

1. INTRODUCTION 1

2. LITERATURE SURVEY AND BACKGROUND 6

2.1 Multivariate Adaptive Regression Splines (MARS) 12

    2.1.1. Forward Selection 16

    2.1.2. Backward Elimination 18

2.2. Conic MARS (CMARS) 20

2.3. Self Organizing Map (SOM) 24

3. PROPOSED APPROACH 28

3.1. Motivation 28

3.2. Proposed Approach 31

3.3 Algorithm 33

3.4 Parameters of S-FMARS Approach 34

    3.4.1 Grid Size 36

    3.4.2. Threshold Value 40

4. APPLICATIONS 44

4.1 Background on Applications 44

    4.1.1 Datasets and Validation Techniques 45

    4.1.2 Software 45

    4.1.3 Performance Criteria and Measures 47

4.2 Selection of S-FMARS Parameters for Datasets... 49
4.3. Comparison Study 1... 52
    4.3.1. Artificial datasets 53
    4.3.2 Real Life Data 59
    4.3.3. Performance on Noisy Data 64
4.4. Comparison Study 2... 69
    4.4.1. Artificial Datasets.. 70
    4.4.2. Real Datasets.. 72
4.5. Comparison Study 3... 76
    4.5.1. Artificial Datasets.. 79
    4.5.2. Real Datasets.. 83
    4.5.3. Performance on Noisy Data 88
5. CONCLUSION AND FURTHER RESEARCH.. 93
REFERENCES 98

APPENDICIES

A. MATHEMATICAL FORMULATIONS FOR DATA GENERATION 104
B. GRID PLOTS OF MATHEMATICAL FUNCTIONS... 105
C. EFFECTS OF GRID SIZE AND THREDHOLD VAUE ON ARTIFICIAL AND REAL DATASETS.. 107
D. COMPARISON OF PROJECTION METHODS FOR ARTIFICIAL AND REAL DATASETS 124
CURRICULUM VITAE.. 128

NOMENCLATURE

y : Response variable.
x : Univariate predictor variable.
$\mathbf{x} = (x_1,..., x_p)^T$: Vector of predictor variables.
p : Number of predictors.
n : Number of data points.
$\mathbf{x}_i = (x_{i,1},..., x_{i,p})^T$: *i*th data vector.
$\mathbf{x}_j = (x_{1,j},..., x_{n,j})^T$: *j*th predictor variable.
$x_{i,j}$: *i*th individual data value in the *j*th predictor variable.
τ : Knot value.
$\psi_m(\mathbf{x}^m)$: *m*th basis function.
$\mathbf{x}^m$: Variable vector for *m*th basis function.
$\mathbf{w}_l = (w_{l,1},...,w_{l,p})$: *l*th weight vector.
$\mathbf{w}_b$: Weight vector of the best matching unit (BMU).
$k(b)$: Topological neighborhood of BMU.
$h_{k(b)}$: Neighborhood function defined around the BMU.
$a(t)$: Learning rate function.
$M_{\max}$: User-specified maximum number of basis functions.
$\tilde{t}$: Threshold value for the number of hits.
g : Grid size of lattice.
m_u : Average number of hits.
std : Standard deviation of number of hits.
u : Number of neurons (units) in the lattice.
$\mathbf{z}_i = (\mathbf{x}_i, y_i)$: *i*th data vector including *i*th predictor vector with *i*th response value.
$\tilde{\mathbf{z}}_s = (\tilde{\mathbf{x}}_s, \tilde{y}_s)$: *s*th projected data vector.
$\tilde{x}_{s,j}$: *s*th individual projected data value in the *j*th predictor variable.

ABBREVIATIONS

Adj-R^2	Coefficient of determination-Adjusted
BFs	Basis functions
BMU	Best-matching unit
BF_{final}	Number of BFs in the final model
CMARS	Conic-MARS
CQP	Conic quadratic programming
CPU	Computational run time
FMARS	Forward Selection step of MARS
GLM	Generalized linear models
GCV	Generalized cross validation
LOF	Lack-of-fit
M_{max}	Maximum number of BFs
MARS	Multivariate adaptive regression splines
MinSpan	Minimum span approach
R^2	Coefficient of determination
RMSE	Root mean square error
PRSS	Penalized residual sum of squares
RSS	Residual Sum of Squares
SOM	Self-organizing maps

LIST OF TABLES

TABLES

Table 1. Features of the datasets. 46
Table 2. Design Values for Grid Size and Threshold Value. 50
Table 3. Best Parameter Values For Artificial Data. 52
Table 4. Best Parameter values for Real Data. 52
Table 5. Performances of FMARS and S-FMARS on the train data. 54
Table 6. Performances of FMARS and S-FMARS on the test data and stabilities. 54
Table 7. Performances of FMARS and S-FMARS for different *n* and M_{max}. 56
Table 8. Performances of FMARS and S-FMARS for different interaction term 59
Table 9. Average performances of FMARS and S-FMARS on the train data. 60
Table 10. Average performances of FMARS and S-FMARS on test and stabilities 61
Table 11. Overall performances of FMARS and S-FMARS methods. 61
Table 12. Average CPU times of methods for different sample size and scale 63
Table 13. Performances of FMARS and S-FMARS on noisy data 1. 65
Table 14. Performances of FMARS and S-FMARS on noisy data 2. 68
Table 15. Performances of MinSpan and S-FMARS on the train data. 71
Table 16. Performances of MinSpan and S-FMARS on the test data and stability. 71
Table 17. Average performances of MinSpan and S-FMARS on the train data. 72
Table 18. Average performances of MinSpan and S-FMARS on test and stability. 73
Table 19. Overall performances of MinSpan and S-FMARS methods. 73
Table 20. Average CPU times of methods for different sample size and scale 75
Table 21. Average performances of methods on train data. 79
Table 22. Average performance of methods on test data and stabilities. 81
Table 23. Average performances of methods on train data. 84

Table 24. Average performances of methods on test data and stabilities. 85
Table 25. Overall performances of methods. 86
Table 26. Average CPU times of methods with respect to sample size 87
Table 27. Average performance of methods on test noisy data. 90
Table 28. Average performances of methods on noisy data. 91
Table C1. Ratio=RMSE/TIME value for different grid sizes in artificial datasets 107
Table C2. Ratio=RMSE/TIME value for different grid sizes in artificial datasets 112
Table C3. Ratio=RMSE/TIME value for different grid sizes in real datasets 116
Table C4. Ratio=RMSE/TIME value for different grid sizes in real datasets. 120

LIST OF FIGURES

FIGURES

Figure 1. The form of reflected pairs of truncated linear functions. 13
Figure 2. Two-way interactions BFs (Based on Hastie et al., 2001). 15
Figure 3. Topology of SOM. 24
Figure 4. Weight vectors (red points) along with original data points (green points). 31
Figure 5. (a) Original data points. (b) Weight vectors of selected BMUs and original data points. (c) Projected weight vectors and original data points. (d) S-FMARS model for which knots are selected from the projected weight vectors. 34
Figure 6. The algorithm of S-FMARS. 35
Figure 7. Examples of grid structures. 37
Figure 8. Graph of RMSE versus grid size for Dataset 2. 38
Figure 9. Graph of Time versus grid size for Dataset 2. 39
Figure 10. Graph of Ratio versus Grid size. 39
Figure 11. The sample hits of a 5x5 hexagonal grid including 25 neurons. 41
Figure 12. Graph of RMSE versus Threshold value. 42
Figure 13. Graph of Time versus Threshold value. 43
Figure 14. Graph of Ratio versus Threshold Value. 43
Figure 15. Grid plot for data set 8. 56
Figure 16. CPU times versus sample sizes (*n*) for M_{max}=60. 57
Figure 17. CPU time versus sample size. 57
Figure 18. CPU time versus number of interaction terms. 59
Figure 19.Interaction plots of size and scale for the CPU times for FMARS and S-FMARS methods. 64
Figure 20. Sinus function with and without noise. 65
Figure 21. Fitted models for sinus function with noise. 67

Figure 22.Interaction plots of size and scale for the CPU times for MinSpan and S-FMARS methods. 76
Figure 23. Algorithm of S-CMARS method. 78
Figure 24. The plot of norm $L\theta$ versus $\sqrt{RSS}$ 78
Figure 25. Interaction plots of size and scale for the CPU times for five methods. 89
Figure A 1. (a) Grid plot of Dataset 1 (b) Grid plot of Dataset 2 (c) Grid plot of Dataset 3 (d) Grid plot of Dataset 4 (e) Grid plot of Dataset 5 (f) Grid plot of Dataset 6 (g) Grid plot of Dataset 7 106
Figure C1. Graph of ratio versus grid size for Dataset 1. 108
Figure C2. Graph of ratio versus grid size for Dataset 2. 108
Figure C3. Graph of ratio versus grid size for Dataset 3. 109
Figure C4. Graph of ratio versus grid size for Dataset 4. 109
Figure C5. Graph of ratio versus grid size for Dataset 5. 110
Figure C6. Graph of ratio versus grid size for Dataset 6. 110
Figure C7. Graph of ratio versus grid size for Dataset 7. 111
Figure C8. Graph of ratio versus grid size for Dataset 8. 111
Figure C9. Graph of ratio versus threshold value for Dataset 1. 112
Figure C10. Graph of ratio versus threshold value for Dataset 2. 113
Figure C11. Graph of ratio versus threshold value for Dataset 3. 113
Figure C12. Graph of ratio versus threshold value for Dataset 4. 114
Figure C13. Graph of ratio versus threshold value for Dataset 5. 114
Figure C14. Graph of Ratio versus Threshold Value for Dataset 6. 115
Figure C15. Graph of ratio versus threshold value for Dataset 7. 115
Figure C16. Graph of ratio versus threshold value for Dataset 8. 116
Figure C17. Graph of ratio versus threshold value for AutoMpg. 117
Figure C18. Graph of ratio versus grid size for Comm.Crime. 117
Figure C19. Graph of ratio versus grid size for Conc.Compress. 118
Figure C20. Graph of ratio versus grid size for Parkinsons. 118
Figure C21. Graph of ratio versus grid size for PM10. 119

Figure C22. Graph of ratio versus grid size for Red Wine. 119
Figure C23. Graph of ratio versus threshold value for AutoMpg. 120
Figure C24. Graph of ratio versus threshold value for Comm.Crime. 121
Figure C25. Graph of ratio versus threshold value for Conc.Compress. 121
Figure C26. Graph of ratio versus threshold value for Parkinsons. 122
Figure C27. Graph of ratio versus threshold value for PM10. 122
Figure C28. Graph of ratio versus threshold value for Red Wine. 123

CHAPTER 1

INTRODUCTION

Multivariate adaptive regression spline (MARS) is a powerful nonparametric regression method for constructing flexible models by introducing truncated linear functions. Due to its simplicity and effectiveness for handling high-dimensional data settings, MARS has recently become a popular tool for solving various classification and regression problems including prediction mobile radio channels (Kubin, 1999), credit scoring (Lee et al. 2006), detecting disease risk (York et al., 2006), environmental modeling (Leathwick et al., 2005, 2006), direct response modeling (Deichmann et al., 2002).

Regression splines provide a flexible model estimate with the help of piecewise functions (splines), so that the nonlinearity of a model is approximated through the use of separate regression models defined over the distinct subintervals of the range of predictor variables. The intervals that define the pieces are separated by a sequence of knots or breaking points whose number and locations are practically unknown in advance. The simplest method considers knots as fixed and equally spaced (Keele, 2008). In this method, the total number of knots is selected first, and then, knots are allocated “equally-spaced” in the sense that either the distance or the number of distinct design points between two consecutive knot points remains the same throughout the whole data domain. The number of equally-spaced knots is increased iteratively until a satisfactory estimate is obtained. In this case, however, an unfortunate knot placement may lead to misleading results (Yao et al., 2008). Motivated by the work of Smith (1982), an adaptive approach has been proposed for regression splines. Friedman and Silverman (1989), Friedman (1991) and Denison et al. (1998) automatically selected the number and the location of knots from a set of distinct design points via a model selection criterion.

In adaptive regression splines, knots are selected through a two-stage algorithm called **forward selection** and **backward elimination**. In forward selection, knots are added into a model in a stepwise manner as long as a lack-of-fit (LOF) criterion is decreased, and then, the ones contributing less to the model are eliminated via backward elimination. A natural strategy of knot selection during the forward process is to consider every distinct data point as a candidate knot location. This strategy estimates the underlying data structure by evaluating every data points as breaking points where the regression models change their slopes. This can give reasonable results for low noisy settings; however, increase the local variance for highly noisy data. Besides, it is true for all cases (data with low and high noises) that the computational run time increases significantly.

In this study, a new knot selection procedure is proposed for adaptive regression splines to make it computationally efficient without increasing the local variability. To decrease the computing time of adaptive regression splines, the set of points searched is restricted to a small subset of data points during the forward step. Hence, less number of data points is evaluated as candidate knot locations for the function estimation. Here, the way of subsetting is a critical issue. The points can be selected randomly or equally spaced with a partial search; however, it may result in a poor performance for the spline regression (Lou and Wahba, 1997) depending on the form of underlying true function. For example, the functions including nonhomogeneous smoothness may be approximated better with many unevenly distributed knots instead of using fixed interval of knots. In this respect, a data-driven subsetting reflecting the underlying data structure is offered in this study. To provide such a subsetting procedure, a mapping approach that uses **self-organizing maps** (SOM) is proposed. In this approach, a large set of data points can be reduced or compressed into smaller set of units through a nonlinear mapping. This kind of mapping transforms the high-dimensional data into a low-dimensional map of units via weight vectors. The data vectors are mapped into a new lattice by an updating formula based on the distance between data and weight vectors. During the mapping procedure, the relative distance between data points is preserved by a topological order (grid structure) of the units; so that, the original data structure can be properly approximated by the distribution of the weight vectors. Here, the weight vectors can be considered as representatives or pointers of data points. In the proposed approach, therefore, knots are determined by considering weight vectors as the reference. The data points projected by the weight vectors are then used as the candidate knots in the function estimation. In the proposed method, candidate knots are obtained by considering the whole data values of the predictors, instead of searching locally on each predictor as in the MARS algorithm does. Once the candidate knots are determined as the projections of the weight vectors, knots are then selected among them through a forward selection method via piecewise linear functions. This approach is actually a modified version of the forward selection step of MARS algorithm; thus, the new approach is called as S-FMARS, where S stands for subsetting and FMARS is used for forward selection of MARS algorithm.

The current thesis consists of five chapters and four appendices, and each one is organized in the following way. In Chapter 2, we provide a brief background on the spline functions and their use in regression splines. As the critical issues in regression splines, selection of the number and location of knot points are discussed through some common approaches by pointing out their associated advantages and disadvantages. This chapter concludes with a short overview of the adaptive regression splines which propose an automatic knot selection procedure and a summary of the usage of different versions of adaptive regression splines in the literature. Since the proposed approach developed over the forward step of MARS algorithm, the main idea behind the MARS is also given in this chapter. Two complementary strategies of the MARS algorithm are examined in the subsections named *Forward Stepwise Selection* and *Backward Stepwise Elimination*. In the section of the forward selection procedure, a Minimum Span (MinSpan) approach proposed by Friedman (1993) for the forward step of MARS algorithm to optimize the knot points is discussed. The performance of the proposed approach is then compared with MinSpan in the subsequent sections. In the backward elimination section, the pruning procedure is mentioned by emphasizing different model selection criteria. In the same section, a modified version of MARS algorithm with uses as an alternative backward step, called Conic MARS (CMARS) approach is discussed in detail. The proposed approach is then implemented to MARS and CMARS in place of their forward selection step to make them computationally efficient.

Chapter 3 introduces the proposed approach. Firstly, the motivation behind the proposed idea is given by discussing the computational complexity of MARS algorithm and evaluating the mapping idea with its appropriate properties for knot selection purpose. The steps of the proposed knot selection algorithm are then explained in detail with the help of figures and mathematical formulations. In addition, two important parameters of the proposed approach are studied to make the proposed approach more accurate and time efficient. These parameters are the grid size and the threshold value set for the number of data points assigned to units during the mapping. This chapter also presents a way of selecting the best values for the parameters to obtain a time efficient regression spline model without loss of accuracy.

Chapter 4 presents a background for the application process. Firstly, the datasets for which the proposed approach is applied and compared with other methods are described. Then, the software used to implement the proposed approach and execute the other methods is presented. In the next section, the performance criteria and measures used to evaluate the performance of models produced by the proposed approach and other regression spline methods are described. Then, the best parameter values of the proposed approach are determined for the datasets under study. Once the parameters of the proposed approach are determined, the performance of the proposed approach is evaluated and compared with the other methods through three comparison study. In the first comparison study, proposed approach is compared with the forward selection algorithm of MARS via some artificial datasets, real datasets and noisy setting. In the second one, based on the same datasets, the proposed approach is compared with a minimum-span knot selection scheme. Finally, the backward elimination procedure of MARS and penalized strategy of CMARS are combined with the proposed approach, and the performances of the hybrid methods are compared with the original MARS, MARS with MinSpan approach and CMARS methods. The corresponding findings are explained in detail in this chapter.

Based on the findings given in Chapter 4, a comprehensive discussion, as well as the conclusion and further studies are presented in the last section.

Appendix A presents the mathematical functions of the problems utilized in this thesis. While some functions have low-order nonlinear behavior, some have highly nonlinear form. The figures in Appendix B are the grid plots of the problems given in Appendix A. Related with the study performed in Chapter 4, ratio values calculated by both using the root mean squared error (RMSE) values of the models obtained by the proposed approach and the corresponding computational run times (CPU time) are given for different grid sizes for artificial and real datasets in Table C.1 and Table C.3 of Appendix C, respectively. Similar tables obtained for different threshold values are shown in Table C.2 for artificial datasets and in Table C.4 for real datasets. Besides, the graphs of “Ratio versus grid size” and “Ratio versus threshold values” are displayed for all artificial and real datasets. In Appendix D, performances of three projection methods offered to be used in the proposed approach are compared with respect to some performance criteria.

CHAPTER 2

LITERATURE SURVEY AND BACKGROUND

In statistical modeling, the relationship that may exist between a response variable y and a vector of predictors $\mathbf{x} = (x_1, ..., x_p)^T$ is approximated with the following general type of model

$$y = f(\mathbf{x}) + \varepsilon \, , \tag{1}$$

Here, ε indicates the error term with zero mean, and p denotes the number of predictor variables.

In statistical framework, the function in (1) is generally approximated by a parametric model assuming a linear form of the predictor variables. Different linear models are used in the literature depending on the nature of the response variable. If the response is continuous, a linear regression model is estimated using least squares (LS). For the discrete responses, generalized linear models (GLM) including logistic or Poisson regressions are estimated.

In GLM, a model is specified by selecting a sampling distribution for the response variable and a functional form for the predictors. For example, for a linear regression model, the **normal** distribution $N(\mu, \sigma^2)$ with expected value μ and constant variance is chosen for the continuous response and a **linear form** of the predictors $\eta = \mathbf{x}^T \boldsymbol{\beta}$ is determined for a given vector of predictors $\mathbf{x}$ with $p \times n$ dimensions. Here, η is an $n \times 1$ vector of linear predictions and $\boldsymbol{\beta}$ is a vector of unknown parameters that must be estimated. In fact, the GLM generalizes the linear regression models. The stochastic component (response variable) can follow any distribution from the exponential family, and the linear functional form of predictors is generalized with a link function $\eta = h(\mathbf{x}^T\boldsymbol{\beta})$. For example, for the logistic regression, y is a binary response and follows a binomial distribution. The relation between linear predictors $\mathbf{x}^T\boldsymbol{\beta}$ and y is supplied by the following logistic link function (Hastie and Tibshirani, 1990; Keele, 2008)

$$p = 1/(1+e^{-\eta}) \tag{2}$$

where p is the probability that $y=1$ for given values of $\mathbf{x}_i$ $(i=1,...,n)$. As in (2), many models which are considered as nonlinear have linear functional form of predictors. This is why they retain a linearity assumption. However, in practice, the form of the relation between variables generally is not linear, so that aforementioned parametric assumptions turn out to be too restrictive for many practical applications. In order to estimate a nonlinear functional form, a variety of transformations can be applied to predictor variables. Power transformations are often reasonable methods for representing the nonlinear functional forms. Nevertheless, they have some limitations. For a given univariate variable, x; for example, they provide global estimates for the relationship between x and y by assuming the relation to be constant over the range of x. However, the relationship between x and y is generally local. Namely, the statistical relationship between two variables changes over the range of x. For such cases, the assumption of global estimate often disregards the underlying true relation. Moreover, for more complex nonlinear forms, global estimates like power transformations are not sufficient.

In the absence of strong theory for the assumed functional form, the underlying relationship between predictor variables $\mathbf{x}$ and the y response is estimated from the data. In data-based modeling, global estimates are changed places with local estimates which refer to nonparametric regression models in statistics. Nonparametric regression allows one to estimate nonlinear fits between variables with a few assumptions about the functional form of the nonlinearity. In this framework, the function which defines the dependency of y on $\mathbf{x}$ is generalized from linear functions to any smooth function $g(\mathbf{x})$, and it is typically estimated using additive models given in (3).

$$g(\mathbf{x}) = \sum_{i=1}^{p} g_i(x_i). \tag{3}$$

Although the assumption of additivity is more restrictive than a fully multivariate nonparametric regression model, it is a common way of extending nonparametric estimation for high dimensional data (p>1) subject to the problem of curse of dimensionality (Hastie and Tibshirani, 1990).

In (3), $g_1,\ldots,g_p$ are arbitrary smooth or unspecified functions which are typically estimated using spline functions or local averaging smoothers (Cleveland, 1993; Silverman, 1985). Since the underlying relation is practically inhomogeneous and the degree of smoothness is unknown in advance, it is common to estimate the smooth function g_i by using spline functions due to their good numerical properties (de Boor, 1978; Schumaker, 1981; Green and Silverman, 1994).

Spline functions refer to piecewise regression models defined over the intervals in the range of univariate x. The intervals that define the pieces are separated from each other by a sequence of points, called breaking points or knots. The slopes of the regression models are forced to change from one interval to another over the range of x at knots. Hence, a flexible model estimate is achieved by the help of many local fits. There are various types of splines: regression splines, cubic splines, B-splines, P-splines, natural splines, thin-plate splines, smoothing splines, and the ones which are the combinations of different types such as natural cubic B-splines.

In nonparametric framework, to estimate the smooth terms in regression models using splines, two main approaches are basically followed: smoothing splines and regression splines. Smoothing splines are advanced and well-known local averaging smoothers, and appear as a solution to an optimization problem. It tries to minimize a penalized residual sum of squares (PRSS) by using a roughness parameter which controls the smoothness of the model fit. For nonadaptable smoothers, the smoothing parameter is specified by the user or set by an automated procedure like cross validation (Hastie and Tibshirani, 1990). Smoothing splines are more complex than piecewise polynomial, however, they become very popular in statistics with the help of studies conducted by Wahba (1983, 1990), Wahba and Wold (1975), Silverman (1985), Green and Silverman (1994), or by Eubank (1999) who provided an excellent overview of smoothing spline techniques and their applications in statistics.

Regression splines provide a flexible model estimate by the help of piecewise functions. The regions that define the pieces are separated by a sequence of knots (breaking points). The number and the location of knots, which are unknown in advance, have a critical importance in controlling the amount of smoothness and flexibility during the function estimation. Standard practice is to consider knots as fixed and place knots at evenly spaced intervals in the data. To ensure an adequate data for each interval to get a smooth fit, knots are placed at either quartiles or quintiles in the data by default (Keele, 2008). In practice, this approach may lead to an unfortunate knot placement resulting in misleading results. Actually, the number of knots is more crucial than the place of knots (Stone, 1986). The number of knots acts as a span parameter denoting the width of the intervals for splines, and affects the amount of smoothing applied to the data by controlling the number of piecewise fits. The spline with less number of knots provides globally smooth fit. However, the flexibility of the model is increased as the number of knots increases. Moreover, number of knots governs the trade-off between bias and variance of the estimate. Increasing the number of knots increases the local variability while decreasing the bias. This situation is called undersmoothing. On the other hand, decreasing the number of knots increases the bias while decreasing the variability in the fit, which refers to oversmoothing. In selecting the number of knots, the cases of oversmoothing and undersmoothing should be taken into consideration. The main goal in the selection of knots, therefore, should be to produce as smooth fit as possible without departing from the underlying true regression function.

In this context, many efficient methods have been studied in the literature. Some studies consider the knot selection procedure as a model selection approach. Since each knot is an additional parameter being added to the model, some model selection criteria such as the Cp statistic (Mallows, 1973), AIC (Akaike, 1973) or GCV (Craven and Wahba, 1979) are recommended to select the number of knots. For knot selection, AIC was used by Atilgan (1988) and recommended by Eilers and Marx (1996). More recently, an adaptive strategy, originally proposed by Smith (1982), was used by Friedman and Silverman (1989), Friedman (1991), Stone et al. (1997), Lou and Wahba (1997) and Breiman (1993) to select the number and location of the knots. In these approaches, knots were selected via a stepwise procedure using a model selection criterion. In the first step, called **forward selection**, the knot that reduces the criterion the most is included into a model and a rich set of knots are allowed to be selected through this step. In the second step, called **backward elimination**, the knots contributing less to the model are removed.

There are many different versions of adaptive regression splines which uses the same adaptive strategy. He and Ng (1996) developed a stepwise knot selection algorithm in the quantile regression context. This can be viewed as a variation of the algorithm of Stone et al. (1997). The TURBO algorithm of Friedman and Silverman (1989) and its subsequent generalization, the MARS algorithm (Friedman, 1991), include knot selection for univariate scatterplot smoothing as a special case. However, TURBO and MARS are tailored for multivariate smoothing and their computational overhead requires restriction to piecewise linear **basis functions** (BFs) for practical implementation. Denison, Mallick and Smith (1998) have developed an alternative

Bayesian approach to regression spline fitting. The main difference between their approach and the approach of Smith and Kohn (1996) is that the number of knots, and their locations, are not fixed in advance, but instead, are considered as random components of a Bayesian model. The Markov Chain Monte Carlo (MCMC) strategy for selecting the model involves knots being added and deleted, and therefore, a change in the dimension of the model. More recent contributions to additive modeling have been made by Lou and Wahba (Hybrid Adaptive Splines [HAS], 1997); Stone, Hansen, Kooperberg, and Troung ([POLYMARS], 1997), Weber et al. (CMARS, 2012), Taylan, Weber and Beck (2007), Özmen (2010) and Özmen et al. (R-CMARS, 2010). HAS performs forward knot selection via GCV with a “cost” term, as in MARS, but replaces backward deletion by ridge regression. POLYMARS is a multi-response version of MARS which has been customized for computational efficiency. CMARS is a hybrid method as HAS. It generates forward knot selection via GCV and use ridge regression instead of backward deletion. Different from HAS, however, it solves the penalized splines by a Tikhonov regularization. Based on CMARS, R-CMARS is proposed as the robustification of CMARS with robust optimization to decrease the estimation error in CMARS.

In this thesis, a new forward selection algorithm is proposed to decrease the computing time of adaptive regression splines. Since the proposed method has some common properties with the forward step of MARS algorithm. MARS is explained in detail in this chapter. Additionally, the proposed approach is compared with a MinSpan approach proposed to optimize the knots in the forward selection of MARS algorithm. Although the main purpose behind this idea is to decrease the local variability, it also decreases the computing time effectively. This is why the MinSpan method is examined in detail in this study.

The proposed approach is developed as a new forward selection part; therefore, it can be followed by a backward elimination step as in MARS, or other methods using an alternative method for backward step like in CMARS. For all these approaches, the computing time can be decreased significantly by using the proposed approach beforehand. In this chapter, therefore, background information on MARS and CMARS are given in the following subsection.

2.1 Multivariate Adaptive Regression Splines (MARS)

MARS is a popular nonparametric regression technique developed by Friedman (1991) particularly for approximating nonlinear relationship within the data with the help of splines. Splines refer to a wide class of piecewise defined functions used to provide local fits for estimating the underlying form of functions using the data. The nonlinearity between the response and predictors is then estimated by having different regression slopes in the corresponding intervals of each predictor. These intervals are distinct and separated by breaking points, called knots. MARS uses piecewise linear functions for local approximations which are easy to implement. The form of the truncated linear functions are given for a univariate variable, x, as follows (Hastie et al., 2001)

$$[(x-\tau)]_+ = \begin{cases} (x-\tau), & x > \tau \\ 0, & \text{otherwise} \end{cases}, \quad [-(x-\tau)]_+ = \begin{cases} (\tau - x), & x < \tau \\ 0, & \text{otherwise} \end{cases} \qquad (4)$$

Two functions in (4) are called **reflected pairs** and characterized by the breaking points τ, called **knots**. The first expression takes the value of zero for all x values less than or equal to the threshold value τ and takes $(x - \tau)$ for all values greater than τ. On the other hand, the second expression results in zero for all x values greater than or equal to τ and gets $(\tau - x)$ otherwise. The "+" sign represents positive part of the function.

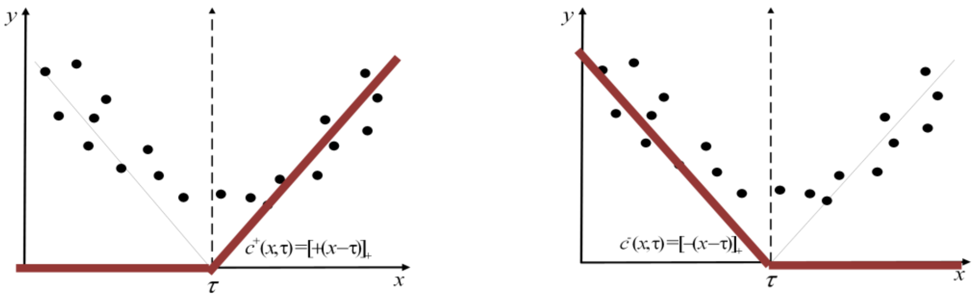


Figure 1. The form of reflected pairs of truncated linear functions.

MARS builds a flexible model by fitting piecewise linear functions by which the nonlinearity of a model is approximated through linear functions in distinct intervals of the predictor space. The knot (breaking) points where behaviors of the function changes play a key role in the function approximation but the number and location of knots are practically unknown. In classical spline, knot points are usually predefined or equally spaced. In MARS, however, knots are determined by a search procedure.

For a given vector of predictor variables, $\mathbf{x}=(x_1,x_2,...,x_p)^T$, all distinct individual data values, $x_{i,j}$, $(i=1,...,n)$ of the corresponding predictor variable x_j $(j=1,...,p)$ are considered as knot points, and introduced into the model via a reflected pair given in (4). The set of all possible reflected pairs with the corresponding knots can be expressed with set C in (5).

$$C=\{(x_j-\tau)_+,(\tau-x_j)_+ \mid \tau\in\{x_{1j},x_{2j},...x_{nj}\},\ j\in\{1,...,p\}\}. \tag{5}$$

MARS generates its model by using the **basis functions** (BFs) defined over the functions in the set C. In additive MARS models, every elements of C can be considered as one BF. For highly nonlinear datasets requiring interaction effects, MARS modeling can be generalized with the BFs including tensor product of two or more functions from the set C. Therefore, the general form of BFs defined over the subvector of predictor variables, $\mathbf{x}^m$ can be defined as follows

$$\psi_m(\mathbf{x}^m) = \prod_{v=1}^{K_m} [s_{(v,m)}.(x_{j(v,m)} - \tau_{(v,m)})]_+ , \tag{6}$$

where K_m is the number of truncated linear functions in the mth BF; $x_{j(v,m)}$ is the jth predictor variable corresponding to the vth truncated linear function in the mth BF; $\tau_{(v,m)}$ represents the knot value corresponding to the predictor variable $x_{j(v,m)}$ in the mth BF. The quantities $s_{(v,m)}$ take values from the set $\{\pm 1\}$.

There is a limitation in the construction of the BFs; the ones built by the multiplication of truncated linear functions must include distinct predictor variables. This prevents the occurrence of higher-order degrees of a variable which increase or decrease too sharply near the boundaries of the factor space. A piecewise linear function can approximate the higher-order powers in a more stable way.

Multiplication of two BFs produces a result which is nonzero only over the factor space where both components are nonzero (Figure 2). Thus, the regression surface is obtained by using only nonzero components locally- only when they are needed. If polynomial BFs are used, then the multiplication of BFs would be nonzero everywhere and would not work as well. The BF in Figure 2 is defined as the multiplication of two BFs such as

$$\psi(x_1, x_2) = (x_1 - \tau_{4,1})_+ (\tau_{3,2} - x_2),$$

where, $\tau_{4,1} \in \{x_{1,1}, x_{2,1}, ..., x_{n,1}\}$ and $\tau_{32} \in \{x_{1,2}, x_{2,2}, ..., x_{n,2}\}$.

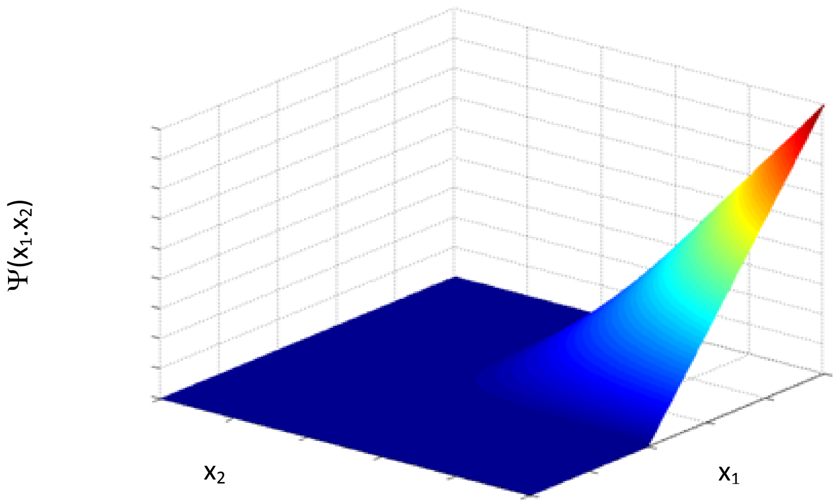


Figure 2. Two-way interactions BFs (Based on Hastie et al., 2001).

The model developed by MARS is similar to the one developed in classical linear regression; however, BFs or their products are used instead of the original predictor variables. For a given vector of predictor variables, $\mathbf{x} = (x_1,..., x_p)^T$ and the target variable y, the model has the form

$$y = c_0 + \sum_{m=1}^{M} c_m \psi_m(\mathbf{x}^m) + \varepsilon \,, \tag{7}$$

where c_0 is the intercept term; $\psi_m(\mathbf{x}^m)$ is the *m*th BF with a coefficient c_m; M is the number of BFs in the current model (Friedman and Silverman, 1989; Friedman 1991).

The estimates of the coefficients $(c_0,...,c_m)$ in (7) are calculated by a $(M+1)$-parameter LS fit of the response y on the fixed BFs, $\psi_m(\mathbf{x}^m)$ $(m=1,..., M)$. Since optimizing the (averaged) squared residuals over all BFs defined for all possible knots is a fairly difficult task computationally, especially for large M, a stepwise strategy is adapted for BF selection in the MARS modeling.

Stepwise strategy of MARS includes two steps: **forward selection** and **backward elimination**. In the forward selection, the algorithm starts with a model consisting of intercept term c_0 and adds a reflected pair from the set C iteratively until the maximum number of terms specified by the user is reached by the model. At the end of this step, a large model typically overfitting the data is obtained. Then, a backward elimination is implemented to refine the model. In this pruning step, the BFs contributing less to the model are eliminated. Detailed descriptions for these two phases are given in the following sections.

2.1.1. Forward Selection

In classical forward stepwise regression, each predictor is added into the model via some model selection criteria such as C_p, AIC or F statistic. The main purpose behind the method is to identify a useful subset of the predictors for a better approximation. In adaptive regression spline, each BF is considered as a new predictor. The forward stepwise algorithm searches for the BFs and at each step the split that minimizes some lack-of-fit criterion from all possible splits on each BF is chosen. The algorithm deliberately overfit the data by inserting large number of BFs into the model. So that, all types of curvatures are tried to be estimated by adding BFs with the corresponding knot points where the curvature exists.

The forward step has a critical role in knot selection. In general, all distinct data points are evaluated as a knot point through BFs and their contribution to the model is checked via a lack-of-fit criterion. The aim of a lack-of-fit criterion is to provide a data-based estimate of the future prediction error which is then minimized with respect to the parameters of the procedure (Friedman, 1991). In main effect models, each BF built on one predictor variable with the corresponding predictor value as knot. Hence, each distinct data value is introduced to the model predictor wisely. In interaction models, tensor products of two or more distinct predictor variables are added to the model. In this case, breaking points are represented with a vector of the corresponding predictor values. Evaluating every distinct data value as a knot enables one to catch the curvatures truly; however, it is computationally expensive and increases the local variability. Especially, in highly noisy data, evaluating noises as knots for function estimation leads to a redundant effort, and decreases the model accuracy.

In order to prevent the situations mentioned above to be happen, a MinSpan approach has been proposed to restrict the candidate knot locations (Friedman, 1991). Its simplest version is to make every other distinct *rth* observation (in order of ascending univariate *x*-value) eligible for a knot placement. For noisy settings, this implementation can lead to decrease in the local variability. Additionally, the computing time is reduced by a factor of n/r in the absence of ties. In conventional splines, the value for r is taken as fixed or calculated in such a way as to make the number of distinct design points between any two adjacent knots equal (Ruppert, 2002; Ruppert et al., 2003). This method is simple and easy to implement. Nevertheless, the knots may not be placed at all critical locations (Yao and Lee, 2008). Friedman and Silverman (1989) proposed a data-adaptive value (as a function of *n*) for the number of distinct design points between any two adjacent knots by using a coin tossing argument. The proposed value *L*, based on the assumption of having symmetric distributed error terms, is defined as the solution of (Friedman, 1991)

$$P(L) = \alpha, \tag{8}$$

where $P(L)$ is the probability of observing a run of length *L* or longer in pn_m tosses of a fair coin and α is a small number (e.g. $\alpha = 0.01$or 0.05). The quantity n_m is the number of observations for which $\psi_m(\mathbf{x}^m) > 0$, and p denotes the number of predictors. Hence, pn_m represents the number of potential locations for each new knot for each BF $\psi_m(\mathbf{x}^m)$. Setting $r = L(\alpha)/2.5$ would give the smoother resistance to run of positive and negative error values with probability α. The reason for using 2.5 (or 3 to be more conservative) in the denominator is the fact that a piecewise linear smoother must place between two and three knots in the interval of the run to respond to it and not degrade the fit anywhere else.

For $pn_m \geq 10$ and $\alpha < 0.1$, a good approximation to $L(\alpha)$ is

$$L(\alpha) = -\log\left[-\frac{1}{pn_m}\ln(1-\alpha)\right], \tag{9}$$

so that the reasonable number of observations between knots is given by

$$r = -\log\left[-\frac{1}{pn_m}\ln(1-\alpha)\right]\Big/2.5. \tag{10}$$

The MinSpan approach provides a local search around the current knot over a specific predictor variable. The approach does not consider the whole data structure; instead, it selects the knot predictor wisely. Its main objective is to decrease the local variability in function estimation but at the same time it consequently decreases the computing time significantly.

2.1.2. Backward Elimination

Backward stepwise is a pruning step which eliminates the redundant BFs selected in the forward step. Hence, the overfitting problem is aimed to be removed. To estimate the model with the optimal number of BFs or knots, MARS uses a model selection criterion called **generalized cross validation** (GCV). This criterion depends on the idea of minimizing the average-squared residuals of the fit by considering a model complexity, which is the number of BFs in the model. For a given data vector $\mathbf{z}_i = (\mathbf{x}_i, y_i)$ $(i = 1,..., n)$ the criterion proposed by Craven and Wahba (1979) is given as follows

$$GCV(M) = \frac{1}{n} \frac{\sum_{i=1}^{n} (y_i - \hat{f}_M(\mathbf{x}_i))^2}{(1 - P(M)/n)^2} \ , \tag{11}$$

where, y_i is the *i*th observed response value; $\hat{f}_M(\mathbf{x}_i)$ is the fitted response value obtained for the *i*th observed predictor vector $\mathbf{x}_i = (x_{i,1},..., x_{i,p})^T$ $(i = 1,..., n),$ n is the number of data points; M represents the maximum number of BFs in the model.

In general, $P(M)$ is calculated by using the formula given below

$$P(M) = trace(\mathbf{B}(\mathbf{B}^T\mathbf{B})^{-1}\mathbf{B}^T) + 1, \tag{12}$$

and represents the cost penalty measure of a model where there are M BFs (Friedman, 1991). Here, $\mathbf{B}$ is the matrix of BFs with dimension $M \times n.$

$P(M)$ in (12) represents effective number of parameters which is a penalty measure for complexity. There are different representations for $P(M)$; commonly used one is: $P(M) = r + dK,$ where r is the number of linearly independent BFs in the model, and K is the number of knots selected in the forward process. Note that if the model is additive then d is taken to be two; if the model is an interaction model then d is taken to be three (Friedman, 1991; Hastie, 2001). If the value of $P(M)$ is small, it produces a model with many BFs. Otherwise, a smaller model with less BFs is obtained. This procedure continues for all number of BFs and then the best model that has minimum GCV is chosen.

In some studies, alternative methods are proposed for the backward step of MARS (Lou and Wahba, 1997; Weber et al, 2012). In these studies, a penalty term is added to the lack-of-fit criterion. CMARS uses up all BFs generated by the forward algorithm of MARS, and minimize a penalized residual sum of squares (PRSS) value. Hence, both the accuracy and complexity of the model are tried to be controlled through a penalty parameter. One of the main drawbacks of CMARS mentioned in the paper of Weber et al. (2012) is that it is not as efficient as the MARS method. To improve CMARS algorithm for reducing computational run time, the proposed approach is also implemented to CMARS algorithm. Beforehand, detailed information on CMARS is given in the following section.

2.2. Conic MARS (CMARS)

CMARS, where "C" stands for "*conic*", "*convex*" and "*continuous*", is a modified version of MARS algorithm which uses a PRSS approach instead of the backward elimination step of MARS algorithm. By using a penalty term in addition to the lack-of-fit criterion, PRSS can control the complexity of the model estimation. CMARS algorithm is built on the set of BFs selected through the forward algorithm of MARS; thereby, they share the same forward selection step. However, CMARS modifies the MARS algorithm by taking into account the nearby placement of knots. The BFs with knots $\boldsymbol{\tau}_i = (\tau_{i,1}, \tau_{i,2}, \ldots, \tau_{i,p})^T$ are constructed at $\mathbf{x}_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,p})^T$ or just nearby the data vector $\bar{x}_i = (\bar{x}_{i,1}, \bar{x}_{i,2}, \ldots, \bar{x}_{i,p})^T$. Namely, knot points may not be taken as one of the data points ($\tau_{i,j} \neq \bar{x}_{i,j}$ for all $(i = 1,2,\ldots,n)$ and $(j = 1,2,\ldots,p)$) in CMARS. The aim of this modification is to take the derivatives during optimization process of the PRSS with the following form:

$$PRSS = \sum_{i=1}^{n} \; y_i - f(\bar{\boldsymbol{x}}_i) \;\; ^{2} + \sum_{m=1}^{M_{\max}} \lambda_m \sum_{\substack{|\boldsymbol{\alpha}|=1 \\ \boldsymbol{\alpha}=(\alpha_1,\alpha_2)^T}}^{2} \; \sum_{\substack{r<s \\ r,s\in V(m)}} \int \theta_m^2 \left[D_{r,s}^{\alpha} \psi_m(\boldsymbol{t}^m) \right]^2 d\boldsymbol{t}^m, \qquad (13)$$

where $M_{\max}$ is the number of BFs reached at the end of the forward algorithm; $V(m) = \{K_j^m \mid j = 1,2,...,K_m\}$ is the variable set associated with *m*th BF, $\psi(m)$, and $\mathbf{t}^m = (t_{m_1}, t_{m_2}, ..., t_{m_{K_m}})^T$ represents the variables which contribute to the *m*th BF. The λ_m values are the nonnegative **penalty parameters** assigned for each BFs $m = (1,..., M_{\max})$. Moreover, $D_{r,s}^{\boldsymbol{\alpha}} \psi_m(\mathbf{t}^m)$ is denoted as in (14) for $\boldsymbol{\alpha} = (\alpha_1, \alpha_2)^T$, $|\boldsymbol{\alpha}| = \alpha_1 + \alpha_2$, where $\alpha_1, \alpha_2 \in \{0,1\}$.

$$D_{r,s}^{\boldsymbol{\alpha}} \psi_m(\mathbf{t}^m) = \frac{\partial^{|\boldsymbol{\alpha}|} \psi_m}{\partial^{\alpha_1} t_r^m \partial^{\alpha_2} t_s^m}(\mathbf{t}^m). \qquad (14)$$

Here, if $\alpha_i = 2$, the derivative $D_{r,s}^{\boldsymbol{\alpha}} \psi_m(\mathbf{t}^m)$ vanishes, and by addressing indices $r < s$, the Schwarz's Theorem is applied.

The PRSS approach bases on a tradeoff between the accuracy and complexity, and it is established with the help of a penalty parameters, λ_m in (13). In this equation, while the first term controls the **accuracy** which refers to small sum of squares errors, the second term controls the **complexity**.

In equation (13), the second part of the PRSS includes multi-dimensional integrals, which are usually difficult to handle. Therefore, discretization techniques are preferred generally. The PRSS problem is simplified by applying discretization in the multidimensional integral in (13) as follows (see Yerlikaya, (2008) and Taylan et al. (2007), for more detail).

$$PRSS \approx \left\| \mathbf{y} - \boldsymbol{\psi}(\bar{\mathbf{d}})\boldsymbol{\theta} \right\|_2^2 + \sum_{m=1}^{M_{\max}} \lambda_m \sum_{i=1}^{(n+1)^{K_m}} L_{im}^2 \theta_m^2, \qquad (15)$$

where $\boldsymbol{\psi}(\overline{\mathbf{d}}) = (\boldsymbol{\psi}(\overline{\mathbf{d}}_1),...,\boldsymbol{\psi}(\overline{\mathbf{d}}_n))^T$ is a matrix with dimensions of $(n \times (M_{\max} + 1))$; $\| \cdot \|_2$ denotes the Euclidean norm, and the numbers L_{im} are defined as

$$L_{im} = \left[\left(\sum_{\substack{|\boldsymbol{\alpha}|=1 \\ \boldsymbol{\alpha}=(\alpha_1,\alpha_2)^T}}^{2} \sum_{\substack{r<s \\ r,s \in V(m)}} \left[D^{\boldsymbol{\alpha}}_{r,s} \psi_m (\hat{\boldsymbol{x}}_i^m) \right]^2 \right) \Delta \hat{\boldsymbol{x}}_i^m \right]^{1/2} . \tag{16}$$

Here, $\hat{\mathbf{x}}_i^m$ and $\Delta\hat{\mathbf{x}}_i^m$ are related to the predictor data used for discretization.

The linear systems of equations, $\mathbf{y} = \boldsymbol{\psi}(\overline{\mathbf{d}})\boldsymbol{\theta}$, can be solved approximately by using the PRSS. The problem is classified as ill-posed, which means irregular or unstable. Thus, *Tikhonov regularization problem* is considered for the solution of PRSS problem because it is the most widely used method for converting the ill-posed problems to well-posed (regular or stable) ones. The PRSS in (15) is rearranged as given in (17) to be handled as a Tikhonov regularization problem.

$$PRSS \approx \left\| \mathbf{y} - \boldsymbol{\psi}(\overline{\mathbf{d}})\boldsymbol{\theta} \right\|_2^2 + \lambda \left\| \mathbf{L}\boldsymbol{\theta} \right\|_2^2 , \tag{17}$$

where $\mathbf{L}$ is an $(M_{\max} + 1) \times (M_{\max} + 1)$ -diagonal matrix with first column $\mathbf{L}_0 = \mathbf{0}_{(n+1)^{K_m}}$ and the other columns being the vectors $\mathbf{L}_M$ introduced in (16). Here, $\boldsymbol{\theta}$ with the dimension of $((M_{\max} + 1) \times 1)$ is a vector consisting of the parameters to be estimated.

In (15), there is a sequence of penalty parameters $\boldsymbol{\lambda} = (\lambda_1,..., \lambda_{M_{\max}})^T$ which makes the PRSS problem still far away from the Tikhonov regularization approach. To represent the PRSS as a Tikhonov regularization problem as in (17), a single penalty parameter should be defined. This is why the same λ value is assigned for each derivative term as $\lambda_1 = \lambda_2 = ... = \lambda_{M_{\max}} = \lambda.$

The Tikhonov regularization problem tries to minimize two objective functions, $\| \mathbf{y} - \boldsymbol{\psi}(\overline{\mathbf{d}})\boldsymbol{\theta} \|_2^2$ and $\| \mathbf{L}\boldsymbol{\theta} \|_2^2$ by combining them into a single functional form using a linear sum of the functions with a weight, λ .

The Tikhonov regularization problem is rearranged by using conic quadratic programming (CQP), which uses the advantages of both continuous and convex optimization techniques. The form of the CQP is as given below

$$\begin{aligned} &\min_{t,\boldsymbol{\theta}} \; t, \\ &\text{subject to } \left\| \boldsymbol{\psi}(\overline{\mathbf{d}})\boldsymbol{\theta} - \mathbf{y} \right\|_2 \le t, \\ &\left\| \mathbf{L}\boldsymbol{\theta} \right\|_2 \le \sqrt{Z} \; . \end{aligned} \tag{18}$$

In general form, the CQP in (18) can expressed with the following form

$$\begin{aligned} &\min_{x} \mathbf{c}^T \mathbf{x}, \\ &\text{subject to} \left\| \mathbf{D}_i \mathbf{x} - \mathbf{d}_i \right\|_2^2 \le \mathbf{p}_i^T \mathbf{x} - q_i \quad (i = 1,...,k) \; . \end{aligned} \tag{19}$$

where,

$$\mathbf{c} = (\mathbf{1}, 0_{M_{\max}+1}^T)^T, \;\; \mathbf{x} = (t, \boldsymbol{\theta}^T)^T, \; \mathbf{D}_1 = (\mathbf{0}_n, \boldsymbol{\psi}(\overline{\mathbf{d}})), \, \mathbf{d}_1 = \mathbf{y}, \, \mathbf{p}_1 = (1,0,...,0)^T, \; q_1 = 0$$
$$\mathbf{D}_2 = (\mathbf{0}_{M_{\max}+1}, \mathbf{L}), \, \mathbf{d}_2 = \mathbf{0}_{M_{\max}+1}^T, \, \mathbf{p}_2 = \mathbf{0}_{M_{\max}+2}, \;\; q_2 \le -\sqrt{Z}.$$

Here, *k* represents the number of cone in the optimization problem.

2.3. Self Organizing Map (SOM)

SOM is developed as an effective neural network technique for analysis and visualization of high-dimensional data (Kohonen, 1988). It adaptively transforms high-dimensional data into a lower dimensional discrete map of units as in Figure 3. Here, the discrete output space is called **grid**, and the nodes placed on the grid represent the **neurons**. The output neurons are generally arranged in a two-dimensional lattice providing a neighboring relation between neurons. The neurons on the lattice are positioned according to a particular shape: rectangular or hexagonal. Therefore, different neighboring relations can be formed among neurons on the grid.

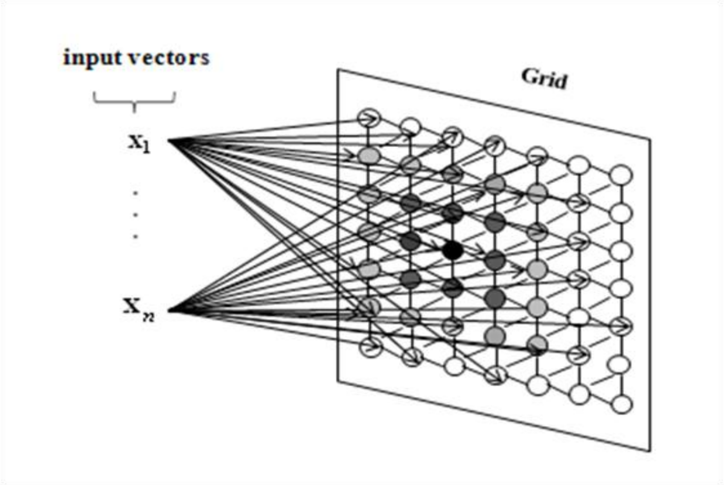


Figure 3. Topology of SOM.

Figure 3 shows the schematic diagram of the two-dimensional lattice of neurons. Each neuron has a specific topological position in the lattice and is represented by a p -dimensional weight vector, $\mathbf{w}_l = (w_{l,1},...,w_{l,p})^T \; (l = 1,...,u)$, where p is the dimension of input space; u denotes the number of neurons in the lattice, and $u < n$. Each neuron is fully connected to all input values, $\mathbf{x}_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,p})^T$ $(i = 1, \ldots, n)$; and the corresponding weight is updated from one data realization to another.

The algorithm of SOM is based on a competitive learning and is trained iteratively. It proceeds, first, by initializing the weight vectors of the neurons. This process can be done by assigning small random values to each weight vectors or by using the eigen values of the given data. After initialization, training process is achieved either by processing the input vector sequentially or as a batch. At each iteration of the training, one data point $\mathbf{x}_i$ $(i = 1, \ldots, n)$ from the original space is introduced into the grid, and the most similar neuron to the current data point is found by using a similarity measure. The closest neuron for the corresponding input vector is called **best-matching-unit** (BMU) and its weight vector is represented by $\mathbf{w}_b \in \Re^p$, where b represents the BMU. The similarity between a neuron and the input vector is usually found by using the Euclidean distance measure between the corresponding weight vector and the input vector as follows (Kohonen, 1988)

$$\mathbf{w}_b = \underset{l=1,\ldots,u}{\arg\min} \{\| \mathbf{x}_i - \mathbf{w}_l \|_2\}. \tag{20}$$

Once the BMU is found at the current iteration, t, the weight vectors of the neurons within the topological neighborhood of BMU are updated with the rule given below

$$\mathbf{w}_{l(b)}(t+1) = \mathbf{w}_{l(b)}(t) + a(t)\, h_{l(b)}(t)\, (\mathbf{x}_i(t) - \mathbf{w}_{l(b)}(t))\ (l = 1, \ldots, u), \tag{21}$$

where $\mathbf{w}_{l(b)} \in \Re^p$ represents the weight vector of the neuron inside the topological neighborhood of BMU labeled as $l(b)$; $h_{l(b)}$ is the neighborhood function defined around the BMU, and $a(t)$ is a learning rate function.

The BMU locates at the center of a topological neighborhood of neurons $h_{l(b)}$ and this neighborhood around the BMU decays smoothly with a distance measure $d_{(b,l)}$ defined between the BMU, b and the lth neuron. A typical choice of $h_{l(b)}$ is the Gaussian function in (22) due to the facts that it locates the BMU at centers and decreases monotonically as $d_{(b,l)} \to \infty$, which is a necessary condition for convergence.

$$h_{l(b)} = \exp\left(-\frac{d^2_{(b,l)}}{2\sigma^2} \right). \tag{22}$$

Here, the parameter σ is the width of the topological neighborhood which should shrink within discrete number of iterations. A popular width parameter is described by Ritter et al. (1992) as

$$\sigma(t) = \sigma_0 \exp\left(-\frac{t}{\eta_1} \right) \quad t = 0,1,2,..., \tag{23}$$

where σ_0 is the initial value of σ , η_1 is a time constant parameter , and t represents the iteration number.

As well as the neighborhood function $h_{l(b)}(t)$; the learning-rate parameter should also be time varying to provide a convergence in equation (21). In particular, it should be started with an initial value, and then, decrease gradually with increasing time (i.e. number of iteration). A common function satisfying these requirements is the exponential function denoted as

$$a(t) = a_0 \exp\left(-\frac{t}{\eta_2} \right) \quad t = 0,1,2,..., \tag{24}$$

where η_2 is another time constant parameter of the SOM algorithm (see Haykin (1999) for more details).

CHAPTER 3

PROPOSED APPROACH

3.1. Motivation

In an adaptive regression spline, the scope is to produce a good set of BFs (with optimal number of knots and their locations) for approximating the output function f in (1) with an efficient algorithm and feasible computation time. In MARS method, the greatest computational burden is in the forward selection part. During this process, BFs are added with a hierarchical manner into the model. The model starts with an intercept term, and at each successive step, a new reflected pair from the set C in (5) with the corresponding knot is introduced into the model (7) using the form $\psi_m(\mathbf{x}^m)(x_{\mathrm{j}}-\tau)_+$ and $\psi_m(\mathbf{x}^m)(\tau-x_{\mathrm{j}})_+$. Here, $\psi_m(\mathbf{x}^m)$ represents the BF in the form of (6) selected in the previous step including the product of different variables other than the current x_j $(j=1,..,p)$. Finally, the construction of model terminates when the number of BFs in the model reaches to a preset number, M_{max}.

At each forward selection step, the contribution of a newly added BF pair,

$$c_{M-1}\psi_m(\mathbf{x}^m)(x_{\mathrm{j}}-\tau)_+ + c_M\psi_m(\mathbf{x}^m)(\tau-x_{\mathrm{j}})_+, \tag{25}$$

is evaluated through a LOF criterion depends on the squared error, given in (26) defined over M BFs.

$$\underset{m,v,\tau}{\arg\min}\ \mathrm{LOF}\left((\mathbf{y}-\hat{f}_M(\mathbf{x})^2)\right), \tag{26}$$

where,

$$\hat{f}_M(\mathbf{x}) = \sum_{k=0}^{M\text{-}2} \hat{c}_k \psi_k(\mathbf{x}) + \hat{c}_{M\text{-}1} \psi_m(\mathbf{x})(x_{\mathrm{j}} - \tau)_+ + \hat{c}_M \psi_m(\mathbf{x})(\tau - x_j)_+ . \qquad (27)$$

Namely, if the model with the estimated coefficients $(\hat{c}_0,..., \hat{c}_M)$ can produce the largest decrease in the LOF criterion, the generated model forms a basis for the successive steps.

The forward step is an exhaustive search process of knot selection. Each distinct data value of each predictor variable, $x_{i,j}$ $(i = 1,..., n; j = 1,..., p)$, is a candidate knot point. So at each step of forward selection, pnM number of data points are introduced to the model with the pair of truncated linear functions, and evaluated through the LOF in (26) with a computational complexity of nM^2; here, n is the number of data points, and M is the number of BFs in the model at each step (Friedman, 1993). The computing time associated with each iteration is therefore proportional to pn^2M^3. Finally, in order to reach a final model with $M_{\max}$ BFs, the total time required for the forward selection is proportional to $pn^2M_{\max}^4$, which is then reduced to $pnM_{\max}^3$ by examining the eligible parameter values in a special order (Friedman, 1991; Friedman, 1993).

As well as $M_{\max}$, the strategy of searching knots over all distinct data values, pn makes the training of MARS computationally expensive. For a fixed number of observations, it is possible to decrease the computer time of the forward step by decreasing the number of **candidate knot locations**. In this study, to speed up the forward selection process, a new approach is proposed with changes in the knot selection search over all distinct data values to a much smaller set of data values. A subset of data points representing the original data is chosen by using a mapping approach similar to the one presented in Section 2.3. Here, the way of mapping is important because the selected points should provide a good approximation for the underlying data structure. Due to its following properties (Haykin, 1999), SOM suits for our purpose.

Property 1. *Approximation of input space*

At each iteration of training, the weight vectors of BMU and neighboring neurons come close to the current data points while the weight vectors of others are left unchanged. In this way, different weight vectors tend to become tuned in to different domain of input variables. After sufficient iterations, weight vectors tend to be located in the input space so that an approximation to the distribution of data is achieved in the sense of some minimal residual error. This approximation approach is rooted in vector quantization method which is based on Lloyd algorithm. (see Gersho and Gray (1992) for more details.)

Property 2. *Topological ordering or self-organizing*

The neurons on the lattice have spatial locations, and are connected with a neighborhood relation. This property provides a spatial concentration for network movement at each iteration. After repeated iterations, a particular domain of input space is going to be represented with the neurons topologically close to each others.

The topological order of neurons can be visualized as an elastic net of weight vectors (in red color) in the coordinates of original data points (in green color) shown in Figure 4. The lines connecting the weight vectors represent the spatial location of corresponding neurons on the lattice.

Property 3. *Density matching*

The density distribution of the underlying data can be matched by self-organizing maps. The dense regions in the input space from which the data points are drawn with a high probability of occurrence are mapped onto larger domain of the space of neurons. In Figure 4, the neurons tend to drift where the data is dense while only a few neurons are located where the data is sparse.

On this account, the resulting weight vectors of mapping can form a base for a data-driven subset of data points (representing the structure of the underlying data) that will be used as candidate knot points

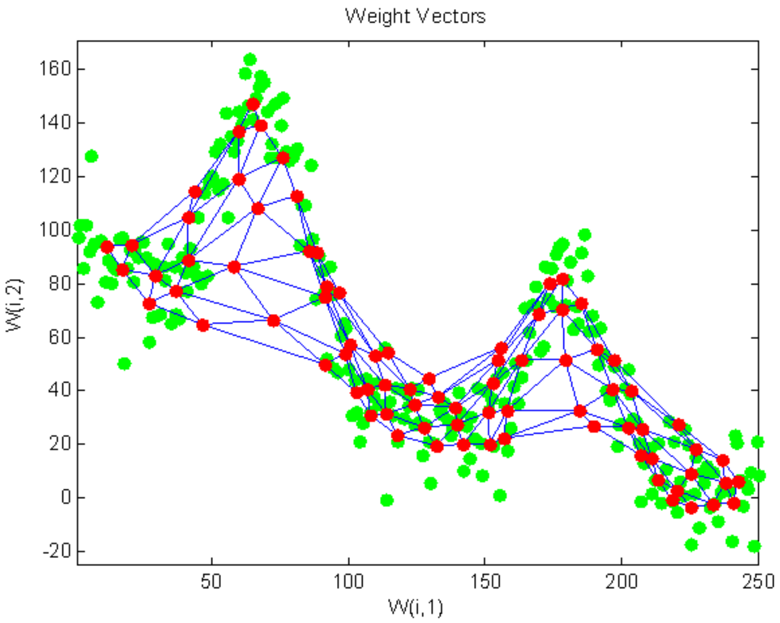


Figure 4. Weight vectors (red points) along with original data points (green points).

3.2. Proposed Approach

In this thesis, to select the candidate knot points in a more efficient way, a mapping idea mentioned in Section 2.3 is proposed. Due to the good properties of SOM emphasized in the previous section, the underlying data structures can be approximated properly with some representative weight vectors. Selecting the knot points by the help of these weight vectors can decrease the computing time of model building significantly without decreasing the accuracy.

In the proposed method, called S-FMARS, each data point $\mathbf{z}_i = (\mathbf{x}_i, y_i)$ $(i = 1,..., n)$ are considered as input data and mapped from the original data space, $\Re^{(p+1)}$ into a grid via (*p+1*)-dimensional weight vectors $\mathbf{w}_l = (w_{l,1},...,w_{l,p+1})^T$ $(l = 1,..., u)$.The main reason of taking into account the response values with the predictor values during the mapping is to preserve the relation between predictors $\mathbf{x}$ and response variable y in the new space. Once the weight vectors are updated according the underlying dataset, the weight vectors exposed to at least one data point (called taking a hit) are selected as the representatives of the corresponding data points. Then, piecewise-linear regression splines are built at the knot selected from the set of values represented by the selected weight vectors. For this set, the selected weight vectors can be directly used as a candidate knot points or used as a reference for any other points in data space to be evaluated as the potential knot location. For example, instead of using weight vectors, the original data points referred by the weight vectors can be considered as candidate knot point. The way of determining a point in data space by the help of weight vectors is named **projection** in this thesis. Namely, the weight vectors are projected from the grid to the original data space. Two ways of projection are studied in this study: the **nearest data point** and **the mean of k-nearest data points**. While the nearest data method finds the closest data vectors to the selected weight vectors, *k*-nearest data method takes the average of the *k* data points close to weight vectors. Here, *k* denotes the number of hits of the corresponding neuron. Due to its accuracy and prediction performances, weight vectors are projected onto data space using the nearest-data method. The results of the corresponding analyses are given in Appendix D.

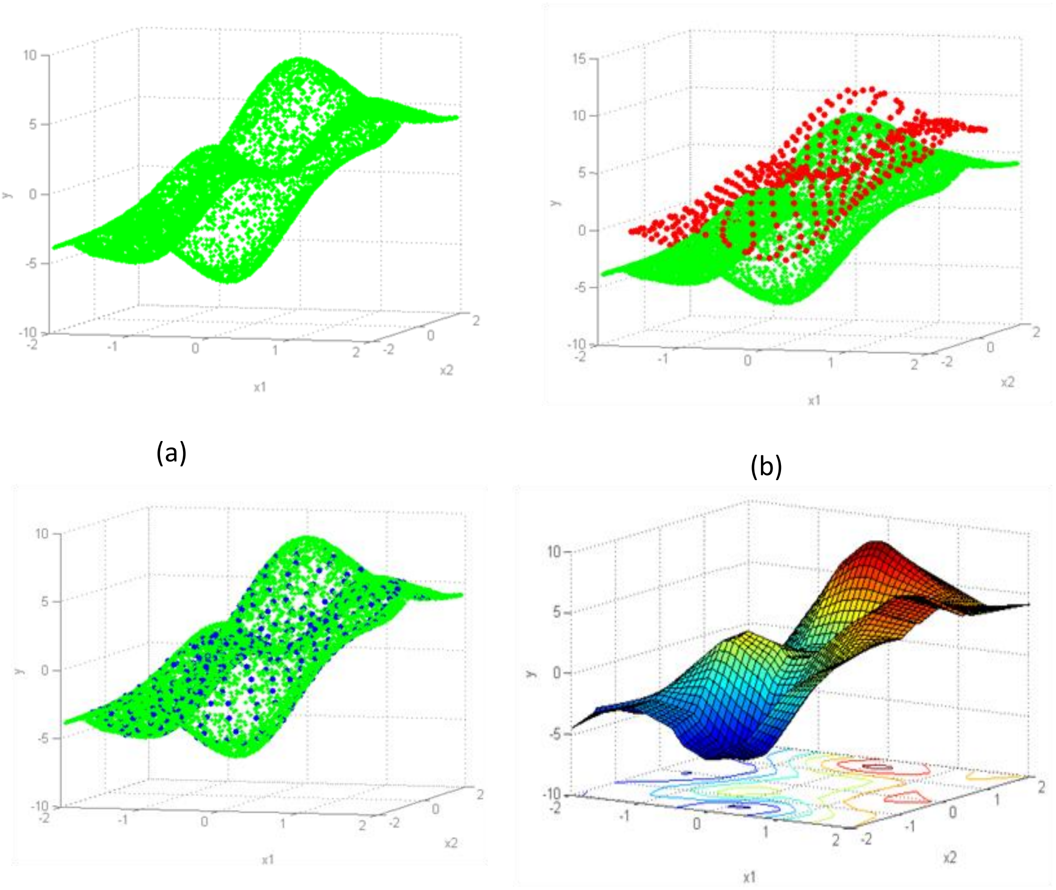
The steps of the proposed approach S-FMARS are described below through an example given in Figure 5, and the pseudo code for the S-FMARS algorithm is presented in Figure 6.

3.3 Algorithm

1. (*Mapping*). Data points $\mathbf{z}_i = (\mathbf{x}_i, y_i)$ $(i = 1,..., n)$ (Figure 5.a) are mapped into a space of neurons by an iterative algorithm given in Section 2.3.
2. (*Selection of neurons*). In Step 1, if a neuron is selected as a BMU during the training, it means making a **hit**. In this step, the neurons with at least one hit are selected.
3. (*Projection*). The weight vectors associated with the neurons selected in Step 2 (Figure 5.b) may not be one of the original data points. So the weight vectors are projected onto the original data space (Figure 5.c), where the projected data point is represented by $\tilde{\mathbf{z}}_s = (\tilde{x}_{s,1},...,\tilde{x}_{s,p},\tilde{y}_s)$ $(s = 1,..., S)$, where S denotes the number of selected neurons, and $S \le u$.
4. (*Knot selection and model building via the Forward Selection*). The estimated model is built on the truncated linear functions in which the knots are the values of the predictor data, $\tilde{\mathbf{x}}_s$, projected in Step 3. Here, every distinct value of the corresponding predictor variable, $\tilde{x}_{s,j}$ $(s = 1,..., S;\ j = 1,..., p)$, are considered as candidate knot points for BFs. The new set of truncated linear functions is given as

$$D = \{\ (x_j - \tau^*)_+, (\tau^* - x_j)_+ \,|\, \tau^* \in \{\tilde{x}_{s,j}\}\ ,\ j = \{1,..., p\}\ , s = \{1,..., S\}\ \}. \qquad (28)$$

S-FMARS constructs a model by regressing *y* on the BFs developed over the set of *D* in a stepwise manner, and the significant BFs with the corresponding knots are selected via the lack-of-fit criterion in (26) (Figure 5.d).



(a)

(b)

(c)

(d)

Figure 5. (a) Original data points. **(b)** Weight vectors of selected BMUs and original data points. **(c)** Projected weight vectors and original data points. **(d)** S-FMARS model for which knots are selected from the projected weight vectors.

3.4 Parameters of S-FMARS Approach

In S-FMARS approach, the set of BFs obtained after running the algorithm presented in Figure 6 contains less number of candidate BFs than that of the set C in (5), so that the computing time of forward step decreases in a significant manner. In this approach, it is also possible to decrease the computing time further by decreasing the size of set D in (28) via two parameters called **grid size** and **threshold value** set for the number of hits of each neuron. However, changing the values of parameters in

```
input: a set of data vectors $\mathbf{z}_i = (\mathbf{x}_i, y_i)$ $(i = 1,\dots,n)$; a threshold value $\tilde{t}$; grid size $g$.
lattice: a grid with a specified size and a set of weight vectors, $\mathbf{w}_l$ $(l = 1,\dots,u)$.
begin for mapping
  initialize each weight vector $\mathbf{w}_l$.
  repeat
        select one data vector, $\mathbf{z}_i = (\mathbf{x}_i, y_i)$.
        find the BMU such that $\mathbf{w}_b = \arg\min_{l=1,\dots u}\{d(\mathbf{z}_i, \mathbf{w}_l)\}$.
        for all weight vectors of neighboring neurons, $w_{l(b)}$, do
            $\mathbf{w}_{l(b)}(t+1) = \mathbf{w}_{l(b)}(t) + a(t)\, h_{l(b)}(t)(\mathbf{z}_i(t) - \mathbf{w}_{l(b)}(t))$
    until the termination condition holds (until a specified number of training epochs).
end
select the BMUs whose number of hits is greater than a specified threshold value, $\tilde{t}$. The corresponding weight vectors are denoted as $\mathbf{w}_s$ $(s = 1,\dots,S)$.
project the weight vector of the selected BMUs, $\mathbf{w}_s$ $(s = 1,\dots,S)$ to the data point $\tilde{\mathbf{z}}_s = (\tilde{\mathbf{x}}_s, \tilde{y}_s)$ such that $d(\mathbf{w}_s, \tilde{\mathbf{z}}_s) = \arg\min_{i=1,\dots n}\{\, d(\mathbf{w}_s, \mathbf{z}_i)\}$.
begin model building with $B_1(\mathbf{x}) = 1$.
 $M = 2$
    while $M < M_{\max}$
        for $m = 1$ to $M - 1$ do:
            for $j \notin \{j(k,m) | 1 \le k \le K_m\}$,
                for $\tau^* \in \{\tilde{x}_{s,j} \mid B_m(\mathbf{x}) > 0\}$,
  $g \leftarrow \sum_{k=0}^{M-2} a_k \psi_k(\mathbf{x}) + a_{M\text{-}1}\psi_m(\mathbf{x})(x_j - \tau^*)_+ + a_M \psi_m(\mathbf{x})\tau^* - x_j(\tau^* - x_j)_+$,
                    $LOF \leftarrow \min_{a_1,\dots,a_{M+1}} LOF(g)$.
                end
            end
        end
    end
```

Figure 6. The algorithm of S-FMARS.

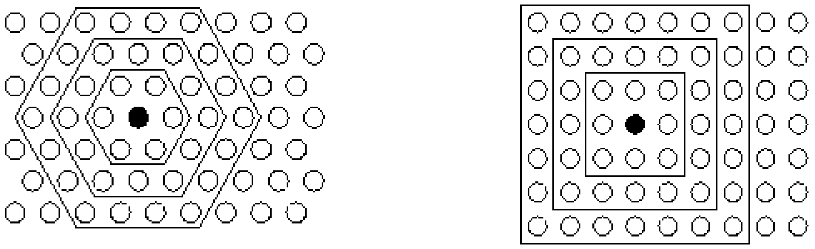
decreasing the computing time should be achieved carefully by considering the model accuracy.

Since some representative knot points could be eliminated while decreasing the size of map, accuracy of the corresponding models may become worse. In the following sections, the effect of these parameters both on computing time and model accuracy are evaluated through a sensitivity analysis.

Since some representative knot points could be eliminated while decreasing the size of map, accuracy of the corresponding models may become worse. In the following sections, the effect of these parameters both on computing time and model accuracy are evaluated through a sensitivity analysis.

3.4.1 Grid Size

In S-FMARS approach, mapping starts with a grid topology that can be hexagonal or a rectangular whose size is preset in advance (see Figure 7). The grid size represents the dimension of a lattice in terms of total number of neurons (Vasento et al., 2000). Cardinality of neurons has an important effect on the mapping quality; so the approximation capability. If the grid size is large enough, SOM builds a dense lattice with a large number of BMUs, and approximates the underlying data distribution better than the lattice with a small number of map units. However, SOM with a large number of BMUs produces a large subset of candidate knot points, so that it needs more computing time. Therefore, the size of a lattice is considered as a trade-off between the less computing time and a good approximation both in mapping and modeling.



(a) Hexagonal (b) Rectangular grid

Figure 7. Examples of grid structures.

The size of the grid can be either specified by the user, or can be defined heuristically. In general, a heuristic for grid size $g = 5\sqrt{n}$ introduced by Vesanto et al. (2000) is used for adequate approximation of the original data points, where n represents the number of original data points. In this study, for each dataset, the effect of grid size on computing time and model accuracy are observed by running the S-FMARS approach for 10 different grid sizes as in Figure 8, where $g = \sqrt{n}$. The best number of grid size is then determined by observing the changes in RMSE and computing time in seconds.

Figure 8 and Figure 9 displays the results of a sensitivity analysis constructed for the Dataset 2 in Table 1. The results obtained for other problems are given in Appendix C in the same order as in Table 1. In Figure 8, the RMSE of a model obtained after the run of S-FMARS approach gets smaller as the grid size for the approach gets larger. This is due to the fact that large grid sizes provide better approximation of the underlying distribution than those of the small grid sizes. However, more computing time is required for both mapping and modeling as the grid size gets larger (see Figure 9).

To select the best grid size for the underlying dataset, the changes in both RMSE and CPU time should be evaluated carefully. As it is seen in Figure 8, the change in

RMSE becomes stable at grid size 5g/4, while the change in CPU time significantly increases after the grid size g. Therefore, to render a decision on the best grid size, a ratio taking into account both the model accuracy and CPU time simultaneously can be stated as follows,

$$r = \text{rmse/time}. \qquad (29)$$

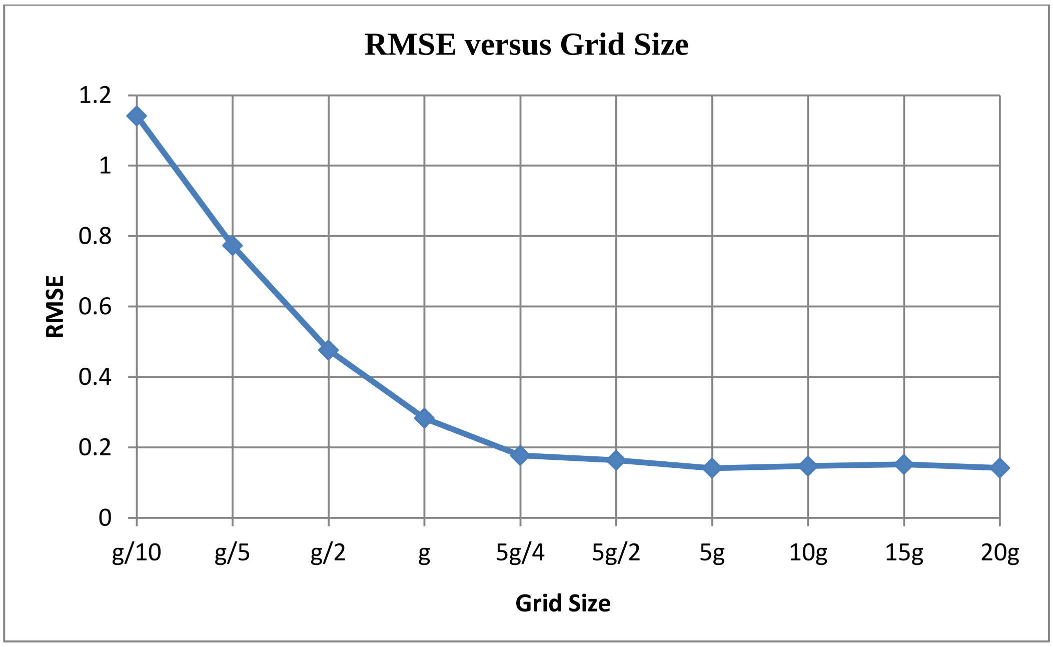


Figure 8. Graph of RMSE versus grid size for Dataset 2.

The graph of "Ratio versus grid size" (Figure 10) for the underlying dataset can provide an intuition about the best grid size for the S-FMARS approach. The grid size where the ratio does not change significantly can be determined as the best size for the accurate model with efficient computing time.

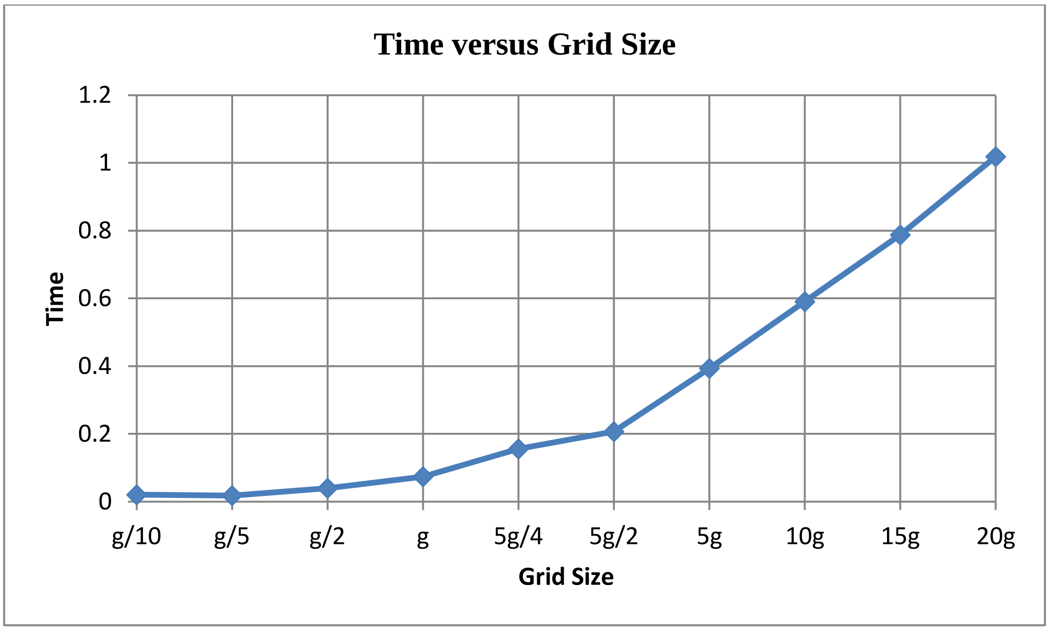


Figure 9. Graph of Time versus grid size for Dataset 2.

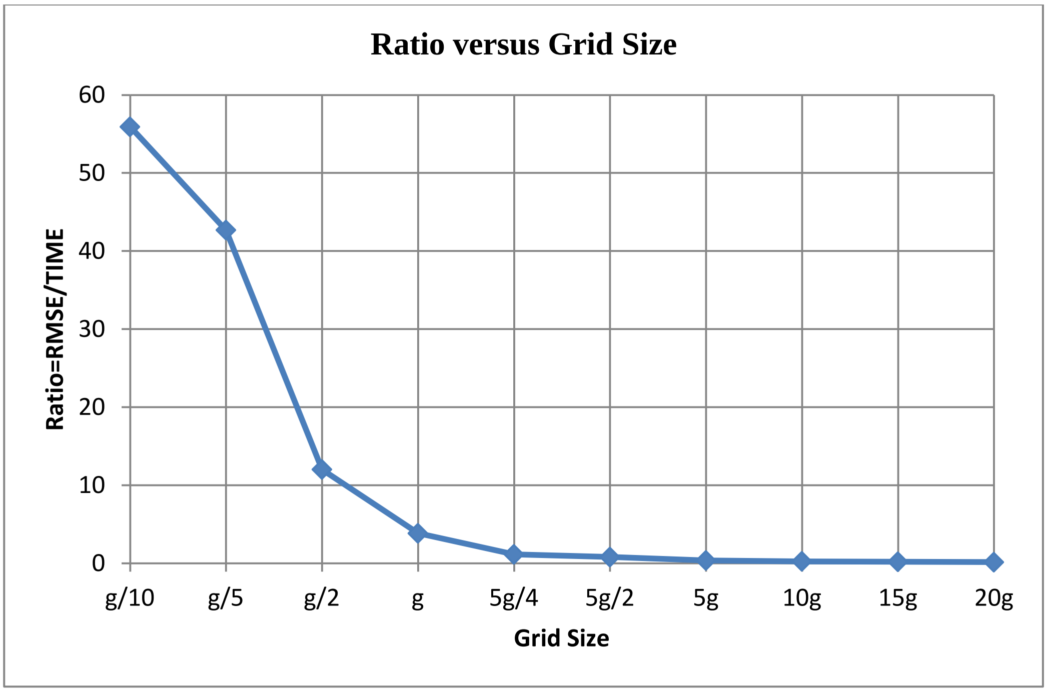


Figure 10. Graph of Ratio versus Grid size.

Moreover, the best grid size can also be determined by setting a stopping value for the slope of the ratio. For instance, the best grid size (*g*) for the dataset used in this study can be taken as *5g*, if the stopping value is set to 0.15.

3.4.2. Threshold Value

In S-FMARS approach, another affecting parameter on computing time is the number of hits. Here, the hit represents the data point coming close to a neuron by an updating rule as stated in Section 2.3 during mapping. In the proposed approach, neurons taking at least one hit (the neurons determined as BMUs) are selected, and their corresponding weight vectors are used as candidate knot points. On this account, the set of candidate knot points (set D in (28)), so the number of neurons selected can be controlled by a threshold value, $\tilde{t}$ set for the number of hits.

The number of hits owned by each neuron can be visualized via a graph of sample hits (see Figure 11). The number in each cell gives the frequency of data points mapped from the input space to the corresponding neuron. While neurons with large number of data points represent the dense regions of data, the ones with small number of data points represent the sparse regions or outliers. The neurons with zero data points are not BMUs of any original data point, so they are disregarded.

Setting a threshold value for the number of hits can control the selection of neurons or weight vectors for the knot placement. A high threshold value reduces the number of neurons, so the candidate knots. Besides, a high threshold value leads the S-FMARS algorithm to select the neurons representing the dense regions of data points. Thus, the neurons attained to sparse regions or outliers are automatically eliminated before the knot placement. This elimination procedure however may decrease the model quality although it decreases the CPU time.

A low threshold value leads S-FMARS algorithm to produce a large number of candidate knot points. Therefore, the model built after implementing the S-FMARS approach is generally more accurate than the models obtained after running the S-FMARS approach processed with a high threshold value (see Figure 12). On the other hand, S-FMARS with low threshold requires more CPU time to build a model (see Figure 13).

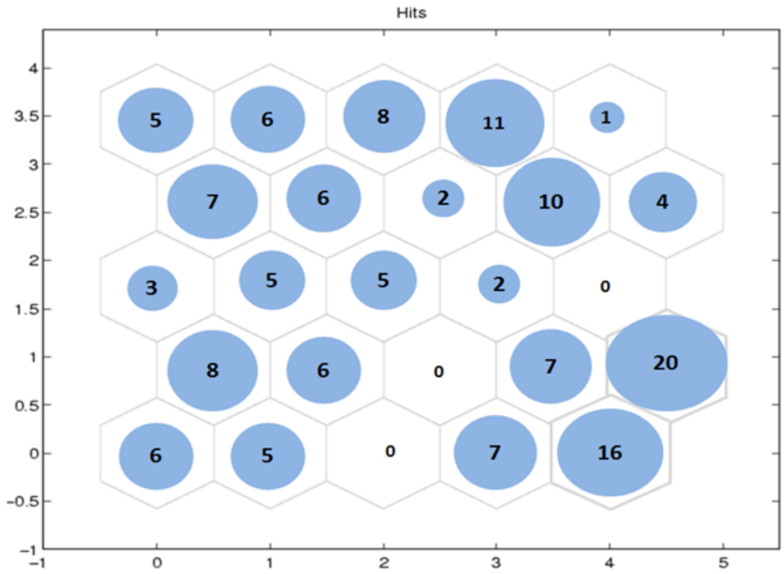


Figure 11. The sample hits of a 5x5 hexagonal grid including 25 neurons.

When $\tilde{t} = 1$, more weight vectors are selected, and a large set of candidate knot locations is obtained. The model constructed on this set of knot points can approximate the underlying function more accurately as in Figure 12, and the model accuracy (RMSE) becomes worse as the threshold value gets larger. However, the computing time required for model building is high for $\tilde{t} = 1$, and it decreases for the large threshold values as presented in Figure 13.

The aim here is to increase the threshold value without decreasing the accuracy of models. On this account, to see the effect of threshold value on both model accuracy and computing time, the slope of ratio given in (30) is examined for six different threshold levels starting $\tilde{t} = 1$ to three standard deviations of hits (std) above the average hits (m_u) calculated as follows

$$m_u = n/u, \tag{30}$$

where n is the number of data points, and u is the number of neurons in the map.

As the threshold value for the number of hits gets larger, the ratio increases; hence, inaccurate models are obtained (see Figure 14). Therefore, the point where the ratio is settled and starts to increase further can be determined as the best threshold value. Similarly, like in grid size, a cut off value set for the slope of ratio can be used to determine the best threshold value of the number of hits. For the dataset in Figure 14, the point *mu+std* can be used set as the best threshold value where the slope is no more than 0.15.

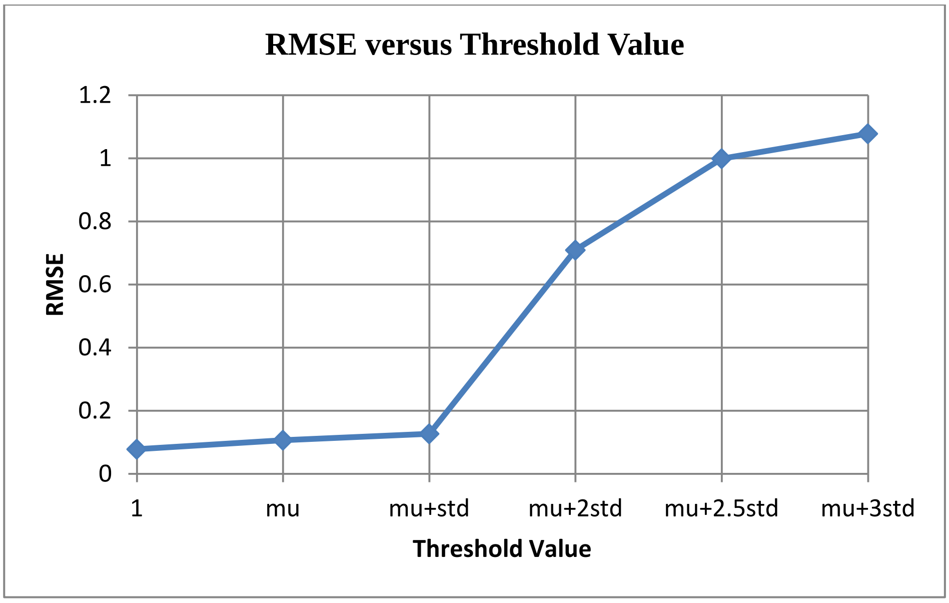


Figure 12. Graph of RMSE versus Threshold value.

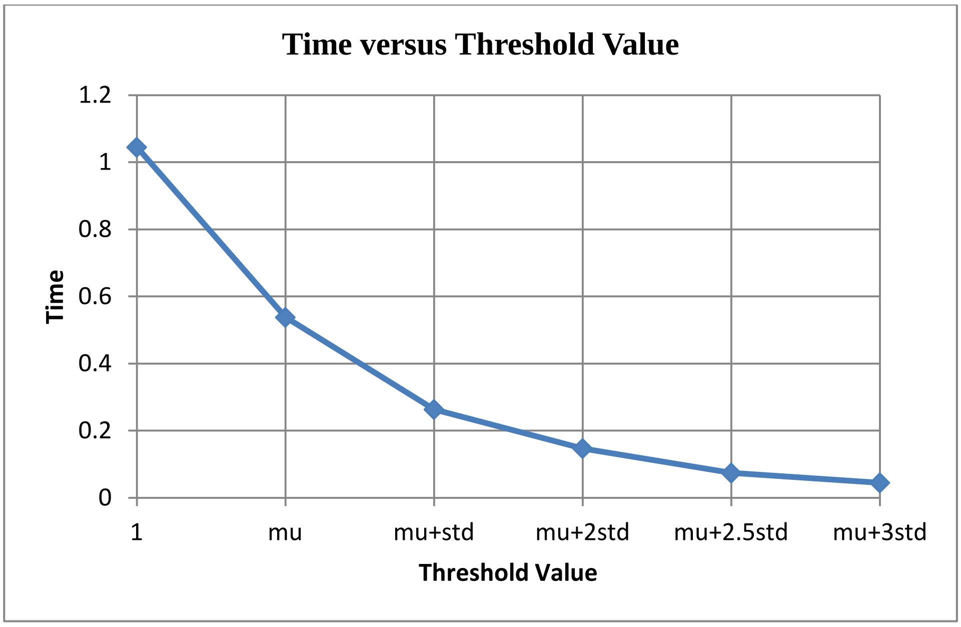


Figure 13. Graph of Time versus Threshold value.

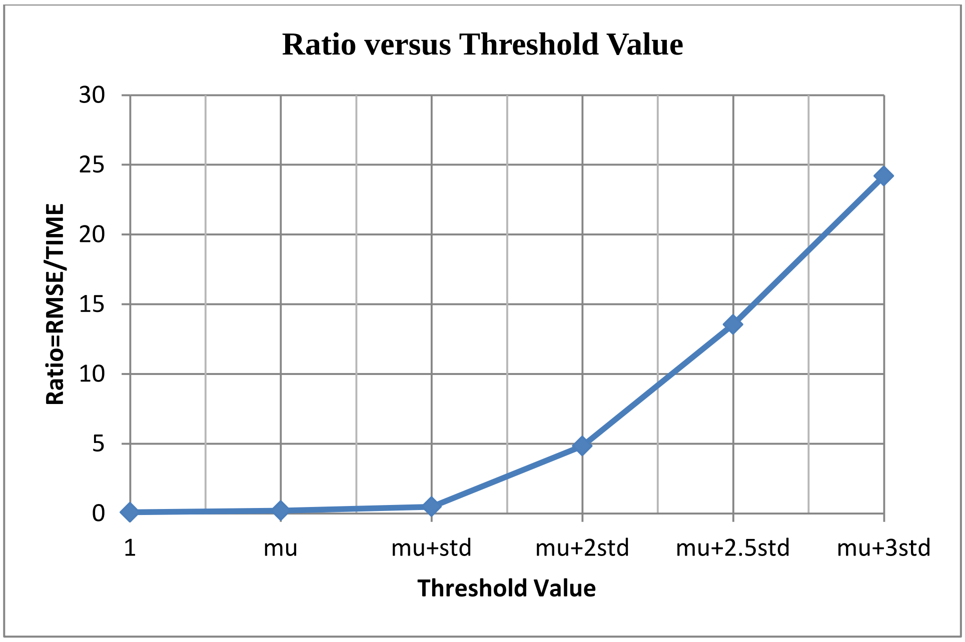


Figure 14. Graph of Ratio versus Threshold Value.

CHAPTER 4

APPLICATIONS

4.1 Background on Applications

The proposed algorithm given in Section 3.3 includes two main steps: mapping and model building. Firstly, a set of candidate knots is determined via a mapping and projection, and then, a regression spline model is developed by searching the knots over the set of data points gathered in the first step. The implementation of mapping idea prior to the model building is proposed to decrease the computational burden of adaptive regression spline mainly caused by the forward step. Together with the mapping and model building strategy, the proposed approach can be considered as a modified forward selection algorithm of MARS. The performance of the proposed approach, S-FMARS, is evaluated and compared with the forward selection algorithm of MARS and MinSpan approaches (for detailed information see Section 2.1.1) through various applications with respect to different performance criteria. The proposed approach is compared with FMARS and MinSpan approach separately in Section 4.3 and 4.4, respectively. In addition, to control the complexity of the S-FMARS model and to prevent the overfitting problem, the backward elimination strategy of MARS and the idea behind the CMARS are implemented to the proposed forward selection algorithm. Performance of five methods called MARS, MARS with MinSpan, SMARS, CMARS, S-CMARS are compared with respect to accuracy, complexity, stability and robustness criteria in Section 4.5. Here, SMARS refers to the method including the proposed forward selection algorithm and backward elimination step of MARS. S-CMARS denotes the modified version of CMARS which is built on the BFs selected by S-FMARS.

4.1.1 Datasets and Validation Techniques

The experiments are conducted on 10 artificial and six real datasets. The list of the datasets with their four different features including number of data points (**size**), number of predictor variables (**scale**), degree of interaction (**nonlinearity**) and **noisy** behavior are given in Table 1. In all datasets, predictor variables and response variable are all taken as continuous. The first six datasets together with the Dataset 10 are generated from the functions originally given by Jin et al. (2001), (see Appendix A and Appendix B for the function descriptions and the grid plots of the functions, respectively). Dataset 7 is the robot arm example used by Friedman (1993), and Dataset 8 is taken from the MATLAB user's quide (2010). The function used for Dataset 9 is the sinus function and the corresponding data is generated with some noise. The last six datasets belong to real life problems, and are originally taken from the UCI repository (Frank, 2010). These datasets are selected according to their size, *n* and number of predictor variables, *p*. Before the construction of the models, all datasets are preprocessed and standardized to become comparable.

For artificial data sets, to compare and validate the performance of the methods, a test data is generated by the same function used for training data. In real data sets, however, 3-fold with three replications cross validation approach is used for model validation. In this approach, the original data set is randomly divided into three part (fold). At each time, one part is retained for testing, and other two parts are used for model building. Hence, three models are built at each time. This process is replicated three times with new partitions.

4.1.2 Software

The proposed approach S-FMARS is written entirely in MATLAB^R (Matlab, 2010) with the assistance of SOM Toolbox (Vesanto, 2000) and ARESLab Toolbox. ARESLab Toolbox is created by Jekabsons (2011) as a collection of Matlab codes for implementing MARS algorithm. This toolbox implements the main functionality of MARS technique close to the description in the Friedman's paper (Friedman, 1991). It should be indicated that the model building is not accelerated using "Fast MARS" queuing (Friedman, 1993) together with the "fast least square update technique" in this code. SOM Toolbox (Vesanto, 2000) is another Matlab library created for self-organizing maps.

To develop a CMARS model, first, the Matlab code written for S-FMARS is used to obtain the BFs provided from the proposed forward selection algorithm. Then, the code written in MATLAB (2009a, The MathWorks, U.S.A.) by Yerlikaya (2008) and developed further by Batmaz et al. (2010) is used to obtain CMARS models. For optimization process in CMARS, the MOSEK optimization software (6. MOSEK ApS, Denmark) is utilized.

Table 1. Features of the datasets.

Datasets	Sample Size (n)	# of inputs (p)	Nonlinearity	Noisy Behaviour
1	1000	7	high	no
2	1000	5	low	no
3	1000	10	low	no
4	10000	2	high	no
5	10000	3	high	no
6	10000	3	low	no
7	1000	5	high	no
8	10000	2	high	no
9	100	1	low	yes
10	100	2	low	yes
Parkinsons	578	21	high	-
Red Wine	1599	11	high	-
Com.Crime	879	24	high	-
Conc. Comp.	1030	8	high	-
PM10	500	7	low	-
Auto Mpg	398	7	low	-

4.1.3 Performance Criteria and Measures

The performance of each method is measured with respect to accuracy, complexity, stability, robustness and efficiency criteria. To evaluate the goodness of the model fit, Root Mean Square Error (RMSE), Adjusted-Multiple Coefficient of Determination (Adj-R^2) and GCV given in (11) are used for training data. The equations for RMSE and Adj-R^2 are given in (31) and (32), respectively. RMSE indicates the grossly inaccurate estimates. Namely, the smaller the RMSE is, the better the model fits to the data. Adj-R^2 is a penalized form of R^2 with respect to the number of predictors in the model. It gives the amount of variation in response which is explained by the model. Thus, the higher the Adj-R^2, the better the model is. As stated in Equation (11), GCV criterion takes the number of BFs in the model into account as well as the model accuracy. Hence, the model complexity can be evaluated and compared with respect to the GCV measure.

$$RMSE = \left(\frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \right)^{1/2}, \qquad (31)$$

where y_i is the *i*th observed response value, $\hat{y}_i$ is the *i*th fitted response, and *n* denotes the number of observations.

$$Adj\text{-}R^2 = 1 - \left(\frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2} \right) \left(\frac{n-1}{n-p-1} \right), \qquad (32)$$

where $(n-p-1) \neq 0$. Here, $\bar{y}$ is the mean response, and *p* denotes the number of predictors in the model.

Since the measures obtained for the training data are not sufficient to access the accuracy of newly predicted points, a test data is also used to verify the prediction accuracy of the models. RMSE and Adj-R^2 measures are used to examine the prediction performances. Furthermore, to measure the change in the performance of methods between the training and test datasets, a stability measure defined below is used

$$\min\left\{\frac{MR_{TR}}{MR_{TE}},\frac{MR_{TE}}{MR_{TR}}\right\}, . \qquad (33)$$

where MR_{TR} and MR_{TE} represents the performance measures (RMSE or Adj-R^2) for the test and training data sets, respectively. The model whose stability measure is close to one represents a stable model. Furthermore, robustness of the methods under different data sets is also evaluated with the help of the spread of performance measures used.

The efficiency of each method is measured by recording the computational run times (the CPU time) of models to make the results comparable. Both methods are tested on the same platform (Intel Core2 Duo CPU T7250@2.00 GHz 2.00 GB RAM). For each dataset, the CPU times of the models are recorded in **seconds**. A detailed analysis is performed on CPU times of methods with respect to sample size and number of predictor variables (refers scale of data). To achieve such kind of an analysis, real datasets in Table 1 are categorized into two groups as medium/large and small/large according to sample size and scale, respectively. Moreover, the differences between CPU times of methods according to sample size and scale are also tested statistically by a nonparametric method called Mann-Whitney Test. Mann-Whitney is a nonparametric version of two-sample t test used for independent samples when the normality assumption is violated (Lehmann, 1975).

To compare the performances of two models, **one-sample sign test** (Gibbons and Chakraborti, 2003) is used in our experiments. Here, the datasets used by two models are considered as paired sample, which means there is a dependency between them.

One-sample sign test is a nonparametric test, which makes little assumptions about the nature of underlying distributions. Generally, it is used as an alternative to one-sample paired t-test and Wilcoxon signed-rank test, respectively when the normality assumption is violated and population distribution is not assumed to be symmetric. In this study, one-sample sign test is interpreted for α=0.05 significance level in the comparison studies of two methods. To compare the performances of more than two models, repeated analysis of variance (RANOVA) test (Davis, 2003) is used. RANOVA is a statistical test used for mean comparison. The hypothesis stated for model comparison is as follows:

$$\begin{aligned} &H_{0:}\ \mu_1 = \mu_2 = \mu_3 ... = \mu_k \\ &\text{versus} \\ &H_{1:}\ \textit{at least one is different} \end{aligned} \tag{34}$$

Here, μ stands for the expected value of a performance measure such as RMSE, GCV, etc. used in the comparisons. Once the test is rejected at α=0.05 significance level, the differences between models are tested pairwisely using Fisher’s Least Significant Differences (LSD) test. One-sample sign test and RANOVA teat are applied for training and test datasets as well as stabilities of the measures for real life data sets via the statistical software SPSS^{TM}. These tests are not applied to artificial datasets since the underlying normality and variance equality assumptions are not satisfied. The reason for lack of normality and variance inequality is the fact that the measures obtained are in different orders (or scales).

4.2 Selection of S-FMARS Parameters for Datasets

As mentioned in Section 3.4, S-FMARS has two important parameters that has effect in decreasing the computational run time and increasing the model accuracy. The grid size is the parameter that controls the approximation quality. If the underlying data points are mapped into a grid with a large number of neurons, then the underlying input pattern can be approximated well; otherwise, more information is lost during the mapping. The other parameter is the number of data points assigned to each neuron after mapping. If a high threshold value is set for the number of data points grouped around the neuron, then less number of weight vectors is selected as the reference for candidate knot points. That is, the set of candidate knots is restricted to more than the case for which the low threshold value is set. In Chapter 3.4, the ways of selecting the best values for these parameters are given.

In this chapter, the best S-FMARS parameter values are determined for all datasets utilized in this study. That is, the performance of S-FMARS method with respect to model accuracy and time efficiency is determined as a result of a sensitivity analysis performed on 10 different grid size and six distinct threshold values. The design levels determined for the grid size and the threshold value are given in Table 2. Here, $g=\sqrt{n}$, and m_u and std represent the mean and standard deviation of data points assigned to neurons, respectively.

Table 2. Design Values for Grid Size and Threshold Value.

Grid Size	Threshold Value
$g/10$	1
$g/5$	m_u
$g/2$	m_u+std
g	$m_u+2\,std$
$5g/4$	$m_u+2.5\,std$
$5g/2$	$m_u+3\,std$
$5g$	
$10g$	
$15g$	
$20g$	

The computational run time and model accuracy of the S-FMARS method is observed for each design value given in Table 2. As mentioned in Section 3.4.1, as the grid size of the lattice to which the original data points are assigned increases, the approximation of the underlying input pattern become well, so that model accuracy increases. On the other hand, the computing time of the method increases. The effect of the threshold value on the performance of S-FMARS method is the exact opposite of grid size (see Section 3.4.2). Namely, as the threshold value increases, the accuracy of the model and the computational run time decrease. This is due to the fact that less number of weight vectors is selected as a reference for the candidate knot points which leads the model to be built on the less number of candidate knot points. Hence, model accuracy and computing time decreases. This stated effects of grid size and threshold value on the performance of S-FMARS with respect to model accuracy and computing time is valid and observed for all datasets. The performance measures of S-FMARS calculated for the design points in Table 2 are given in Appendix C for all datasets.

In determination of the best parameter values, the graphs including “Ratio versus grid size” and “Ratio versus threshold value” are used. Here, the measure of “Ratio” denotes the ratio between RMSE and computing time (see Equation 29). With the help of this measure, the change both in model accuracy and computing time can be observed for different parameter values. The breaking point where the lines become stable can be used as the best parameter values. As well as the performance table, the graphs of “Ratio versus grid size” and “Ratio versus threshold value” are presented in Appendix C for all datasets.

As a result of the sensitivity analysis, the best grid sizes and threshold values of S-FMARS method are determined for artificial datasets and real datasets, in Table 3 and Table 4, respectively.

Table 3. Best Parameter Values For Artificial Datasets.

Datasets	Grid Size	Threshold Value
1	$5g/2$	1
2	$5g/4$	$m_u + std$
3	$g/2$	$m_u + std$
4	$g/2$	$m_u + 2std$
5	$g/2$	1
6	$g/2$	1
7	$g/2$	m_u
8	$5g/2$	$m_u + std$

Table 4. Best Parameter values for Real Datasets.

Datasets	Grid Size	Threshold Value
AutoMpg	$5g/2$	1
ComCrime	$5g/4$	m_u
ConcComp	$5g$	m_u
Parkinsons	$5g$	1
PM10	$5g$	1
Redwine	$5g$	1

4.3. Comparison Study 1

In this section, the performance of the proposed approach, S-FMARS is evaluated and compared with the forward selection algorithm of MARS, named as FMARS. The performances are evaluated through eight artificial data, six real data and two noisy data with respect to **accuracy**, **complexity**, **robustness** and **time efficiency**. Analyses are given under the name of artificial datasets, real dataset and noise analysis.

4.3.1. Artificial datasets

This section evaluates and compares the methods for Datasets 1-8. Both method use the same number of interaction terms (*Int.*) for each data set, and allow their models grow up to the same preset number of BFs (M_{max}), which is 100 for all cases. The accuracy and complexity measures of models calculated for each training data and the CPU time required for the corresponding models are given in Table 5, as well as the number of BFs found in the final model ($\mathrm{BF}_{\mathrm{final}}$).

The number of BF in the final model denotes the complexity of the model. For almost all models, two models have similar complexity. The accuracy measures calculated for each method seem close to each other in Table 5. For data sets three, four, six and eight, S-FMARS performs better than FMARS with respect to RMSE. For three data sets, S-FMARS overperforms FMARS with respect to complexity measure (GCV). It is noted that Adj-R^2 values of all models are very high for all cases. This may be due to the overfitting problem or smoothness of the underlying datasets, which do not include noise. The prediction performances of both methods and their stabilities of measures are compared via the RMSE and Adj-R^2 measures as given in Table 6. It can be indicated that the prediction performance of S-FMARS is slightly better than that of FMARS for five datasets (see Table 6) and more stable than FMARS for four data sets.

The models are compared with respect to time efficiency via CPU times in seconds given in the last column of Table 5. It is seen that S-FMARS is much more efficient than FMARS for all datasets. It provides at least 83 % decrease in the CPU times. In addition, S-FMARS achieve this reduction in time without losing much in accuracy.

Table 5. Performances of FMARS and S-FMARS on the train data.

Datasets	Methods	Int.	BF$_{final}$	RMSE	Adj-R^2	GCV	CPU Time(sec.)	Decrease in time (%)
1	FMARS	4	100	1062.9*	0.977*	2013650*	6210.9	95
	S-FMARS		100	1089.1	0.976	2114441	311.8*	
2	FMARS	1	43	4271.9*	0.999	21796917*	44.1	83
	S-FMARS		43	4359.8	0.999	22703916	7.5*	
3	FMARS	2	47	24.1	0.999	740.5	1934.5	88
	S-FMARS		47	24.0*	0.999	740.1*	231.1*	
4	FMARS	2	77	0.026	0.998	0.001	16904.8	94
	S-FMARS		81	0.025*	0.998	0.001	999.7*	
5	FMARS	2	43	513.3*	0.999	269180*	9756.7	95
	S-FMARS		45	514.2	0.999	270354	516.8*	
6	FMARS	2	23	2.969	0.999	8.913	3521.9	99
	S-FMARS		23	2.892*	0.999	8.456*	51.0*	
7	FMARS	4	100	0.028*	0.992	0.001*	3311.4	94
	S-FMARS		100	0.030	0.991*	0.002	186.7*	
8	FMARS	2	81	0.166	0.998	0.029	93483.0	99
	S-FMARS		78	0.151*	0.998	0.024*	484.7*	

Note: * indicates better performance.

Table 6. Performances of FMARS and S-FMARS on the test data and stabilities.

Datasets	TEST				STABILITY			
	RMSE		Adj-R^2		RMSE		Adj-R^2	
	FMARS	S-FMARS	FMARS	S-FMARS	FMARS	S-FMARS	FMARS	S-FMARS
1	1120.7*	1126.7	0.959	0.959	0.948	0.967*	0.982	0.983*
2	3849.4	3834.1*	0.999	0.999	0.901*	0.879	1.000	1.000
3	23.0*	23.1	0.999	0.999	0.954	0.963*	1.000	1.000
4	0.026	0.026	0.998	0.998	1.000*	0.962	1.000	1.000
5	538.5	529.7*	0.999	0.999	0.953	0.971*	1.000	1.000
6	2.981	2.869*	0.999	0.999	0.996*	0.992	1.000	1.000
7	0.029	0.028*	0.986	0.986	0.966*	0.933	0.994	0.995*
8	0.168	0.151*	0.998	0.998	0.988	1.000*	1.000	1.000

Note: * indicates better performance.

4.3.1.1 Effects of Maximum number of BFs and sample size on the CPU time

The computational run time of both methods depends on the problem size (*n*) and a user-specified maximum number of BFs (M_{max}). To obtain the performance of both methods for different *n* values, five different datasets with n=400, 800, 1600, 3200 and 6400 are generated using the function in Dataset 8 (Figure 15). Additionally, FMARS and S-FMARS models are built for three different M_{max} values 20, 40 and 60.

The results presented in Table 7 show that as *n* and M_{max} increase, the CPU time required for model building drastically increases for both methods. Moreover, one-sample sign test signifies that the accuracy and complexity measures of two models are not statistically different for each M_{max} and *n* values (*p*-values > 0.05). Computing time of S-FMARS is less than that of FMARS for all sample size and M_{max} combinations (see Table 7). Especially for large datasets, the decrease in CPU time is more drastic than for small ones. As it is seen in Figure 16, which displays the run times of methods recorded for the models with M_{max}=60, the difference between the CPU times of two methods become noticeable as the sample size increases. Correspondingly, while the decrease in CPU times is 70 % for the dataset with smallest *n* and M_{max} value, the decrease in CPU is 98% for the largest dataset with large number of M_{max}.

In addition, CPU times of both methods change in a similar manner according to different M_{max} and *n* values (see Figure 17). As the values increase, CPU times of both methods also increase. However, regardless of M_{max} and *n* values, the computing time required for model building in S-FMARS is drastically less than that of FMARS, which is figured out by different y-scales in Figure 17. At least 70 % decrease is achieved by S-FMARS method.

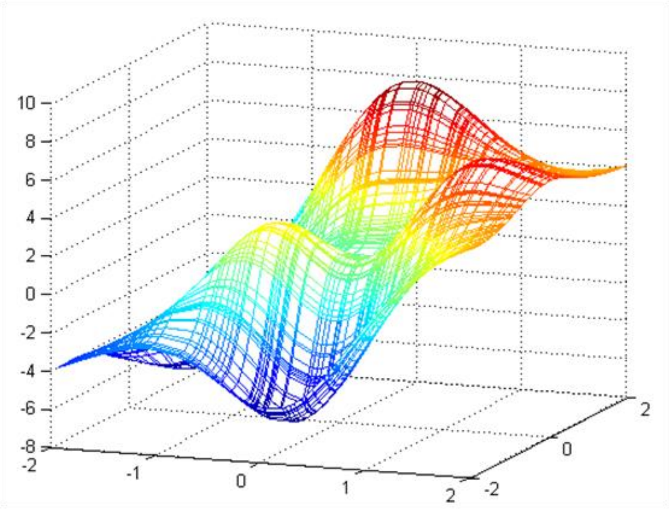


Figure 15. Grid plot for data set 8.

Table 7. Performances of FMARS and S-FMARS for different *n* and M_{max}.

n	M_{max}	RMSE		Adj-R^2		GCV		CPU Time (sec.)		Difference in time (%)
		F MARS	S-FMARS	F MARS	S-FMARS	F MARS	S-FMARS	F MARS	S-FMARS	
400	20	1.207*	1.229	0.850	0.852*	1.914*	1.985	3.7	1.1*	70
	40	0.994*	1.057	0.893*	0.891	1.767*	2.001	13.8	2.3*	83
	60	0.911*	0.935	0.905	0.915*	2.139*	2.254	35.7	4.5*	87
800	20	1.302	1.293*	0.829	0.835*	1.934	1.908*	8.3	1.3*	84
	40	1.053	1.046*	0.885	0.892*	1.453	1.433*	40.3	3.4*	92
	60	0.993	0.983*	0.895	0.905*	1.500	1.469*	107.9	8.5*	92
1600	20	1.268*	1.285	0.846*	0.843	1.715*	1.762	24.0	1.7*	93
	40	1.057*	1.072	0.891	0.891	1.272*	1.308	109.7	5.6*	95
	60	1.006*	1.011	0.900	0.903*	1.233*	1.247	308.7	14*	95
3200	20	1.227*	1.233	0.855	0.855	1.555*	1.570	72.5	3.5*	95
	40	1.028	1.027*	0.898	0.899*	1.126	1.124*	357.4	14.5*	96
	60	0.985	0.987	0.906	0.907*	1.069*	1.073	1088.6	35.6*	97
6400	20	1.274	1.266*	0.844	0.846*	1.650	1.628*	339.1	7.7*	98
	40	1.060*	1.072	0.892*	0.890	1.160*	1.187	1609.1	37.5*	98
	60	1.013*	1.014	0.901	0.901	1.077*	1.079	6228.6	114.8*	98

Note: * indicates better performance.

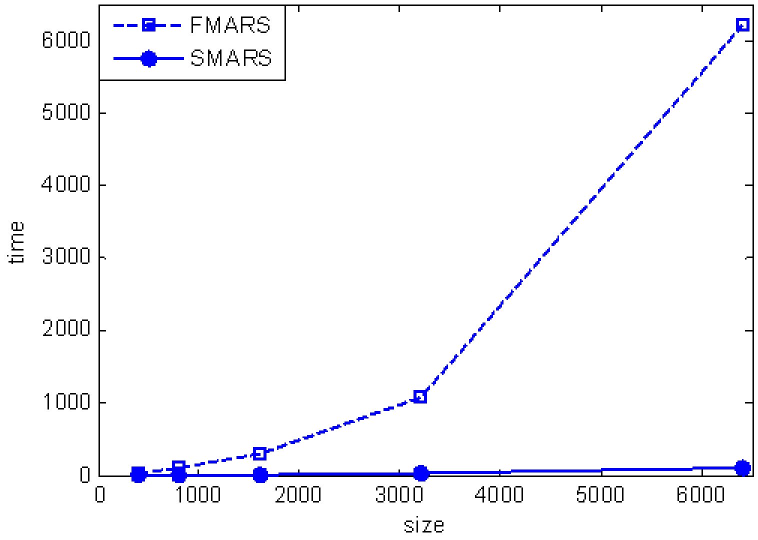


Figure 16. CPU times versus sample sizes (n) for M_{max}=60.

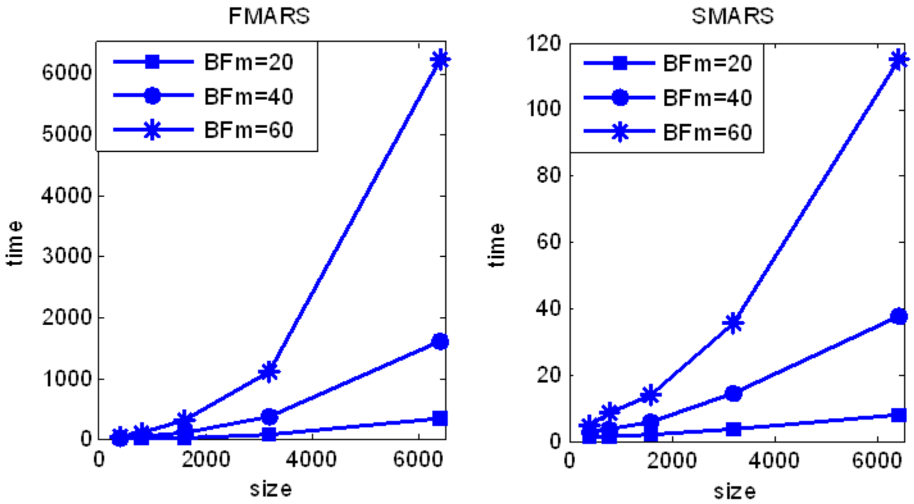


Figure 17. CPU time versus sample size.

Note: BFm represents the number of BFs in the final model built for a preset M_{max} value.

4.3.1.2 *Effect of interaction terms on the CPU time*

In general, interaction models require more CPU time than additive models (Friedman. 1993). The effect of interation terms on the computing time of FMARS and S-FMARS is tested on Dataset 7, which is a robot arm example used by Friedman (1993). This data is taken from a hypothetical robot arm free to move in three dimensions (x, y, z). It includes five input variables which are taken to be the lengths of upper and forearm l_1, l_2, respectively, and three angles θ_1, θ_2, ϕ. The response is the distance from the origin (J_1) to the end of the forearm (x, y, z) opposite to the joint (J_2), the location of which is given by

$$\begin{aligned} x &= l_1 \cos\theta_1 - l_2 \cos(\theta_1 + \theta_2)\cos\phi, \\ y &= l_1 \sin\theta_1 - l_2 \sin(\theta_1 + \theta_2)\cos\phi, \\ z &= l_2 \sin\theta_2 \sin\phi. \end{aligned} \tag{35}$$

The distance (response) is then obtained by

$$d = (x^2 + y^2 + z^2)^{1/2}. \tag{36}$$

The best model describing the nonlinear relationship between the response and predictor variables of robot arm data is an interaction model. The CPU times spend for building models with different degree of interaction terms are observed and compared in Table 8, and summarized in Figure 18. As a result, as the number of interaction term increases, the CPU times of both methods increases correspondingly, but S-FMARS is much more efficient than FMARS for all cases. It provides approximately the same percent of decrease in CPU times for all number of interaction terms, which is 95%. Moreover, the performances of S-FMARS and FMARS are not statistically different with respect to accuracy and complexity for different number of interaction terms.

Table 8. Performances of FMARS and S-FMARS for different interaction term

#	RMSE		Adj-R^2		GCV		CPU Time (sec.)		Decrease
int.	FMARS	S-FMARS	FMARS	S-FMARS	FMARS	S-FMARS	FMARS	S-FMARS	in CPU time (%)
1	0.160	0.157*	0.761	0.771*	0.024*	0.025	159.1	5.5*	97
2	0.082	0.079*	0.999	0.999	0.006	0.006	814.9	39.3*	95
3	0.064*	0.065	0.999	0.999	0.004	0.004	1323.4	61.*6	95
4	0.063	0.062*	0.999	0.999	0.004	0.003	1448.7	73.1*	95
5	0.063	0.062*	0.999	0.999	0.004	0.003	1670.9	70.5*	96

Note: * indicates better performance.

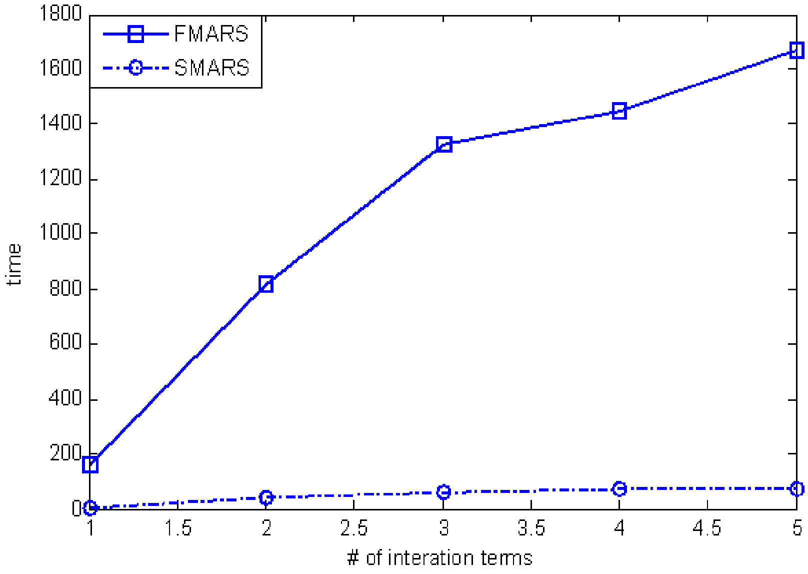


Figure 18. CPU time versus number of interaction terms.

4.3.2 Real Life Data

Two methods are also compared on real life datasets presented in Table 1. Before the model construction, all datasets are preprocessed and standardized to increase the model performances and make them comparable. 3-fold and three times replicated cross validation approach is used to validate the performance of the methods. The averages of nine performance measures for the corresponding models are listed in Table 9. Algorithms are run for different M_{max} values presented in the 3th column of Table 9. The best models obtained for *Parkinsons Telemonitoring*, *Communities and Crime*, *AutoMp*g and *PM10* are additive models (with no interaction), whereas for *Red Wine Quality* and *Concrete Compressive Strength* datasets, respectively, two and three way interaction models are found to be the best for both methods.

In order to decide whether or not both methods are statistically different, one-sample sign test is applied to the performance measures calculated for train and test data sets displayed in Table 9 and Table 10, respectively. The significance of differences between the stabilities of the measures calculated on train and test performance measures given in Table 10 is also checked statistically with one sample-sign test. Moreover, to evaluate the overall performances of methods, mean and standard deviation of all accuracy and complexity measures are given in Table 11 for train and test datasets as well as stabilities of measures. Here, standard deviation is used for indicating the robustness of the methods.

Table 9. Average performances of FMARS and S-FMARS on the train data.

Datasets	Models	M_{max}	Int.	RMSE	Adj-R^2	GCV	CPU Time (sec.)	Decrease in CPU time (%)
Parkinson	FMARS	50	-	0.348	0.863	0.195	29.51	90
	S-FMARS			0.347*	0.864*	0.194*	3.09*	
Red Wine	FMARS	90	2	0.619*	0.569*	0.603*	338.91	79
	S-FMARS			0.672	0.513	0.665	72. 68*	
Com. Crime	FMARS	150	-	0.365*	0.817*	0.580*	216.10	71
	S-FMARS			0.412	0.766	0.800	62.66*	
Conc.Comp.	FMARS	100	3	0.196*	0.955*	0.102	1122.53	85
	S-FMARS			0.202	0.952	0.103*	168.51*	
AutoMpg	FMARS	100	-	1.847*	0.994*	64.66*	16.75	80
	S-FMARS			2.184	0.878	90.42	3.32*	
PM10	FMARS	50	-	0.649*	0.503*	0.867*	11.11	81
	S-FMARS			0.665	0.477	0.911	2.16*	

Note: * indicates better performance.

Table 10. Average performances of FMARS and S-FMARS on test data and stabilities.

Datasets	TEST				STABILITY			
	RMSE		Adj-R^2		RMSE		Adj-R^2	
	FMARS	S-FMARS	FMARS	S-FMARS	FMARS	S-FMARS	FMARS	S-FMARS
Parkinson	0.640*	0.685	0.716*	0.683	0.544*	0.507	0.830*	0.791
Red Wine	1.237	0.937*	0.213	0.251*	0.500	0.717*	0.374	0.490*
Com. Crime	0.739	0.681*	0.520	0.569*	0.494	0.605*	0.637	0.743*
Conc.Comp.	0.443*	0.459	0.823*	0.816	0.442*	0.440	0.862*	0.857
Auto Mpg	5.199	2.934*	0.720	0.845*	0.355	0.744*	0.724	0.962*
PM10	1.013	0.808*	0.262	0.369*	0.641	0.823*	0.521	0.774*

Note: * indicates better performance.

Table 11. Overall performances of FMARS and S-FMARS methods.

Methods	TRAIN			TEST		STABILITY	
	RMSE	Adj-R^2	GCV	RMSE	Adj-R^2	RMSE	Adj-R^2
FMARS	0.671* (0.602**)	0.784* (0.203)	11.168* (26.207**)	1.545 (1.812)	0.542 (0.256)	0.496 (0.096**)	0.658 (0.187)
S-FMARS	0.747 (0.728)	0.742 (0.200*)	15.515 (36.697)	1.084* (0.920**)	0.589 * (0.240**)	0.639* (0.148)	0.770* (0.157**)

Notes: * Indicates better performance with respect to mean. ** Indicate better performance with respect to standard deviation in parenthesis.

Depending on the results presented in Table 9, Table 10 and Table 11, the following conclusions can be drawn:

- FMARS produces slightly accurate and less complex models than S-FMARS except Parkinson data. However, according to one-sample sign test, the accuracy of S-FMARS model is not statistically different than that of FMARS considering all performance measures.
- On test data, S-FMARS performs better than FMARS for Red Wine Quality, Communities and Crime, Auto Mpg and PM10 datasets with respect to

RMSE and Adj-R^2. For the same datasets, S-FMARS is more stable than FMARS (see Table 10).

- With respect to overall performance, FMARS models are more accurate and robust than that of S-FMARS on training data. On test data, however, S-CMARS performs better with respect to accuracy and is more robust than FMARS. S-FMARS is more stable with respect to all of the measures (see Table 11).
- S-FMARS is more efficient than FMARS for all data sets. S-FMARS decreases the CPU time at least 71%, which is observed for Communities and Crime.

As mentioned in Section 4.3.1.1, CPU times of methods are affected by sample size and number of predictor variables which gives the scale of data. To observe the effects of sample size and scales on CPU time, the datasets are classified according to these two important features given in Table 12. The levels assigned to scale feature are *small* and *large*. Data with less than or equal to 10 predictor variables is assigned as small, otherwise it is assigned as large. On the other hand, datasets classified into two as medium and large with respect to sample size. Small data has a sample size less than or equal to 600, while large data has more than 600 instances. The average CPU times of methods observed for each level of sample size and scale are given in Table 12. To evaluate the significance of differences between average CPU times of methods obtained for two types of data classified with respect to scale and sample size are tested by using a nonparametric test called Mann-Whitney test. This test is a nonparametric version of two-sample t test used for independent samples where normality assumption is violated.

Depending on the results presented in Table 12, the following conclusions can be drawn:

Table 12. Average CPU times of methods for different sample size and scale

Features of Data		Methods	
		FMARS	S-FMARS
Sample Size	Medium	19.1	2.9
	Large	559.2	101.3
	Percent Difference (%)	97	97
Scale	Small	383.5	58.0
	Large	194.8	46.1
	Percent Difference (%)	49.2	20.4

- Two methods are more efficient for medium-sized datasets than large-sized ones. The difference between the average CPU times of methods obtained for medium-sized and large-sized datasets are found significant by Mann-Whitney test (*p*-values=0.0051). Additionally, S-FMARS performs better than FMARS method on both medium and large datasets. S-FMARS reduces the CPU times by 97 % from medium-sized data to large-sized ones.
- It is interesting that the effect of scale on CPU time seems quite the opposite of sample size. When the number of predictor variable is increases, the CPU time decreases. This may be due to the fact that there is an interaction effect between sample size and scales. Nevertheless, CPU times of S-FMARS method are less than that of FMARS for both small-scaled and large-scaled datasets. In addition, the difference between CPU times between two types of data are not statistically significant according to Mann-Whitney test (*p*-values=0.4712).

Due to the significant three-way interaction effects including sample size, scale and methods, a typical pattern for the CPU times of the methods is hard to detect. Nevertheless, interaction plots in Figure 19 can be helpful for determining the best size-scale combination for a method in relation with CPU time. To exemplify, with respect to CPU time, two methods are more efficient on medium-sized training samples regardless of scale. However, for large sample sizes, the largest CPU times are observed for small-scaled datasets for both methods.

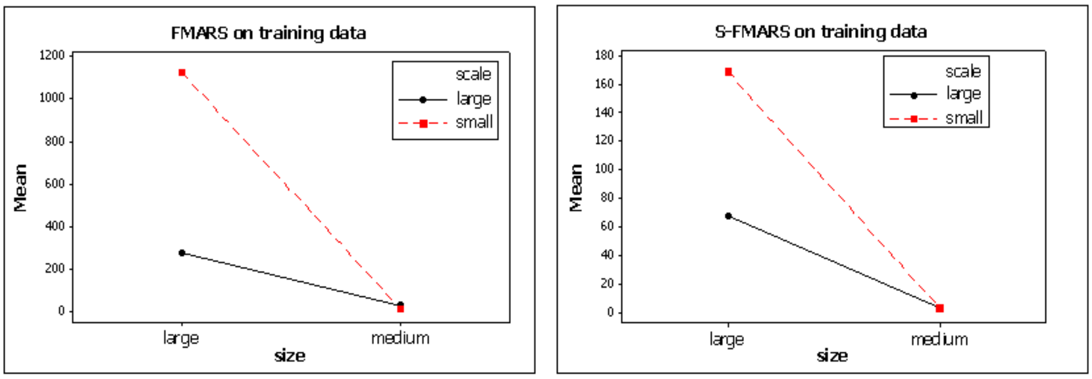


Figure 19.Interaction plots of size and scale for the CPU times for FMARS and S-FMARS methods.

4.3.3. Performance on Noisy Data

In this section, to see the effect of noise on performance of both methods, two simulation studies are carried out.

4.3.3.1 Noisy Data 1 (Dataset 9)

Using the *sinus* function two data sets are generated with and without noise with 100 observations (see Figure 20). Two methods are fitted to them and then the accuracy and complexity measures are calculated (see Table 13). The performance measures of models obtained for noise-free data are given in the first row of Table 13. The other rows are related with noise data, and the measures correspond to the fits obtained for different M_{max} values. The main reason of analyzing the performance of methods for different number of $\mathrm{M}_{\mathrm{max}}$ values on noisy data is to observe the sensitivity of methods against noise. Moreover, to measure the sensitivity of the model fits to noisy data, noise-free data is used as a test data, and the performance measures are calculated using the fitted values obtained for the noisy data and the noise-free data points (Table 13).

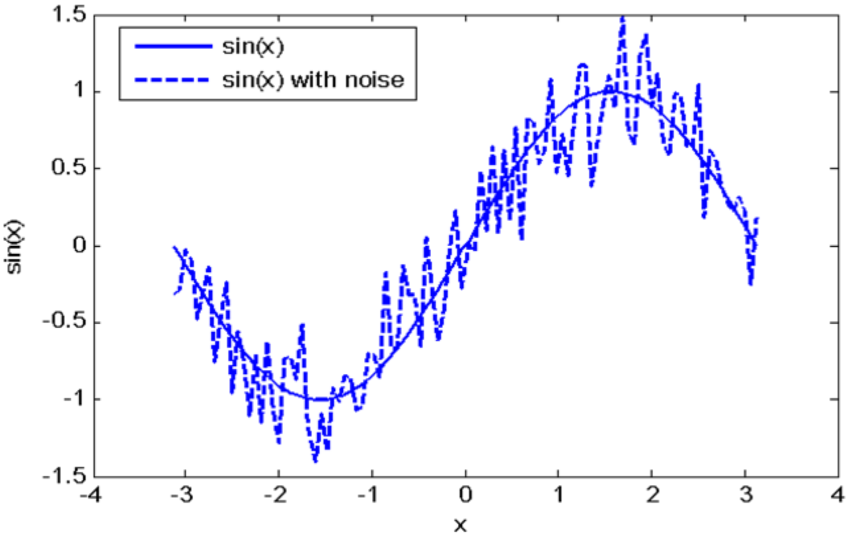


Figure 20. Sinus function with and without noise.

Table 13. Performances of FMARS and S-FMARS on noisy data 1.

	M_{max}	BF_{final}		Datasets	RMSE		Adj-R^2	
		FMARS	S-FMARS		FMARS	S-FMARS	FMARS	S-FMARS
Noise-free data	≥20	19	19	Raw	0.012*	0.014	0.999	0.999
Noisy data	20	20	19	Train	0.228*	0.243	0.881*	0.866
				Test	0.122	0.078	0.962	0.985
	30	30	19	Train	0.216*	0.243	0.877*	0.866
				Test	0.139	0.078	0.944	0.985
	40	40	19	Train	0.201*	0.243	0.877*	0.866
				Test	0.161	0.078	0.915	0.985
	60	60	19	Train	0.145*	0.243	0.867*	0.866
				Test	0.212	0.078	0.782	0.985

Note: * indicates better performance.

According to the results displayed in Tables 13, the following conclusions can be drawn:

- For noise-free data, both methods use 19 BFs in the final model although M_{max} are set to 30. The accuracy and complexity measures of both methods are very close to each other (Table 13, the first row).
- In noisy data, S-FMARS builds its best models with 19 BFs for all M_{max} settings. However, FMARS builds more complex models as M_{max} value increase to rise up the model accuracy. This shows that FMARS is more sensitive to the noise than S-FMARS and tries to model the noise.
- Table 13 exposes that the fits obtained by FMARS is more sensitive to noise than S-FMARS. Although the model fits obtained by FMARS is more accurate on noisy data, and gets better as the *Mmax* value increases, its performance on noise-free data gets worse as the model become complex. S-FMARS can provide a less sensitive model to noise by building a less complex model for noise data. The sensitivity of both fits on noisy data is illustrated in Figure 21. As it is seen that while FMARS prone to model the noise for the predictor values, especially for the interval [-2, 1] in x-axis, S-FMARS tries to fit a noise-free data.

4.3.3.2 Noisy Data 2 (Dataset 10)

In this analysis, another noisy data is created using the following function:

$$f(\mathbf{x}) = 0.5x_1^2 + x_2^2 - x_1x_2 - 7x_1 - 7x_2 \, , \tag{37}$$

where x_1 and x_2 are assumed to have Uniform (-10, 10) distribution. This data refers to noisy-free data. To obtain noisy data, normally distributed noise ε having zero mean is added to the response, $f(\mathbf{x})$. Here, the variance of the noise is assumed to be

1/100 of the variance of $f(\mathbf{x})$ in (37). When -5 ≤ $x_1, x_2 \leq 5$, however, the variance of the noise is assumed to be 1254.9/100=12.55 (Jin et al., 2001).

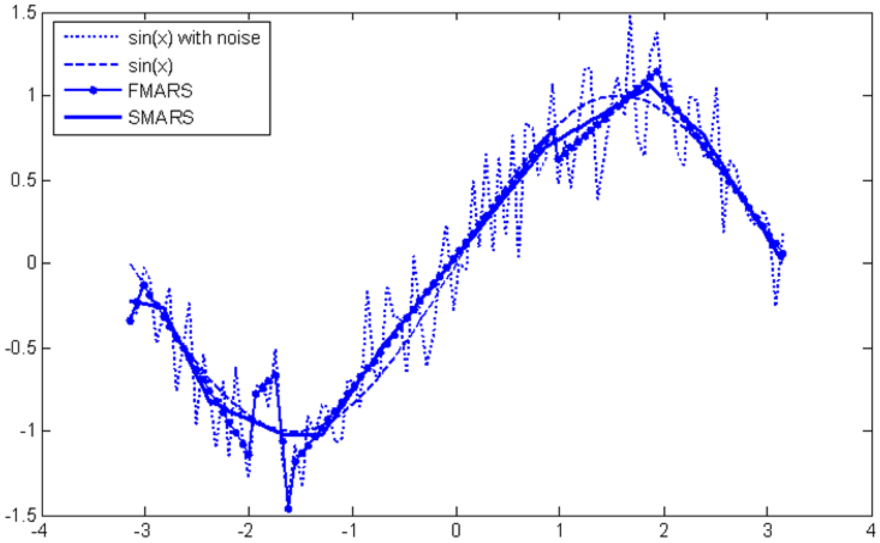


Figure 21.. Fitted models for sinus function with noise.

Two training data sets are created as described above with and without noise. Similar to the analysis mentioned in the previous section, two methods are applied to noise free data with 10 maximum numbers of BFs (M_{max}), and with various M_{max} values to noisy data. Again, the reason of building FMARS and S-FMARS models with various M_{max} values is to observe the sensitivity of methods to noise. In addition, a test data is generated using the function in (37) and the same measures are calculated. The results are given in Table 14. To observe the sensitivity of model fits, again the measures are recalculated for training and test data sets using the fitted values obtained for noisy data and noise-free data points instead of noisy ones.

Table 14. Performances of FMARS and S-FMARS on noisy data 2.

Data	Data sets	M_{max}	BF_{final}		RMSE		Adj-R^2	
			FMARS	S-FMARS	FMARS	S-FMARS	FMARS	S-FMARS
Noise-free data	Train	10	7	9	0.69*	0.75	0.999	0.999
	Test	10	7	9	1.06*	1.08	0.999	0.999
Noisy data	Train	9	9	9	7.27	7.22*	0.992	0.992
	Test	9	9	9	8.63	8.03*	0.990	0.990
	Train	10	10	10	7.14	7.07*	0.992	0.992
	Test	10	10	10	9.88	8.70*	0.987	0.990*
	Train	20	20	10	6.33*	7.07	0.993*	0.992
	Test	20	20	10	11.82	8.70*	0.976	0.990*
	Train	30	30	10	5.61*	7.07	0.993 *	0.992
	Test	30	30	10	16.92	8.70*	0.947	0.990*
	Train	60	55	10	3.93*	7.07	0.995 *	0.992
	Test	60	55	10	30.39	8.70*	0.795	0.990*
Noisy fit vs noise-free data	Train	9	9	9	2.67*	2.58	0.999	0.999
	Test	9	9	9	3.93	2.29*	0.999	0.999
	Train	10	10	10	2.72*	2.84	0.999	0.999
	Test	10	10	10	6.31	3.66*	0.994	0.999*
	Train	20	20	10	4.22	2.84*	0.997	0.999*
	Test	20	20	10	8.89	3.66*	0.986	0.999*
	Train	30	30	10	5.17	2.84*	0.994	0.999*
	Test	30	30	10	15.05	3.66*	0.957	0.999*
	Train	60	55	10	6.55	2.84*	0.987	0.999*
	Test	60	55	10	29.41	3.66*	0.811	0.999*

Note: * indicates better performance.

Based on the result of analysis mentioned above, the following conclusions can be stated:

- FMARS performs slightly better on noise-free data, and its model is less complex than S-FMARS.
- On noisy data, when the M_{max} value is set to large values (M_{max}>10), FMARS performs better than S-FMARS for training data by building more complex

models (models with large number of BFs). However its prediction performance gets worse as the M_{max} value increase.

- On training data with noise, S-FMARS performs better for small M_{max} values. But, the accuracy of models gets worse as the M_{max} increase.
- For all cases, S-FMARS overperfoms FMARS on test data.
- S-FMARS provides a closer fit to noise-free data by the fits obtained for noisy data. Hence, S-FMARS fits is less sensitive to noise than FMARS.

4.4. Comparison Study 2

The CPU of MARS is affected from various parameters such as predefined maximum number of BFs (M_{max}), stopping criteria defined for the difference between two consecutive LOF in (26), degree of interactions and the number of candidate knot points. This paper proposes a new approach to decrease the computational complexity of MARS by restricting the candidate knot points to a small subset of data points by a mapping approach. In the literature, some other knot restriction algorithms are proposed not mainly to decrease the computing time, but to decrease the local variability. However, these approaches still provide less computing time than MARS algorithm. Use of equally-spaced knot locations, use of predefined knot locations or setting an interval value or minimum value for the number of data points between two adjacent knots in the ascending order of predictor-axis (MinSpan) are some of these approaches. Since the MinSpan method described in Section 2.1.1 is a data-adaptive approach and more effective than the other methods, S-FMARS is compared with the MinSpan approach to demonstrate their computing efficiency and model accuracy. In this section, model performances and efficiencies of S-FMARS and MinSpan approach is compared via the performance measures mentioned in Chapter 4.1.3.

4.4.1. Artificial Datasets

For each approach, the same number of interaction terms (*Int.*) is used in model building, and the models are allowed to grow up to the same preset number of BFs (M_{max}) which is 100 for all cases. The accuracy measures of models calculated for training data and the corresponding CPU time required for modeling of each training dataset are given in Table 15, as well as the number of BFs in the final model ($\mathrm{BF}_{\mathrm{final}}$).

As mentioned in the comparison study of FMARS and S-FMARS, the number of BF in the final model denotes the complexity of the model. As a result of this analysis, again two models have similar complexity for almost all models. The accuracy measures calculated for each method seem close to each other in Table 15. For five data sets (one, four, five, six and eight), S-FMARS performs better than FMARS with respect to RMSE. For four data sets, S-FMARS overperforms FMARS with respect to complexity measure (GCV). The prediction performances of both methods and their stabilities of measures are compared via RMSE and Adj-R^2 measures in Table 16. MinSpan performs better in Datasets two and three in terms of RMSE. For the other problems, however, prediction capability of models obtained with S-FMARS approach is higher than MinSpan. Additionally, S-FMARS produces slightly stable models than does the MinSpan for datasets one, two and eight in terms of RMSE.

The models are compared with respect to time efficiency via CPU times given in the last column of Table 15. Although MinSpan decreases the CPU time of MARS algorithm significantly, S-FMARS is still more efficient than MinSpan for all datasets. As it is seen in Table 15, the most significant decreases are observed for datasets six and eight, which are 90 %, while the least one is observed for dataset three as 17%. Moreover, the models of S-FMARS models can compete with the models of MinSpan with respect to accuracy.

Table 15. Performances of MinSpan and S-FMARS on the train data.

Datasets	Methods	Int.	BFm	RMSE	Adj-R^2	GCV	CPU Time (sec.)	Decrease in CPU time (%)
1	MinSpan	4	100	1173.1	0.975	2453059.2	976.7	68
	S-FMARS		100	1089.1*	0.976*	2114441.0*	311.8*	
2	MinSpan	1	45	4244.4*	0.999	21707171.5*	9.8	23
	S-FMARS		43	4359.8	0.999	22703916.0	7.5*	
3	MinSpan	2	47	23.5*	0.999	706.7*	278.5	17
	S-FMARS		47	24.0	0.999	740.1	231.1*	
4	MinSpan	2	77	0.026	0.998	0.001	2509.0	60
	S-FMARS		81	0.025*	0.998	0.001	999.7*	
5	MinSpan	2	43	515.2	0.999	271143.1	1308.7	61
	S-FMARS		45	514.2*	0.999	270354.0*	516.8*	
6	MinSpan	2	23	2.969	1.000	8.912	487.7	90
	S-FMARS		23	2.892*	0.999	8.456*	51.0*	
7	MinSpan	4	100	0.030	0.992*	0.002	3311.4	94
	S-FMARS		100	0.030	0.991	0.002	186.7*	
8	MinSpan	2	83	0.154	0.998	0.025	4784.9	90
	S-FMARS		78	0.151*	0.998	0.024*	484.7*	

Table 16. Performances of MinSpan and S-FMARS on the test data and stability.

	Performance on TEST dataset				STABILITY			
	RMSE		Adj-R^2		RMSE		Adj-R^2	
Datasets	MinSpan	S-FMARS	MinSpan	S-FMARS	MinSpan	S-FMARS	MinSpan	S-FMARS
1	1258.4	1126.7*	0.959	0.959	0.932	0.967*	0.984*	0.983
2	3489.5*	3834.1	0.999	0.999	0.822	0.879*	1	1
3	22.750*	23.1	0.999	0.999	0.968*	0.963	1	1
4	0.026	0.026	0.998	0.998	1.000*	0.962	1	1
5	539.2	529.7*	0.999	0.999	0.973*	0.971	1	1
6	2.980	2.869*	0.999	0.999	0.996*	0.992	1	1
7	0.028	0.028	0.986	0.986	0.933	0.933	0.994	0.995*
8	0.154	0.151*	0.998	0.998	0.999	1.000*	1	1

Note: * indicates better performance.

4.4.2. Real Datasets

Two methods are also compared on real life datasets presented in Table 17. Again, 3-fold and three times replicated cross validation approach is used to validate the performance of the methods and the averages of the corresponding nine performance measures are listed for each method in Table 17. Algorithms are run for different M_{max} values presented in the 3th column of Table 17. As it is seen that both methods use the same number of BFs in the final model which is equal to M_{max}.

In order to test whether or not the accuracy and prediction performances of two methods are statistically different, one-sample sign test is applied to measures obtained for train and test data sets in addition to the stabilities of the measures. In order to evaluate the overall performances of methods, mean and standard deviation of all accuracy and complexity measures are given in Table 19 for train and test datasets as well as stabilities of measures. Here, standard deviation is used for indicating the robustness of the methods.

Table 17. Average performances of MinSpan and S-FMARS on the train data.

Datasets	Methods	Mmax	RMSE	Adj-R^2	GCV	CPU Time (sec.)	Decrease in CPU time (%)
AutoMpg	MinSpan	100	2.022*	0.935*	77.38*	4.9	33
	S-FMARS		2.184	0.878	90.42	3.3*	
Com.Crime	MinSpan	150	0.392*	0.842*	0.673*	133.6	53
	S-FMARS		0.412	0.766	0.800	62.7*	
Conc.Comp.	MinSpan	100	0.203	0.958*	0.101	303.6	44
	S-FMARS		0.202*	0.952	0.100*	168.5*	
Parkinsons	MinSpan	50	0.325*	0.900*	0.190*	11.5	73
	S-FMARS		0.347	0.864	0.194	3.1*	
PM10	MinSpan	50	0.653*	0.581*	0.856*	3.1	29
	S-FMARS		0.665	0.477	0.911	2.2*	
Redwine	MinSpan	90	0.637*	0.596*	0.603*	338.9	79
	S-FMARS		0.672	0.513	0.665	72. 7*	

Note: * indicates better performance.

Table 18. Average performances of MinSpan and S-FMARS on test data and stability.

Datasets	TEST				STABILITY			
	RMSE		Adj-R^2		RMSE		Adj-R^2	
	MinSpan	S-FMARS	MinSpan	S-FMARS	MinSpan	S-FMARS	MinSpan	S-FMARS
Auto Mpg	2.642	2.934*	0.869*	0.845	0.765*	0.744	0.929	0.962*
Com. Crime	0.710	0.681*	0.563	0.569*	0.552	0.731*	0.669	0.743*
Conc.Comp.	0.500	0.459*	0.803	0.816*	0.406	0.440*	0.838	0.857*
Parkinson	0.470	0.685	0.746*	0.683	0.692*	0.507	0.829*	0.791
PM10	0.893	0.808*	0.286	0.369*	0.731	0.823*	0.493	0.774*
Red Wine	0.940	0.937*	0.245	0.251*	0.677	0.717*	0.411	0.489*

Note: * indicates better performance.

Table 19. Overall performances of MinSpan and S-FMARS methods.

Methods	TRAIN			TEST		STABILITY	
	RMSE	Adj-R^2	GCV	RMSE	Adj-R^2	RMSE	Adj-R^2
MinSpan	0.705* (0.669**)	0.802 (0.170**)	13.301* (31.394**)	1.026* (0.815**)	0.585* (0.268)	0.637 (0.134**)	0.695 (0.208)
S-FMARS	0.747 (0.728)	0.742* (0.200)	15.515 (36.697)	1.084 (0.920)	0.589 (0.240**)	0.660* (0.151)	0.769* (0.158**)

Notes: * Indicates better performance with respect to mean. ** Indicate better performance with respect to standard deviation.

Depending on the results presented in Table 17, Table 18 and Table 19, the following conclusions can be drawn:

- MinSpan performs better than S-FMARS for all data sets except the Concrete Compress data with respect to accuracy and complexity measures. However, according to one-sample sign test, the accuracy of S-FMARS model is not statistically different than that of FMARS considering all performance measures (p-value>0.05).

- On new observations, S-FMARS performs better than MinSpan for all data sets except Parkinsons with respect to both RMSE and Adj-R^2.
- The prediction performance on the test data and stability of the models results for both methods are displayed in Table 18. The performance of models obtained by S-FMARS is slightly better than the performance of models produced by MinSpan with respect to prediction capability and stability for all datasets except Parkinsons data.
- In most of the data sets, S-FMARS is more stable than MinSpan with respect to all measures.
- The overall accuracy and prediction performances of MinSpan models are better than S-FMARS models.
- In the overall, S-FMARS is more stable than MinSpan with respect to all of the measures.
- The differences between the CPU times of S-FMARS and Minspan are not as much as the differences between S-FMARS and FMARS. However, S-FMARS is again more efficient than MinSpan for all data sets. As it is seen in Table 16, the most significant decrease is observed for Red Wine, which is 79 %, while the least decrease is observed for PM10 as 29 %.

To observe the effects of sample size and scales on CPU time, the average CPU times of methods obtained for medium/large-sized and small/large-scaled datasets are given in Table 20. Additionally, the significance of differences between the average CPU times of methods obtained for two types of data classified with respect to scale and sample size are tested by using Mann-Whitney test. Depending on the results presented in Table 20, the following conclusions can be drawn:

- Two methods are more efficient for medium-sized datasets than large-sized ones. The difference between the average CPU times of methods obtained for medium-sized and large-sized datasets are found significant by Mann-Whitney test (p=0.0051). Additionally, S-FMARS performs better than

FMARS method on both medium and large datasets. However, both methods provide approximately the same percent decrease in CPU times from medium-sized data to large-sized ones, which are more than 95%.

- MinSpan is more efficient on small-scaled data than large-scaled ones, while it is quite opposite for S-FMARS approach. When the number of predictor variable is increases, the CPU time of S-FMARS decreases. This may indicates the existence of is an interaction effect between sample size and scales. Nonetheless, CPU times of S-FMARS method are less than that of Minspan for both small-scaled and large-scaled datasets. In addition, the difference between CPU times between two types of data are not statistically significant according to Mann-Whitney test (p=0.4233).

Table 20. Average CPU times of methods for different sample size and scale

Features of Data		Methods	
		MinSpan	F-SMARS
Sample Size	Medium	6.5	2.9
	Large	258.7	101.3
	Percent Difference (%)	98	97
Scale	Small	103.9	58.0
	Large	161.3	46.2
	Percent Difference (%)	35.6	20.4

In order to determine the best size-scale combination for two methods in relation with CPU time, three-way interactions effects including sample size, scale and methods are displayed in Figure 22. As it is seen in Figure 22, both methods are more efficient on medium-sized training samples regardless of scale with respect to CPU time. However, for large sample sizes, the largest CPU times are observed for small-scaled datasets for both methods. Moreover, the effect of scale seems more significant on large-sized datasets for S-FMARS method.

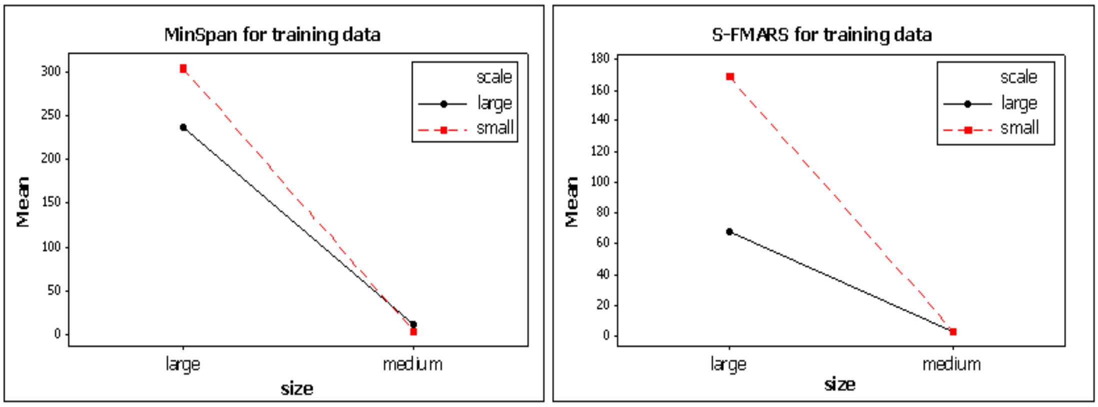


Figure 22. Interaction plots of size and scale for the CPU times for MinSpan and S-FMARS methods.

4.5. Comparison Study 3

As in MARS algorithm, many adaptive regression splines include a backward elimination step to determine the optimum number of terms in the final model so that the overfitting problem caused by the forward step can be prevented. The same strategy can also be implemented to S-FMARS. Since S-FMARS method is consist of forward selection strategy, a model deliberately overfitting the underlying function with a large number of BFs is obtained. On the other hand, S-FMARS is a time efficient forward selection method by which a multivariate regression spline models can be obtained in less time. When it is compared with the forward step of MARS algorithm with and without MinSpan approach, it is observed that S-FMARS method is much more efficient in time than the other methods. Therefore, it can be offered as an alternative to the forward step of adaptive regression splines in which data points are searched for the proper knot locations.

In addition to MARS algorithm, S-FMARS can also be implemented to CMARS algorithm to make it efficient in time. CMARS is a modified version of MARS algorithm in which instead of backward elimination step, a penalized residual sum of squares is used, and solved with CQP. However, they are based on the same forward selection strategy and use the same BFs for the second part. That is, CMARS applies its penalization strategy to the BFs obtained from the forward part of MARS. As stated in the study of Weber et al (2012), the only drawback of CMARS method is its high computational run times. In that study, performance of CMARS is compared with MARS algorithm and stated that CMARS is not as efficient MARS. Since CMARS decrease the complexity of the model by applying a penalized residual sum of squares and solves it by CQP, the method becomes computationally expensive. For this reason, S-FMARS which is proposed as a revised version of forward step of MARS algorithm can be implemented to improve the CMARS algorithm to reduce its computational run time.

In this study, S-FMARS method is applied to MARS and CMARS algorithm to improve them by reducing their computational run times. Implementation of S-FMARS to MARS algorithm is straight forward. The backward strategy of MARS algorithm is applied to the models obtained by S-FMARS (The algorithm of S-FMARS is given in Chapter 4.3). The resulting method is called SMARS.

The S-FMARS algorithm is combined with CMARS methods with the following steps given in Figure 23. The new method is called S-CMARS. The performances of the S-CMARS, SMARS are evaluated and compared with MARS, MARS with MinSpan approach and CMARS in the following subsections.

Step 1: A S-FMARS method is constructed and the best parameter values of S-FMARS methods are determined for the underlying dataset.

Step 2: The set of BFs are obtained by applying S-FMARS method with best parameter values obtained in Step 1.

Step 3: A CMARS Model is constructed for the BFs obtained in Step 2, and the optimal value of bound $\sqrt{Z}$ in (18) is found. To achieve this, the curve of $\sqrt{RSS}$ versus norm of $L\theta$ in the log-log scale is obtained (see Figure 24). The optimal value of this curve is the corner point which is demonstrated by a red point. The selected value gives the best solution for both accuracy and complexity.

Step 4: CMARS is rerun for the optimal value of $\sqrt{Z}$, and the final model is obtained.

Figure 23. Algorithm of S-CMARS method.

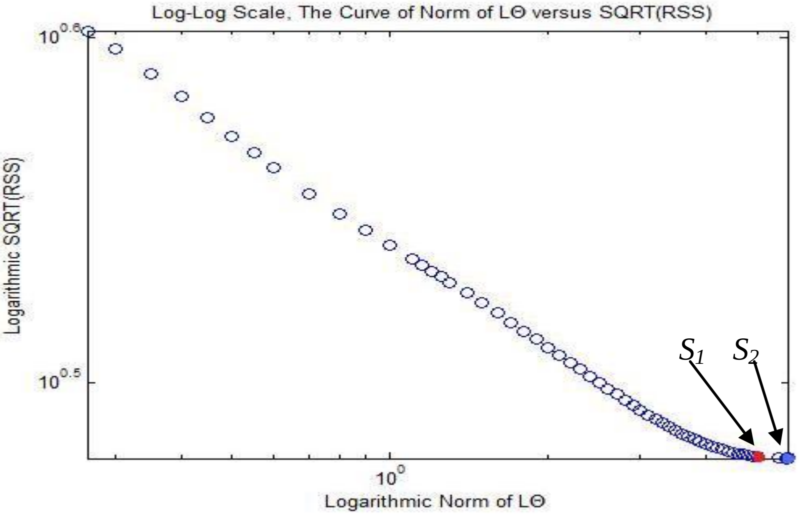


Figure 24. The plot of norm $L\theta$ versus $\sqrt{RSS}$.

4.5.1. Artificial Datasets

Five adaptive regression methods are evaluated and compared for all artificial datasets with respect to RMSE, Adj-R^2, GCV and CPU time calculated for training and test data sets. The stabilities of the measures are also calculated. The maximum number of BFs is set to 100 for all datasets. The same number of interaction terms is used for all datasets mentioned. The measures obtained for all artificial datasets are given in Table 21. The order of the measures calculated for each dataset is different for some datasets. This is because of the fact that the mean of datasets are in different orders. Because of this reason, the methods cannot be compared on the average. The performance of methods are evaluated and compared separately for each datasets. Depending on the results presented in Table 21 and Table 22, the following conclusions can be stated:

Table 21. Average performances of methods on train data.

Datasets	Methods	BF_{final}	RMSE	Adj-R^2	GCV	CPU Time (sec.)
1	MARS	69	1103.8*	0.976*	1772987.9*	6022.21
	MinSpan	76	1196.5	0.972	2173914.8	2000.39
	SMARS	70	1135.5	0.975	1887498.7	317.86*
	CMARS	101	5928.1	0.293	2013650.0	6320.33
	S-CMARS	101	5617.7	0.429	2114441.3	424.55
2	MARS	27	4271.9	0.999*	20348729.4	75.04
	MinSpan	27	4248.1*	0.999*	20122417.8*	14.33
	SMARS	24	5975.6	0.998	39317130.7	5.54*
	CMARS	43	4271.8	0.999*	21796916.8	75.52
	S-CMARS	41	5993.2	0.998	42147837.8	7.44
3	MARS	35	24.1*	0.999	693.4	3210.70
	MinSpan	35	23.5	0.999	661.6*	411.79
	SMARS	33	28.1	0.999	937.1	116.71*
	CMARS	47	24.1*	0.999	740.5	3191.42
	S-CMARS	43	28.1	0.999	990.2	124.14

Table21. Cont.

Datasets	Methods	BF_{final}	RMSE	Adj-R^2	GCV	CPU Time (sec.)
4	MARS	49	0.023*	0.998	0.001	295.99
	MinSpan	46	0.026	0.998	0.001	84.45
	SMARS	52	0.025	0.998	0.001	33.54*
	CMARS	87	0.023*	0.998	0.001	548.58
	S-CMARS	95	0.025	0.998	0.001	67.99
5	MARS	32	467.6*	0.999*	257450.5	129.37
	MinSpan	27	524.1	0.999*	314813.2	30.97
	SMARS	33	467.4*	0.999*	222016.0*	184.76
	CMARS	45	1031.7	0.997	276350.1	219.45
	S-CMARS	41	1044.3	0.997	286406.0	17.33*
6	MARS	14	3.2	1.000*	10.6	60.14
	MinSpan	14	3.1*	1.000*	10.3*	10.06
	SMARS	15	3.4	1.000*	12.4	4.37*
	CMARS	21	176.1	0.533	11.0	60.71
	S-CMARS	21	176.7	0.529	12.8	6.50
7	MARS	80	0.029*	0.992*	0.001	5952.52
	MinSpan	78	0.031	0.991	0.001	1043.46
	SMARS	77	0.030	0.991	0.001	360.44
	CMARS	101	0.030	0.991	0.001	3378.96
	S-CMARS	101	0.030	0.992*	0.002	258.61*
8	MARS	44	0.162	0.998*	0.033*	491.63
	MinSpan	48	0.165	0.998*	0.035	95.68
	SMARS	43	0.185	0.997	0.043	30.59
	CMARS	75	0.160*	0.998*	0.039	481.90
	S-CMARS	73	0.185	0.997	0.051	28.53*

- The number of BFs in the final model gives information about the complexity of the final model. For all datasets, CMARS and S-CMARS models seem to include the maximum number of BFs provided by the forward selection part. This property is stated as a disadvantage of the method in the study of Weber et. al. (2012), which is still valid for the S-CMARS method. However, MARS do no remove the BFs even if they have approximately zero coefficients (Yerlikaya, 2008).

- For training and test data sets one, five and six, MARS, MinSpan and SMARS perform better than CMARS and S-CMARS with respect to RMSE and Ajd-R^2. Although SMARS and S-CMARS methods are based on the same forward selection algorithm (S-FMARS), their accuracy performances are also different.
- For training and test data sets two and eight, MARS, MinSpan and SMARS slightly overperfoms CMARS and S-CMARS. This result attributed to the mapping approach; during the mapping, the underlying data structure may not be approximated properly. Hence, important knot points can be ignored.
- With respect to stability, SMARS is more stable than the other methods for data sets one, two and three. For problem three, S-CMARS is as stable as SMARS method, as well as for the data set seven. While MinSpan is more stable than other methods for data sets four and eight, stable models are obtained by CMARS and MARS for data sets five and six, respectively.
- Run times are related to the sample size, number of predictors, number of interaction terms and maximum number of BFs set by the user. Since the same parameter values are set for all methods, efficiency of methods can be compared for each dataset separately. For all training datasets, the most efficient method is SMARS which is then followed by S-CMARS. The CPU time required for model building is much more less in these methods than the CPU times of other methods. The MinSpan approach also decreases the CPU time of MARS algorithm significantly.

Table 22. Average performance of methods on test data and stabilities.

Datasets	Methods	TEST		STABILITY	
		RMSE	Adj-R^2	RMSE	Adj-R^2
1	MARS	1147.4	0.978	0.962	0.998
	MinSpan	1291.3	0.973	0.927	0.999
	SMARS	1153.5	0.978	0.984	0.997
	CMARS	6526.8	0.298	0.908	0.985
	S-CMARS	6079.7	0.390	0.924	0.910

Table22. Cont.

Datasets	Methods	TEST		STABILITY	
		RMSE	Adj-R^2	RMSE	Adj-R^2
2	MARS	3849.4	0.999	0.901	1.000
	MinSpan	3475.7	1.000	0.818	1.000
	SMARS	5722.6	0.999	0.958	1.000
	CMARS	3849.5	0.999	0.901	1.000
	S-CMARS	5703.4	0.999	0.952	1.000
3	MARS	23.1	0.999	0.959	1.000
	MinSpan	22.7	0.999	0.965	1.000
	SMARS	27.7	0.999	0.986	1.000
	CMARS	23.0	0.999	0.956	1.000
	S-CMARS	27.7	0.999	0.986	1.000
4	MARS	0.026	0.998	0.872	0.999
	MinSpan	0.027	0.998	0.984	1.000
	SMARS	0.026	0.998	0.960	1.000
	CMARS	0.026	0.998	0.866	1.000
	S-CMARS	0.027	0.998	0.943	1.000
5	MARS	441.1	0.999	0.943	1.000
	MinSpan	491.9	0.999	0.939	1.000
	SMARS	481.9	1.000	0.970	1.000
	CMARS	1023.7	0.997	0.992	1.000
	S-CMARS	944.6	0.998	0.905	0.999
6	MARS	3.3	1.000	0.957	1.000
	MinSpan	3.3	1.000	0.940	1.000
	SMARS	3.2	1.000	0.937	1.000
	CMARS	193.6	0.526	0.910	0.987
	S-CMARS	194.4	0.521	0.909	0.985
7	MARS	0.028	0.993	0.990	0.999
	MinSpan	0.029	0.993	0.935	0.998
	SMARS	0.029	0.993	0.957	0.998
	CMARS	0.030	0.992	0.994	0.999
	S-CMARS	0.030	0.992	1.000	1.000
8	MARS	0.189	0.997	0.854	0.999
	MinSpan	0.172	0.998	0.959	1.000
	SMARS	0.200	0.997	0.926	1.000
	CMARS	0.189	0.997	0.849	0.999
	S-CMARS	0.200	0.997	0.925	1.000

4.5.2. Real Datasets

Five methods are evaluated and compared for six real datasets with different sizes and scales (Table 1). Different maximum number of BF is set for each data set (Table 23), but the same number of interaction terms is used by the methods within each data set. The average of measures obtained for 3-folds and three replications of each train datasets are given in Table 23. The results obtained for test datasets are given in Table 24.

Since the data sets are standardized before the application, it becomes possible to compare the overall averages of measures calculated for each method. Then, the mean and standard deviation values of all accuracy measures obtained for training and test dataset are given in Table 25 as well as those of the stability of the measures for evaluating the overall performances of methods. In order to compare the performance of methods statistically, RANOVA is performed. The test results is evaluated at α=0.05 significance level. This test procedure is applied for training and test datasets, as well as stabilities of the measures.

Depending on the results presented in Tables 23, 24 and 25, the following conclusions can be drawn:

- Due to the number of BFs in the final model, the models built by CMARS and S-CMARS for training data seem more complex than the other methods.
- The accuracy measures of methods are close to each other for training data sets except the Com.Crime training data. For this data set, MARS, MinSpan and SMARS overperforms CMARS and S-CMARS.
- With respect to RMSE measure, CMARS performs better than the other methods. With respect to Adj-R^2 and GCV values, however, MARS shows better performance.

Table 23. Average performances of methods on train data.

Datasets	Methods	BF_{final}	Measures			
			RMSE	Adj-R^2	GCV	CPU Time (sec.)
Com.Crime	MARS	59	0.405*	0.824*	0.257*	250.44
	MinSpan	46	0.446	0.790	0.277	234.77
	SMARS	36	0.475	0.765	0.292	137.50*
	CMARS	151	0.527	0.665	0.588	413.35
	S-CMARS	151	0.519	0.674	0.739	196.73
Con.Comp.	MARS	60	0.220	0.948*	0.079*	873.97
	MinSpan	61	0.221	0.948*	0.080	338.56
	SMARS	60	0.228	0.944	0.087	156.20
	CMARS	101	0.216*	0.948*	0.107	871.18
	S-CMARS	101	0.235	0.938	0.117	153.71*
Parkinsons	MARS	25	0.347	0.877*	0.158*	62.24
	MinSpan	23	0.354	0.872	0.161	12.25
	SMARS	30	0.373	0.857	0.169	6.70*
	CMARS	51	0.345*	0.872	0.200	80.42
	S-CMARS	51	0.354	0.866	0.219	31.25
AuoMpg	MARS	26	2.157	0.917*	7.235*	24.84
	MinSpan	19	2.435	0.897	8.100	11.90
	SMARS	13	2.555	0.888	8.039	10.51*
	CMARS	101	2.154*	0.897	64.266	27.53
	S-CMARS	101	2.229	0.890	90.690	10.83
PM10	MARS	24	0.671	0.541*	0.602*	12.43
	MinSpan	21	0.688	0.520	0.612	3.96
	SMARS	20	0.696	0.510	0.618	3.35*
	CMARS	51	0.665*	0.522	0.853	16.28
	S-CMARS	51	0.670	0.514	0.895	6.92
Red Wine	MARS	43	0.678	0.531	0.567	770.12
	MinSpan	44	0.672*	0.538*	0.562*	406.91
	SMARS	37	0.686	0.521	0.563	187.31*
	CMARS	91	0.682	0.507	0.671	780.74
	S-CMARS	91	0.694	0.491	0.676	196.62

Note: * indicates better performance.

Table 24. Average performances of methods on test data and stabilities.

Datasets	Methods	TEST		STABILITY	
		RMSE	Adj_R^2	RMSE	Adj_R^2
Com.Crime	MARS	0.694	0.505	0.584	0.613
	MinSpan	0.637	0.582	0.700	0.737
	SMARS	0.637	0.584	0.746	0.764
	CMARS	0.563	0.676	0.936*	0.984
	S-CMARS	0.560*	0.678*	0.926	0.995*
Con.Comp.	MARS	0.347	0.877	0.633	0.925
	MinSpan	0.329*	0.892*	0.671*	0.941*
	SMARS	0.381	0.845	0.599	0.894
	CMARS	0.338	0.885	0.639	0.933
	S-CMARS	0.366	0.858	0.642	0.915
Parkinsons	MARS	0.481	0.745	0.721	0.849
	MinSpan	0.452*	0.780*	0.785	0.894
	SMARS	0.457	0.773	0.815*	0.903*
	CMARS	0.487	0.737	0.709	0.844
	S-CMARS	0.478	0.750	0.742	0.866
AuoMpg	MARS	3.311	0.780	0.651	0.850
	MinSpan	2.583	0.871*	0.943*	0.971*
	SMARS	2.725*	0.856	0.938	0.964
	CMARS	3.017	0.817	0.714	0.911
	S-CMARS	2.780	0.852	0.802	0.957
PM10	MARS	0.839	0.242	0.800	0.448
	MinSpan	0.847	0.229	0.813	0.440
	SMARS	0.794	0.322	0.877*	0.632
	CMARS	0.811	0.291	0.819	0.558
	S-CMARS	0.788*	0.332*	0.850	0.646*
Red Wine	MARS	0.908	0.161	0.746	0.302
	MinSpan	0.903	0.169	0.744	0.315
	SMARS	0.909	0.155	0.755	0.297
	CMARS	0.847*	0.267*	0.806*	0.526*
	S-CMARS	0.872	0.221	0.796	0.451

Note: * indicates better performance.

Table 25. Overall performances of methods.

Methods	TRAIN			TEST		STABILITY	
	RMSE	Adj-R^2	GCV	RMSE	Adj-R^2	RMSE	Adj-R^2
MARS	0.746*	0.773*	1.483*	1.097	0.530*	0.689	0.632
	(0.715**)	(0.189)	(2.826**)	(1.105)	(0.308)	(0.080**)	(0.272)
MinSpan	0.803	0.761	1.632	0.958*	0.570	0.776	0.689
	(0.820)	(0.187**)	(3.176)	(0.826**)	(0.331)	(0.097)	(0.299)
SMARS	0.835	0.747	1.628	0.984	0.573	0.788*	0.718
	(0.862)	(0.189)	(3.148)	(0.876)	(0.300)	(0.118)	(0.263)
CMARS	0.765	0.735	11.114	1.010	0.563	0.770	0.722
	(0.704)	(0.196)	(26.040)	(1.002)	(0.282**)	(0.105)	(0.237**)
S-CMARS	0.784	0.729	15.556	0.974	0.569	0.793	0.733*
	(0.730)	(0.197)	(36.809)	(0.905)	(0.286)	(0.096)	(0.243)

Notes: * Indicates better performance with respect to mean. ** Indicate better performance with respect to standard deviation.

- CMARS and S-CMARS perform better than the other methods for Communities Crime test data, although their performances are worse than others for training data set. For the four test data, MinSpan performs better than the other methods with respect to RMSE and Adj-R^2. While the performance of S-CMARS is better than the others for PM10 test data, CMARS overperforms others for Red Wine test data.
- With respect to stabilities of RMSE values, CMARS produce more stable models for Communities Crime and Red Wine data. The models of SMARS built for Parkinsons and PM10 are more stable than the other models. For rest of the test data, MinSpan seems more stable. On the other hand, S-CMARS seems more stable for Communities Crime and PM10 with respect to Adj-R2 values. Again, while MinSpan produce more stable models for Concrete Compression and AutoMpg, SMARS is more stable for Parkinsons.
- RANOVA test results obtained for training and test data sets as well as stabilities of measures conclude that there are no cases where one method is statistically significantly better than the others with respect to RMSE, Adj-R^2 and GCV measures.

- SMARS seems the most efficient method with minimum CPU time. S-CMARS comes the second. MinSpan is more efficient than MARS algorithm, but not as much as SMARS.

To observe the effects of sample size and scales on CPU time, the average CPU times of methods obtained for the datasets classified with respect to scale and sample sizes as in Section 4.3.2 are given in Table 26.

Table 26. Average CPU times of methods with respect to sample size

Features of Data		Methods				
		MARS	MinSpan	SMARS	CMARS	S-CMARS
Sample Size	Medium	33.2	9.4	6.9	41.4	16.3
	Large	631.5	326.7	160.3	688.4	182.4
	Percent Decrease (%)	95	97	96	94	91
Scale	Small	303.7	118.1	56.7	305.0	57.2
	Large	360.9	218.0	110.5	424.8	141.5
	Percent Decrease (%)	16	46	49	28	60

Additionally, the significance of differences between the average CPU times of methods obtained for two types of data classified with respect to scale and sample size are tested by using Mann-Whitney test. Depending on the results presented in Table 26, the following conclusions can be drawn:

- All methods are more efficient for medium datasets than large ones. The difference between the average CPU times of methods obtained for medium and large datasets are found significant by Mann-Whitney test. Additionally, SMARS performs better than all other methods on both medium and large datasets.

- The most significant decrease in CPU time is observed for MinSpan approach between medium and large datasets.
- The effect of scale on CPU time is not as significant as sample size. The most significant decrease is observed for S-CMARS method, which is 60 %. Although methods seem slightly efficient for small scaled data than the large scaled ones, the difference between CPU times between two types of data are not statistically significant according to Mann-Whitney test (*p*-value>0.05).

Three-way interaction effects including sample size, scale and methods are examined through interaction plots displayed in Figure 25. for determining the best size-scale combination for a method in relation with CPU time. Figure 25 shows that all methods are more efficient on small scale and medium training samples with respect to CPU time. While the scale affects the performance of MARS, CMARS and S-CMARS methods with respect to time efficiency, scale has no significant effect on CPU time of MinSpan and SMARS. In addition, for large training samples, MARS and CMARS are more efficient for large-scaled data than small-scaled ones.

4.5.3. Performance on Noisy Data

To evaluate the sensitivity of five methods on noisy data, the same data sets used in Section 4.3.3 are also used in this comparison study. Two data sets are generated with and without noise. For the first analysis, a sinus function is used.

4.5.3.1. Noisy Data 1

Finally, a simulation study is performed to see the effect of noise on performance of all methods. For this purpose, data having 100 observations are generated using the sinus function with and without noise. Five models are fitted to them, and both the accuracy and complexity measures are obtained as in Table 27. In addition, to measure the sensitivity of the fits to the noisy data, the performance measures are recalculated using the fitted values obtained for noisy data, and noise-free data points. The conclusions drawn from the analysis for noisy data are given as follows:

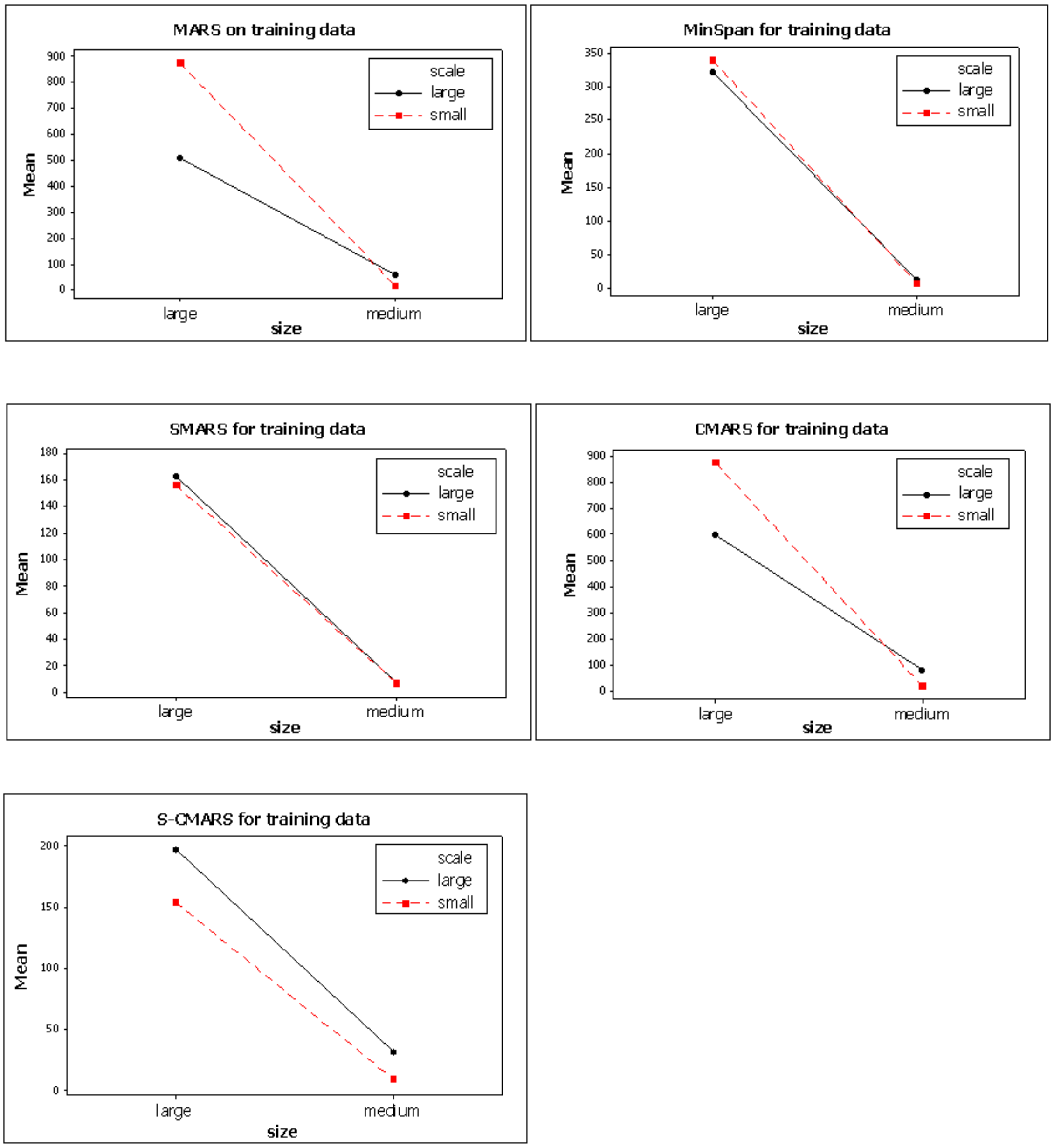


Figure 25. Interaction plots of size and scale for the CPU times for five methods**.**

- CMARS and S-CMARS built more complex models both noisy and noise-free data.
- CMARS and MARS methods overperform other methods on noisy data with respect to RMSE and Adj-R^2 measures, respectively. They perform better on noise-free data as well.
- The fit obtained by MinSpan for noisy data is able to predict the noise-free data well. Thus, it is less sensitive to noise than the other methods.

Table 27. Average performance of methods on test noisy data.

	Methods	Measures		
		BF_{final}	RMSE	Adj-R^2
Noise-free data	MARS	11	0.012*	1.000
	MinSpan	10	0.015	0.999
	SMARS	11	0.014	1.000
	CMARS	19	0.012*	1.000
	S-CMARS	19	0.014	1.000
Noisy data	MARS	6	0.235	0.892*
	MinSpan	5	0.245	0.884
	SMARS	4	0.257	0.873
	CMARS	21	0.228*	0.879
	S-CMARS	19	0.243	0.866
Noisy vs noise-free data	MARS	6	0.084	0.984
	MinSpan	5	0.071*	0.989*
	SMARS	4	0.121	0.968
	CMARS	21	0.118	0.952
	S-CMARS	19	0.078	0.980

Note: * indicates better performance

4.5.3.2. Noisy Data 2

A second simulation study is carried on using the function in (37), and a similar study is constructed for all methods as in the previous section. Namely, the performances of models built by five methods are compared on noisy data with respect to accuracy and complexity measures (Table 28), and the sensitivity of fits on noisy data is revealed in Table. Additionally, a new test data is generated using the same function and the same measures are calculated to evaluate the prediction performances of fits for new observations.

Table 28. Average performances of methods on noisy data.

	Methods	Train			Test	
		BF_{final}	RMSE	Adj-R^2	RMSE	Adj-R^2
Noise-free data	MARS	7	0.691*	0.999	1.060	0.999
	MinSpan	7	1.012	0.999	2.964	0.999
	SMARS	8	0.754	0.999	1.081	0.999
	CMARS	7	0.692	0.999	3.579	0.999
	S-CMARS	9	0.754	0.999	1.081	0.999
Noisy data	MARS	8	6.961	0.992	8.398	0.989
	MinSpan	5	7.466	0.991	8.134*	0.990*
	SMARS	4	7.449	0.991	8.380	0.988
	CMARS	21	6.999	0.991	8.159	0.987
	S-CMARS	19	6.960*	0.993*	8.826	0.988
Noisy vs noise-free data	MARS	8	2.506	0.999	3.089	0.998
	MinSpan	5	2.666	0.999	2.646	0.999*
	SMARS	4	3.134	0.998	2.825	0.999*
	CMARS	21	2.321*	0.999	2.477*	0.998
	S-CMARS	19	2.908	0.998	3.921	0.996

Note: * indicates better performance

The analysis results can be summarized as follows:

- MARS performs better on noise-free data with respect to accuracy and prediction capability.
- S-CMARS performs better than other methods on noisy training data with respect to RMSE and Adj-R^2.

- The prediction performance of MinSpan is better than other methods on noisy test data.
- The CMARS fit obtained for noisy data can also predict the noise-free data better than the other methods. Hence, the sensitivity of SMARS to noisy data is less than the one of other methods.

CHAPTER 5

CONCLUSION AND FURTHER RESEARCH

In the spline smoothing, one of the critical issue is determining the proper knots, especially, for curves having varying shapes. In this study, we propose a two-stage knot selection procedure for adaptive regression splines. Firstly, a potential set of knots is selected by a mapping approach with the intension to locate points according to the data distribution. The final knot selection is then made by a stepwise model fitting algorithm. The combination of these two procedures, so called S-FMARS, is a modified forward selection step of MARS which provides a time efficient model building strategy for adaptive regression splines without degrading the model accuracy and prediction performance.

In S-FMARS, two important parameters have special effects on model building and CPU time: grid size and threshold value set for the number of data points assigned to each of the map unit. As mentioned in Section 3.4, the grid size (number of neurons in the lattice) is as a trade-off between less computing time and a good approximation both in mapping and modeling. As the grid size increases, the approximation of underlying data structure become well, but the CPU time required for mapping and modeling increases. During the mapping, similar data points are grouped around the neurons having a neighborhood relation in the lattice. Among these neurons, while some units carry more data points, some others are just attained to one data point. The neurons assigned to less number of data points most probably represent outliers or sparse regions in the data space. By setting a threshold value for the number of data points assigned to one neuron, outliers and data points, where less data structure occurs can be eliminated. For better approximation and best subsetting with more representative data points, a sensitivity analysis is studied for the best grid size and threshold value. In this sensitivity analysis, a measure (RMSE/TIME) is proposed to be used in the determination of best parameter values. By observing the change in this ratio value against different grid sizes and threshold values, the best grid size and the threshold value is determined.

Once the parameters of the S-FMARS are found, the method can be applied to any datasets with continuous response. Especially for high-dimensional and large datasets, S-FMARS can be considered as a strong alternative to the conventional forward selection procedures in spline fitting. The performance of S-FMARS is evaluated and compared with the forward selection algorithm of MARS (FMARS) and also MinSpan with respect to accuracy, complexity, stability and robustness criteria via a set of artificial and real datasets. The analyses conclude that the performance of S-FMARS models is not statistically different from the models obtained by FMARS and MinSpan approaches with respect to all criteria. Moreover, it is obviously clear that the S-FMARS approach is much more efficient than the other methods. For noisy settings, the fit obtained by S-FMARS for noisy data can also approximate the noise-free data well. Hence, the S-FMARS fits seems to be less sensitive to noise than those of FMARS.

The forward selection approach of regression splines builds a large model which deliberately overfits the data. This property is also valid for the proposed forward selection algorithm. In general approach, a backward elimination step is applied to prune the model comes from the forward step. In this strategy, contributions of model terms are evaluated through a complexity measure; MARS uses this strategy. In some studies, however, contributions of model terms are examined via a penalized term added to accuracy measure. In this strategy, parameter estimation is achieved through a PRSS; CMARS bases on this strategy. Based on two purposes, both the backward elimination strategy of MARS and the PRSS strategy of CMARS can be applied to S-FMARS. The first one is to eliminate the overfitting problem, and to provide a complete adaptive regression spline method. The second one is to solve the main drawbacks of CMARS approach, which is being inefficient in time. CMARS construct PRSS problem as a Tikhonov regularization problem and solves it using a CQP, which make the method computationally expensive. In addition, the PRSS problem is based on the knot points selected among the data points by the forward step of MARS. During the knot selection process, as the number of data points is increased, more data points are evaluated as knot points, which leads to an increase in the computing time of CMARS, significantly. In this respect, S-FMARS can be a good alternative by selecting a representative data points to be evaluated as knots. In this thesis, the proposed forward selection algorithm is implemented to both the CMARS and MARS algorithms, which are named as S-CMARS and SMARS, respectively. Their performances are evaluated via many performance criteria and compared with MARS, MARS with MinSpan and CMARS methods. The results of the analysis indicate that SMARS and S-CMARS are obviously the most efficient two methods with respect to time. Their CPU times are significantly less than those of the other methods. Even CMARS is improved by the proposed forward selection algorithm as being more efficient than MARS. The performances of SMARS and S-CMARS seem not to be as good as the other methods for some real datasets; however, the accuracy loss is small in absolute values compared to the run times. Moreover, RANOVA test results obtained for the real life data sets show that performances of SMARS and S-CMARS are not statistically significantly different from the other methods. Actually, for the real data sets under study, there are no cases where a method seems effective with respect to all performance measures. For artificial datasets, the performance measures of methods are evaluated within each data separately due to different problem scales. For some generated data the performance of SMARS and S-CMARS does not seem as good as the performances of the other methods. This may be resulted from the inadequate approximation of underlying data structure caused by mapping or the projection of weight vectors to original data points. The proper knot points could be ignored during the mapping or projection. To make the SMARS and S-CMARS methods more accurate, one has to provide a good approximation of underlying data. Besides, for same cases, the reason of bad performances may not be rooted from mapping idea, but from the strategy behind the CMARS. Before the application of CMARS, the optimal value of the bound set in the optimization problem (18) should be found by investigating the corner points. In some cases, however, it is difficult to catch the corner points properly.

The sensitivity of methods on noisy setting is also examined in the study. The analysis results show that CMARS and MinSpan seem to be less sensitive to noise than the other methods. MinSpan overperforms the other methods on noisy test data. In addition, MARS performs better than other methods on noise-free data with respect to accuracy and prediction capability.

The models build by S-CMARS and CMARS are more complex than the models of other methods. As mentioned in Yerlikaya (2008), even though the BFs having coefficients zero or near to zero, they are remained in the final model of CMARS. Namely, the BFs contributing less to model are not removed. The same property is also valid for S-CMARS. In this respect, a bootstrapping strategy is proposed by Yazıcı (2011) to decrease the model complexity of CMARS. By a bootstrapping approach, the contribution of each BFs to the model can be determined by drawing bootstrap samples from the data sets, and computing the corresponding coefficients for each sample. Bootstrapping is a computer-intensive method due to its high dependence on computers. As a whole, CMARS with bootstrapping approach requires more CPU time. However, S-CMARS is more efficient in time than CMARS; so that, the bootstrapping can be implemented to S-CMARS to decrease the model complexity and computing time, which is left as a future work.

As another future study, the mapping strategy can be studied as a feature reduction method of the proposed approach, S-FMARS, to decrease the model complexity (i.e.number of terms in model). If the predictor variables can be reduced by considering their importance to model through the mapping, the knot selection process of spline fitting can be applied on the new set of predictor variables. So that, the computational complexity caused by the forward selection step can also be decreased.

As a final conclusion, the newly developed knot selection scheme can be implemented to any kind of adaptive regression splines including a forward knot selection algorithm. This study only covers the estimation of continuous responses. However, the idea behind the proposed approach can also be studied as a future work for the responses with discrete levels such as binary, nominal or ordinal.

REFERENCES

Akaike, H. (1973). Information Theory and an Extension of the Maximum Likelihood Principle, *2nd International Symposium on Information Theory* (ed. B. F. Petrov and F. Cs´aki), Academiai Kiado, Budapest.

Atilgan, T. (1988). Basis Selection for Density Estimation and Regression. AT&T Bell Laboratories technical memorandum.

Batmaz, İ., Yerlikaya-Özkurt, F., Kartal-Koç, E., Köksal, G. and Weber, G. W. (2010). Evaluating the CMARS Performance for Modeling Nonlinearities. *Proceedings of the 3rd Global Conference on Power Control and Optimization, Gold Coast (Australia),* 1239, 351-357.

Breiman, L. (1993). Fitting Additive Models to Regression Data. *Computational Statistics and Data Analysis*, 15, 13-46.

Cleveland, W.S. (1993). Robust Locally Weigthed Regression and Smoothing Scatterplots. *Journal of the American Association*, 74 (368), 829-836.

Craven, P., and Wahba, G. (1979). Smoothing Noisy Data with Spline Functions: Estimating the Correct Degree of Smoothing by the Method of Generalized Cross Validation. *Numerische Mathematik*, 31, 377-403.

Davis, C.S. (2003). *Statistical Methods for the Analysis of Repeated Measures*. Springer-Verlag, New York.

de Boor, C. (1978). A Practical to Guide to Splines. Springer-Verlag, New York.

Deichmann, J., Eshghi, A., Haughton, D., Sayek, S., and Teebagy, N. (2002). Application of Multiple Adaptive Regression Splines (MARS) in Direct Response Modeling. *Journal of Direct Marketing*, 16, 15–27.

Di, W. (2006). Long Term Fixed Mortgage Rate Prediction Using Multivariate Adaptive Regression Splines. School of Computer Engineering, Nanyang Technological University.

Denison, D.G.T, Mallick, B.K. and Smith, A.F.M. (1998). Automatic Bayesian Curve Fitting. *J. Roy. Statist. Soc.*, 60, 333–350.

Eilers, Paul H.C., and Marx, B.D. (1996). Flexible Smoothing with B-splines and Penalties. *Statistical Science*, 11, 98-102.

Eubank, R. L. (1999). *Nonparametric Regression and Spline Smoothing*. Marcel Dekker, New York.

Friedman, J.H., and Silverman, B.W. (1989). Flexible Parsimonious Smoothing and Additive Modelling. *Technometrics*, 31, 3–21.

Friedman, J.H. (1991). Multivariate Adaptive Regression Splines. *The Annals of Statistics*, 19, 1-67.

Friedman, J.H. (1993). *Fast MARS*. Stanford University Department of Statistics, Technical report 110.

Fox, J. (2002). Nonparametric Regression, *An R and S-PLUS Companion to Applied Regression*. Sage, Thousand Oaks CA.

Frank, A., and Asuncion, A. (2010). UCI Machine Learning Repository. Irvine, CA: University of California, School of Information and Computer Science. Available at http://archive.ics.uci.edu/ml.

Gersho, A., and Gray, R.M. (1992). *Vector Quantization and Signal Compression*. Kluwer ,Norwell.

Gibbons, J.D., and Chakraborti, S. (2003). *Nonparametric Statistical Inference*. Marcel Dekker, New York.

Green, P.H., and Silverman, B.W. (1994). *Nonparametric Regression and Generalized Linear Models*. Chapman Hall,Boca Raton, FL.

Hastie, T.J., and Tibshirani, R.J. (1990). *Generalized Additive Models*, Chapman and Hall, New York.

Hastie, T.J., Tibshirani, R.J., and Friedman, J. (2001). *The Elements of Statistical Learning, Data Mining, Inference and Prediction*. Springer, New York.

Haykin, S. (1999). *Neural Networks: A Comprehensive Foundation*. Prentice-Hall, New Jersey.

He, X. and Ng, P. (1996). COBS: Constrained Smoothing Made Easy. Unpublished manuscript.

Jekabsons, G. (2011). ARESLab: Adaptive Regression Splines Toolbox for Matlab/Octave. Available at http:// www.cs.rtu.lv/jekabsons/

Jin, R., Chen, W., and Simpson, T.W. (2001). Comparative Studies of Metamodeling Techniques under Multiple Modeling Criteria. *Structural and Multidisciplinary Optimization*, 23, 1-13.

Keele, L. (2008). *Semiparametric Regression for the Social Sciences*. John Wiley and Sons, Chichester, UK.

Kohonen, T. (1988). *Self-Organizing and Associative Memory*. Springer-Verlag, New York.

Kubin, G., (1999). Nonlinear Prediction of Mobile Radio Channels: Measurments and Mars Model Designs, IEEE Proc. International Conference on Acoustics, Speech, and Signal Processing, 5, 15-19, 2667-2670.

Leathwick, J.R., Rowe,D., Richardson, J., Elith, J., and Hastie, T. (2005). Using Multivariate Adaptive Regression Splines to Predict the Distributions of New Zealand’s Freshwater Diadromous Fish. *Freshwater Biology*, 50, 2034–2052.

Leathwick, J.R., Elith, J., and Hastie, T. (2006). Comparative Performance of Generalized Additive Models and Multivariate Adaptive Regression Splines for Statistical Modelling of Species Distributions. *Ecological Modelling*, 199, 188–196.

Lee, T.S., Chiu, C.C., Chou, Y.C., and Lu, C.J. (2006). Mining the Customer Credit using Classifcation and Regression Tree and Multivariate Adaptive Regression Splines. *Computational Statistics and Data Analysis,* 50, 1113–1130.

Lehmann, Erich L. (1975); *Nonparametrics: Statistical Methods Based on Ranks*.

Lou, Z., and Wahba, G. (1997). Hybrid Adaptive Splines. *Journal of the American Statistical Association*, 92, 107–116.
Mallows, C. L. (1973). Some comments on *Cp*, *Technometrics*, 15, 661–675.
Özmen, A. (2010). Robust Conic Quadratic Programming Applied to Quality Improvement- A Robustification of CMARS, MSc., Middle East Technical University.

Özmen, A., Weber, G-W., and Batmaz, İ. (2010). The new robust CMARS (RCMARS) method, 24th Mini EURO Conference-On Continuous Optimization and Information-Based Technologies in the Financial Sector, MEC EurOPT Selected Papers, ISI Proceedings, 362–368.

Ruppert, D. (2002). Selecting the Number of Knots for Penalized Splines. *Journal of Computational and Graphical Statistics*, 11, 735-757.

Ruppert, D., Wand, M.P., and Carroll, R.J. (2003). *Semiparametric Regression*. Cambridge University Press.

Schoenberg, I. (1964a). On Interpolation by Spline Functions and its Minimum Properties. *International Series of Numerical Analysis.* 5, 109–129.

Schoenberg, I. (1964b). Spline Functions and the Problem of Graduation. *Natural Academy of Science*, 52, 947–950.

Schumaker, L. L. (1981). *Spline Functions: Basic Theory*. John Wiley & Sons, Inc.

Silverman, B. W. (1985). Some Aspects of the Spline Smoothing Approach to Nonparametric Regression Curve Fitting. *Journal of the Royal Statistical Society*, Series B, 47, 1–52.

Smith, P.L. (1982). Curve Fitting and Modeling with Splines using Statistical Variable Selection Techniques. Report NASA 166034, NASA, Langley Research Center, Hampton.

Smith, M. and Kohn, R. (1996). Nonparametric Regression Using Bayesian Variable Selection, *Journal of Econometrics*, 75, 317–343.

Stone, C.J., and Koo, CY. (1985). Additive Splines in Statistics. *Proceeding of the Statistical Computing section. Alexandria, VA: American Statistical Association*, 45-48.

Stone, C.J. (1986). Comment: Generalized additive models. *Statistical Science*, 2:312-314.

Stone, C., Hansen, M., Kooperberg, C., and Troung, Y. (1997). Polynomial Splines and their Tensor Products in Extended Linear Modeling. *Annals of Statistics*, 25, 1371-1470.

Taylan,P., Weber G-W, and Beck, A. (2007). New Approaches to Regression by Generalized Additive Models and Continuoues Optimization for Modern Applications in Finance, Science and Technology, Journal of Optimization. 56, 675-698.

Vesanto, J., Himberg, J., Alhoniehmi, E., and Parhankangas, J. (2000). SOM toolbox for Matlab 5, Report A57. Available at http:// www.cis.hut.fi // projects/ somtoolbox/

Yao F., and Lee Thomas, C.M. (2008). An Improved Knot Placement Scheme for Penalized Spline Regression. *Journal of the Korean Statistical Society*. 37, 259-267.

Yerlikaya, F. (2008). *A New Contribution to Nonlinear Robust Regression and Classication with MARS and its Application to Data Mining for Quality Control in Manufacturing*. Master Thesis, Graduate School of Applied Mathematics, Department of Scientific Computing, METU, Ankara, Turkey.

Yazici, C. (2011). A Computational Approach to Nonparametric Regression: Bootstrapping the CMARS Method. Master Thesis, Department of Statistics, METU, Ankara, Turkey.

York, T.P., Eaves, L.J., and Van den Oord, E.J.C.G. (2006). Multivariate Adaptive Regression Splines: a Powerful Method for Detecting Disease-risk Relationship Differences Among Subgroubs, *Statistics in Medicine*, 25, 8, 1355-1367.

Wahba, G. (1983). Bayesian Confidence Intervals for the Cross-Validated Smoothing Spline. *Journal of the Royal Statistical Society*, Series B 45 133–150.

Wahba, G. (1990). Spline Models for Observational Data. CBMS-NSF 59, Regional Conference Series in Applied Mathematics.

Wahba G., and Wold, S. (1975). A Completely Automatic French Curve: Fitting Spline Functions by Cross Validation. *Commun. Statistics*, 4, 1-17.

Weber, G-W., Batmaz, İ., Köksal, G., Taylan, P., Yerlikaya-Özkurt, F. (2012). CMARS: A New Contribution to Nonparametric Regression with Multivariate Adaptive Regression Splines Supported by Continuous Optimization. *Inverse Problems in Science and Engineering*, 20 (3), 371-400.

Whittaker, E. T. (1923). On a New Method of Graduation. *Proceedings of the Edinburgh Mathematical Society*, 41, 63–75.

Wong, C., and Kohn, R. (1996). A Bayesian Approach to Additive Semi-Parametric Regression. *Journal of Econometrics*, 74, 209–235.

APPENDIX A

MATHEMATICAL FORMULATIONS FOR DATA GENERATION

Mathematical functions used for generation of the artificial datasets given in Table 1.

P1. $f(\mathbf{x}) = \sum_{i=1}^{7}[(\ln(x_i - 2))^2 + (\ln(10 - x_i))^2] - \left(\prod_{i=1}^{7} x_i\right)^2 \quad 2.1 \le x_i \le 9.9$

P2. $f(\mathbf{x}) = \sum_{j=1}^{10} \exp(x_j)\left(c_j + x_j - \ln\left(\sum_{k=1}^{10}\exp(x_k)\right)\right),$

cj = -6.089,-17.164,-34.054,-5.914,-24.721,-14.986,-24.100,-10.708,
-26.662,-22.179

P3.

$$f(\mathbf{x}) = x_1^2 + x_2^2 + x_1x_2 - 14x_1 - 16x_2 + (x_3 - 10)^2 - 4(x_4 - 5)^2 + (x_5 - 3)^2 + 2(x_6 - 1)^2$$
$$+ 5x_7^2 + 7(x_8 - 11)^2 + 2(x_9 - 10)^2 + (x_{10} - 7)^2 + 45$$

P4. $f(\mathbf{x}) = \sin(\pi x_1/12)\cos(\pi x_2/16)$

P5. $f(\mathbf{x}) = (x_1 - 1)^2 + (x_1 - x_2)^2 + (x_2 - x_3)^4$

P6. $f(\mathbf{x}) = 5.3578547x_2^2 + 0.8356891x_1x_3 + 37.293239x_1 - 40792.141$

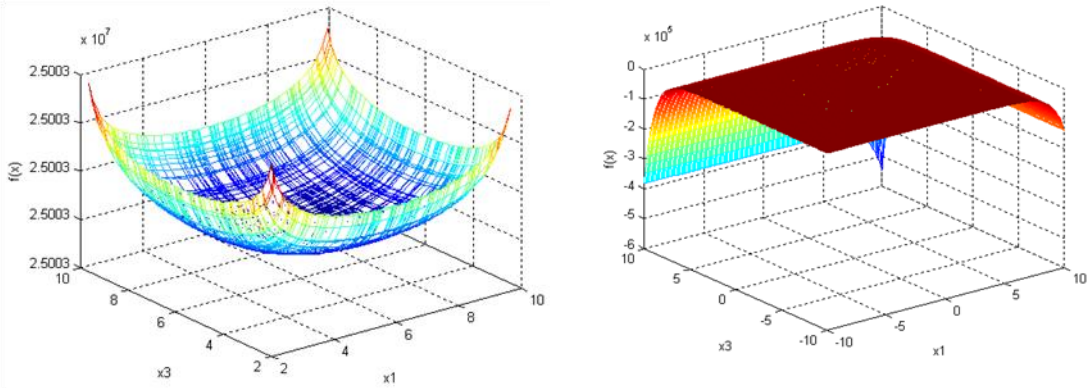
P8.

$$f(\mathbf{x}) = 3(1-x)^2\exp(-x^2 - (y+1)^2) - 10(x/5 - x^3 - y^5)\exp(-x^2 - y^2)$$
$$-1/3\exp(-(x+1)^2 - y^2) + 2x$$

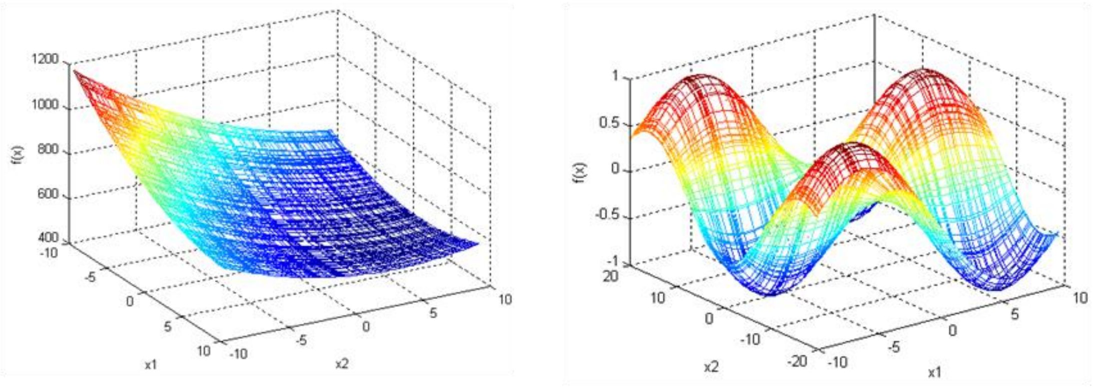
APPENDIX B

GRID PLOTS OF MATHEMATICAL FUNCTIONS

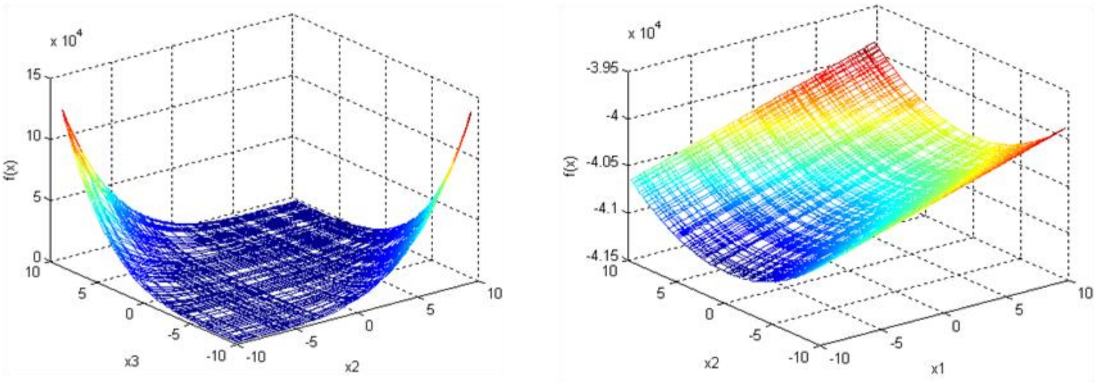
Grid plot of mathematical functions used for generation of the artificial datasets given in Table 1.



(a) P1(x_1, x_3) other x_i=3. **(b)** P2(x_1, x_3) other x_i=3.

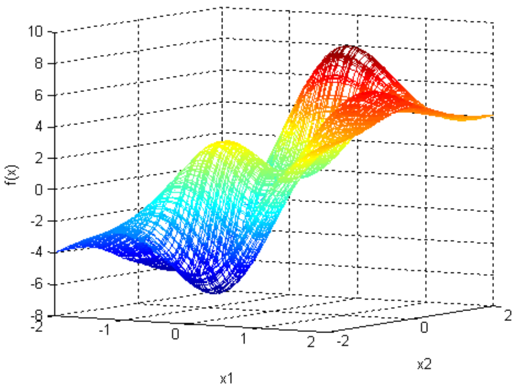


(c) P3**(**x_1**,** x_2**)** other x$_i$=4. **(d)** P4.



(e) **P$_5$(** x_2 **,** x_3 **)** other x_i =3.

(f) **P$_6$(** x_1**,** x_2 **)** other x_3 =3.



(g) P$_8$

Figure A 1. (a) Grid plot of Dataset 1 (b) Grid plot of Dataset 2 (c) Grid plot of Dataset 3 (d) Grid plot of Dataset 4 (e) Grid plot of Dataset 5 (f) Grid plot of Dataset 6 (g) Grid plot of Dataset 7

APPENDIX C

EFFECTS OF GRID SIZE AND THREDHOLD VAUE ON ARTIFICIAL AND REAL DATASETS

Table C1. Ratio=RMSE/TIME value for different grid sizes in artificial datasets

Grid Size	Data sets							
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
g/10	234.511	55.916	193.472	148.304	418.472	394.535	78.812	377.920
g5	132.066	42.688	129.288	44.832	29.840	34.856	53.413	39.875
g/2	53.302	12.016	12.827	11.220	8.847	11.272	14.426	12.633
g/5	30.924	3.835	5.584	5.862	5.401	5.844	6.630	6.894
5g/4	24.034	1.139	6.813	5.854	3.201	3.704	5.159	5.550
5g/2	12.165	0.792	2.677	1.780	2.065	3.698	2.526	2.196
5g	6.453	0.359	1.278	0.970	0.937	0.967	1.221	1.146
10g	4.301	0.249	0.859	0.543	0.569	0.516	0.762	0.727
15g	3.275	0.193	0.606	0.259	0.386	0.336	0.552	0.403
20g	2.788	0.139	0.582	0.199	0.310	0.253	0.591	0.359

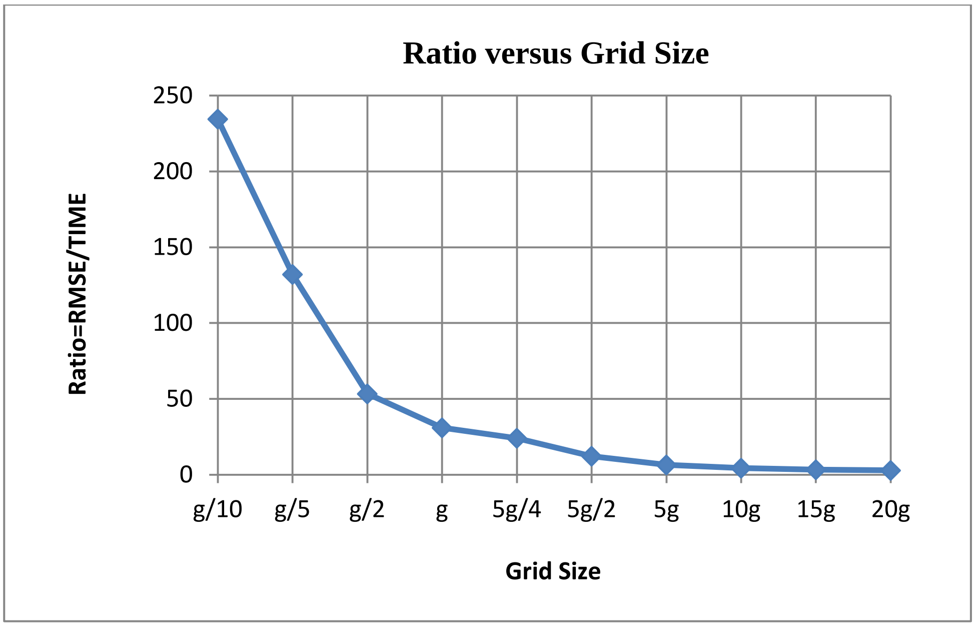


Figure C1. Graph of ratio versus grid size for Dataset 1.

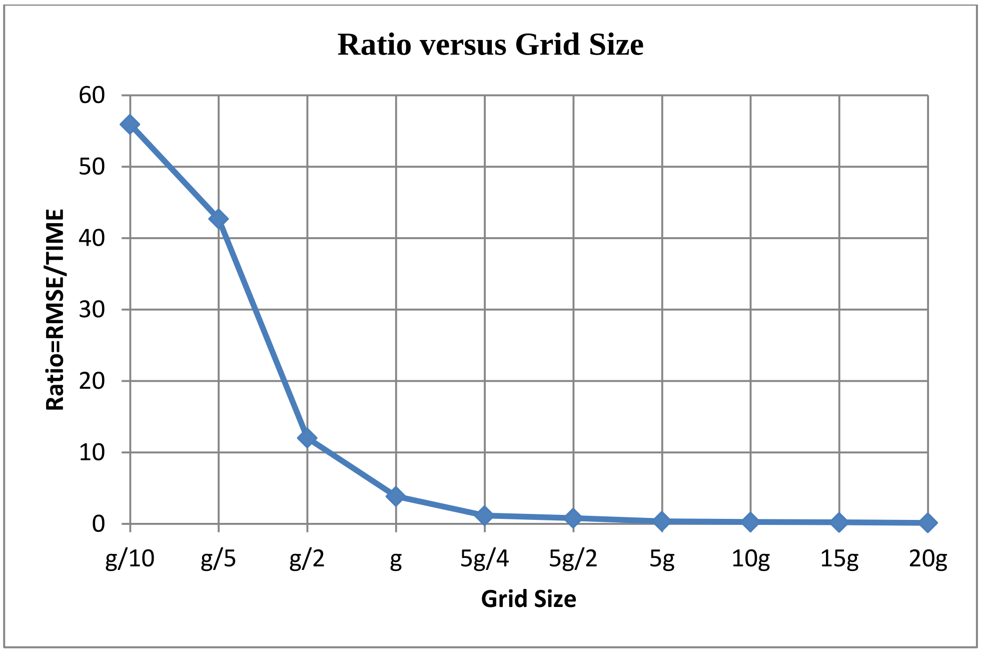


Figure C2. Graph of ratio versus grid size for Dataset 2.

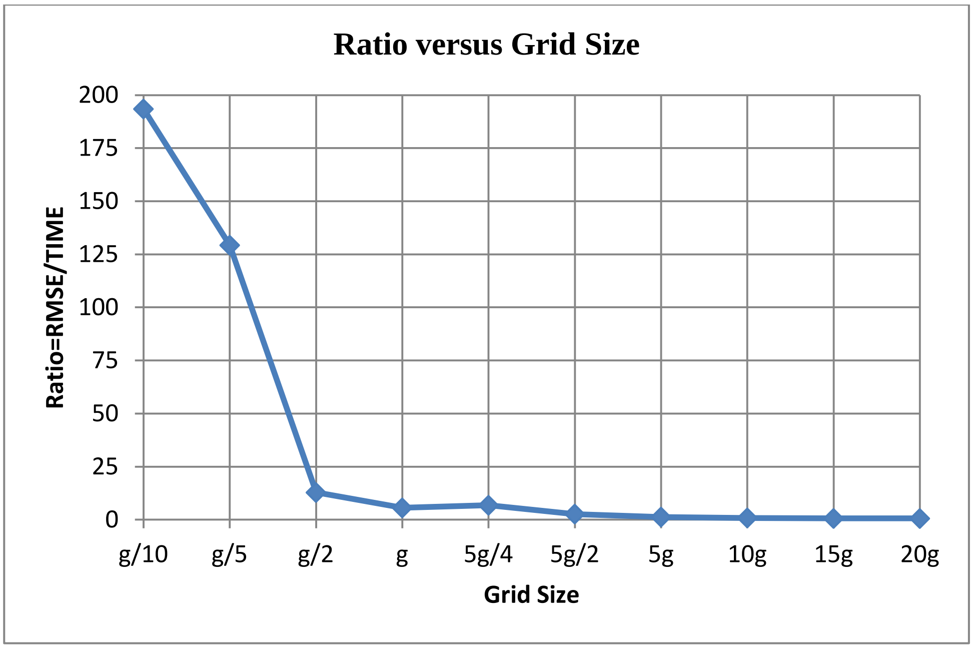


Figure C3. Graph of ratio versus grid size for Dataset 3.

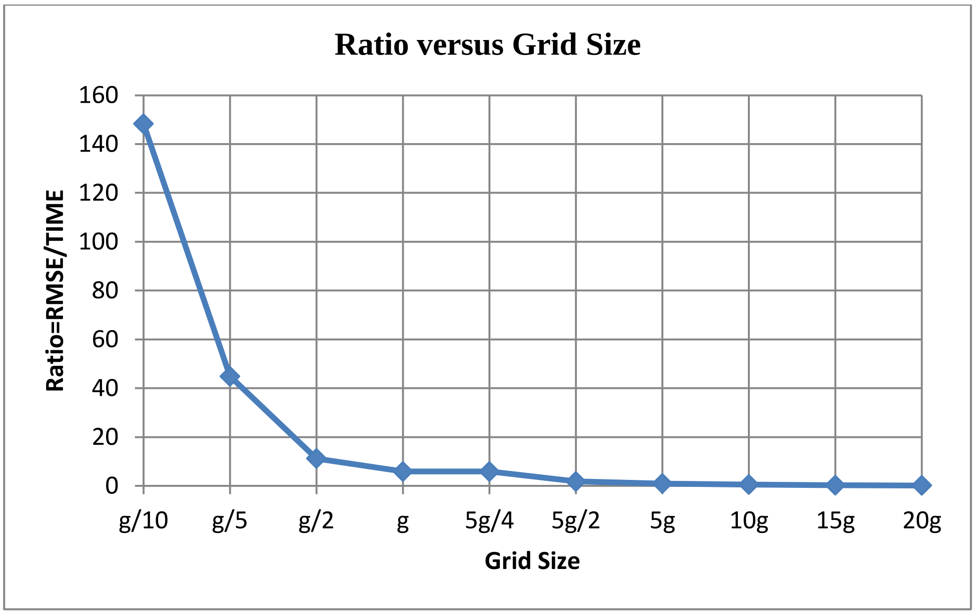


Figure C4. Graph of ratio versus grid size for Dataset 4.

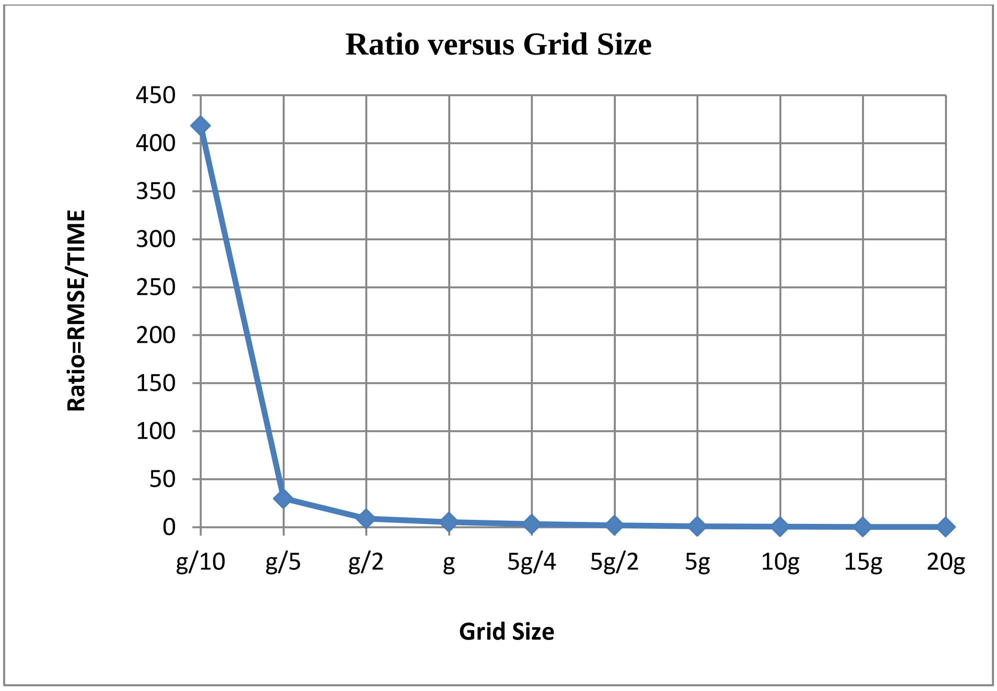


Figure C5. Graph of ratio versus grid size for Dataset 5.

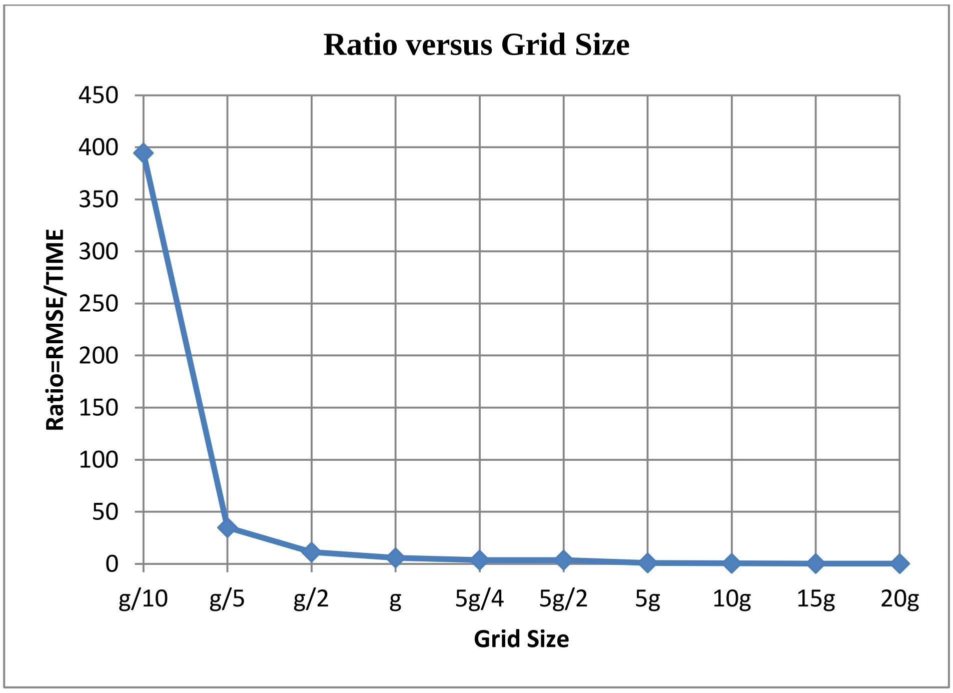


Figure C6. Graph of ratio versus grid size for Dataset 6.

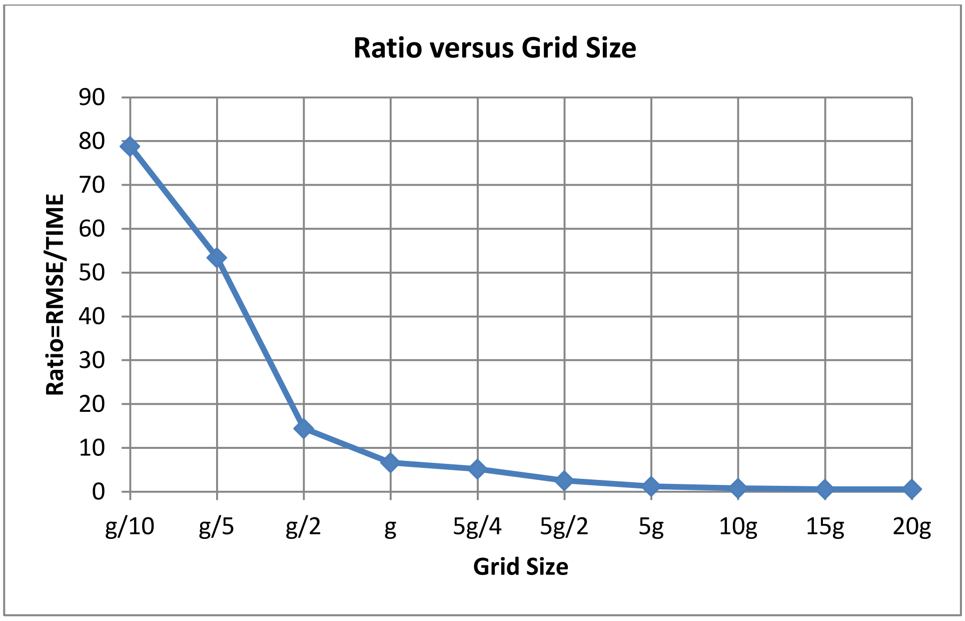


Figure C7. Graph of ratio versus grid size for Dataset 7.

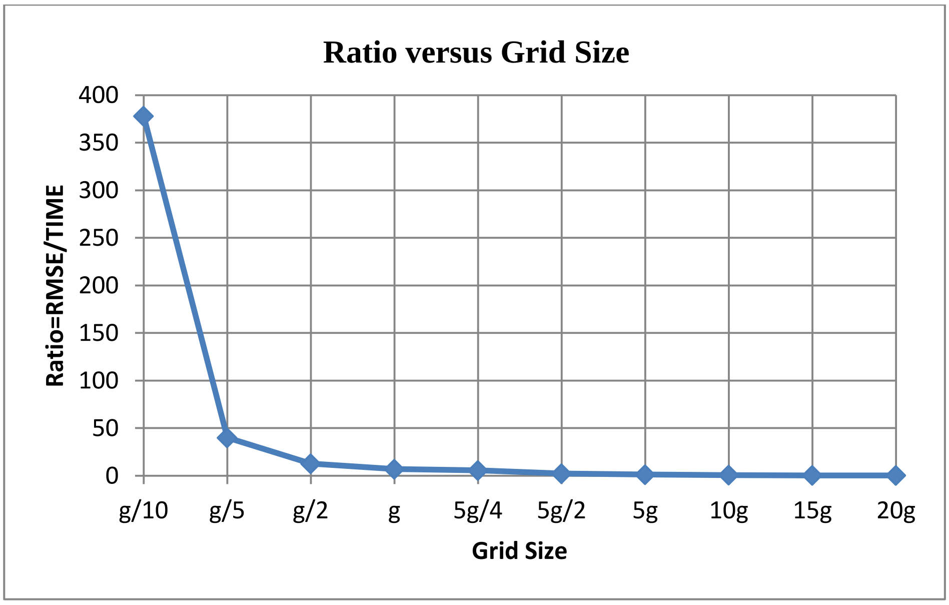


Figure C8. Graph of ratio versus grid size for Dataset 8.

Table C2. Ratio=RMSE/TIME value for different grid sizes in artificial datasets

Threshold Value	Data sets 1	2	3	4	5	6	7
1	2.395	0.074	0.133	0.056	0.296	0.271	0.666
m_u	5.422	0.198	0.416	0.072	0.591	0.611	1.519
$m_u + std$	15.847	0.481	0.775	0.405	2.261	2.226	3.420
$m_u + 2\,std$	36.043	4.831	12.125	0.689	12.162	6.464	10.962
$m_u + 2.5\,std$	61.765	13.544	20.680	6.719	11.807	23.055	31.555
$m_u + 3\,std$	104.437	24.196	89.023	27.050	15.872	42.770	31.504

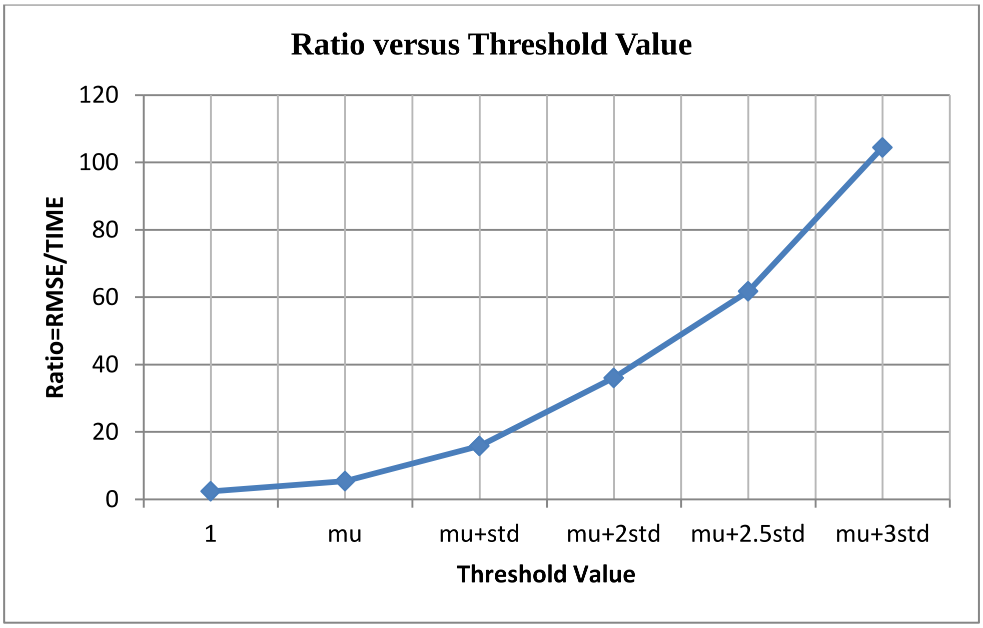


Figure C9. Graph of ratio versus threshold value for Dataset 1.

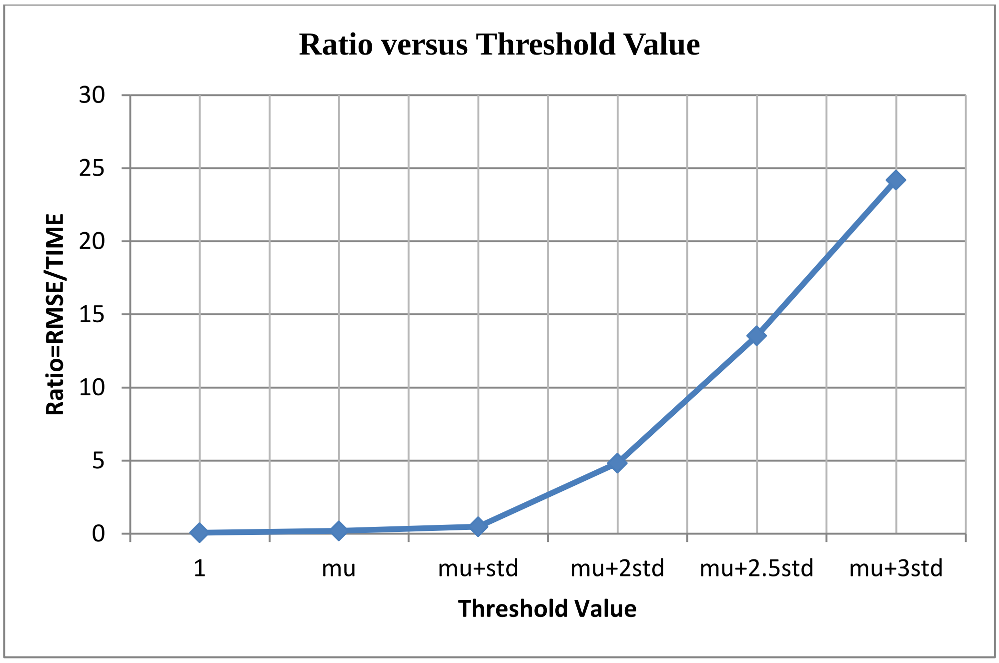


Figure C10. Graph of ratio versus threshold value for Dataset 2.

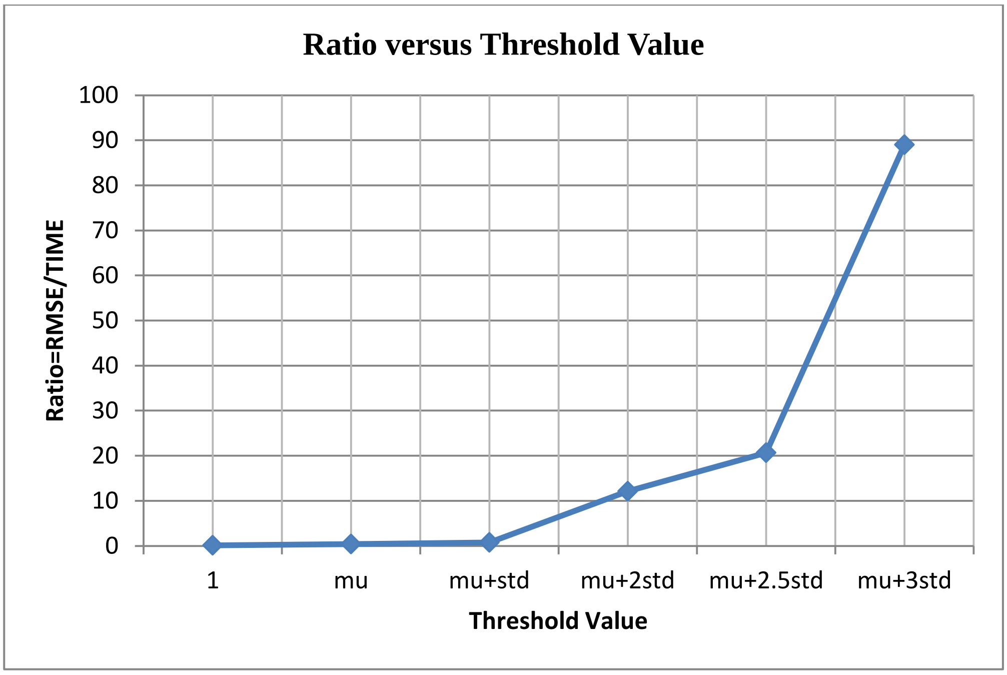


Figure C11. Graph of ratio versus threshold value for Dataset 3.

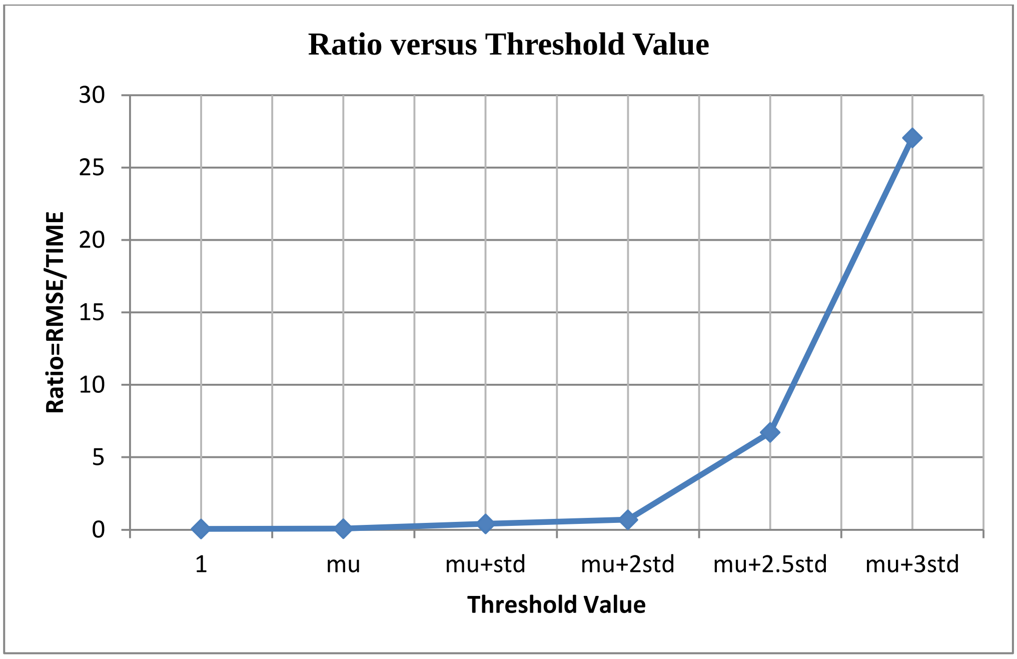


Figure C12. Graph of ratio versus threshold value for Dataset 4.

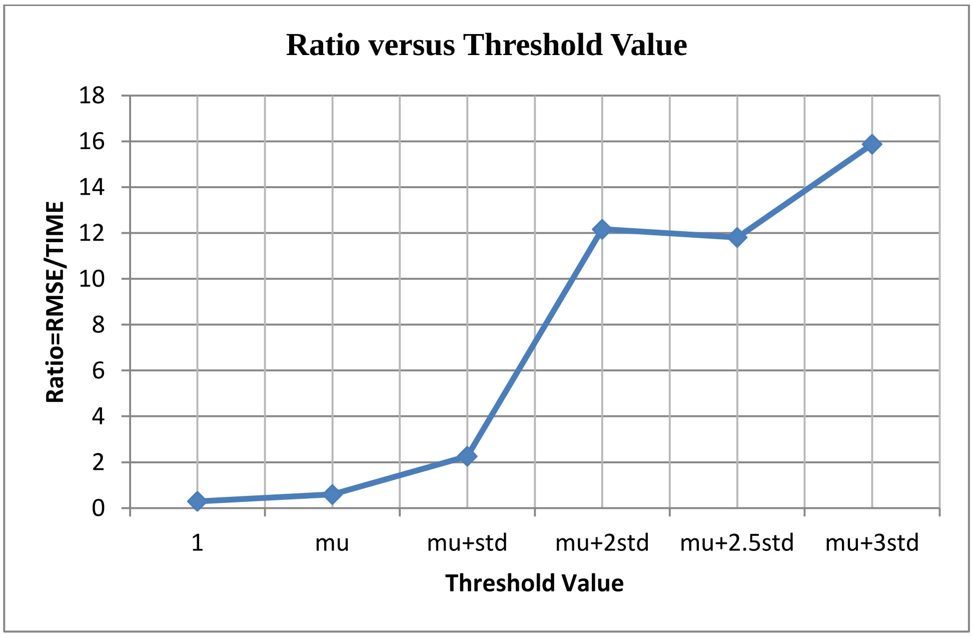


Figure C13. Graph of ratio versus threshold value for Dataset 5.

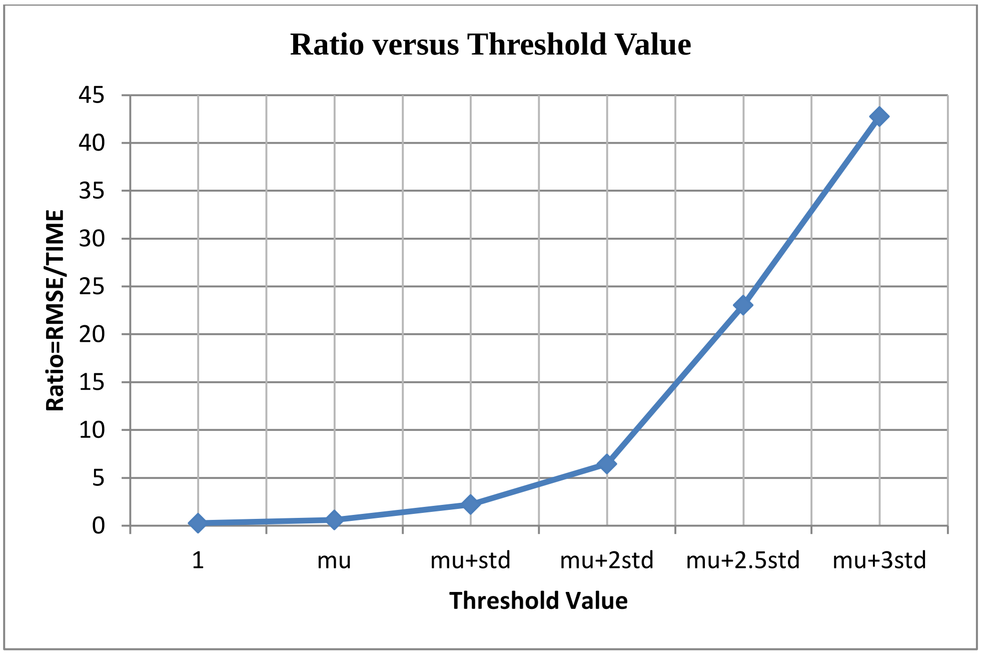


Figure C14. Graph of Ratio versus Threshold Value for Dataset 6.

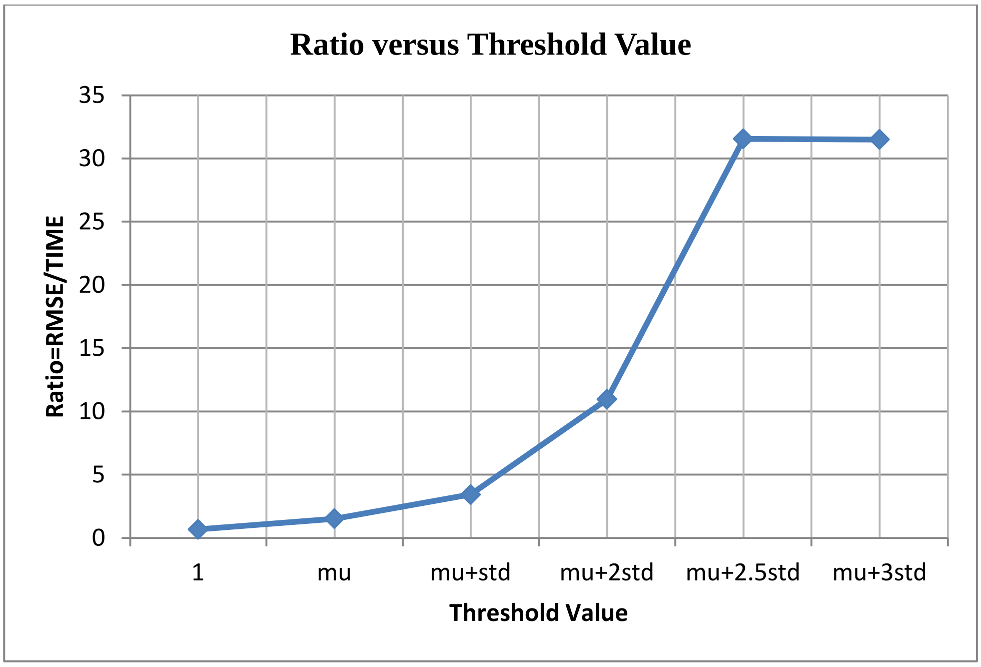


Figure C15. Graph of ratio versus threshold value for Dataset 7.

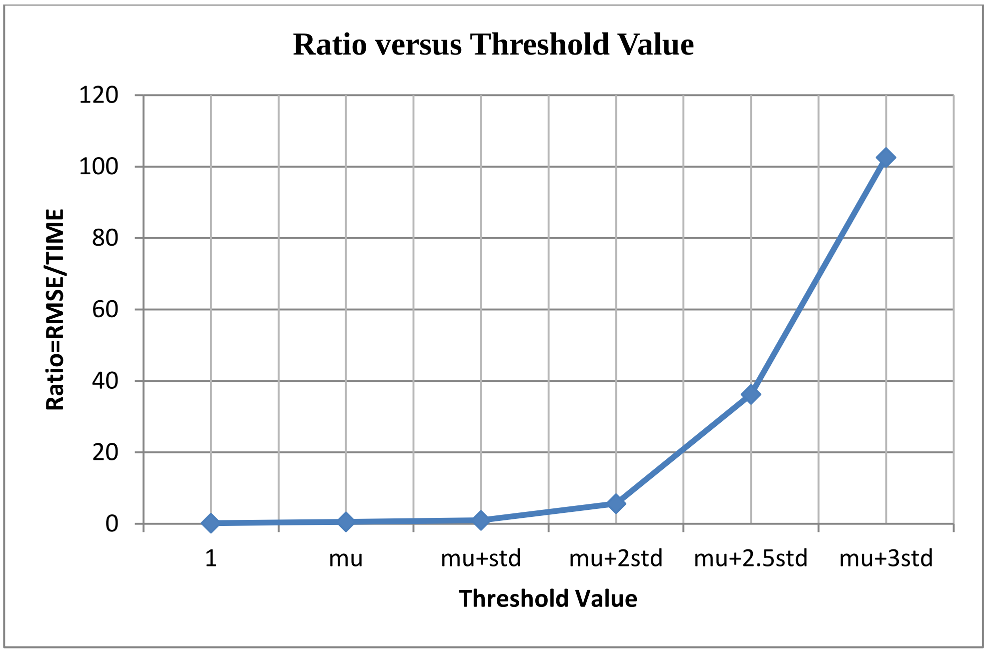


Figure C16. Graph of ratio versus threshold value for Dataset 8.

Table C3. Ratio=RMSE/TIME value for different grid sizes in real datasets

Grid Size	Data Sets					
	AutoMpg	Com.Crime	Conc.Comp	Parkinsons	PM10	Red Wine
g/10	756.460	587.648	153.489	194.726	239.026	191.580
g/5	498.185	97.102	104.352	112.206	103.290	150.800
g/2	99.874	15.062	54.201	81.292	44.141	76.340
g/5	15.046	11.672	25.090	42.799	27.643	47.018
5g/4	15.423	10.908	22.397	40.047	21.555	43.890
5g/2	10.934	7.852	13.230	24.870	14.626	31.102
5g	8.808	5.685	7.467	15.315	9.123	22.731
10g	6.985	4.494	5.444	9.124	6.184	16.990
15g	6.325	4.110	4.325	6.512	4.658	13.971
20g	5.709	3.650	3.725	5.111	4.094	12.493

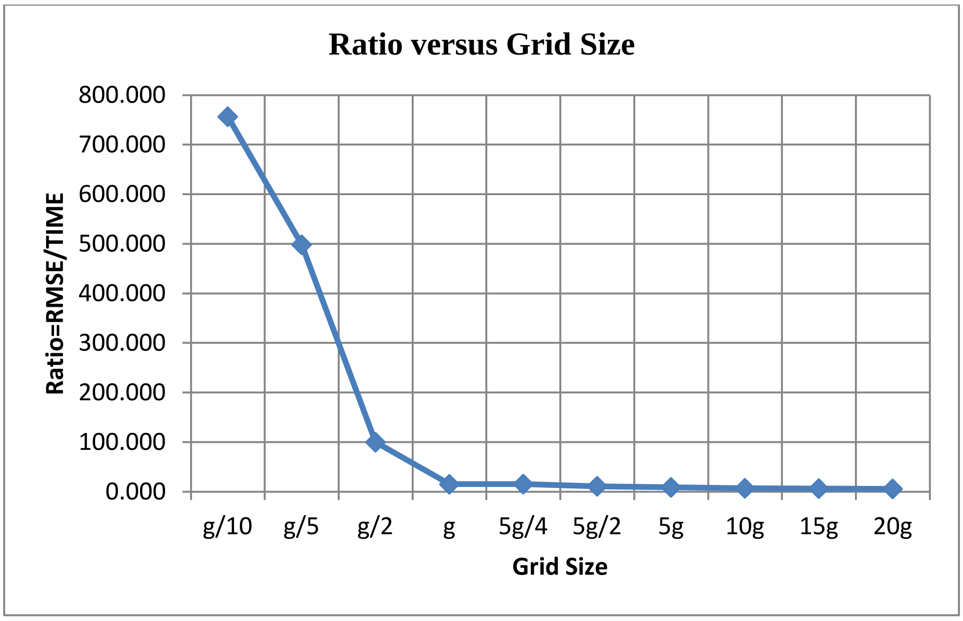


Figure C17. Graph of ratio versus threshold value for AutoMpg.

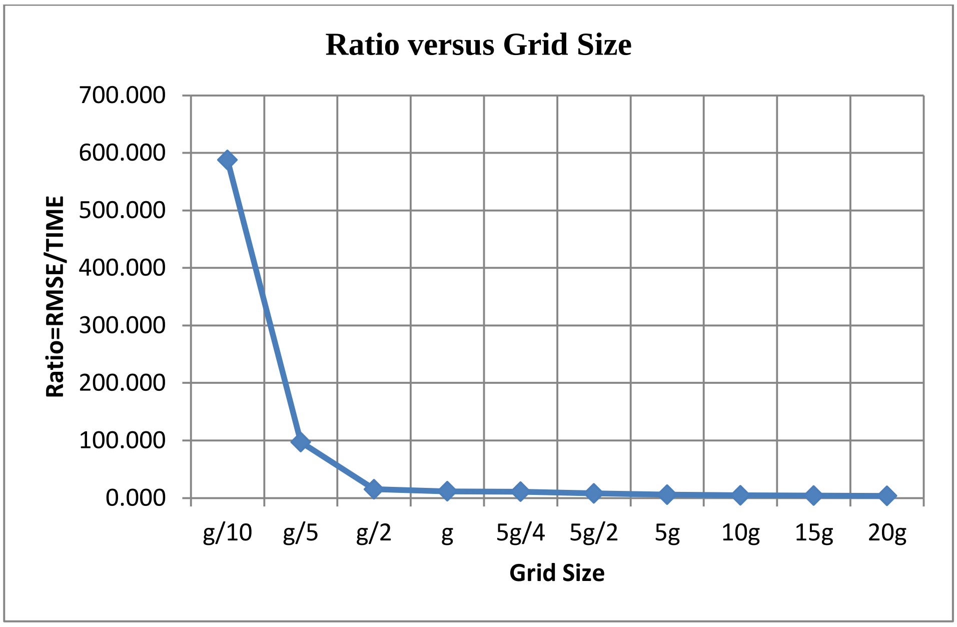


Figure C18. Graph of ratio versus grid size for Comm.Crime.

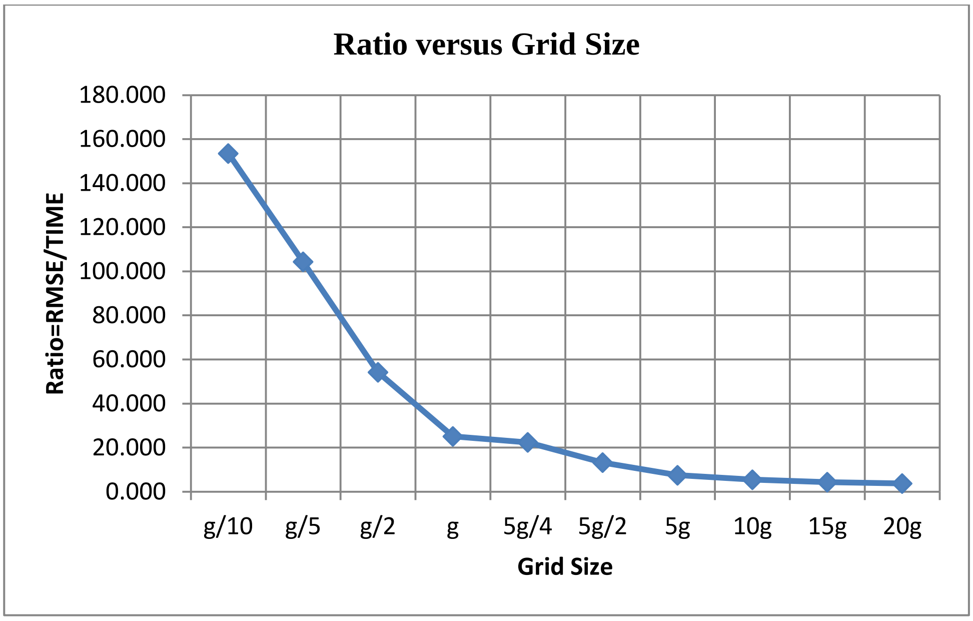


Figure C19. Graph of ratio versus grid size for Conc.Compress.

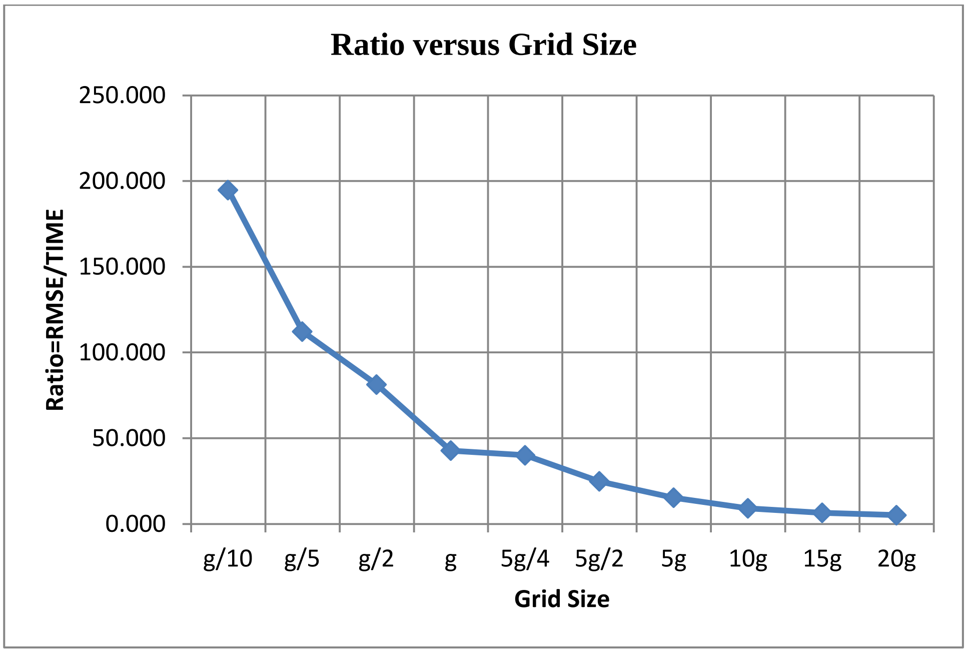


Figure C20. Graph of ratio versus grid size for Parkinsons.

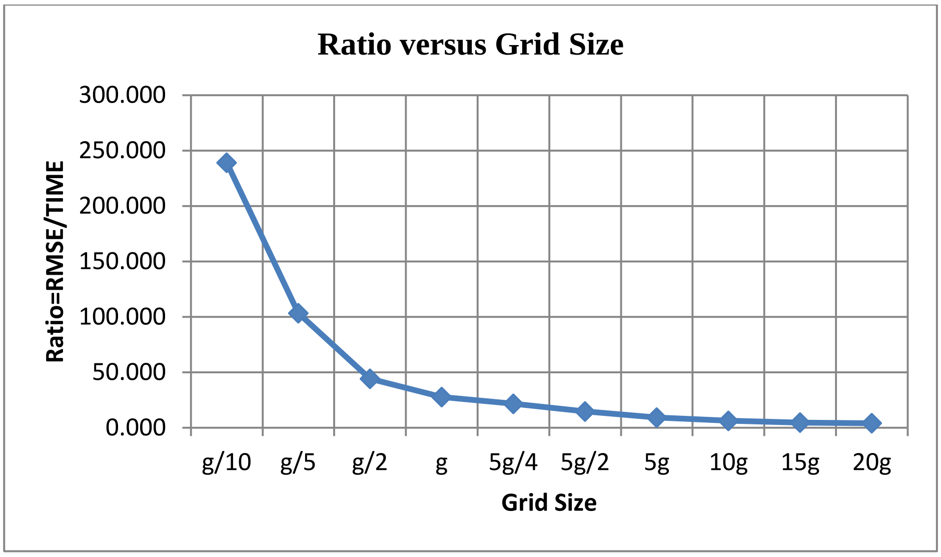


Figure C21. Graph of ratio versus grid size for PM10.

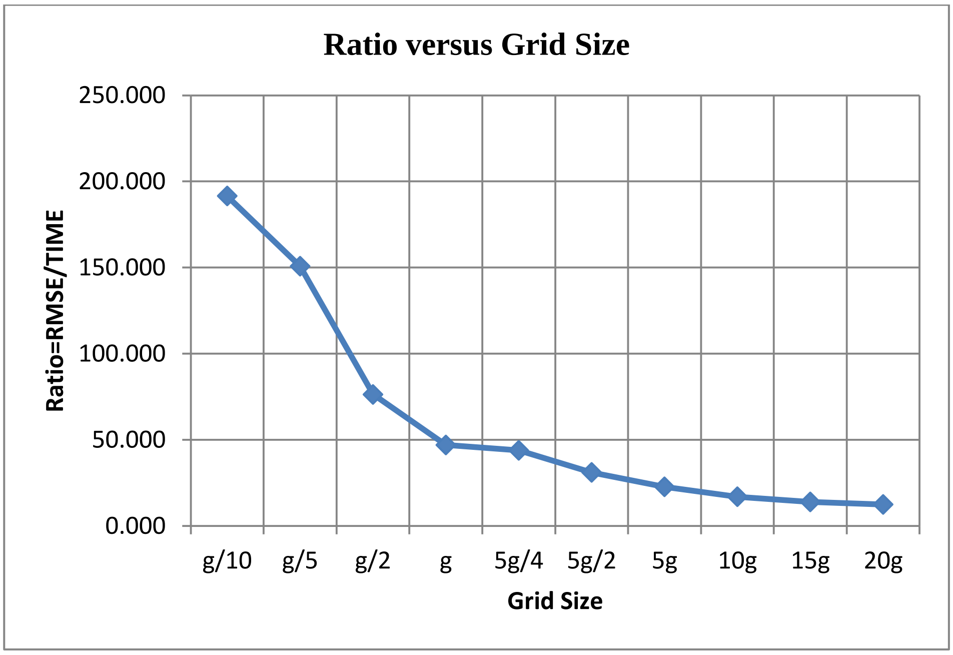


Figure C22. Graph of ratio versus grid size for Red Wine.

Table C4. Ratio=RMSE/TIME value for different grid sizes in real datasets.

Threshold Value	**Data Sets**					
	AutoMpg	**ComCrime**	**ConcComp**	**Parkinsons**	**PM10**	**Red Wine**
1	4.530	4.896	2.988	4.875	4.705	13.215
m_u	5.821	6.239	4.896	6.880	7.101	17.186
m_u+std	13.027	8.350	9.340	11.196	8.670	29.748
m_u+2std	61.635	15.042	24.619	23.265	17.681	64.606
$m_u+2.5std$	94.935	95.540	53.522	32.538		95.909

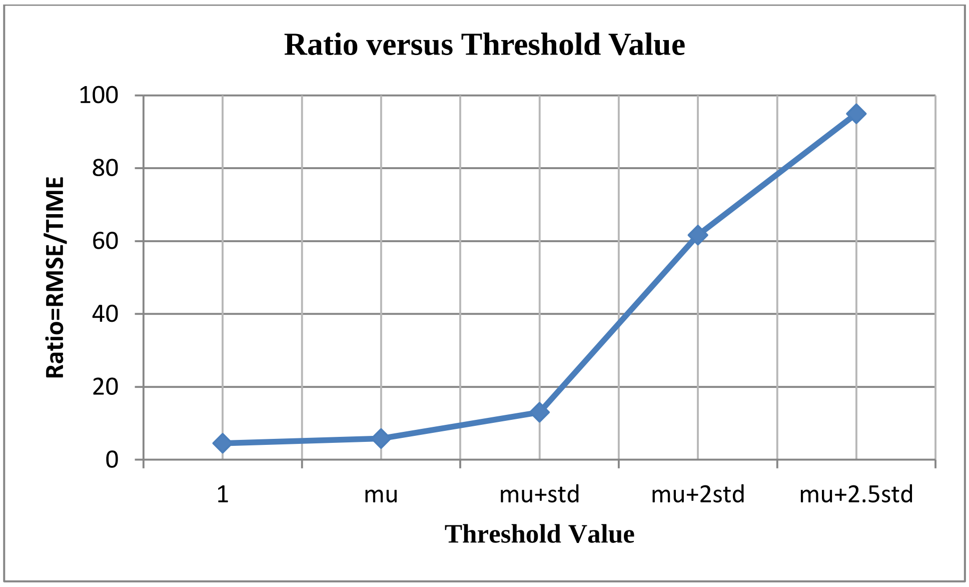


Figure C23. Graph of ratio versus threshold value for AutoMpg.

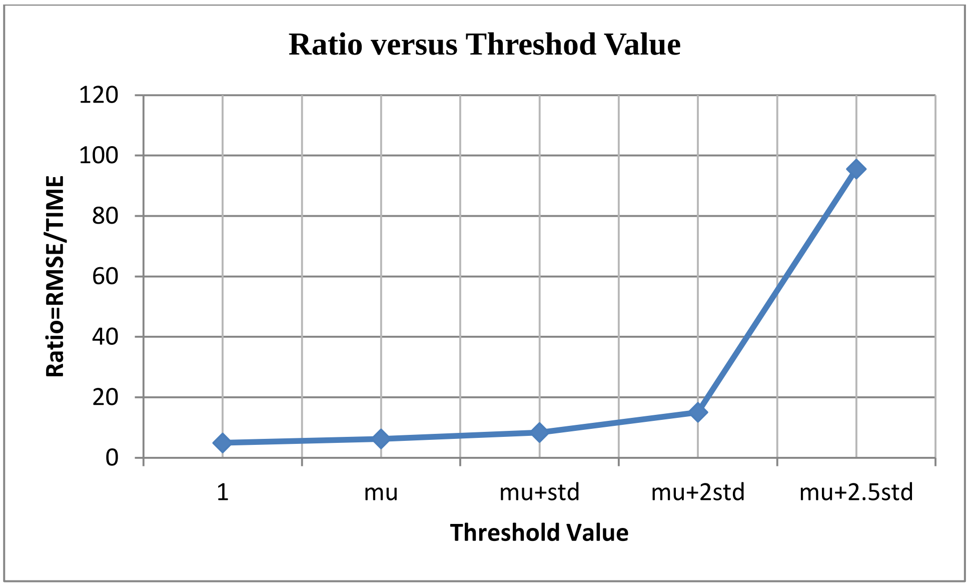


Figure C24. Graph of ratio versus threshold value for Comm.Crime.

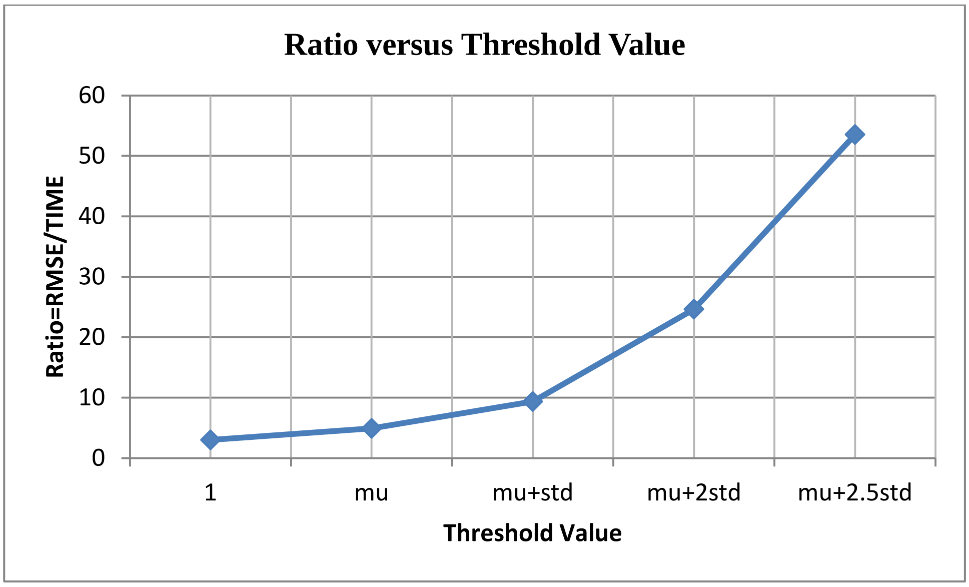


Figure C25. Graph of ratio versus threshold value for Conc.Compress.

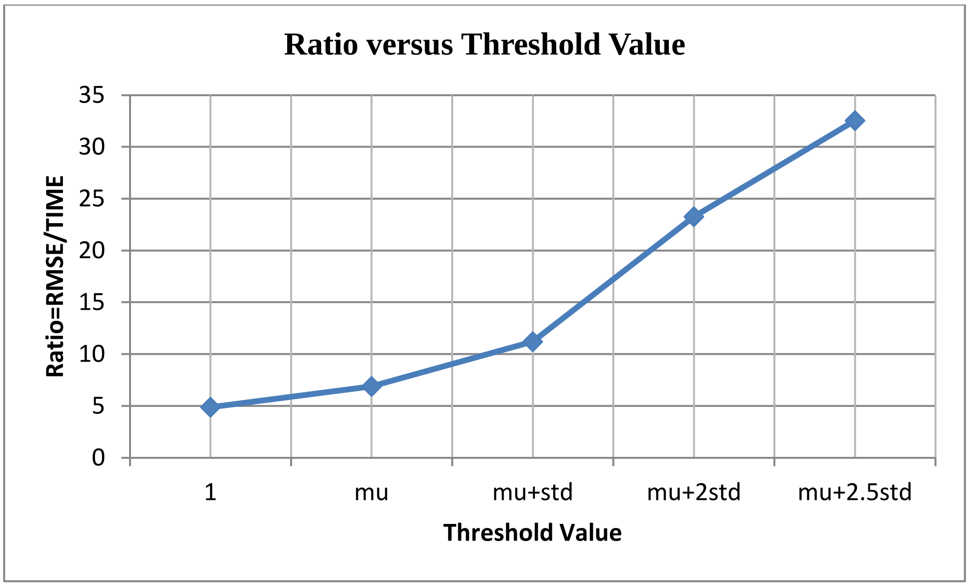


Figure C26. Graph of ratio versus threshold value for Parkinsons.

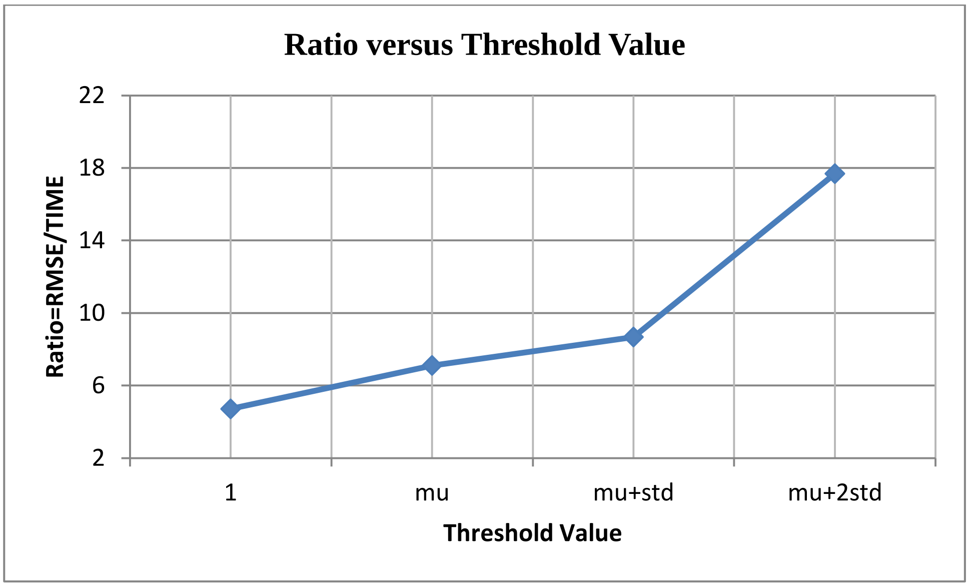


Figure C27. Graph of ratio versus threshold value for PM10.

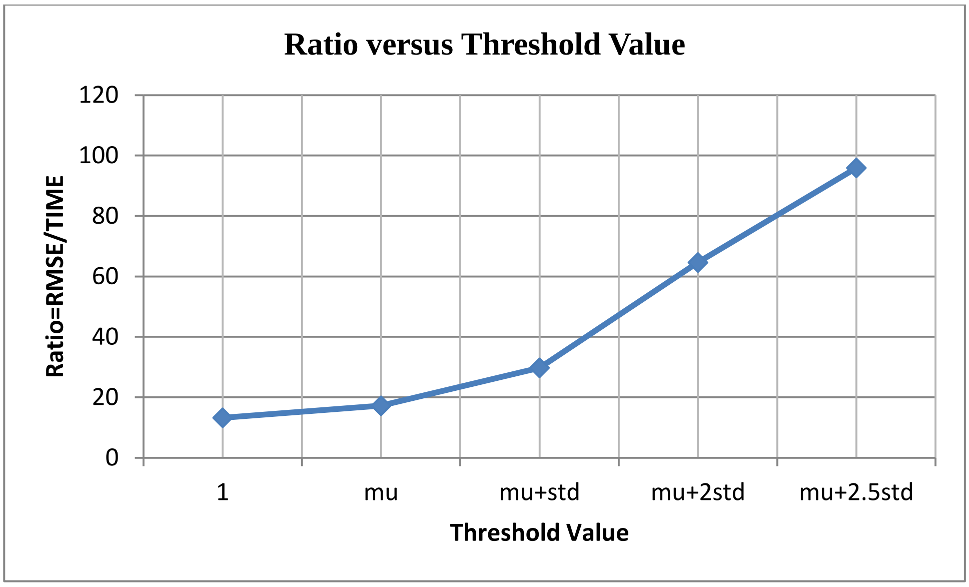


Figure C28. Graph of ratio versus threshold value for Red Wine.

APPENDIX D

COMPARISON OF PROJECTION METHODS FOR ARTIFICIAL AND REAL DATASETS

Table D1. Comparison of projection methods for artificial training data.

Datasets	Methods	BF_{final}	RMSE	Adj-R2	GCV
1	mean of *k*-nearest	101	1469.9	0.961	3851488.1
	nearest	101	1089.1*	0.979*	2114441.3*
	no projection	101	1424.5	0.963	4043347.5
2	mean of *k*-nearest	53	62532.8	0.835	4881686406.6
	nearest	41	5966.3*	0.999*	42147837.8*
	no projection	27	18941.0	0.985	400043347.5
3	mean of *k*-nearest	47	25.889	0.999	857.7
	nearest	43	28.132	0.999	990.2
	no projection	47	23.967*	0.999*	735.1*
4	mean of *k*-nearest	89	0.029	0.998	0.001
	nearest	77	0.028	0.998	0.001
	no projection	81	0.026*	0.998	0.001
5	mean of *k*-nearest	31	845.3	0.998	725504.0
	nearest	45	467.4*	0.999*	223365.1*
	no projection	41	563.2	0.999*	323648.9
6	mean of *k*-nearest	23	2.955	0.999	8.830
	nearest	23	2.892*	0.999	8.456*
	no projection	23	3.019	0.999	9.215
7	mean of *k*-nearest	101	0.032	0.991	0.002
	nearest	101	0.030*	0.992	0.002
	no projection	101	0.030*	0.992	0.002
8	mean of *k*-nearest	77	0.162	0.998	0.027
	nearest	79	0.151*	0.998	0.024*
	no projection	83	0.155	0.998	0.025

Table D2. Comparison of projection methods for test data and stabilities.

Datasets	Methods	TEST		STABILITY	
		RMSE	**Adj-R2**	**RMSE**	**Adj-R2**
1	mean of *k*-nearest	1409.5	0.967	0.959	0.993
	nearest	1126.7*	0.979*	0.967	0.999
	no projection	1417.1	0.967	0.995*	0.996
2	mean of *k*-nearest	61228.7	0.864	0.979*	0.966
	nearest	5738.1*	0.999*	0.962	1.000
	no projection	16237.5	0.990	0.857	0.994
3	mean of *k*-nearest	25.889	0.999	1.000*	1.000
	nearest	27.724	0.999	0.986	1.000
	no projection	22.717*	0.999	0.948	1.000
4	mean of *k*-nearest	0.027	0.998	0.899*	0.999
	nearest	0.028	0.998	0.987	1.000
	no projection	0.026*	0.998	0.988	1.000
5	mean of *k*-nearest	863.6	0.998	0.979	1.000
	nearest	481.9*	0.999*	0.970	1.000
	no projection	572.3	0.999	0.984*	1.000
6	mean of *k*-nearest	2.960	0.999	0.998*	1.000
	nearest	2.869*	0.999	0.992	1.000
	no projection	3.040	0.999	0.993	1.000
7	mean of k-nearest	0.033	0.991	0.995*	1.000
	nearest	0.028*	0.993	0.941	0.999
	no projection	0.029	0.993	0.948	0.999
8	mean of *k*-nearest	0.163	0.998	0.998	1.000
	nearest	0.151*	0.998	1.000*	1.000
	no projection	0.156	0.998	0.996	1.000

Table D3. Comparison of projection methods for training data of real datasets

Datasets	**Methods**	**BF$_{final}$**	**RMSE**	**Adj-R2**	**GCV**
AutoMpg	mean of *k*-nearest	101	2.193	0.897	90.979
	nearest	101	2.184	0.891	90.420
	no projection	101	2.143*	0.902*	86.885*
Com.Crime	mean of *k*-nearest	151	0.406	0.795	0.722
	nearest	151	0.409	0.792	0.734
	no projection	151	0.401*	0.800*	0.707*
Con.Comp	mean of *k*-nearest	101	0.209	0.951	0.110
	nearest	101	0.221	0.946	0.123
	no projection	101	0.206*	0.953*	0.106*
Parkinsons	mean of *k*-nearest	51	0.335	0.883	0.202
	nearest	50	0.330*	0.887*	0.193*
	no projection	51	0.337	0.882	0.205
PM10	mean of *k*-nearest	51	0.662	0.520	0.879
	nearest	51	0.656*	0.527*	0.866*
	no projection	51	0.683	0.488	0.937
Red Wine	mean of *k*-nearest	91	0.649	0.555	0.679
	nearest	91	0.645*	0.561*	0.671*
	no projection	91	0.652	0.551	0.685

Table D4. Comparison of projection methods for test data and stabilities.

Datasets	Methods	TEST		STABILITY	
		RMSE	Adj-R2	RMSE	Adj-R2
AutoMpg	mean of *k*-nearest	2.798	0.843	0.784	0.940
	nearest	2.642*	0.869*	0.827*	0.975*
	no projection	2.859	0.837	0.750	0.928
Com.Crime	mean of *k*-nearest	0.653*	0.591*	0.621*	0.743*
	nearest	0.686	0.547	0.596	0.691
	no projection	0.716	0.508	0.561	0.636
Con.Comp	mean of *k*-nearest	0.348	0.879	0.601	0.925
	nearest	0.673	0.550	0.329	0.581
	no projection	0.331*	0.891*	0.622*	0.935*
Parkinsons	mean of *k*-nearest	0.579	0.615	0.580	0.696
	nearest	0.477*	0.738*	0.692*	0.832*
	no projection	0.675	0.543	0.500	0.616
PM10	mean of *k*-nearest	0.846	0.253	0.782	0.487
	nearest	0.807*	0.320*	0.813*	0.607*
	no projection	0.850	0.246	0.804	0.503
Red Wine	mean of *k*-nearest	0.973	0.032	0.667	0.058
	nearest	0.927	0.121	0.696	0.217
	no projection	0.894*	0.184*	0.730*	0.334*

CURRICULUM VITAE

PERSONAL INFORMATION

Surname. Name	:	KARTAL KOÇ. ELÇİN
Date & Place of Birth	:	16.09.1980 - KAYSERİ
Nationality	:	Turkish (TC)
Phone	:	+90 312 210 2979
E- mail	:	kartalelcin@gmail.com

EDUCATION

Degree	Institution	Year of Graduation
M.S.	MIDDLE EAST TECHNICAL UNIVERSITY Department of Statistics	**2007**
B.S.	MIDDLE EAST TECHNICAL UNIVERSITY Department of Statistics	**2004**
High School	Ankara Milli Piyango Anatolian High School	**1999**

WORK EXPERIENCE

Year	Place	Enrollment
2007-2012	Department of Statistics. METU	Research and Teaching Assistant.
2004-2007	S.P.A.C. METU-Technopolis.	Researcher

FOREIGN LANGUAGE

English

PUBLICATIONS &PROCEEDINGS

- Determining the Climate Zones of Turkey by Center-Based Clustering Methods. Nonlinear Dynamics of Complex Systems: Applications in Physical. Biological and Financial Systems (Eds.) J.A. Tenreiro Machado. Dumitru Baleanu. Albert Luo. Springer.

- Robust Regression Metamodeling of Complex Systems: The case of Solid Rocket Motor Performance Metamodeling. Advances in Intelligent Modelling and Simulation: Simulation Tools and Applications (Eds) A. Byrski. Z.Oplatkova. Marco Carvalho. Marek Kisiel-Dorohinicki. Janusz Kacprzyk. Springer.

- Evaluating the CMARS Performance for Modeling Nonlinearities. Batmaz, İ., F. Yerlikaya-Özkurt, E. Kartal-Koç, G. Köksal and G. W. Weber. Global Conference on Power Control and Optimization PCO 2010. Queensland, Australia, February 2-4, 2010. ISBN: 978-983-44483-1-8. CDROM.

- Classification Models Based on Tanaka's Fuzzy Linear Regression Approach: The Case of Customer Satisfaction Modeling. Özer. G.. G.. Köksal. İ. Batmaz. Ö. Türker-Bayrak. T. Kılıç and E. Kartal-Koç. 1st International Fuzzy Systems Symposium Proceeding. Ankara. October 1-2. 2009. 233-240.

- Türkiye İklim Bölgelerinin Hiyerarşik Kümeleme Yöntemi ile Belirlenmesi. Kartal-Koc.E.. İyigun C.. Fahmi. M..F.. Yozgatlıgil C.. Purutcuoğlu V.. Batmaz I.. Köksal G.. Türkeş. M.. *İstatistik Araştırma Dergisi (Journal of Statistical Research).* 08:13-25. 2010. Ankara.

PROJECTS PARTICIPATED

- Determination of Climate Zones and Development of Rainfall Prediction Models for Turkey by Data Mining (supported by METU Research Fund.)

- Use and Development of Data Mining in Quality Improvement (supported by TUBITAK and METU Research Fund.)

MEMBERSHIPS

- *EUROPT* - The Continuous Optimization Working Group of EURO (http://www.iam.metu.edu.tr/EUROPT/)