SIViP (2017) 11:391–398
DOI 10.1007/s11760-016-0972-8

ORIGINAL PAPER

Highly efficient nonlinear regression for big data with lexicographical splitting

Mohammadreza Mohaghegh Neyshabouri[1] · **Oguzhan Demir**[1] · **Ibrahim Delibalta**[2] · **Suleyman Serdar Kozat**[1]

Received: 31 December 2015 / Revised: 22 August 2016 / Accepted: 26 August 2016 / Published online: 19 September 2016

Abstract This paper considers the problem of online piecewise linear regression for big data applications. We introduce an algorithm, which sequentially achieves the performance of the best piecewise linear (affine) model with optimal partition of the space of the regressor vectors in an individual sequence manner. To this end, our algorithm constructs a class of 2^D sequential piecewise linear models over a set of partitions of the regressor space and efficiently combines them in the mixture-of-experts setting. We show that the algorithm is highly efficient with computational complexity of only $O(mD^2)$, where m is the dimension of the regressor vectors. This efficient computational complexity is achieved by efficiently representing all of the 2^D models using a "lexicographical splitting graph." We analyze the performance of our algorithm without any statistical assumptions, i.e., our results are guaranteed to hold. Furthermore, we demonstrate the effectiveness of our algorithm over the well-known data sets in the machine learning literature with computational complexity fraction of the state of the art.

Keywords Online learning · Nonlinear regression · Piecewise linear · Lexicographical splitting

✉ Mohammadreza Mohaghegh Neyshabouri
mohammadreza@bilkent.edu.tr

Oguzhan Demir
odemir@ee.bilkent.edu.tr

Ibrahim Delibalta
ibrahim.delibalta@turktelekom.com.tr

Suleyman Serdar Kozat
kozat@ee.bilkent.edu.tr

[1] Department of Electrical and Electronics Engineering, Bilkent University, 06800 Bilkent, Ankara, Turkey

[2] Turk Telekom Communications Services Inc., Istanbul, Turkey

1 Introduction

We study online nonlinear regression and introduce a highly efficient and effective algorithm suitable for big data applications. Nonlinear regression and adaptive filtering are extensively studied problems in the signal processing [1,2] and machine learning [3] literatures, especially for applications where linear modeling [4] does not provide satisfactory results. However, in contemporary big data applications, the amount, dimensionality, and streaming rate of the data to be processed are significantly large compared to the classical frameworks [5]. Hence, we need online algorithms, which process the data in a sequential manner and then discard the processed data. Moreover, for such applications, the computational complexity is an important issue [5]. To this end, we introduce an online nonlinear regression algorithm with a significantly low computational complexity while providing superior modeling power.

In the regression problem, the objective is to find a function to model the relationship between given regressor vectors and desired data. As detailed later, this generic setup can be applied in a wide range of problems including prediction and modeling problems. Since the class of nonlinear models is too powerful to optimize for [2], we consider piecewise linear models, which can avoid overfitting while accurately modeling the nonlinear relationship between the desired data and the regressor vectors [6]. In such models, the space of regressor vectors is divided into several disjoint regions, and in each region, an independent linear model is used. Note that in order to provide satisfactory results, the optimal partition of the regressor space should be known. However, in real-world applications, there is no prior information about the optimal partition of the regressor space. Using tree-based regression algorithms [2,7] is a common approach in such scenarios. These algorithms form a class of models over a

set of partitions of the regressor space and adaptively combine these models, such that the combination asymptotically achieves the performance of the best model in the class. In particular, in nonstationary environments, where the optimal partition varies over time, this adaptive combination significantly improves the performance, in comparison to the models over fixed partitions. Furthermore, using hierarchical characteristics of trees leads to drastic reduction in computational complexity of the models combination [2] and makes these algorithms appropriate for big data applications. However, the partitioning technique used in most of previous studies on tree-based nonlinear regression problem is binary partitioning [2,7]. It is shown in [2] that given a set of D fixed splitting positions in the regressor space, the binary tree represents roughly 1.5^{D+1} different partitions of the regressor space and the piecewise linear models over these partitions. However, 2^D different partitions can be built using D given splitters, most of which are not represented by the binary trees.

In this paper, we use the lexicographical splitting graph [8], in the regression context for the first time in the literature. We propose an algorithm, which given a set of D fixed splitting positions, builds a sequential piecewise linear model over all of the 2^D possible partitions, and combines them with computational complexity of only $O(D^2)$. Hence, we asymptotically achieve the performance of the optimal piecewise linear model, in the accumulated squared error sense, without any statistical assumptions on the data. Note that when the true partition of the regressor space is one of the lexicographical partitions, but none of the binary partitions, the performance of our algorithm is significantly better than binary tree-based algorithms. Hence, our algorithm is more generic, compared to the binary tree-based algorithms. We point out that our results are guaranteed to hold in an individual sequence manner [9].

Our main contributions include: (1) We propose a novel, highly efficient and effective nonlinear regression algorithm suitable for big data applications; (2) we show that our algorithm asymptotically achieves the performance of the best piecewise linear model, over the optimal partition of the regressor space demonstrating the optimality in a strong mathematical sense without any statistical assumptions; (3) we show that even though our algorithm combines 2^D different sequential piecewise linear models, its computational complexity is just of $O(D^2)$; (4) in our simulations, we show that the proposed algorithm provides significantly better performance compared to other well-known nonlinear filters and regression algorithms.

The remainder of the paper is as follows. In Sect. 2, we provide the problem description and explain the challenges using an example. We introduce and discuss the performance of our algorithm in Sect. 3. Finally in Sect. 4, we demonstrate the performance of the introduced algorithm over the well-known data sets in the machine learning literature and a synthetic scenario.

2 Problem description

In this paper, all vectors are column vectors and denoted by boldface lower case letters. For a K-element vector $\mathbf{u}$, $\mathbf{u}^{\{i\}}$ represents the ith element, $\mathbf{u}^T$ is the ordinary transpose, and $\|\mathbf{u}\|_1 = \sum_{i=1}^{K} |u^{\{i\}}|$ is the L^1-norm of $\mathbf{u}$, where $|u^{\{i\}}|$ is the absolute value of $u^{\{i\}}$. We study sequential nonlinear regression, where at each round t, we observe a regressor vector $\mathbf{x}_t$, $\mathbf{x}_t \in [-A, A]^m$, and predict a desired data $d_t \in \mathbb{R}$, using a sequential nonlinear regression function $f_t(\cdot)$, such that our predict $\hat{d}_t$ is $\hat{d}_t = f_t(\mathbf{x}_t)$. After predicting $\hat{d}_t$, we observe the desired data d_t and suffer a loss according to d_t and $\hat{d}_t$, i.e., $l(d_t, \hat{d}_t)$. Then, using the observed d_t, we update $f_t(\cdot)$. Using this update procedure, we seek to minimize the accumulated loss function in an n round trial, i.e., $\sum_{t=1}^{n} l(d_t, \hat{d}_t)$. Note that n is not known or used in this paper for optimization, i.e., we work in a truly sequential manner [2]. In this paper, we use the squared error function as the loss function due to notational simplicity, i.e., $l(d_t, \hat{d}_t) = (d_t - \hat{d}_t)^2$. In Sect. 3.6, we show how to modify the algorithm to use different loss functions.

The regret of a sequential regression algorithm $\mathscr{A}$, which produces regression functions $f_t(\cdot)$, against a class of fixed regression functions, i.e., $\mathscr{S} = \{f^{(1)}, f^{(2)}, \ldots\}$, is defined as

$$R(\mathscr{A}, \mathscr{S}) = \sum_{t=1}^{n} l(d_t, f_t(\mathbf{x}_t)) - \min_{f^{(i)} \in \mathscr{S}} \sum_{t=1}^{n} l(d_t, f^{(i)}(\mathbf{x}_t)). \tag{1}$$

Note that if the regret grows sublinearly as n increases, i.e., $R(\mathscr{A}, \mathscr{S}) < O(n)$, then the average regret per round, i.e., $R(\mathscr{A}, \mathscr{S})/n$, tends to zero as n goes to infinity, which means that the sequential algorithm $\mathscr{A}$ asymptotically achieves the performance of the best fixed regression function in $\mathscr{S}$.

Given a partition of the regressor space, i.e., R_i for $i = 1, 2, \ldots, P$, where $R_i \subset [-A, A]^m$, $\cup_{p=1}^{P} R_p = [-A, A]^m$, and $R_i \cap R_j = \varnothing$ for $\forall i \neq j$, we define a sequential piecewise linear regression (SPLR) function as

$$f_t(\mathbf{x}_t) = \mathbf{v}_{t,I(\mathbf{x}_t)}{}^T \mathbf{x}_t + b_{t,I(\mathbf{x}_t)}, \tag{2}$$

where $I(\mathbf{x}_t) = p$ if $\mathbf{x}_t \in R_p$ for $p = 1, 2, \ldots, P$. For the sake of notational simplicity, with an abuse of notation, we rearrange (2) as

$$f_t(\mathbf{x}_t) = \mathbf{v}_{t,I(\mathbf{x}_t)}{}^T \mathbf{x}_t, \tag{3}$$

where $\mathbf{x}_t = [\mathbf{x}_t; 1]$ and $\mathbf{v}_{t,I(\mathbf{x}_t)} = [\mathbf{v}_{t,I(\mathbf{x}_t)}; b_{t,I(\mathbf{x}_t)}]$.

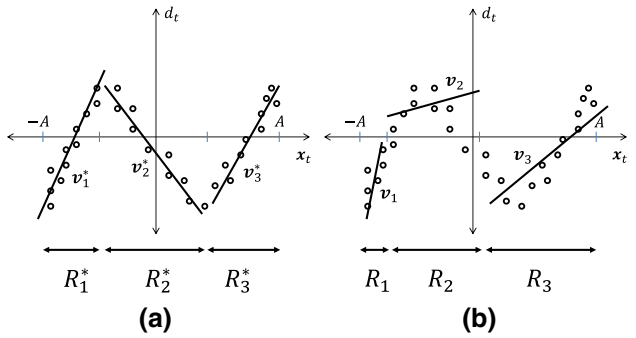

Fig. 1 Nonlinear regression with given partition and arbitrary partition

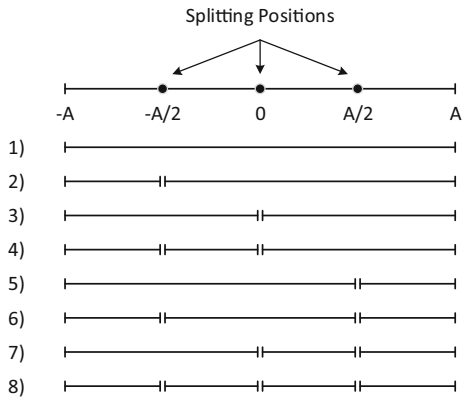

Fig. 2 Lexicographical partitions of the regressor space with $D = 3$ when $x_t \in [-A, A]$

We use SPLR functions as our nonlinear regression function. In piecewise linear regression with squared error loss function, if the optimal P-region partition of the regressor space is given, we can consider the problem as a collection of P independent sequential linear regression problems. Hence, the well-known least mean square (LMS) algorithm [10] can be used for each region of the given partition in order to search for the optimal linear regression function in a sequential manner.

We emphasize that the performance of the piecewise linear regression algorithm heavily depends on the correct selection of the partition of the regressor space. As an example, in Fig. 1, a set of particular d_t and $\mathbf{x}_t$ pairs for scalar regressors $\mathbf{x}_t = x_t$ are shown, where the underlying relationship is defined using the partition shown in Fig. 1a. If one uses the regions of the partition determined in Fig. 1a, i.e., $\{R_1^*, R_2^*, R_3^*\}$, and runs an LMS algorithm on each region of this partition, the SPLR function determined by $\{\mathbf{v}_{t,1}, \mathbf{v}_{t,2}, \mathbf{v}_{t,3}\}$, seeks the piecewise linear function determined by $\{\mathbf{v}_1^*, \mathbf{v}_2^*, \mathbf{v}_3^*\}$ as shown in Fig. 1a. However, if one wrongly commits to the partition given in Fig. 1b, i.e., $\{R_1, R_2, R_3\}$, the performance severely degrades, because the SPLR function seeks the wrong function determined by $\{\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3\}$, in Fig. 1b. Here, we assume no priori information about the correct partition of the regressor space. Hence, we seek to find an algorithm to search for the best fixed piecewise linear regression function over the unknown optimal partition of the regressor space.

3 A lexicographical sequential algorithm

In order to search for the best SPLR function, we form a class of lexicographical partitions of the regressor space and build SPLR model over each one of them. As detailed in the following sections, each model runs an LMS algorithm in each region of its corresponding partition in order to seek the best linear regression function in each region, and forms an SPLR function. At each round, each model uses its corresponding regression function to predict the desired data. We use a mixture-of-experts approach to combine the predictions of all models and produce our final prediction $\hat{d}_t$. We show that our combination of models converges to the best convex combination of the models. The key point of our algorithm is the efficient implementation of this combination.

To this end, first the lexicographical partitioning of the regressor space is introduced in Sect. 3.1. In Sect. 3.2, the SPLR models constructed over these partitions are explained. Section 3.3 explains the method used to combine the predictions of these models in a sequential manner. Section 3.4 shows how all of the lexicographical partitions can be compactly represented using a lexicographical splitting graph. Using this graph leads to an efficient implementation of the proposed combination of models, which is explained in Sect. 3.5. For the sake of representation simplicity, we explain the lexicographical partitioning for one-dimensional regressor space, i.e., $\mathbf{x}_t = x_t \in [-A, A]$. However, in Sect. 3.6, it is shown that the proposed algorithm can be directly extended to m-dimensional regressor space case. It is also explained that our algorithm can be easily modified to use different loss functions.

3.1 Lexicographical partition of the 1-dimensional regressor space

In this section, we illustrate the lexicographical partitioning of the regressor space. As shown in Fig. 2, for the 1-dimensional regressor data $\{x_t\}_{t\geq 1}$, we assume that $x_t \in [-A, A]$. In lexicographical partitioning with D-split, we consider D fixed splitting positions in the regressor space, i.e., $[-A, A]$. Splitting of the regressor space can just happen in these D points; hence, one can build 2^D different partitions of the regressor space with these D splitting positions. As an example, in Fig. 2, we represent lexicographical partitions with 3-split, which results $K = 2^3 = 8$ different possible partitions. Note that we choose equispaced splitting

positions. However, in general, one can select the splitting positions in an arbitrary manner. Note that if we select a sufficiently large D, then one of these K partitions will sufficiently approximate the optimal partition of the regressor space. For example, one can observe that the 6th partition in Fig. 2 fits well to the accurate partition of the regressor space for (x_t, d_t) pairs given in Fig. 1. In this paper, we use lexicographical partitioning with parameter D and build $K = 2^D$ SPLR models using produced lexicographical partitions as described in the following section.

3.2 Sequential piecewise linear models on the partitions

For each lexicographical partition described previously, we form an SPLR model, which runs an LMS algorithm on each region of its corresponding partition of the regressor space. Suppose a specific partition has P regions denoted by $R_1, R_2, \ldots, R_P$, e.g., say the last partition in Fig. 2, where $P = 4$. The SPLR function over this partition is defined as (3), where we update $\mathbf{v}_{t,i}$ for $i = 1, 2, \ldots, P$, using the LMS algorithm. At round t, after we made our prediction, i.e., $\hat{d}_t = \mathbf{v}_{t,I(\mathbf{x}_t)}{}^T \mathbf{x}_t$, and observed the true desired data d_t, we calculate $\mathbf{v}_{t+1,i}$ as follows:

$$\mathbf{v}_{t+1,i} = \begin{cases} \mathbf{v}_{t,i} + \mu(d_t - \hat{d}_t)\mathbf{x}_t, & \text{if } i = I(\mathbf{x}_t) \\ \mathbf{v}_{t,i}, & \text{if } i \neq I(\mathbf{x}_t), \end{cases}$$

where μ is a small step size. Using this update rule, each model searches for the best fixed piecewise linear regression function over its corresponding partition.

3.3 Sequential combination of the piecewise linear models defined by the lexicographical splittings

We combine predictions of all $K = 2^D$ different SPLR models defined using the lexicographical splittings, to produce our final prediction. Our final prediction at each round will be a convex combination of predictions of all K models. The weights of this combination will be updated based on the performance of the models in a sequential manner. We show later that using this update rule, we asymptotically achieve the performance of the best fixed convex combination of the model predictions. Denoting the prediction of kth SPLR model at round t by $\hat{d}_t^{\{k\}}$, for $k = 1, 2, \ldots, K$, we define $\hat{\mathbf{d}}_t = (\hat{d}_t^{\{1\}}, \hat{d}_t^{\{2\}}, \ldots, \hat{d}_t^{\{K\}})$. We produce the final prediction $\hat{d}_t$, using the well-known EG algorithm [9] by

$$\hat{d}_t = \mathbf{w}_t{}^T \hat{\mathbf{d}}_t, \tag{4}$$

where $\mathbf{w}_t = \left(w_t^{\{1\}}, w_t^{\{2\}}, \ldots, w_t^{\{K\}}\right)$ is the weights vector at round t, where the kth element $w_t^{\{k\}}$ is the weight corresponding to the kth model at round t. We update these weights recursively by

$$w_{t+1}^{\{k\}} = \frac{w_t^{\{k\}} \exp\left(-\eta \epsilon_t \hat{d}_t^{\{k\}}\right)}{\sum_{j=1}^{K} w_t^{\{j\}} \exp\left(-\eta \epsilon_t \hat{d}_t^{\{j\}}\right)}, \tag{5}$$

where η represents a positive, constant learning rate, and $\epsilon_t = l'(d_t - \hat{d}_t)$ is the first derivative of $l(d_t, \hat{d}_t)$ with respect to $\hat{d}_t$. In this paper we use $l(d_t, \hat{d}_t) = (d_t - \hat{d}_t)^2$; therefore, $\epsilon_t = 2(\hat{d}_t - d_t)$. Now we show that the aforementioned sequential convex combination achieves the performance of the best fixed convex combination of the model predictions. Suppose that predictions of the models are bounded, i.e., there exists a fixed positive number B, such that $-B \leq d_t^{\{k\}} \leq B$ holds for every k and t. Let $\mathscr{Z}$ be the class of all fixed K-element vectors $\mathbf{w} = (w^{\{1\}}, \ldots, w^{\{K\}})$, with nonnegative elements and L^1-norm equal to one. Theorem 5.10 in [9] implies that if we set $w_1^{\{k\}} = 1/K$, for $k = 1, \ldots, K$, and run EG algorithm as described in (4) and (5),

$$\begin{aligned} R(\mathscr{A}, \mathscr{Z}) &= \sum_{t=1}^{n} (d_t - \hat{d}_t)^2 - \min_{\mathbf{w} \in \mathscr{Z}} \sum_{t=1}^{n} (d_t - \mathbf{w}^T \hat{d}_t)^2 \\ &\leq O(\sqrt{n}). \end{aligned} \tag{6}$$

As discussed before, this sublinear bound implies that our algorithm asymptotically achieves the performance of the best fixed convex combination of the models. Note that each model is searching for the best piecewise linear regression function; hence, the proposed combination searches for the best piecewise linear regression function over the best partition between our lexicographical partitions. If the D parameter in our lexicographical partitioning is large enough, the best partition in our partitions set will be close enough to the true optimal partition.

3.4 Hierarchical nature of the lexicographical partitioning

In this section, we represent the 2^D different partitions using a lexicographical splitting graph. In the lexicographical splitting graph, we partition the regressor space in a hierarchical manner. For example in Fig. 3, a lexicographical splitting graph with parameter $D = 3$ is shown. In this figure, we have the first partition at top of the graph, i.e., $\{[-A, A]\}$. We can split this region in 3 ways and make a new partition of the regressor space. For example, if we choose dashed lines on the figure, we will produce a new partition, $\{[-A, A/2], [A/2, A]\}$. As shown in the figure, we cannot split $[A/2, A]$, but the interval $[-A, A/2]$ can be splitted again in two different ways, making two new intervals which alongside $[A/2, A]$ will be a new possible partition of the

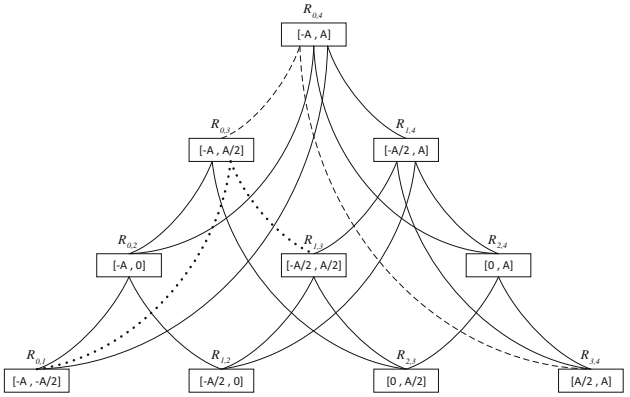

Fig. 3 Lexicographical splitting graph with $D = 3$ when $x_t \in [-A, A]$

whole regressor space. For instance, by choosing the dotted lines in Fig. 3, we can produce the partition $\{[-A, -A/2], [-A/2, A/2], [A/2, A]\}$. In lexicographical splitting graph with parameter D, different choices of splitting or not splitting on the lexicographical splitting graph will lead to 2^D different lexicographical partitions of the regressor space. We label the nodes of the graph with $R_{i,j}$, $0 \leq i < j \leq D+1$, such that $R_{i,j} = R_{i,k} \cup R_{k,j}$.

We use this graph to implement and combine our SPLR models efficiently. Since all regions that form the models corresponding partitions are represented as nodes in the lexicographical splitting graph, running an independent sequential linear regression algorithm in each node of the graph suffices to form all of the models. Therefore, even though using lexicographical partitioning with parameter D, we have $K = 2^D$ different sequential piecewise linear models, it suffices to run only $0.5(D+1)(D+2)$, i.e., the number of nodes in a lexicographical splitting graph with D-splits, sequential linear regression algorithms on nodes of the graph to form all 2^D models. Furthermore, when we observe x_t at round t, we do not need to update all of the linear regression functions. In fact, if $\mathbf{x}_t \in R_{i,i+1}$ in the finest partition, then $\mathbf{x}_t \in R_{k,i+1}, R_{k,i+2}, \ldots, R_{k,D+1}$ for $k = 0, 1, ..., i$, therefore we need to update only $(i+1)(D-i+1)$ sequential regression functions at these nodes, which include $\mathbf{x}_t$. We point out that the number of these nodes, which need to be updated, is approximately half of the total number of nodes at the worst-case scenario. In the following section we use the hierarchical nature of the lexicographical splitting graph to implement the combination given in (4) and (5) efficiently.

3.5 Efficient calculation of the weighted EG algorithm

In order to implement the EG algorithm effectively, first we observe that (5) can be reduced to the following, by taking equal initial weights and applying some straightforward algebra:

$$w_t^{\{k\}} = \frac{\exp\left(-\eta \sum_{\tau=1}^{t-1} \epsilon_\tau \hat{d}_\tau^{\{k\}}\right)}{\sum_{j=1}^{K}\left(\exp\left(-\eta \sum_{\tau=1}^{t-1} \epsilon_\tau \hat{d}_\tau^{\{j\}}\right)\right)}. \tag{7}$$

Now we represent an efficient calculation of the denominator of (7) using the lexicographical splitting graph. As mentioned before, each node in this graph corresponds to one of the $0.5(D+1)(D+2)$ sequential linear regression models. We denote the prediction of the sequential linear regression function in $R_{i,j}$ at time t, by $d_t^{\{i,j\}}$. In order to calculate the denominator of (7), we define $G_t{}^{\{k\}}$ for each model and $G_t{}^{\{i,j\}}$ for each node of the graph, as

$$G_t^{\{k\}} = \exp\left(-\eta \sum_{\tau=1}^{t-1} \epsilon_\tau \hat{d}_\tau^{\{k\}}\right), \tag{8}$$

$$G_t^{\{i,j\}} = \exp\left(-\eta \sum_{\tau=1}^{t-1} 1_{(x_t \in R_{i,j})} \epsilon_\tau \hat{d}_\tau^{\{k\}}\right), \tag{9}$$

where

$$1_{(x_t \in R_{i,j})} = \begin{cases} 1, & \text{if } x_t \in R_{i,j} \\ 0, & \text{if } x_t \notin R_{i,j}. \end{cases} \tag{10}$$

Now, denoting the denominator of (7) by G_t, we have

$$G_t = \sum_{k=1}^{K} G_t^{\{k\}}. \tag{11}$$

Also, we can express (8) as

$$G_t{}^{\{k\}} = \prod_{R_{i,j} \in \Gamma^{\{k\}}} G_t{}^{\{i,j\}}, \tag{12}$$

where $\Gamma^{\{k\}}$ represents the set of regions composing the kth lexicographical partition of the regressor space. Note that if $x_t \in R_{i,j}$ and $R_{i,j} \in \Gamma^{\{k\}}$, then $d_t^{\{k\}} = \hat{d}_t{}^{\{i,j\}}$. Now, the observation in (12) and the hierarchical nature of lexicographical partitioning lead to a recursive calculation of (11). We define an intermediate variable H on every node and calculate it as

$$H_t{}^{\{i,j\}} = G_t{}^{\{i,j\}} + \sum_{k=i+1}^{j-1} H_t{}^{\{i,k\}} G_t{}^{\{k,j\}}. \tag{13}$$

We observe that using above recursion $G_t = H_t^{\{0,D+1\}}$. Following the definition of $G_t^{\{i,j\}}$, the sequential update of $G_t^{\{i,j\}}$ can be expressed as

$$G_{t+1}^{\{i,j\}} = \begin{cases} G_t^{\{i,j\}} exp(-\eta \epsilon_t \hat{d}_t{}^{\{i,j\}}), & \text{if } x_t \in R_{i,j} \\ G_t^{\{i,j\}}, & \text{if } x_t \notin R_{i,j}. \end{cases} \tag{14}$$

We also have $H_{t+1}^{\{i,j\}} = H_t^{\{i,j\}}$ for $x_t \notin R_{i,j}$. Hence, the update rule for H variables is given by

$$H_{t+1}^{\{i,j\}} = \begin{cases} G_{t+1}{}^{\{i,j\}} + \sum_{k=i+1}^{j-1} H_{t+1}{}^{\{i,k\}} G_{t+1}{}^{\{k,j\}}, & \text{if } x_t \in R_{i,j} \\ H_t^{\{i,j\}}, & \text{if } x_t \notin R_{i,j}. \end{cases} \tag{15}$$

Note that at $t = 0$, we have to calculate G and H at all nodes, but at next rounds, when we observe x_t, we just need to update G and H variables at the nodes, which include x_t. Now we represent an efficient calculation of the final prediction of algorithm $\hat{d}_t$. Note that we can express $\hat{d}_t$ as

$$\hat{d}_t = \frac{\sum_{k=1}^{K} G_t^{\{k\}} \hat{d}_t^{\{k\}}}{G_t}. \tag{16}$$

In (16), we have calculated the denominator previously. We can calculate the numerator using an approach similar to the approach used to update G_t and calculate G_{t+1}. The only difference between calculation of G_{t+1} and the numerator of $\hat{d}_t$ is that $G_t^{\{k\}}$ gets multiplied by an exponential term in the former and by $\hat{d}_t{}^{\{k\}}$ in the latter. We define intermediate variables $\tilde{G}_t^{\{i,j\}}$ and $\tilde{H}_t^{\{i,j\}}$ at all nodes as

$$\tilde{G}_t^{\{i,j\}} = \begin{cases} G_t^{\{i,j\}} \hat{d}_t^{\{i,j\}}, & \text{if } x_t \in R_{i,j} \\ G_t^{\{i,j\}}, & \text{if } x_t \notin R_{i,j}, \end{cases} \tag{17}$$

$$\tilde{H}_t^{\{i,j\}} = \begin{cases} \tilde{G}_t^{\{i,j\}} + \sum_{k=i+1}^{j-1} \tilde{H}_t^{\{i,k\}} \tilde{G}_t^{\{k,j\}}, & \text{if } x_t \in R_{i,j} \\ H_t^{\{i,j\}}, & \text{if } x_t \notin R_{i,j}. \end{cases} \tag{18}$$

Note that with these definitions, $\hat{d}_t = \tilde{H}_t^{\{0,D+1\}} / H_t^{\{0,D+1\}}$.

The final algorithm is summarized in Table 1. Note that using this efficient implementation of the combination of the models, instead of updating 2^D sequential piecewise linear models and their combination weights, at worst-case scenario we update $D^2/4$ sequential linear models, combine them using $D^2/2$ intermediate variables, and update half of these intermediate variables. Hence, the computational complexity of the proposed algorithm is of $O(D^2)$.

3.6 Extension to m-dimensional regressor space and different loss functions

In this section, we extend our discussion to m-dimensional regressor space and different loss functions. Actually, extension to m-dimensional regressor space is straightforward. Throughout the text, we constructed our algorithm using 1-dimensional regressor data for illustration purposes. In the extension to m-dimensional case, the only difference is that we need to use $(m-1)$-dimensional disjoint hyperplanes as separators. For instance, if the regressor data are two dimensional, we need disjoint lines to separate our regressor space. The extension to different loss functions is also straightforward. In order to use a different loss function, we have to change the LMS algorithm in the nodes, to a sequential linear regression algorithm, which is appropriate for our loss function. Also the variable $\epsilon_t = l'(d_t, \hat{d}_t)$ in EG algorithm (5) should be changed by selecting different loss functions.

Table 1 Algorithmic description of lexicographical models weighting algorithm

Variables:
μ : Learning rate of LMS Algorithm
η : Learning rate of EG Algorithm
D : Depth of the Lexicographical Splitting Graph
$A : |\{x_t\}_{t\geq 1}| < A$
Initialization:
$L_0^{\{i,j\}} = 1$, Calculate $H_0^{\{i,j\}}$ using (13)
Algorithm:
for $t = 1$ **to** n **do**
  $V_t = \{R_{i,j} : x_t \in R_{i,j}\}$
  for all $\{R_{i,j}\} \in V_t$ **do**
    $\hat{d}_t^{\{i,j\}} = \mathbf{v}_{t,i,j}^T \mathbf{x}_t$ (extend scaler x_t by concatenating 1)
  end for
  for all $\{R_{i,j}\}$ **do**
    Calculate $\tilde{L}_t^{\{i,j\}}$ using (17)
  end for
  for all $\{R_{i,j}\}$ **do**
    Calculate $\tilde{H}_t^{\{i,j\}}$ using (18)
  end for
  $\hat{d}_t = \tilde{H}_t^{\{0,D+1\}} / H_t^{\{0,D+1\}}$
  $\varepsilon_t = l'(d_t, \hat{d}_t)$
  for all $\{R_{i,j}\}$ **do**
    Calculate $L_{t+1}^{\{i,j\}}$ using (14)
    Update the model as described in Section 3.2
  end for
  for all $\{R_{i,j}\}$ **do**
    Calculate $H_{t+1}^{\{i,j\}}$ using (15)
  end for
end for

4 Simulations

In this section, we demonstrate the performance of our algorithm, which we denote by lexicographical models weighting (LMW), over the well-known data sets, California housing and Abalone [11], and a synthetic scenario. Since the computational complexity of the algorithms is a critical issue in the big data applications, we have shown the computational complexity of all of the algorithms used in this section in Table 2.

Table 2 Comparison of the computational complexities of the proposed algorithms

Algorithm	Computational complexity
LMW	$O(mD^2)$
CTW	$O(mln(D))$
VF	$O(m^r)$
B-SAF	$O(mr^2)$
CR-SAF	$O(mr^2)$
FNF	$O((mr)^r)$

In the table, m represents the dimensionality of the regressor space, D represents the number of splitting positions in the respective algorithms, and r represents the order of the corresponding filters and algorithms

4.1 Matched partitions

In the first setup, we compare the performance of our algorithm with other tree-based algorithms such as context tree weighting (CTW) [2] and other well-known nonlinear regression algorithms such as Volterra filter (VF) [12] and Fourier nonlinear filter (FNF) [13]. To this end, we created a scenario such that there is a piecewise linear relationship between desired data and regression vectors described by

$$d_t = \begin{cases} \mathbf{v}_1^T \mathbf{x}_t + \pi_t, & \text{if } \boldsymbol{\phi}^T \mathbf{x}_t \leq -0.5 \\ \mathbf{v}_2^T \mathbf{x}_t + \pi_t, & \text{if } -0.5 < \boldsymbol{\phi}^T \mathbf{x}_t \leq 0.5 \\ \mathbf{v}_3^T \mathbf{x}_t + \pi_t, & \text{if } 0.5 < \boldsymbol{\phi}^T \mathbf{x}_t, \end{cases} \tag{19}$$

where $\boldsymbol{\phi} = [1; 0]$, π_t is a sample function from a zero mean white Gaussian process with variance 0.1, and $\mathbf{x}_t^{\{1\}}$ and $\mathbf{x}_t^{\{2\}}$ are sample functions from two independent zero mean Gaussian processes with variance 1. We normalized the regressor vectors such that they are bounded to 1, i.e., $|\mathbf{x}_t^{\{1\}}| \leq 1$ and $|\mathbf{x}_t^{\{2\}}| \leq 1$. In order to observe the effect of nonstationarity, we selected $\mathbf{v}_1 = [7; 7]$, $\mathbf{v}_2 = [-7; -7]$, $\mathbf{v}_3 = [7; -7]$ in the first half of the data and $\mathbf{v}_1 = [-11; -11]$, $\mathbf{v}_2 = [11; 11]$, $\mathbf{v}_3 = [-11; 11]$ in the second half. For CTW and LMW algorithms we used 2-dimensional hyperplanes $\mathbf{x}^{\{1\}} = -0.5$, $\mathbf{x}^{\{1\}} = 0$ and $\mathbf{x}^{\{1\}} = 0.5$, in 3-dimensional $(\mathbf{x}^{\{1\}}, \mathbf{x}^{\{2\}}, d_t)$ space, as splitting positions. In order to make a fair comparison we set the order of FNF and VF algorithms to 3 and 2, respectively. The average of the normalized mean squared error prediction performance of aforementioned algorithms, over 100 different realizations of the described model, is shown in Fig. 4. This figure shows that our LMW algorithm has superior performance both before and after the abrupt change of the model.

4.2 Benchmark real-life data sets

In the first experiment, the regressor space is 2 dimensional. In order to show that our algorithm also works good in the higher-dimensional regressor spaces, we tested our algorithm using the well-known data sets California housing and Abalone [11], which have high-dimensional regressor space. We compared the performance of our algorithm with other well-known nonlinear regression algorithms Bezier spline adaptive filter (B-SAF) [15], Catmull-Rom spline adaptive filter (CR-SAF) [15], FNF [13], CTW [2], and VF [12]. In both "California housing" and "Abalone" experiment, we set the learning rates of the adaptive filters to 0.01. In order to make a fair comparison between these methods, we set the order of all the algorithms accordingly. For instance, we set the parameter of the LMW as $D = 3$ and the depth of the CTW as 2, to make the number of regions in the finest partitions of this algorithms equal. In Figs. 5 and 6, the normalized mean squared errors in all of the aforementioned algorithms are shown on the "California housing" and "Abalone" data sets, respectively. In both figures, we observe that the tree-based algorithms LMW and CTW have superior performance with respect to the other algorithms. However, we observe that the CTW algorithm and the LMW algorithm performance is almost equal in these two experiments as opposed to the previous experiments. Note that the LMW and CTW algorithms are both mixture of types algorithms. However, when the finest partition of these two algorithms includes equal number of regions, the lexicographical partitioning method used in LMW produces more partitions than partitioning technique used in CTW. Hence, in the case of prediction of a data sequence whose underlying partition is one of the lexicographical partitions but none of the partitions created in CTW, the LMW algorithm has better performance as in the previous experiments. However, if the underlying partition is one of the partitions of both LMW and CTW algorithms, then they perform in a similar way.

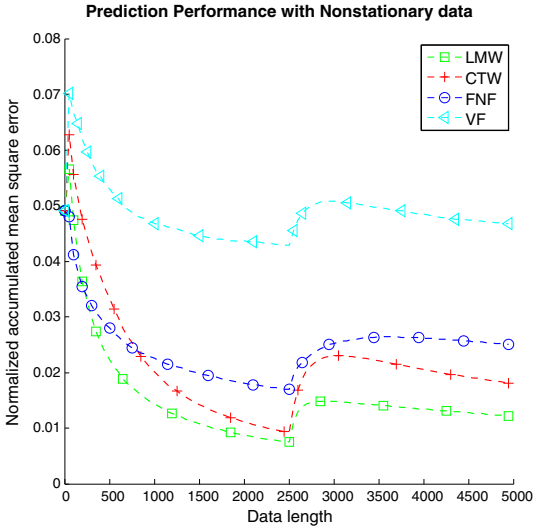

Fig. 4 Performance of LMW, CTW, FNF, and VF algorithms for a nonstationary data set, generated by (19), with abrupt change of parameters at beginning of the second half

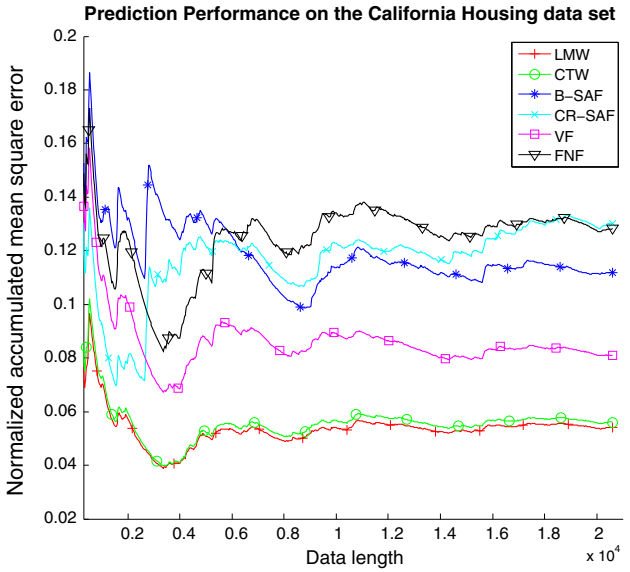

Fig. 5 Performance of LMW, CTW, B-SAF, CR-SAF, FNF, and VF algorithms over well-known California housing data set

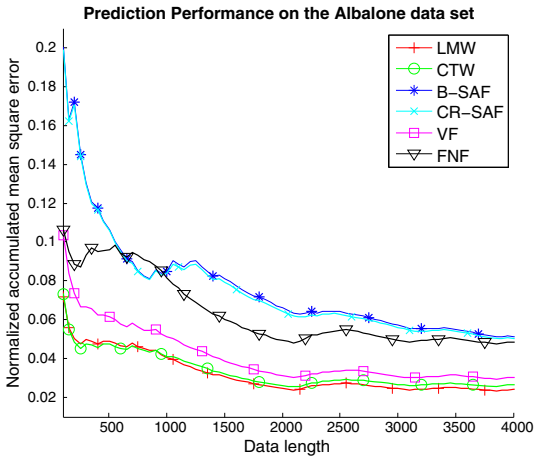

Fig. 6 Performance of LMW, CTW, B-SAF, CR-SAF, FNF, and VF algorithms over well-known Abalone data set

5 Conclusion

In this paper, we consider the problem of online piecewise linear regression. We introduce an algorithm, which sequentially achieves the performance of the best piecewise linear model with optimal partition of the regressor space in an individual sequence manner. Our algorithm uses a "lexicographical splitting graph" to compactly represent a wide set of piecewise linear models over different partitions of the regressor space. Using this graph, we combine the piecewise linear models with a significantly low computational complexity. We show that this combination asymptotically achieves the performance of the best piecewise linear model over the optimal partition of the regressor space. We avoid any statistical assumption in the analysis of the performance of our algorithm; hence, our results hold for any individual sequence of data. In addition, we demonstrate the effectiveness of our algorithm over the well-known data sets in the machine learning literature. Our algorithm achieves this performance with computational complexity fraction of the state of the art.

Acknowledgements This work is in part supported by Turkish Academy of Science Outstanding Researcher Programme, Tubitak Contract No 113E517, and Turk Telekom Inc.

References

1. Samah, H.A., Isa, N.M., Toh, K.K.: Automatic false edge elimination using locally adaptive regression kernel. Signal Image Video Process. **9**(6), 1339–1351 (2015)
2. Kozat, S.S., Singer, A.C., Zeitler, G.C.: Universal piecewise linear prediction via context trees. IEEE Trans. Signal Process. **55**(7), 3730–3745 (2007)
3. Helmbold, D.P., Schapire, R.E.: Predicting nearly as well as the best pruning of a decision tree. Mach. Learn. **27**(1), 51–68 (1997)
4. Singer, A.C., Feder, M.: Universal linear prediction by model order weighting. Signal Proces. IEEE Trans. **47**(10), 2685–2699 (1999)
5. Malik, P.: Governing big data: principles and practices. IBM J. Res. Dev. **57**(3/4), 1:1–1:13 (2013)
6. Michel, O.J.J., Hero, A.O., Badel, A.-E.: Tree-structured nonlinear signal modeling and prediction. IEEE Trans. Signal Process. **47**(11), 3027–3041 (1999)
7. Vanli, N.D., Kozat, S.S.: A comprehensive approach to universal piecewise nonlinear regression based on trees. Signal Process. IEEE Trans. **62**(20), 5471–5486 (2014)
8. Willems, F.M.J., Shtarkov, Y.M., Tjalkens, T.J.: Context weighting for general finite-context sources. IEEE Trans. Inform. Theory **42**, 42–1514 (1996)
9. Kivinen, J., Warmuth, M.K.: Exponentiated gradient versus gradient descent for linear predictors. J. Inf. Comput. **42**(5), 1514–1520 (1996)
10. Sayed, A.H.: Fundam. Adapt. Filter. Wiley, New Jersey (2003)
11. Blake, C., Merz, C.: UCI repository of machine learning databases (1998). http://www.ics.uci.edu/~mlearn/MLRepository.html
12. Mathews, V.J.: Adaptive polynomial filters. Signal Process. Mag. IEEE **8**(3), 10–26 (1991)
13. Carini, A., Sicuranza, G.L.: Recursive even mirror fourier nonlinear filters and simplified structures. Signal Process. IEEE Trans. **62**(24), 6534–6544 (2014)
14. Hénon, M.: A two-dimensional mapping with a strange attractor. Commun. Math. Phys. **50**(1), 69–77 (1976)
15. Scarpiniti, M., Comminiello, D., Parisi, R., Uncini, A.: Novel cascade spline architectures for the identification of nonlinear systems. Circ. Syst. I Regul. Pap. IEEE Trans. **62**(7), 1825–1835 (2015)