J Glob Optim (2014) 60:79–102
DOI 10.1007/s10898-013-0107-5

Restructuring forward step of MARS algorithm using a new knot selection procedure based on a mapping approach

Elcin Kartal Koc · Cem Iyigun

Received: 20 December 2012 / Accepted: 31 August 2013 / Published online: 26 September 2013

Abstract In high dimensional data modeling, Multivariate Adaptive Regression Splines (MARS) is a popular nonparametric regression technique used to define the nonlinear relationship between a response variable and the predictors with the help of splines. MARS uses piecewise linear functions for local fit and apply an adaptive procedure to select the number and location of breaking points (called knots). The function estimation is basically generated via a two-stepwise procedure: forward selection and backward elimination. In the first step, a large number of local fits is obtained by selecting large number of knots via a lack-of-fit criteria; and in the latter one, the least contributing local fits or knots are removed. In conventional adaptive spline procedure, knots are selected from a set of all distinct data points that makes the forward selection procedure computationally expensive and leads to high local variance. To avoid this drawback, it is possible to restrict the knot points to a subset of data points. In this context, a new method is proposed for knot selection which bases on a mapping approach like self organizing maps. By this method, less but more representative data points are become eligible to be used as knots for function estimation in forward step of MARS. The proposed method is applied to many simulated and real datasets, and the results show that it proposes a time efficient forward step for the knot selection and model estimation without degrading the model accuracy and prediction performance.

Keywords Data mining · Multivariate adaptive regression splines (MARS) · Computational efficiency · High dimensional data reduction · Mapping · Self-organizing maps

E. K. Koc
Department of Statistics, Middle East Technical University, Ankara, Turkey
e-mail: ekartal@metu.edu.tr

C. Iyigun (✉)
Department of Industrial Engineering, Middle East Technical University, Ankara, Turkey
e-mail: iyigun@metu.edu.tr

1 Introduction

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

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

where, ε indicates the error term, and p denotes the number of predictor variables.

The main objective of the model in (1) is to approximate the function $f(\mathbf{x})$ with high accuracy and effective generalization (prediction) capability without making any assumption about the form of underlying relationships [15]. In this respect, nonparametric regression techniques have been proposed for function estimation. In the nonparametric framework, the function $f(\mathbf{x})$ is assumed to be a smooth and typically estimated using spline functions due to their good numerical properties [11,22].

One of the popular approaches in spline estimation is the regression splines. They provide a flexible model estimate by the help of piecewise functions defined on distinct intervals of predictor space, whose end points are determined by breaking points, called knots. In the literature of regression splines, there have been many studies extended in several directions including smoothing splines with locally adaptive smoothing parameters [32], adaptive regression splines with variable knots such as TURBO [17] and Multivariate Adaptive Regression Splines (MARS) [18].

MARS is a powerful nonparametric regression method for constructing flexible models by introducing truncated linear splines [23]. 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 [28], credit scoring [31], detecting disease risk [52], environmental modeling [29,30], direct response modeling [12]. The powerful performance of MARS makes it deserve to further improvements. Many studies have been performed to generalize the MARS method for extended usages and to improve its performance on model accuracy and prediction capability. For example, Hybrid Adaptive Splines (HAS) is developed by combining some features of MARS and traditional smoothing splines to obtain a hybrid smoothing procedure by [32]. POLYMARS is developed by [42] for multivariate response variables. Studies in [13,49] and [41] incorporate a bayesian approach and empirical bayes method in MARS, respectively. CMARS [48,51] modifies the MARS algorithm by introducing the penalized splines and solves it by Tikhonov regularization and conic quadratic programming. Based on CMARS, R-CMARS is proposed as the robustification of CMARS with robust optimization to decrease the estimation error in CMARS [36,37].

MARS is a flexible statistical modeling method that employs forward and backward search algorithms to identify the combination of basis functions that best fits the data. It is a non-parametric regression technique that constructs the relationship from inputs to outputs variables through spline curve [18,23]. The input space is divided into regions containing their own regression eqnarray which contains knots vectors which need to be optimized. In optimization, MARS has been used successfully to estimate the value function in stochastic dynamic programming and it can be potentially useful in many real world optimization problems where the objective functions needs to be estimated from data such as simulation optimization.

MARS is suitable for problems with high input dimensions and has been applied in stochastic dynamic programming [7] and global optimization [10]. In the literature, there are many implementations of MARS algorithm in design and analysis of computer experiments based approaches for continuous-state stochastic dynamic programming [4,5,7,44], Markov decision processes [6], and two-stage stochastic programming [38,39].

MARS also provides a flexible statistical method for high-dimensional problems involving interactions and curvature. It approximates the underlying function with generating a non-linear function by taking products of univariate functions. The interaction terms of a MARS model are transformed to piecewise linear forms, (see Sect. 2, for more details). Recently, a mixed integer programming model is proposed to optimize the MARS function and it is solved using branch and bound [34].

The algorithm (MARS) is based on non-linear optimization and /or stepwise selection. The procedure is spatially adaptive and the stepwise knot selection is necessarily suboptimal. As it is stated in [47], determining the best model over the space of knot splines is a very poorly behaved non-linear optimization problem. This study focuses on the stepwise selection part of MARS and proposes a preprocessing for knot selection using an approach of artificial intelligence based optimization.

In adaptive regression splines, the number and the location of knots have a critical importance in controlling the amount of smoothness for function estimation; however, they are unknown in advance. In many studies, knots have been selected by a model selection approach. Since each knot is considered as an additional parameter being added to the model, some model selection criteria such as Cp statistic [33], Akaike Information Criterion [1] or cross validation [9] are recommended for selecting the number of knots [3,14]. Several adaptive selection approaches for knot selection are proposed in the literature (see [17,18,32,43]). All these methods generate the model by selecting knots via a forward selection and backward elimination method. Generally, a model with too many knots is generated first, and then knots that contribute least to the overall fit are removed. In conventional adaptive procedure, it requires utilizing all distinct data points for knot selection which makes the procedure computationally expensive and increase the local variability. In order to avoid the local variability, a minimum span (MinSpan) approach was proposed to restrict the candidate knot locations with a local search around the current knot point over a specific predictor variable [18]. Another search based approach has been recently developed by [35] based on simulated annealing. All these knot selection approaches mainly search the appropriate knot points while generating the estimation model.

In this study, a new knot selection procedure is proposed for adaptive regression splines considering the overall structure of the underlying dataset in a computationally efficient way. Instead of evaluating all data points, a subset of data points is considered as the appropriate knot locations for model generation. Here, the way of sub-setting is crucial. Instead of selecting the points randomly or equally spaced, before model generation, the points are selected with considering the structure of the data

Since the adaptive regression splines tend to choose more knots in the data dense regions where the underlying true function has more structure, the way of sub-setting should serve to this behavior. To provide such a sub-setting procedure, a mapping approach similar to self-organizing maps (SOM) [27] is proposed. In this approach, the original data points are mapped to a network of nodes in a low dimensional space through a nonlinear mapping. During the mapping procedure, the relative distance between the original data points is preserved by the topological order of the nodes in the new space; so that, the original data structure can be approximated. In the proposed method, the nodes in the mapped space are considered as a reference for selecting the candidate knot locations for model building. Since the cardinality of the nodes is much smaller than the size of the original data set, this referencing avoids searching all distinct data points for knot selection. After restricting the candidate knot points, the proposed approach follows a stepwise model building strategy similar to adaptive regression splines.

The implementation of the new procedure is achieved through the forward step of MARS. We call the proposed procedure as S-FMARS, where S stands for sub-setting, and FMARS stands for the forward step of MARS algorithm. The performance of S-FMARS is evaluated via nine artificial datasets, generated using multivariate functions, and six real datasets extensively used in the literature. The S-FMARS models are compared with the ones built after the forward step of MARS in terms of time efficiency, as well as accuracy, prediction capability and stability.

The present paper is organized as follows. MARS is described briefly in Sect. 2. In Sect. 3, the proposed approach (S-FMARS) is presented. Section 4 presents the numerical results via real examples and simulated datasets according to prediction accuracy, stability and computational efficiency of the method. The paper concludes with remarks and future studies in Sect. 5.

2 Multivariate adaptive regression spline (MARS)

MARS is a popular nonparametric regression technique developed by Friedman [18] particularly for approximating nonlinear relationship within the data with the help of piecewise linear functions, the basis of which are given for a univariate variable, x as follows

$$[+(x-\tau)]_+ = \begin{cases} (x-\tau), & x > \tau \\ 0, & otherwise \end{cases}, \quad [-(x-\tau)]_+ = \begin{cases} (\tau - x), & x < \tau \\ 0, & otherwise \end{cases} \tag{2}$$

The functions in (2) are called truncated linear functions. Two functions are also called reflected pairs and are piecewise linear functions combined with a threshold value τ called knot. 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.

MARS builds a flexible model by fitting piecewise linear functions in the distinct intervals of predictor space. The boundaries of the corresponding intervals are determined by the knot points, but the number and location of knots are practically unknown. In classical spline, knot points are usually predefined or taken as equally spaced. In MARS, however, knots are determined by a search procedure.

For a given vector of predictor variables, $\mathbf{x} = (x_1, x_2, \ldots, x_p)^T$, all distinct individual data values, $x_{i,j}$, $(i = 1, \ldots, n)$ of the corresponding predictor variable $x_j (j = 1, \ldots, p)$ are considered as knot points, and introduced into the model via a reflected pair given in (2). The set of all possible reflected pairs with the corresponding knots can be expressed with set C in (3).

$$C = \{(x_j - \tau)_+,\ (\tau - x_j)_+ |\ \tau \in \{x_{1,j}, x_{2,j}, \ldots x_{n,j}\},\ j \in \{1, \ldots, p\}\}. \tag{3}$$

MARS generates its model by using the basis functions (BFs) defined over the functions in 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 set C. Therefore, the general form of BFs defined over the subvector of predictor variables, $\mathbf{x}^m$ can be defined as follows

$$\psi_m(\mathbf{x}^m) = \prod_{v=1}^{K_m} [s_{(v,m)}.(x_{j(v,m)} - \tau_{(v,m)})]_+, \tag{4}$$

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 $\{\pm 1\}$.

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, $\mathbf{x} = (x_1, \ldots, x_p)^T$ and the target variable y, the model has the form

$$y = c_0 + \sum_{m=1}^{M} c_m \psi_m(\mathbf{x}^m) + \varepsilon \ , \tag{5}$$

where c_0 is the intercept term; $\psi_m(\mathbf{x}^m)$ is the mth BF with a coefficient c_m; M is the number of BFs in the current model [17,18].

The estimates of the coefficients $(c_0, \ldots, c_m)$ in (5) are calculated by a $(M+1)$- parameter least square fit of the response y on the fixed BFs, $\psi_m(\mathbf{x}^m)$ $(m = 1, \ldots, M)$. 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 set C iteratively until a maximum number of terms specified by the user is reached within 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 via a generalized cross validation (GCV) criterion which depends on the idea of minimizing the average-squared residual of fit by considering a model complexity [18].

3 Proposed approach: S-FMARS

3.1 Motivation

In adaptive regression spline, the scope is to produce a good set of BFs (with optimal number of knots and their locations) for approximating the function $f(\mathbf{x})$ in (1) with an efficient algorithm and feasible computation time. In MARS method, the greatest computational burden is in the forward selection step because all distinct data points need to be processed for selecting the right knot points. During this process, corresponding BFs from set C in (3) are added with a hierarchical manner into the model. The model starts with an intercept term. At each successive step, a new reflected pair is introduced into the model (5) 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 (4) selected in the previous step including the product of different variables other than the current x_j $(j = 1, \ldots, p)$. 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,

$$c_{M-1}\psi_m(\mathbf{x}^m)(x_\mathrm{j} - \tau)_+ + c_M\psi_m(\mathbf{x}^m)(\tau - x_\mathrm{j})_+, \tag{6}$$

is evaluated through a lack of fit (LOF) criterion depends on the squared error, given in (7) defined over M BFs.

$$\underset{m,v,\tau}{\arg\min} \, \mathrm{LOF} \, ((\,\mathbf{y} - \hat{f}_M(\mathbf{x}))^2), \tag{7}$$

where,

$$\hat{f}_M(\mathbf{x}) = \sum_{k=0}^{M-2} \hat{c}_k \psi_k(\mathbf{x}) + \hat{c}_{M-1} \psi_m(\mathbf{x})(x_{\mathrm{j}} - \tau)_+ + \hat{c}_M \psi_m(\mathbf{x})(\tau - x_j)_+. \tag{8}$$

Namely, if the model with the estimated coefficients $(\hat{c}_0, \ldots, \hat{c}_M)$ produces 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, $x_{i,j}$ $(i = 1, \ldots, n;\ j = 1, \ldots, p)$, 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 (7) 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. 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 [17,19].

As well as $M_{\max}$, the strategy of searching knots over pn distinct data values makes the training of MARS computationally expensive. For a fixed number of observations, it is possible to decrease the computing time of the forward step by decreasing the number of candidate knot locations. In this study, a new knot reduction approach for the forward step is proposed. The method is founded on the idea of clustering (or grouping) the candidate knots (all distinct data points) by mapping them into a low dimensional space and selecting the representatives of each clusters to be used as candidate knot locations. Since the mapping procedure provides a good approximation to the distribution of the underlying data, the representatives of each cluster can be a reference to select the candidate knot locations. Due to its good approximation capability, a mapping approach similar to self-organizing maps (SOM) [27] is used.

3.2 Proposed method

The proposed algorithm transforms (maps) the original data points $\mathbf{z}_i = (\mathbf{x}_i, y_i)$ $(i = 1, \ldots, n)$, $\mathbf{x} \in R^p$ (input space) into a lower dimensional discrete map of units, called neurons or nodes. The neurons which are located with equal space on a network form a lattice structure. The size and the structure of the grid of neurons are preset in advance. Each neuron has a specific topological position in the lattice and is represented by a $(p+1)$-dimensional weight vector $\mathbf{w}_l = (w_{l,1}, \ldots, w_{l,p+1})^T$ $(l = 1, \ldots, u)$, where $(p+1)$ is equal to the dimension of the original data points and $u < n$. Each unit is fully connected to all $z_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,p}, y_i)^T$ $(i = 1, \ldots, n)$; and the corresponding weight vectors are updated while the data points are introduced.

The mapping procedure processes the data iteratively. At each iteration, one data point $\mathbf{z}_i$, $(i = 1, \ldots, n)$ from the input space is introduced into the map space, and the closest unit to the current data point (called best matching unit (BMU)), is found by using a distance measure, (i.e. Euclidean distance) between the data point and the weight vectors. Once the selected unit is found at the current iteration t, the associated weight vectors of the units within the topological neighborhood of BMU are updated with the rule

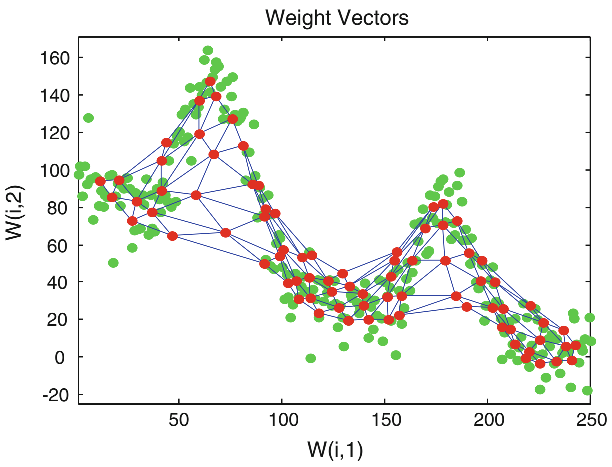


Fig. 1 Weight vectors along with original data points

$$\mathbf{w}_{l(b)}(t+1) = \mathbf{w}_{l(b)}(t) + a(t)\ h_{l(b)}(t)\ (\mathbf{x}_i(t) - \mathbf{w}_{l(b)}(t))\ (l = 1, \ldots, u), \tag{9}$$

where $\mathbf{w}_{l(b)} \in R^{p+1}$ 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 (see [24] for more details).

With the iterations, the weight vectors of BMU and neighboring neurons come close to the current data points while the weight vectors of others are left unchanged. After sufficient iterations, weight vectors tend to be located in the input space so that they can approximate the distribution of data in the sense of some minimal residual error [20]. As in Fig. 1, the weights of neurons (red points) and their corresponding neighbors (connected with blue lines) tend to shift where the data (green points) is dense while only a few weight vectors tend to place where the data is sparse. On this account, the resulting weight vectors can be used as a reference point for the candidate knot locations.

After the approximation is obtained by mapping, the associated weight vectors of BMUs are taken as reference points for candidate knot locations. The reference points can be out of the space defined by the original dataset. To guarantee that the weight vectors are in the space of original data points, weight vectors of BMUs are projected to the original dataset through a projection approach (heuristic). The nearest[1] original data point to the weight vectors $\mathbf{w}_l = (w_{l,1}, \ldots, w_{l,p+1})$ $(l = 1, \ldots, u)$ are selected as the best candidate knot locations, and represented by $\tilde{\mathbf{z}}_s = (\tilde{x}_{s,1}, \ldots, \tilde{x}_{s,p}, \tilde{y}_s)$ $(s = 1, \ldots, S)$. Once selecting the candidate knot locations, the model estimation is constructed by defining the piecewise linear fits over the values of each predictor in the corresponding candidate knot locations, $\tilde{x}_{s,j}$ $(s = 1, \ldots, S;\ j = 1, \ldots, p)$, where $S \leq n$. The new set of truncated linear functions is now represented with a set similar to (3) as

$$D = \{(x_j - \tau^*)_+,\ (\tau^* - x_j)_+ |\ \tau^* \in \{\tilde{x}_{s,j}\}\ ,\ j = \{1, \ldots, p\ ,\ s = \{1, \ldots, S\}\ \}. \tag{10}$$

The significant BFs from set D are selected with the corresponding knot points using the criterion in (7), and then a model is built with the chosen BFs.

[1] Here, for simplicity, the weight vectors associated with BMUs are projected to the nearest original data point. But other projection approaches may also be used.

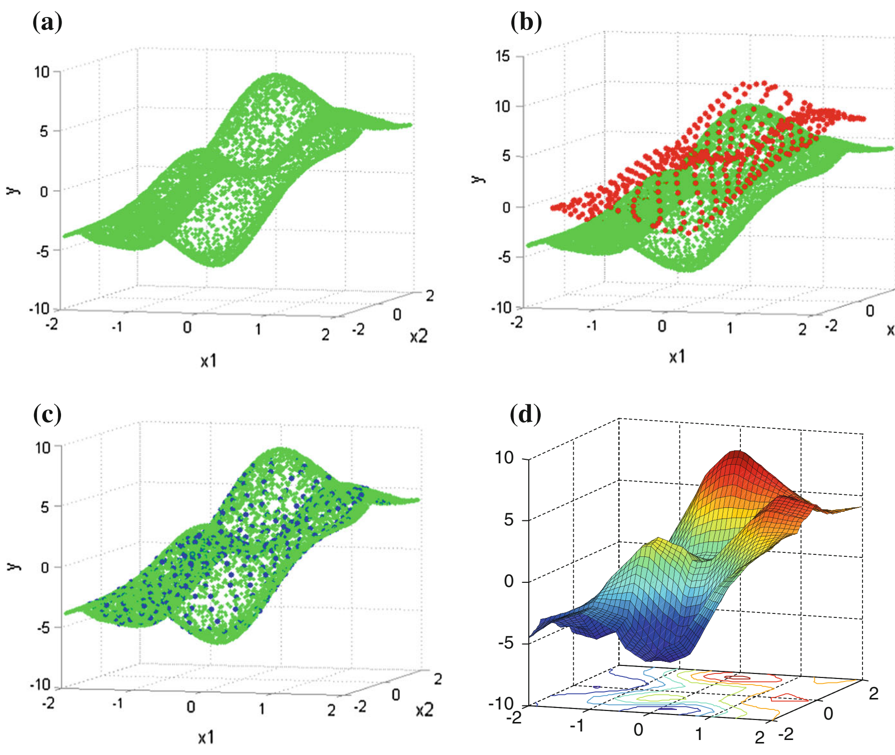


Fig. 2 **a** Original data points. **b** Weight vectors of selected BMUs and original data points. **c** Nearest data points to the map space in the original data points. **d** S-MARS model built using the knot locations in **c**

The steps of S-FMARS are illustrated on an example data in Fig. 2. Figure 2a shows the original data. The reference map space and the nearest data points to the reference map space forming the candidate knot locations are presented in Fig. 2b, c, respectively. The model built with the BFs using the candidate knots of Fig. 2c is given in Fig. 2d.

3.3 Parameters of S-FMARS

3.3.1 Grid size

S-FMARS starts with a grid topology that can be hexagonal or a rectangular. The grid size represents the dimension of a lattice in terms of total number of neurons or sidelengths [46]. Grid size has an important effect on the mapping quality; so the approximation capability. If the map space has a large number of units, it approximates the underlying data distribution better than the one with a small number of map units. However, the map space with a large number of units produces a large subset of candidate knot points and leads to longer computational time for model building. Therefore, the size of a map space is a tradeoff between a good approximation (both in mapping and modeling) and less computing time for model building. The size of the grid can be either specified by the user, or can be defined heuristically. In this study, a heuristic $g_s = 5\sqrt{n}$ introduced by [46] is used for adequate approximation of the original data points, where n represents the number of original data points.

3.3.2 Threshold value

Once the original data is mapped on to the grid of neurons, some units are either not selected as a BMU or are selected by a few data points. The number of points linked to a BMU is called as hit. In the map space, the neurons with large number of data points represent the dense regions of data, while the ones with small number of data points represent the sparse regions or outliers. After the approximation is obtained by the mapping, it is possible to eliminate the BMUs which have a small number of hits by setting a threshold value ($\tilde{t}$). The neurons taking zero hits are naturally disregarded. The threshold value for hits can be evaluates as

$$\tilde{t} = n/u, \tag{11}$$

where u is the number of neurons in the map.

Consequently, setting a threshold value for the number of hits can reduce the number of candidate knot locations, and automatically eliminate the outliers before the knot selection. So that, the CPU time of method can be decreased further.

The pseudo code of S-FMARS algorithm is given in Fig. 3.

4 Numerical study and findings

The proposed algorithm 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 aimed to handle the computational burden of adaptive regression spline mainly caused by the forward step. Since the mapping approach affects the forward step primarily, the new algorithm is compared with the forward step of MARS (called FMARS in this paper). The backward elimination step; however, is not considered in this study.

The experiments are conducted on nine artificial and seven 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. The first six datasets are generated from the functions originally given by [26]. (See "Appendix A" and "Appendix B" for the problem descriptions and grid plots, respectively). Dataset 7 is the robot arm example used by [19], and Dataset 8 is taken from the MATLAB user's quide (2010). Dataset 9 is generated with some noise using the *sinus* function. The last seven datasets refer real life problems (see "Appendix D" for data description), and six of them are originally taken from the UCI repository [16].

FMARS model is generated by ARESLab [25] toolbox written entirely in MATLAB$^{(R)}$ (2010). This toolbox implements the main functionality of MARS technique described by Friedman [18]. S-FMARS model is also developed by MATLAB with the assistance of SOM Toolbox [46] and ARESLab Toolbox.[2]

The performance of both methods is evaluated and compared with respect to accuracy, stability and efficiency criteria. To evaluate the accuracy of models for training data, Root Mean Square Error (RMSE), Adjusted-Multiple Coefficient of Determination (Adj-R^2) and Generalized Cross Validation (GCV) measures are used. The explanations of measures are given in "Appendix C". 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

[2] The code is available on request from the first author.

```
Step 0: Initilization
1 input: a set of data vectors $\mathbf{z}_i = (\mathbf{x}_i, y_i)$ $(i = 1,...,n)$, a threshold value $\tilde{t}$ ; grid
        size $g$.
2 lattice: a grid with a specified size and a set of weight vectors, $\mathbf{w}_l$ $(l = 1,...,u)$.
Step 1: Mapping
3 begin for mapping
4    initialize each weight vector $\mathbf{w}_l$.
5    repeat
6          select one data vector, $\mathbf{z}_i = (\mathbf{x}_i, y_i)$.
7          find the BMU such that $\mathbf{w}_b = \arg\min_{l=1,...u} \{d(\mathbf{z}_i, \mathbf{w}_l)\}$.
8          for all weight vectors of neighboring neurons, $w_{l(b)}$, do
9              $\mathbf{w}_{l(b)}(t+1) = \mathbf{w}_{l(b)}(t) + a(t)\, h_{l(b)}(t)(\mathbf{z}_i(t) - \mathbf{w}_{l(b)}(t))$
10     until the termination condition holds (until a specified number of
           training epochs).
11 end
Step 2: Selection of BMUs
12 select the BMUs whose number of hits is greater than a specified threshold value, $\tilde{t}$ .
       The corresponding weight vectors are denoted as $\mathbf{w}_s$ $(s = 1,...,S)$.
Step 3: Projection
13 project the weight vector of the selected BMUs, $\mathbf{w}_s$ $(s = 1,...,S)$ to the nearest data
       point $\tilde{\mathbf{z}}_s = (\tilde{\mathbf{x}}_s, \tilde{y}_s)$ such that $d(\mathbf{w}_s, \tilde{\mathbf{z}}_s) = \arg\min_{i=1,...n} \{ d(\mathbf{w}_s, \mathbf{z}_i)\}$.
Step 4: Knot selection and model building
14 begin model building with $B_1(\mathbf{x}) = 1$.
15 $M = 2$
16    while $M < M_{\max}$
17       for $m = 1$ to $M - 1$ do:
18          for $j \notin \{j(k,m) | 1 \le k \le K_m\}$,
19             for $\tau^* \in \{\tilde{x}_{s,j} \mid B_m(\mathbf{x}) > 0\}$,
20 $g \leftarrow \sum_{k=0}^{M-2} a_k \psi_k(\mathbf{x}) + a_{M-1}\psi_m(\mathbf{x})(x_j - \tau^*)_+ + a_M \psi_m(\mathbf{x})\tau^* - x_j(\tau^* - x_j)_+$,
21                $LOF \leftarrow \min_{a_1,...,a_{M+1}} LOF(g)$.
22             end
23          end
24       end
25    end
```

Fig. 3 Steps of S-FMARS

accuracy of the models. RMSE and Adj-R^2 measures are used to examine the prediction performances. Furthermore, in order to measure the change in the performance of methods between the training and test datasets, stabilities of measures are calculated (see “Appendix C”). Test datasets are also generated using the functions in “Appendix B” with the same size of training datasets. Finally, the efficiency of each method is measured by the computational run times (CPU time) in seconds on the same platform (Intel Core2 Duo CPU T7250@2.00 GHz 2.00 GB RAM).

In the comparison study, the significance of differences between the performance measures of two methods is tested via one-sample sign test [21]. It is a nonparametric test, which makes little assumptions about the nature of underlying distributions. Generally, it is used as an alternative to one-sample t test and Wilcoxon signed-rank test, respectively when the normality assumption is violated and population distribution is not assumed to be symmetric.

Table 1 Features of the datasets

Datasets	Sample size (n)	# of inputs (p)	Nonlinearity	Noisy behavior
1	1,000	7	High	No
2	1,000	5	Low	No
3	1,000	10	Low	No
4	10,000	2	High	No
5	10,000	3	High	No
6	10,000	3	Low	No
7	1,000	5	High	No
8	10,000	2	High	No
9	100	1	Low	Yes
Parkinson's	578	21	–	–
Red Wine	1, 599	11	–	–
Com. Crime	879	24	–	–
Conc. Comp.	1, 030	8	–	–
PM10	500	7	–	–
Auto Mpg	398	7	–	–
Energy load	1,730	5	–	–

In this study, one-sample sign test is interpreted for $\alpha = 0.05$ significance level for all comparison studies.

4.1 Artificial datasets

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

The number of BFs in the final model denotes the complexity. For almost all data sets, two models produce model with similar complexities. The accuracy measures calculated for each method seem close to each other (see, Table 2). For datasets 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 GCV criterion. 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 are compared via the RMSE and Adj-R^2 measures as given in Table 3. It can be indicated that the prediction performance of S-FMARS is slightly better than that of FMARS for five datasets and more stable than FMARS for four data sets.

When the methods are compared with respect to time efficiency via CPU times (given in the last column of Table 2), it is seen that S-FMARS is much more efficient than FMARS for all datasets. It provides at least 83 % decrease in CPU time. In addition, S-FMARS achieve this reduction in time without losing much in accuracy and prediction capability. The

Table 2 Performance results of FMARS and S-FMARS on the train data

Datasets	Methods	Int.	BF_{final}	RMSE	Adj-R^2	GCV	CPU time (s)	Decrease in time (%)
1	FMARS	4	100	1,062.9*	0.977*	2,013,650*	6,210.9	95
	S-FMARS		100	1,089.1	0.976	2,114,441	311.8*	
2	FMARS	1	43	4,271.9*	0.999	21,796,917*	44.1	83
	S-FMARS		43	4,359.8	0.999	22,703,916	7.5*	
3	FMARS	2	47	24.1	0.999	740.5	1,934.5	88
	S-FMARS		47	24.0*	0.999	740.1*	231.1*	
4	FMARS	2	77	0.026	0.998	0.001	16,904.8	94
	S-FMARS		81	0.025*	0.998	0.001	999.7*	
5	FMARS	2	43	513.3*	0.999	269,180*	9,756.7	95
	S-FMARS		45	514.2	0.999	270,354	516.8*	
6	FMARS	2	23	2.969	0.999	8.913	3,521.9	99
	S-FMARS		23	2.892*	0.999	8.456*	51.0*	
7	FMARS	4	100	0.028*	0.992	0.001*	3,311.4	94
	S-FMARS		100	0.030	0.991*	0.002	186.7*	
8	FMARS	2	81	0.166	0.998	0.029	93,483.0	99
	S-FMARS		78	0.151*	0.998	0.024*	484.7*	

* Better performance

Table 3 Performance results of FMARS and S-FMARS on the test data and stability

Datasets	TEST				STABILITY			
	RMSE		Adj-R^2		RMSE		Adj-R^2	
	FMARS	S-FMARS	FMARS	S-FMARS	FMARS	S-FMARS	FMARS	S-FMARS
1	1,120.7*	1,126.7	0.959	0.959	0.948	0.967*	0.982	0.983*
2	3,849.4	3,834.1*	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

* Better performance

differences between the measures obtained in the training and test datasets and also differences between the stabilities of measures are found statistically insignificant by one-sample sign test (p value >0.05), see Table 3.

4.1.1 *Effects of maximum number of BFs (M_{max}) and sample size (n) on 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}) (described in Sect. 3.1). To observe the

Table 4 Performance results of FMARS and S-FMARS for different n and M_{max}

n	M_{max}	RMSE		Adj-R^2		GCV		CPU Time (sec.)		Decrease in time (%)
		FMARS	S-FMARS	FMARS	S-FMARS	FMARS	S-FMARS	FMARS	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
1,600	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
3,200	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	1,088.6	35.6*	97
6,400	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	1,609.1	37.5*	98
	60	1.013*	1.014	0.901	0.901	1.077*	1.079	6,228.6	114.8*	98

* Better performance

performance of both methods for different n values, five different datasets with $n = 400$, 800, 1,600, 3,200 and 6,400 are generated using the function of Dataset 8 (see "Appendix A"). Additionally, for each case, three different M_{max} values 20, 40 and 60 are set and models are compared.

The results presented in Table 4 show that as n and M_{max} increase, the CPU time required for model building drastically increases for both methods. Moreover, the computing time of S-FMARS is less than FMARS for all sample size and M_{max} combinations. Especially for large datasets, the differences in CPU times are more drastic than for small ones. While the decrease in CPU time is 70 % for the dataset with the smallest n and M_{max} value, it is 98 % for the largest dataset with large number of M_{max}. While the computation time decreases in S-FMARS significantly, 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).

4.1.2 Effect of interaction terms (Int) on CPU time

In general, interaction models require more CPU time than additive models. The effect of interaction terms on computing time of FMARS and S-FMARS is tested through an example using Dataset 7, which is a robot arm example used in [19]. The best model describing the nonlinear relationship between the response and predictor variables on this data is an interaction model. The CPU times of the model construction with different degree of interaction terms are observed and compared in Table 5, and summarized in Fig. 4. As the number of interaction terms increases, the CPU times of both methods increases correspondingly, but S-FMARS is much more time efficient than FMARS. It provides approximately the same percentage decrease in CPU time for all cases of interaction terms. Similarly, the prediction

Table 5 Performance results of FMARS and S-FMARS for different interaction term

# int.	RMSE		Adj-R^2		GCV		CPU time (s)		Decrease in CPU time (%)
	FMARS	S-FMARS	FMARS	S-FMARS	FMARS	S-FMARS	FMARS	S-FMARS	
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	1,323.4	61.6*	95
4	0.063	0.062*	0.999	0.999	0.004	0.003	1,448.7	73.1*	95
5	0.063	0.062*	0.999	0.999	0.004	0.003	1,670.9	70.5*	96

* Better performance

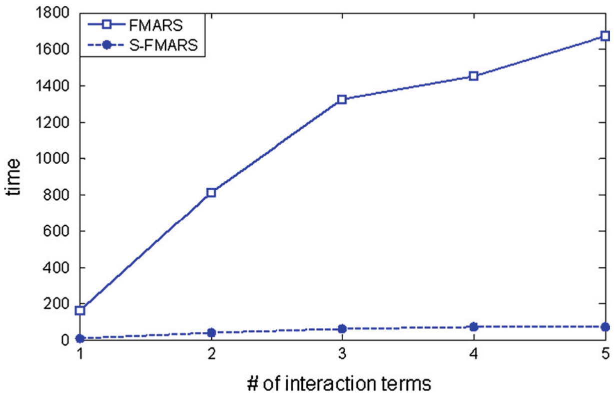


Fig. 4 CPU time versus number of interaction term

performance of both methods is not statistically different in terms of accuracy and complexity measures.

4.2 Real datasets

Both methods are also compared on real life datasets mostly taken from UCI repository [16]. The first six datasets in Table 1 are standardized and the missing values are deleted before the model construction. Energy Load data is not standardized due to its time dependent structure. To compare and validate the methods, 3-times replicated and threefold cross validation approach is used. Here, the data is randomly divided into three sub-samples (folds) and this process is repeated three times. The averages of ninefolds obtained for each measure are given in Table 6. The methods are run for different M_{max} values given in the 2nd column of Table 6.

The results given in Table 6 show that F-MARS produces slightly more accurate models than S-FMARS. However, according to one-sample sign test, the accuracy of S-FMARS model is not statistically different in all performance measures (p value >0.05). On the other hand, S-FMARS is more time efficient than FMARS for all data problems. The decrease in CPU times is at least 71 %, which is observed for Communities and Crime data.

Table 6 Average performance results of F-MARS and S-FMARS on the train data

Datasets	M_{max}	Models	RMSE	Adj-R^2	GCV	CPU time (s)	Decrease in CPU time (%)
Parkinson	50	FMARS	0.348	0.863	0.195	29.51	90
		S-FMARS	0.347*	0.864*	0.194*	3.09*	
Red Wine	90	FMARS	0.619*	0.569*	0.603*	338.91	79
		S-FMARS	0.672	0.513	0.665	72. 68*	
Com. Crime	150	FMARS	0.365*	0.817*	0.580*	216.10	71
		S-FMARS	0.412	0.766	0.800	62.66*	
Conc. Comp.	100	FMARS	0.196*	0.955*	0.102	1,122.53	85
		S-FMARS	0.202	0.952	0.103*	168.51*	
AutoMpg	100	FMARS	1.847*	0.994*	64.66*	16.75	80
		S-FMARS	2.184	0.878	90.42	3.32*	
PM10	50	FMARS	0.649*	0.503*	0.867*	11.11	81
		S-FMARS	0.665	0.477	0.911	2.16*	
Energy load	1,730	FMARS	26,725.9	0.929	977,280,489.5	4,841.6	95
		S-FMARS	26,276.0*	0.932*	944,657,851.9*	236.4*	

Table 7 Average performance results of F-MARS and S-FMARS on test data and stability

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*
Energy load	15,070.7	14,252.94*	0.969	0.972*	0.564*	0.542	0.958	0.959*

The prediction capability of both methods on test data and their stabilities of measures for training and test data are presented in Table 7. With respect to prediction and stability measures, S-FMARS performs slightly better than F-MARS for Red Wine Quality, Communities and Crime, Auto Mpg, PM10 and Energy Load datasets. For the same datasets, S-FMARS is more stable than FMARS. However, one-sample sign test concludes that both methods have similar prediction and stability performances for all datasets in all performance measures (p values $>$0.05).

In order to evaluate the general performances of methods, mean and standard deviation of measures over all training and test datasets are given in Table 8 as well as the stability results. Here, standard deviation is used for indicating the robustness of the methods. It is noted that in the overall analysis, the effect of Energy Load data is disregarded due to its measures in different order (scale) caused by time dependent structure of data.

Table 8 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*	0.784*	11.168*	1.545	0.542	0.496	0.658
	(0.602**)	(0.203)	(26.207**)	(1.812)	(0.256)	(0.096**)	(0.187)
S-FMARS	0.747	0.742	15.515	1.084*	0.589 *	0.639*	0.770*
	(0.728)	(0.200*)	(36.697)	(0.920**)	(0.240**)	(0.148)	(0.157**)

* Better performance with respect to mean
** Better performance with respect to standard deviation in parenthesis

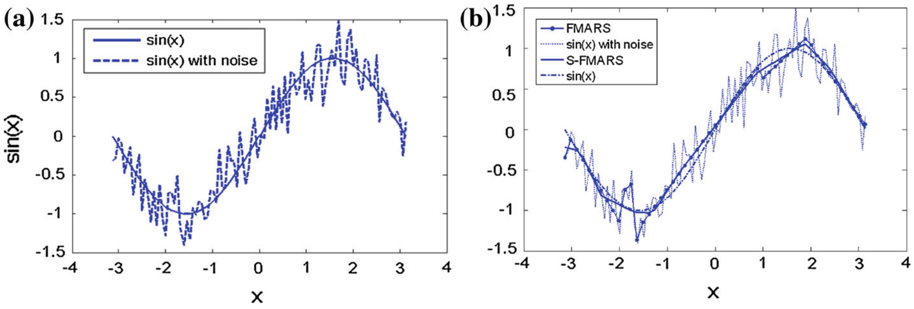


Fig. 5 **a** Sinus function with/without noise. **b** Fitted models for the function with noise

Although the models of FMARS are more accurate and robust than S-FMARS in the training data, S-FMARS shows more accurate predictions in the test data. In addition, the models in S-FMARS are more robust and more stable with respect to all performance measures.

4.3 Experiment on Noisy Data

Using the *sinus* function, two datasets are generated with and without noise using 100 observations (see Fig. 5a). Both methods are applied on the datasets. The accuracy and complexity measures are presented in Table 9. The performance measures calculated for the noise-free data are given in the first row, while the other rows are for noisy data. The methods are run for different M_{max} values on noisy data to observe their sensitivities against noise. Moreover, to measure the sensitivity of the model fitting on noisy data, noise-free setting is used as a benchmark. The accuracy of the fitted values obtained for the noisy data is calculated with respect to noise-free data (see Table 9).

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 9, the first row). In noisy setting, S-FMARS builds its best model with 19 BFs for all M_{max} values. However, as M_{max} value increases, FMARS builds more complex models to rise up the model accuracy. This shows that FMARS is more sensitive to the noise, and actually it tries to model the noise.

The model fits obtained by FMARS is more sensitive to noise than S-FMARS. Although the accuracy of FMARS increases on noisy data and gets better as the M_{max} value increases, its performance on noise-free data gets worse as the model become complex. S-FMARS can

Table 9 Performance results of FMARS and S-FMARS on noisy data

Noise-free data	M_{max}	BF$_{final}$		Datasets	RMSE		Adj-R^2	
		FMARS	S-FMARS		FMARS	S-FMARS	FMARS	S-FMARS
	≥ 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	20	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*

* Better performance

provide a less sensitive model to noise by building a less complex model for noisy data. The sensitivity of both fits on noisy data is illustrated in Fig. 5b. 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.

5 Conclusion

In 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 spline fitting. 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 procedure of MARS. The combination of these two procedures, so called S-FMARS provides a time efficient model building strategy for adaptive regression splines without degrading the model accuracy and prediction performance.

The accuracy, prediction and stability performance of FMARS and S-FMARS are found to be statistically similar to each other. Furthermore, S-FMARS results in less complex and less sensitive models on noisy data. The advantage of S-FMARS on computing time is clearly observed in the experiments. The computational time of S-FMARS is much less than FMARS. Especially, for the datasets requiring more complex models (M_{max} and *Int* are large) with high number of predictors ($p > 20$), S-FMARS generates models with significantly less CPU time than FMARS.

In general, the models produced by S-FMARS are as complex as FMARS in terms of the BFs used in model fitting. The final models of two methods include approximately the same number of BFs for the function approximation. Therefore, the overfitting problem of FMARS is still valid for S-FMARS. To overcome this problem, insignificant input variables can be detected and removed by a feature selection study. Hence the model size can be reduced for complex data structures including excessive number of predictors (p is large).

The choice of parameters (grid size and the threshold value) affects the accuracy and time efficiency of S-FMARS. As mentioned in Sect. 3, the grid size plays a key role for data approximation in mapping and computing time in modeling. For better approximation and best sub-setting with more representative data points, a sensitivity analysis is planned to be implemented for the optimal grid size. In addition, the efficiency of S-FMARS method can be increased by decreasing the number of potential knots with a threshold value as in (11). Further research is being carried out for determining the best threshold value in practice.

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. In this respect, the backward stepwise algorithm developed by Yerlikaya [51] and Weber et al. [48] can be implemented as a future study. In the study of [48], an alternative backward stepwise algorithm is proposed for that of MARS. The approach, called CMARS uses a penalized residual sum of squares for MARS as Tikhonov regularization problem, and threats this with conic quadratic programming. The drawback of the new method is not being as efficient as MARS. In order to increase the efficiency of CMARS, the approach proposed in this article can be a good alternative.

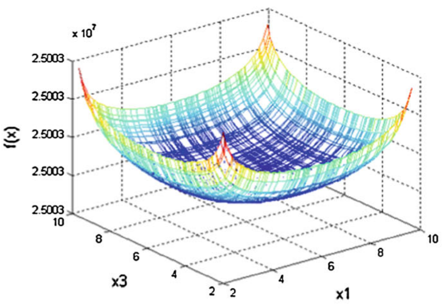
Appendix A

Mathematical formulations for data generation

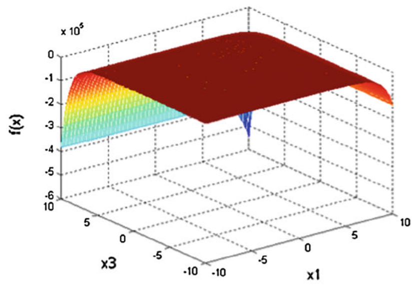
$$
\begin{aligned}
&P_1.f(x) = \sum_{i=1}^{7} \left[(\ln(x_i-2))^2 + (\ln(10-x_i))^2\right] - \left(\prod_{i=1}^{7} x_i\right)^2 2.1 \le x_i \le 9.9\\
&P_2.f(x) = \sum_{j=1}^{10} \exp(x_j)\left(c_j + x_j - \ln\left(\sum_{k=1}^{10} \exp(x_k)\right)\right),\\
&\text{cj} = -6.089, -17.164, -34.054, -5.914, -24.721, -14.986, -24.100, -10.708,\\
&\quad -26.662, -22.179\\
&P_3.f(x) = x_1^2 + x_2^2 + x_1x_2 - 14x_1 - 16x_2 + (x_3-10)^2 - 4(x_4-5)^2 + (x_5-3)^2\\
&\qquad + 2(x_6-1)^2 + 5x_7^2 + 7(x_8-11)^2 + 2(x_9-10)^2 + (x_{10}-7)^2 + 45\\
&P_4.f(x) = \sin(\pi x_1/12)\cos(\pi x_2/16)\\
&P_5.f(x) = (x_1-1)^2 + (x_1-x_2)^2 + (x_2-x_3)^4\\
&P_6.f(x) = 5.3578547x_2^2 + 0.8356891x_1x_3 + 37.293239x_1 - 40792.141\\
&P_8.f(x) = 3(1-x)^2\exp\left(-x^2-(y+1)^2\right) - 10\left(x/5 - x^3 - y^5\right)\exp\left(-x^2-y^2\right)\\
&\qquad - 1/3\exp\left(-(x+1)^2 - y^2\right) + 2x
\end{aligned}
$$

Appendix B

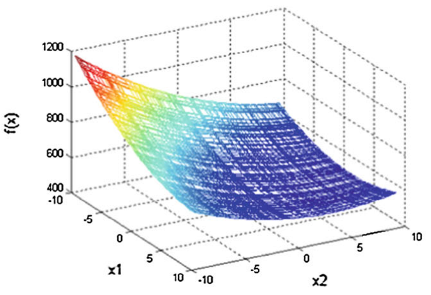
Grid plots of mathematical functions



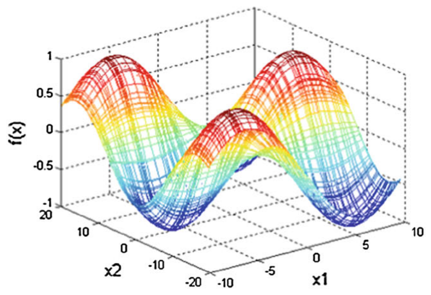
(a) $P_1(x_1, x_3)$ other x_i=3.



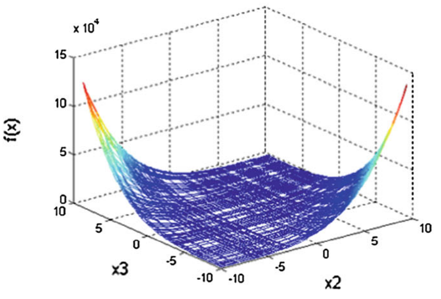
(b) $P_2(x_1, x_3)$ other x_i=3.



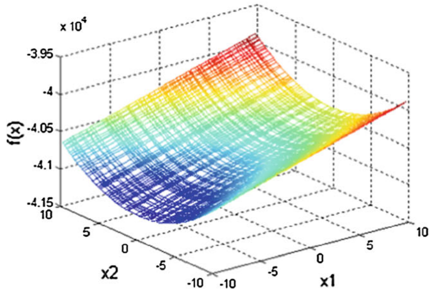
(c) $P_3(x_1, x_2)$ other x_i=4.



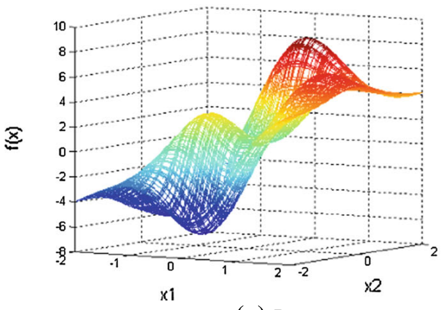
(d) P_4.



(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

Appendix C

Root mean squared error (RMSE)

RMSE indicates the grossly inaccurate estimates. The smaller RMSE, the better it is.

$$RMSE = \sqrt{\frac{1}{n}\sum\left(y_i - \hat{y}_i\right)^2}.$$

where, n is the number of observation, y_i is an ith observed response value and $\hat{y}_i$ is the ith fitted response.

Adjusted-multiple coefficient of determination (Adj-R^2):

The value of coefficient of determination (R^2) indicates how much variation in response is explained by the model. Adj-R^2 is penalized form of R^2 with respect to the number of predictors in model. The higher the Adj-R^2, the better the model fits the data.

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

where $(n-p-1) \neq 0$; p is the number of terms in the model, y_i is an ith observed response value; $\bar{y}_i$ is the mean response and $\hat{y}_i$ is the ith fitted response.

Stability

$$\min\left\{\frac{MR_{TR}}{MR_{TE}}, \frac{MR_{TE}}{MR_{TR}}\right\},$$

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.

Generalized cross validation (GCV)

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

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

The value $P(M)$ is the effective number of parameters which is a penalty measure for complexity. Here, $P(M) = r + dN$, where r is the number of linearly independent BFs in the model, and N 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 $d = 3$.

Appendix D

Red Wine quality

The dataset is related to red type of the Portuguese “Vinho Verde” wine [8]. The inputs include objective tests (e.g. PH values) and the output is based on sensory data (median of at least 3 evaluations made by wine experts). Input variables (based on physicochemical tests) are fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates, and alcohol. There is no missing attribute value in data.

Concrete compressive strength data set

Concrete is the most important material in civil engineering. The concrete compressive strength is a highly nonlinear function of age and ingredients. These ingredients include cement, blast furnace slag, fly ash, water, super plasticizer, coarse aggregate, and fine aggregate [50]. There is no missing attribute value in data.

Communities and crime

The data combines socio-economic data from the 1990 US Census, law enforcement data from the 1990 US LEMAS survey, and crime data from the 1995 FBI UCR. The variables included in the dataset involve the community, such as the percent of the population considered urban, and the median family income, and involving law enforcement, such as per capita number of police officers, and percent of officers assigned to drug units. The per capita violent crimes variable was calculated using population and the sum of crime variables considered violent crimes in the United States: murder, rape, robbery, and assault. It should be noted here that the Communities and Crime dataset is reduced in both size and scale to be appropriate for our analysis.

Auto-Mpg data

The data concerns city-cycle fuel consumption in miles per gallon to be predicted in terms of 3 multi-valued discrete and 5 continuous attributes including cylinders, displacement, horsepower, weight, acceleration, model year and origin [40]. Data has 6 missing values.

Parkinsons telemonitoring

This dataset is composed of a range of biomedical voice measurements from 42 people with early-stage Parkinson’s disease recruited to a six-month trial of a telemonitoring device for remote symptom progression monitoring. The recordings were automatically captured in the patient’s homes. Columns in the table contain subject number, subject age, subject gender, time interval from baseline recruitment date, motor UPDRS, total UPDRS, and 16 biomedical voice measures. Each row corresponds to one of 5,875 voice recording from these individuals. The main aim of the data is to predict the motor and total UPDRS scores (‘motor_UPDRS’ and ‘total_UPDRS’) from the 16 voice measures [45]. It should be noted here that the Parkinsons dataset is reduced in both size and scale to be appropriate for our analysis and the “motor UPDRS” variable is used as dependent variable.

PM10

The data are a subsample of 500 observations from a data set that originates in a study where air pollution at a road is related to traffic volume and meteorological variables, collected by the Norwegian Public Roads Administration. The response variable consists of hourly values of the logarithm of the concentration of PM10 (particles), measured at Alnabru in Oslo, Norway, between October 2001 and August 2003 [2].

Energy load

The data includes hourly electricity load of different regions in Netherlands which is recorded per day between 2008 and 2010. The average electricity load per day is aimed to be predicted by considering the past average values of energy demand. In the analysis, 1730 record is used to predict the daily average energy load.

References

1. Akaike, H.: Information theory and an extension of the maximum likelihood principle. In: Petrov, B.F, Cs'aki, F. (eds.) 2nd International Symposium on Information Theory. Academiai Kiado, Budapest (1973)
2. Aldrin, M.: Improved predictions penalizing both slope and curvature in additive models. Computational Statistics and Data Analysis **50**, 267–284 (2006)
3. Atilgan, T.: Basis Selection for Density Estimation and Regression. AT&T Bell Laboratories Technical Memorandum (1988)
4. Cervellera, C., Chen, V.C.P., Wen, A.: Neural network and regression spline value function approximations for stochastic programming. Comput. Oper. Res. **34**, 70–90 (2006)
5. Chen, V.C.P.: Measuring the goodness of orthogonal array discretization for stochastic programming and stochastic dynamic programming. SIAM J. Optim. **12**(2), 322–344 (2001)
6. Chen, V.C.P., Gunther, D., Johnson, E.L.: Solving for an optimal airline yield management policy via statistical learning. J. R. Stat. Soc. Ser. C **52**(1), 1–12 (2003)
7. Chen, V.C.P., Ruppert, D., Shoemaker, C.A.: Applying experimental design and regression splines to high-dimensional continuous-state stochastic dynamic programming. Oper. Res. **47**, 38–53 (1999)
8. Cortez, P., Cerdeira, A., Almeida, F., Matos, T., Reis, J.: Modeling wine preferences by data mining from physicochemical properties. Decision Support Systems **47**(4), 547–553 (2009)
9. Craven, P., Wahba, G.: Smoothing noisy data with spline functions: estimating the correct degree of smoothing by the method of generalized cross validation. Numerische Mathematik **31**, 377–403 (1979)
10. Crino, S., Brown, D.E.: Global optimization with multivariate adaptive regression splines. IEEE Trans. Syst. Man Cybern. Part B: Cybern. **37**(2), 333–340 (2007)
11. De Boor, C.: A Practical to Guide to Splines. Springer, New York (1978)
12. Deichmann, J., Eshghi, A., Haughton, D., Sayek, S., Teebagy, N.: Application of multiple adaptive regression splines (MARS) in direct response modeling. J. Direct Market. **16**, 15–27 (2002)
13. Denison, D.G.T., Mallick, B.K., Smith, A.F.M.: Automatic bayesian curve fitting. J. R. Stat. Soc. **60**, 333–350 (1998)
14. Eilers, Paul H.C., Marx, B.D.: Flexible smoothing with B-splines and penalties. Stat. Sci. **11**, 98–102 (1996)
15. Fox, J.: Nonparametric Regression, an R and S-PLUS Companion to Applied Regression. Sage, Thousand Oaks (2002)
16. Frank, A., Asuncion, A.: UCI Machine Learning Repository. University of California, School of Information and Computer Science, Irvine, CA. http://archive.ics.uci.edu/ml (2010)
17. Friedman, J.H., Silverman, B.W.: Flexible parsimonious smoothing and additive modeling. Technometrics **31**, 3–21 (1989)
18. Friedman, J.H.: Multivariate adaptive regression splines. Ann. Stat. **19**, 1–67 (1991)
19. Friedman, J.H.: Fast MARS. Stanford University Department of Statistics, Technical Report 110 (1993)
20. Gersho, A., Gray, R.M.: Vector Quantization and Signal Compression. Kluwer, Norwell (1992)
21. Gibbons, J.D., Chakraborti, S.: Nonparametric Statistical Inference. Marcel Dekker, New York (2003)

22. Green, P.H., Silverman, B.W.: Nonparametric Regression and Generalized Linear Models. Chapman Hall, Boca Raton (1994)
23. Hastie, T.J., Tibshirani, R.J., Friedman, J.: The Elements of Statistical Learning, Data Mining, Inference and Prediction. Springer, New York (2001)
24. Haykin, S.: Neural Networks: A Comprehensive Foundation. Prentice-Hall, New Jersey (1999)
25. Jekabsons, G.: ARESLab: Adaptive Regression Splines Toolbox for matlab/Octave. http://www.cs.rtu.lv/jekabsons/ (2011)
26. Jin, R., Chen, W., Simpson, T.W.: Comparative studies of metamodeling techniques under multiple modeling criteria. Struct. Multidiscip. Optim. **23**, 1–13 (2001)
27. Kohonen, T.: Self-organizing and Associative Memory. Springer, New York (1988)
28. Kubin, G.: Nonlinear prediction of mobile radio channels: Measurments and mars model designs. In: IEEE Proceedings of the International Conference on Acoustics, Speech, and Signal Processing, March 15–19, pp. 2667–2670 (1999)
29. Leathwick, J.R., Rowe, D., Richardson, J., Elith, J., Hastie, T.: Using multivariate adaptive regression splines to predict the distributions of New Zealand's freshwater diadromous fish. Freshw. Biol. **50**, 2034–2052 (2005)
30. Leathwick, J.R., Elith, J., Hastie, T.: Comparative performance of generalized additive models and multivariate adaptive regression splines for statistical modelling of species distributions. Ecol. Model. **199**, 188–196 (2006)
31. Lee, T.S., Chiu, C.C., Chou, Y.C., Lu, C.J.: Mining the customer credit using classifcation and regression tree and multivariate adaptive regression splines. Comput. Stat. Data Anal. **50**, 1113–1130 (2006)
32. Luo, Z., Wahba, G.: Hybrid adaptive splines. J. Am. Stat. Assoc. **92**, 107–116 (1997)
33. Mallows, C.L.: Some comments on Cp. Technometrics **15**, 661–675 (1973)
34. Martinez, N., Martinez, D., Rosenberger, J.M., Chen, V.C.P.: Global Optimization for a Piecewise Linear Regression Spline Function (INFORMS Meeting, 2011)
35. Miyata, S., Shen, X.: Free-knot splines and adaptive knot selection. J. Jpn. Stat. Soc. **35**(2), 303–324 (2005)
36. Özmen, A.: Robust Conic Quadratic Programming Applied to Quality Improvement—A Robustification of cmars, MSc. Middle East Technical University (2010)
37. Özmen, A., Weber, G.-W., Batmaz, İ.: The new robust CMARS (RCMARS) method. In: 24th Mini EURO Conference-on Continuous Optimization and Information-Based Technologies in the Financial Sector, MEC EurOPT Selected Papers, ISI Proceedings, pp. 362–368 (2010)
38. Pilla, V.L., Rosenberger, J.M., Chen, V.C.P., Engsuwan, N., Siddappa, S.: A multivariate adaptive regression splines cutting plane approach for solving a two-stage stochastic programming fleet assignment model. Eur. J. Oper. Res. **216**, 162–171 (2012)
39. Pilla, V., Rosenberger, J., Chen, V.: A statistical computer experiments approach to airline fleet assignment. IIE Trans. **40**, 524–537 (2008)
40. Quinlan, R.: Combining instance-based and model-based learning. In Proceedings on the Tenth International Conference of Machine Learning, University of Massachusetts, Amherst. Morgan Kaufmann. 236–243 (1993)
41. Sakamoto, W.: MARS: selecting basis functions and knots with an empirical Bayes method. Comput. Stat. **22**, 583–597 (2007)
42. Stone, C.J., Koo, CY.: Additive splines in statistics. In: Proceeding of the Statistical Computing Section, pp. 45–48. American Statistical Association, Alexandria, VA (1985)
43. Stone, C., Hansen, M., Kooperberg, C., Troung, Y.: Polynomial splines and their tensor products in extended linear modeling. Ann. Stat. **25**, 1371–1470 (1997)
44. Tsai, J.C.C., Chen, V.C.P., Chen, J., Beck, M.B.: Stochastic dynamic programming formulation for a wastewater treatment decision-making framework. Ann. Oper. Res. **132**, 207–221 (2004)
45. Tsanas, A., Little, M.A., McSharry, P.E., Ramig, L.O.: Accurate telemonitoring of Parkinson's disease progression by non-invasive speech tests. IEEE Transactions on Biomedical Engineering **57**(4), 884–894 (2010)
46. Vesanto, J., Himberg, J., Alhoniehmi, E., Parhankangas, J.: SOM toolbox for Mathlab 5, Report A57. http://www.cis.hut.fi//projects/somtoolbox/ (2000)
47. Wahba, G.: In discussion to. J. Ramsay: monotone regression splines in action. Stat. Sci. **3**, 425–462 (1988)
48. Weber, G.W., Batmaz, İ., Köksal, G., Taylan, P., Yerlikaya-Özkurt, F.: CMARS: a new contribution to nonparametric regression with multivariate adaptive regression splines supported by continuous optimization. Inverse Probl. Sci. Eng. **20**(3), 371–400 (2011)
49. Wong, C., Kohn, R.: A Bayesian approach to additive semi-parametric regression. J. Econom. **74**, 209–235 (1996)

50. Yeh, I.-C.: Modeling of strength of high performance concrete using artificial neural networks. Cement and Concrete Research **28**(12), 1797–1808 (1998)
51. Yerlikaya, F.: A New Contribution to Nonlinear Robust Regression and Classification with MARS and Its Application to Data Mining for Quality Control in Manufacturing, MSc.Thesis at the Institute of Applied Mathematics of METU, Ankara (2008)
52. York, T.P., Eaves, L.J., Van den Oord, E.J.C.G.: Multivariate adaptive regression splines: a powerful method for detecting disease-risk relationship differences among subgroubs. Stat. Med. **25**(8), 1355–1367 (2006)