PIECEWISE NONLINEAR REGRESSION VIA DECISION ADAPTIVE TREES

N. Denizcan Vanli[*], *Muhammed O. Sayin*[*], *Salih Ergüt*[†], *and Suleyman S. Kozat*[*]

[*] Department of Electrical and Electronics Engineering
Bilkent University, Bilkent, Ankara 06800, Turkey
{vanli, sayin, kozat}@ee.bilkent.edu.tr

[†] AveaLabs, Istanbul, Turkey
salih.ergut@avea.com.tr

ABSTRACT

We investigate the problem of adaptive nonlinear regression and introduce tree based piecewise linear regression algorithms that are highly efficient and provide significantly improved performance with guaranteed upper bounds in an individual sequence manner. We partition the regressor space using hyperplanes in a nested structure according to the notion of a tree. In this manner, we introduce an adaptive nonlinear regression algorithm that not only adapts the regressor of each partition but also learns the complete tree structure with a computational complexity only polynomial in the number of nodes of the tree. Our algorithm is constructed to directly minimize the final regression error without introducing any ad-hoc parameters. Moreover, our method can be readily incorporated with any tree construction method as demonstrated in the paper.

Index Terms— Nonlinear regression, nonlinear adaptive filtering, adaptive, sequential, binary tree.

1. INTRODUCTION

Adaptive nonlinear regression is extensively studied in the signal processing [1, 2] and machine learning literatures [3], especially for the applications where linear modeling [2] is inadequate, hence, provide unsatisfactory results due to the linearity constraint. Although nonlinear regression methods provide a better modeling compared to the linear regression methods, they usually suffer from overfitting, stability and convergence issues [4], which considerably limit their applicability to signal processing problems. These issues are especially exacerbated in adaptive filtering due to the presence of feedback, which is even hard to control for linear models [4]. Furthermore, for big data problems, in which the regressor space has remarkably large dimensions, nonlinear models are usually avoided due to unmanageable computational complexity increase [5]. To overcome such difficulties, "tree" based nonlinear adaptive filters or regressors are introduced as elegant alternatives to linear models since these highly efficient methods retain the breadth of nonlinear models while mitigating the overfitting and convergence issues [1, 2, 5].

Although the power of regression trees are widely accepted, their usage usually suffers from algorithmic decisions such as the selection of the depth, dimensional submanifold of the regressor space, and discrete settings such as when the data is sparse. In particular, the success of the tree based regressors heavily depends on the "accurate" partitioning of the regressor space. Selection of a good partition, including its depth and regions, from the hierarchy is essential to balance the bias and variance of the regressor [5]. Therefore, in this paper, we introduce an algorithm mitigating such algorithmic decisions and overfitting problems by adaptively reconstructing the partitioning of the tree. Our algorithm, in basic words, performs an adaptive piecewise linear modeling, which is a natural nonlinear extension to linear modeling by partitioning the regressor space into a union of disjoint regions, where these regions are adaptively reconstructed according to the performance of the regressor.

Specifically, we provide a deterministic solution to the problem of nonlinear regression using decision trees. We introduce an algorithm that is shown *i)* to be highly efficient *ii)* to provide significantly improved performance over the state of the art approaches in different applications *iii)* to have guaranteed performance bounds without any statistical assumptions. Our algorithm not only adapts the corresponding regressors in each region, but also learns the corresponding region boundaries, as well as the "best" linear mixture of a doubly exponential number of partitions to minimize the final estimation or regression error, with a computational complexity only polynomial in the number of nodes of the tree. The introduced approach significantly outperforms the other tree based approaches such as [2] as demonstrated in our simulations, since we avoid any artificial weighting of models with highly data dependent parameters and, instead, "directly" minimize the final error, which is the ultimate performance goal. Our methods are generic such that they can readily incorporate random projection (RP) or k-d trees in their framework [5], e.g., the RP trees can be used as the starting partitioning to adaptively learn the tree, regressors and weighting to minimize the final error as data progress.

2. PROBLEM DESCRIPTION

Throughout the paper, all vectors are column vectors and denoted by boldface lower case letters. For a vector $\boldsymbol{x}$, $\boldsymbol{x}^T$ is the ordinary transpose.

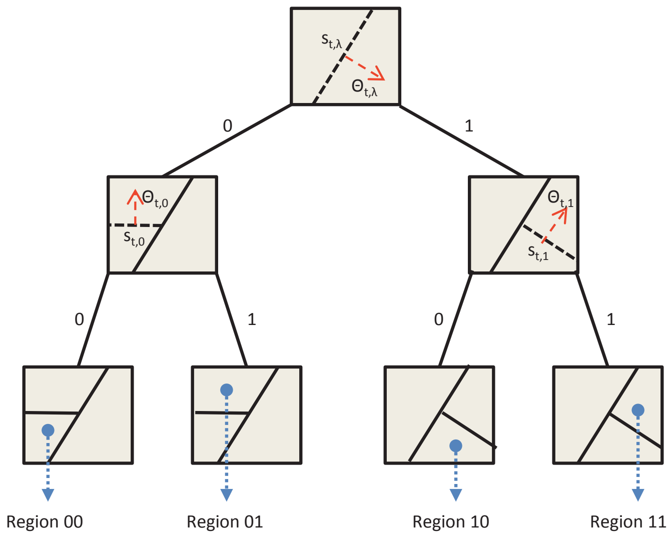

Fig. 1: The partitioning of a two dimensional regressor space using a complete tree of depth-2 with hyperplanes for separation. The whole regressor space is first bisected by $s_{t,\lambda}$, which is defined by the hyperplane $\boldsymbol{\theta}_{t,\lambda}$, where the region on the direction of $\boldsymbol{\theta}_{t,\lambda}$ vector corresponds to the child with "1" label. We then continue to bisect children regions using $s_{t,0}$ and $s_{t,1}$, defined by $\boldsymbol{\theta}_{t,0}$ and $\boldsymbol{\theta}_{t,1}$, respectively.

In this paper, we study sequential nonlinear regression, where we observe a desired signal $\{d_t\}_{t\geq 1}$, $d_t \in \mathbb{R}$, and regression vectors $\{\boldsymbol{x}_t\}_{t\geq 1}$, $\boldsymbol{x}_t \in \mathbb{R}^m$, such that we sequentially estimate d_t by

$$\hat{d}_t = f_t(\boldsymbol{x}_t),$$

and $f_t(\cdot)$ is an adaptive nonlinear regression function. At each time t, the regression error is given by

$$e_t = d_t - \hat{d}_t.$$

Although there exist several different approaches to select the corresponding nonlinear regression function, we particularly use piecewise models such that the space of the regression vectors, i.e., $\boldsymbol{x}_t \in \mathbb{R}^m$, is adaptively partitioned using hyperplanes based on a tree structure. We also use adaptive linear regressors in each region. However, our framework can be generalized to any partitioning of the regression space, i.e., not necessarily using hyperplanes, such as using [5], or any regression function in each region, i.e., not necessarily linear. Furthermore, both the region boundaries as well as the regressors in each region are adaptive.

3. REGRESSOR SPACE PARTITIONING

3.1. A Specific Partition on a Tree

To clarify the framework, suppose the corresponding space of regressor vectors is two dimensional, i.e., $\boldsymbol{x}_t \in \mathbb{R}^2$, and we partition this regressor space using a depth-2 tree as in Fig. 1. A depth-2 tree is represented by three separating functions $s_{t,\lambda}$, $s_{t,0}$ and $s_{t,1}$, which are defined using three hyperplanes with direction vectors $\boldsymbol{\theta}_{t,\lambda}$, $\boldsymbol{\theta}_{t,0}$ and $\boldsymbol{\theta}_{t,1}$, respectively (See Fig. 1). Due to the tree structure, three separating hyperplanes

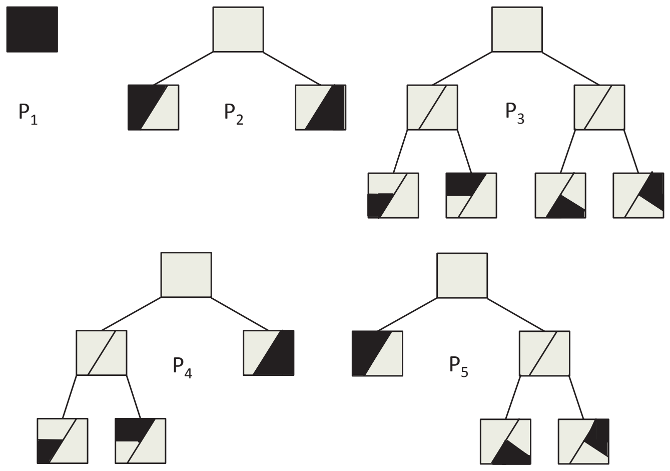

Fig. 2: All different partitions of the regressor space that can be obtained using a depth-2 tree. Any of these partition can be used to construct a piecewise linear model, which can be adaptively trained to minimize the regression error. These partitions are based on the separation functions shown in Fig. 1.

generate only four regions, where each region is assigned to a leaf on the tree given in Fig. 1 such that the partitioning is defined in a hierarchical manner, i.e., $\boldsymbol{x}_t$ is first processed by $s_{t,\lambda}$ and then by $s_{t,i}$, $i = 0, 1$. A complete tree defines a doubly exponential number, $O(2^{2^d})$, of subtrees each of which can also be used to partition the space of past regressors. As an example, a depth-2 tree defines 5 different subtrees or partitions as shown in Fig. 2, where each of these subtrees is constructed using the leaves and the nodes of the original tree. Note that a node of the tree represents a region which is the union of regions assigned to its left and right children nodes [6]. We also emphasize that without loss of generality, the regions pointed by the direction vector $\boldsymbol{\theta}_t$ are labeled as "1" regions on the tree in Fig. 1.

While in each region, one can select various regressors such as linear regressors, Volterra filters, or B-splines, for clarity, we use linear regressors in this paper. Note that linear regressors can also be extended to affine regressors by incrementing the length of the combination vector by one and appending a 1 at the end of the regressor vectors. In this sense, we can obtain the estimates of the all models in Fig. 2 by using the assigned node regressors and the partitioning scheme in Fig. 1. As an example, consider the third model in Fig. 2, i.e., P_3, where this partition is the union of 4 regions each corresponding to a leaf of the original complete tree in Fig. 1, labeled as 00, 01, 10, and 11. At each region, say the 00th region, we generate the estimate $\hat{d}_{t,00} = \boldsymbol{x}_t^T \boldsymbol{v}_{t,00}$, where $\boldsymbol{v}_{t,00} \in \mathbb{R}^m$ is the linear regressor vector assigned to region 00. Considering the hierarchical structure of the tree and having calculated all the region estimates, the final estimate of P_3 is given by

$$\begin{aligned}\hat{d}_t =& s_{t,\lambda} s_{t,0} \hat{d}_{t,00} + s_{t,\lambda}(1 - s_{t,0})\hat{d}_{t,01} \\ &+ (1 - s_{t,\lambda}) s_{t,1} \hat{d}_{t,10} + (1 - s_{t,\lambda})(1 - s_{t,1})\hat{d}_{t,11},\end{aligned}$$

for an arbitrary selection of the separator functions $s_{t,\lambda}, s_{t,0}, s_{t,1}$

and for any $\boldsymbol{x}_t$. We emphasize that any P_i, $i = 1, \ldots, 5$ can be used in a similar fashion to construct a piecewise linear regressor.

3.2. Generic Partitioning with a Tree

In this section, the sequential regressors (as described in Section 3.1) for all partitions in the doubly exponential tree class are combined for some adaptive separator function s_t. For $\beta_d \approx (1.5)^{2^d}$ different models that are embedded within a depth-d tree, we introduce the algorithm in Algorithm 1 achieving asymptotically the same cumulative squared regression error as the optimal combination of the best adaptive models. The algorithm is constructed in the proof of the Theorem 1.

Theorem 1: *Let $\{d_t\}_{t\geq 1}$ and $\{\boldsymbol{x}_t\}_{t\geq 1}$ represents arbitrary and real-valued sequences. The algorithm $\hat{d}_t$ given in Algorithm 1 when applied to any data sequences with an arbitrary length $n \geq 1$ yields*

$$\sum_{t=1}^{n} \big(d_t - \hat{d}_t\big)^2 - \min_{\boldsymbol{w}\in\mathbb{R}^{\beta_d}} \sum_{t=1}^{n} \big(d_t - \boldsymbol{w}^T \hat{\boldsymbol{d}}_t\big)^2 \leq O\big(\ln(n)\big),$$

where $\hat{\boldsymbol{d}}_t = [\hat{d}_t^{(1)}, \ldots, \hat{d}_t^{(\beta_d)}]^T$ and $\hat{d}_t^{(k)}$ represents the estimate of d_t at time t for the adaptive model $k = 1, \ldots, \beta_d$.

This theorem implies that our algorithm given in Algorithm 1, asymptotically achieves the performance of the optimal linear combination of the $O(2^{2^d})$ different "adaptive" regressors partitioning the m-dimensional regressor space that can be represented using a depth-d tree with a computational complexity $O(m4^d)$ (i.e., only polynomial in the number of nodes). We emphasize that while constructing the algorithm, we refrain from any statistical assumptions on the underlying data, and our algorithm works for any sequence of $\{d_t\}_{t\geq 1}$ with an arbitrary length of n.

3.3. Outline of the Proof of Theorem 1 and Construction of the Algorithm

Since we use the stochastic gradient updates in our algorithm, the upper bound proof of Theorem 1 follows similar lines to [7] and a complete proof of Theorem 1 can be obtained in [8]. The outline of the construction of the algorithm is as follows.

We first introduce a labeling for the tree nodes following [6]. The root node is labeled with an empty binary string λ and assuming that a node has a label p, where p is a binary string, we label its upper and lower children as $p1$ and $p0$, respectively. Here we emphasize that a string can only take its letters from the binary alphabet $\{0, 1\}$, where 0 refers to the lower child, and 1 refers to the upper child of a node. We also introduce another concept, i.e., the definition of the prefix of a string. We say that a string $p' = q'_1 \ldots q'_{l'}$ is a prefix to string $p = q_1 \ldots q_l$ if $l' \leq l$ and $q'_i = q_i$ for all $i = 1, \ldots, l'$, and the empty string λ is a prefix to all strings. Let $\mathcal{P}(p)$ represent all prefixes to the string p, i.e., $\mathcal{P}(p) \triangleq \{\nu_1, \ldots, \nu_{l+1}\}$, where $l \triangleq l(p)$ is the length of the string p, ν_i is the string with $l(\nu_i) = i - 1$, and $\nu_1 = \lambda$ is the empty string, such that the first $i - 1$ letters of the string p forms the string ν_i for $i = 1, \ldots, l + 1$.

Algorithm 1 Decision Adaptive Tree (DAT) Regressor

```
1: for $t = 1$ to $n$ do
2:     $\hat{d}_t \Leftarrow 0$
3:     for all $p \in \mathcal{N}_d - \mathcal{L}_d$ do
4:         $s_{t,p} \Leftarrow s^+ + (1 - 2s^+)/(1 + e^{\boldsymbol{x}_t^T \boldsymbol{\theta}_{t,p}})$
5:     end for
6:     for all $p \in \mathcal{L}_d$ do
7:         $\hat{d}_{t,p} \Leftarrow \boldsymbol{v}_{t,p}^T \boldsymbol{x}_t$
8:         $\alpha_{t,p} \Leftarrow 1$
9:         for $i = 1$ to $l(p)$ do
10:            $\alpha_{t,p} \Leftarrow \alpha_{t,p} s_{t,\nu_i}^{q_i}$
11:        end for
12:        $\hat{\delta}_{t,p} \Leftarrow \alpha_{t,p} \hat{d}_{t,p}$
13:        $\kappa_{t,p} \Leftarrow \gamma_d\big(l(p)\big) w_{t,p}$
14:        for all $\acute{p} \in \mathcal{N}_d - (\mathcal{P}(p) \cup \mathcal{S}_d(p))$ do
15:            $\bar{p} \Leftarrow \tilde{p} \in \mathcal{P}(p) \cap \mathcal{P}(\acute{p}) : l(\tilde{p}) = |\mathcal{P}(p) \cap \mathcal{P}(\acute{p})| - 1$
16:            $\kappa_{t,p} \Leftarrow \kappa_{t,p} + \frac{\gamma_d\left(l(p)\right)\gamma_{d-l(\bar{p})-1}\left(l(\acute{p})-l(\bar{p})-1\right)}{\beta_{d-l(\bar{p})-1}} w_{t,\acute{p}}$
17:        end for
18:        $\hat{d}_t \Leftarrow \hat{d}_t + \kappa_{t,p} \hat{\delta}_{t,p}$
19:    end for
20:    $e_t \Leftarrow d_t - \hat{d}_t$
21:    for all $p \in \mathcal{L}_d$ do
22:        $\boldsymbol{v}_{t+1,p} \Leftarrow \boldsymbol{v}_{t,p} + \mu_t e_t \alpha_{t,p} \boldsymbol{x}_t$
23:        $w_{t+1,p} \Leftarrow w_{t,p} + \mu_t e_t \hat{\delta}_{t,p}$
24:    end for
25:    for all $p \in \mathcal{N}_d - \mathcal{L}_d$ do
26:        $\sigma_{t,p} \Leftarrow 0$
27:        for all $\acute{p} \in \mathcal{S}_d(p0)$ do
28:            $\sigma_{t,p} \Leftarrow \sigma_{t,p} + \kappa_{t,\acute{p}} \frac{\hat{\delta}_{t,\acute{p}}}{s_{t,p}}$
29:        end for
30:        for all $\acute{p} \in \mathcal{S}_d(p1)$ do
31:            $\sigma_{t,p} \Leftarrow \sigma_{t,p} - \kappa_{t,\acute{p}} \frac{\hat{\delta}_{t,\acute{p}}}{1-s_{t,p}}$
32:        end for
33:        $\boldsymbol{\theta}_{t+1,p} \Leftarrow \boldsymbol{\theta}_{t,p} - \eta_t e_t \sigma_{t,p} s_{t,p} (1 - s_{t,p}) \boldsymbol{x}_t$
34:    end for
35: end for
```

Hence, we can compactly write the final estimate of the kth model at time t as

$$\hat{d}_t^{(k)} = \sum_{p\in\mathcal{M}_k} \hat{\delta}_{t,p},$$

where

$$\hat{\delta}_{t,p} \triangleq \hat{d}_{t,p} \prod_{i=1}^{l(p)} s_{t,\nu_i}^{q_i},$$

$\mathcal{M}_k$ is the set of all leaf nodes in the kth model, $\hat{d}_{t,p}$ is the regressor of the node p, $l(p)$ is the length of the string p, $\nu_i \in \mathcal{P}(p)$ is the prefix to string p with length $i - 1$, q_i is the ith letter of the string p, i.e., $\nu_{i+1} = \nu_i q_i$, and finally $s_{t,\nu_i}^{q_i}$ denotes the separator function at node ν_i such that

$$s_{t,\nu_i}^{q_i} \triangleq \begin{cases} s_{t,\nu_i}, & \text{if } q_i = 0 \\ 1 - s_{t,\nu_i}, & \text{otherwise} \end{cases}$$

with for some s_{t,ν_i}. We emphasize that we dropped explicit p-dependency of q_i and ν_i to simplify notation.

Since we now have a compact form to represent the tree and the outputs of each partition, we next introduce a method that compactly calculates the adaptive linear combination of $O(2^{2^d})$ piecewise linear regressor outputs.

To achieve a compact representation, we assign a particular linear weight to each node. We denote the weight of node p at time t as $w_{t,p}$ and then we define the weight of the kth model as the sum of weights of its leaf nodes, i.e.,

$$w_t^{(k)} = \sum_{p \in \mathcal{M}_k} w_{t,p},$$

for all $k = 1, \ldots, \beta_d$. Then, we achieve the following stochastic gradient update on the node weights

$$w_{t+1,p} \triangleq w_{t,p} + \mu_t e_t \hat{\delta}_{t,p}.$$

Before stating the algorithm that combines these node weights as well as node estimates, and generates the same final estimate $\hat{d}_t = \boldsymbol{w}_t^T \hat{\boldsymbol{d}}_t$ with a significantly reduced computational complexity, we first let $\mathcal{N}_d$ denote the set of all nodes in a depth-d tree. As an example, for $d = 2$ we obtain $\mathcal{N}_d = \{\lambda, 0, 1, 00, 01, 10, 11\}$. We then observe that for a node $p \in \mathcal{N}_d$ with length $l(p) \geq 1$, there exist a total of

$$\gamma_d\big(l(p)\big) \triangleq \prod_{j=1}^{l(p)} \beta_{d-j}$$

different models in which the node $p \in \mathcal{N}_d$ is a leaf node of that model, where $\beta_0 = 1$ and $\beta_{j+1} = \beta_j^2 + 1$ for all $j \geq 1$. For $l(p) = 0$ case, i.e., for $p = \lambda$, one can clearly observe that there exists only one model having λ as the leaf node, i.e., the model having no partitions, therefore $\gamma_d(0) = 1$.

Hence, after some algebra, the final estimate of our algorithm is given as follows

$$\begin{aligned} \hat{d}_t &= \sum_{k=1}^{\beta_d} w_t^{(k)} \hat{d}_t^{(k)} \\ &= \sum_{k=1}^{\beta_d} \left(\left(\sum_{p \in \mathcal{M}_k} w_{t,p} \right) \left(\sum_{p \in \mathcal{M}_k} \hat{\delta}_{t,p} \right) \right) \\ &= \sum_{p \in \mathcal{N}_d} \kappa(p) \hat{\delta}_{t,p}, \end{aligned}$$

where

$$\kappa(p) \triangleq \sum_{p \in \mathcal{N}_d} \gamma_d\big(l(p)\big) \times \left\{ w_{t,p} + \sum_{\substack{p' \in \mathcal{N}_d - \mathcal{P}(p) \\ \text{with } p \notin \mathcal{P}(p')}} w_{t,p'} \frac{\gamma_{d-l(\bar{p})-1}\big(l(p') - l(\bar{p}) - 1\big)}{\beta_{d-l(\bar{p})-1}} \right\},$$

and $\bar{p}$ denotes the longest prefix to both p and p', i.e., the longest string in the set of nodes $\mathcal{P}(p) \cap \mathcal{P}(p')$.

In a similar fashion, we use a stochastic gradient descent algorithm to update the region boundaries of the separator functions as follows

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \frac{1}{2} \eta_t \nabla_{\boldsymbol{\theta}_t} e_t^2, \tag{1}$$

where $\nabla_{\boldsymbol{\theta}_t} e_t^2$ is the derivative of e_t^2 with respect to $\boldsymbol{\theta}_t$.

In order to obtain a explicit formulation, we first let $\mathcal{S}(p) \triangleq \{p' \in \mathcal{N}_d \,|\, \mathcal{P}(p') = p\}$, i.e., $\mathcal{S}(p)$ denotes the set of all nodes a depth-d tree $p' \in \mathcal{N}_d$ whose set of prefixes include the node p. As an example, for a depth-2 tree, we have $\mathcal{S}(0) = \{0, 00, 01\}$. We also let $\mathcal{L}_d \triangleq \{p \in \mathcal{N}_d \,|\, l(p) = d\}$, i.e., $\mathcal{L}_d$ denotes the set of all nodes of a depth-d tree whose length is d, which can be viewed as the leaf nodes of the depth-d tree. As an example, for a depth-2 tree, we have $\mathcal{L}_2 = \{00, 01, 10, 11\}$.

Hence, after some algebra, the stochastic gradient update in (1) for an inner node $p \in \mathcal{N}_d - \mathcal{L}_d$ is given as follows

$$\boldsymbol{\theta}_{t+1,p} = \boldsymbol{\theta}_{t,p} + \eta_t e_t \left(\sum_{q=0}^{1} \sum_{p' \in \mathcal{S}(pq)} (-1)^q \frac{\hat{\delta}_{t,p'}}{s_{t,p'}} \right) \frac{\partial s_{t,p}}{\partial \boldsymbol{\theta}_{t,p}},$$

where the last term, i.e., $\partial s_{t,p} / \partial \boldsymbol{\theta}_{t,p}$ can be replaced with the corresponding derivative of the separator function with respect to the extended direction vector. For instance, if we choose $s_t = \big(1 + \exp(\boldsymbol{x}_t^T \boldsymbol{\theta}_t)\big)^{-1}$ as our separator function, we obtain $\partial s_{t,p} / \partial \boldsymbol{\theta}_{t,p} = -s_t(1 - s_t)\boldsymbol{x}_t$, where we emphasize that $\partial s_{t,p} / \partial \boldsymbol{\theta}_{t,p}$ includes s_t and $1 - s_t$ terms, hence in order not to slow down the learning rate of our algorithm, we may restrict $s^+ \leq s_t \leq 1 - s^+$ for some $0 < s^+ < 0.5$. According to this restriction we define the separator function as $s_t = s^+ + (1 - 2s^+) \big(1 + \exp(\boldsymbol{x}_t^T \boldsymbol{\theta}_t)\big)^{-1}$. This concludes the outline of the proof and the construction of the algorithm. □

4. SIMULATIONS

In this section, we illustrate the performance of our algorithm when the underlying partitioning of the regressor space does not match any partition represented by the tree to demonstrate the power of our algorithm. For this, the desired signal is generated by the following piecewise model

$$d_t = \begin{cases} \boldsymbol{w}^T \boldsymbol{x}_t + \pi_t, & \text{if } \boldsymbol{\phi}_0^T \boldsymbol{x}_t \geq 0.5 \text{ and } \boldsymbol{\phi}_1^T \boldsymbol{x}_t \geq 1 \\ -\boldsymbol{w}^T \boldsymbol{x}_t + \pi_t, & \text{if } \boldsymbol{\phi}_0^T \boldsymbol{x}_t \geq 0.5 \text{ and } \boldsymbol{\phi}_1^T \boldsymbol{x}_t < 1 \\ -\boldsymbol{w}^T \boldsymbol{x}_t + \pi_t, & \text{if } \boldsymbol{\phi}_0^T \boldsymbol{x}_t < 0.5 \text{ and } \boldsymbol{\phi}_2^T \boldsymbol{x}_t \geq -1 \\ \boldsymbol{w}^T \boldsymbol{x}_t + \pi_t, & \text{if } \boldsymbol{\phi}_0^T \boldsymbol{x}_t < 0.5 \text{ and } \boldsymbol{\phi}_2^T \boldsymbol{x}_t < -1 \end{cases}, \tag{2}$$

where $\boldsymbol{w} = [1,\, 1]^T$, $\boldsymbol{\phi}_0 = [4,\, -1]^T$, $\boldsymbol{\phi}_1 = [1,\, 1]^T$, $\boldsymbol{\phi}_2 = [1,\, 2]^T$, $\boldsymbol{x}_t = [x_{1,t},\, x_{2,t}]^T$, π_t is a sample function from a zero mean white Gaussian process with variance 0.1, $x_{1,t}$ and $x_{2,t}$ are sample functions of a jointly Gaussian process of mean $[0,\, 0]^T$ and variance $\boldsymbol{I}_2$. When initializing the algorithms, we assign the four quadrants of the two dimensional regressor space to the leaf nodes of the tree (i.e., we partition the regressor space using $x_1 = 0$ and $x_2 = 0$ lines). We plot the normalized time accumulated regression error for the Decision Adaptive Tree (DAT) regressor of Algorithm 1, the context-tree weighting (CTW) algorithm [2] (both having depths $d = 2$), the second order Volterra filter (VF) [4], the third order Fourier nonlinear filter (FNF) of [9], the cubic B-Spline Adaptive Filter (B-SAF) of [10] having 21 knots, and the Gaussian-kernel regressor(GKR) that is directly tuned to the underlying sequence.

We use the stochastic gradient descent algorithm in the regressor of each node for all algorithms, and the step sizes μ_t are set to 0.005 for the DAT (where $\eta_t = s^+(1-s^+)\mu_t$) and the CTW, 0.1 for the FNF, 0.025 for the B-SAF, 0.05 for the VF, and 1 for the GKR. The GKR is constructed using 4 node regressors, say $\hat{d}_{t,1}, \ldots, \hat{d}_{t,4}$, and a fixed Gaussian mixture weighting (that is selected according to the underlying sequence), giving $\hat{d}_t = \sum_{i=1}^{4} f(\boldsymbol{x}_t; \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)\, \hat{d}_{t,i}$, where $f(\boldsymbol{x}_t; \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$ is the multivariate Gaussian probability density function with mean $\boldsymbol{\mu}_i$ and variance $\boldsymbol{\Sigma}_i$ for $i = 1, \ldots, 4$, $\boldsymbol{x}_t$ is the original regressor vector, i.e., $\boldsymbol{x}_t = [x_{1,t},\ x_{2,t}]^T$ and $\hat{d}_{t,i} = \boldsymbol{v}_{t,i}^T \boldsymbol{x}_t$. In order to match the underlying partition that generates the sequence in (2), the mass points of the GKR are set to $\boldsymbol{\mu}_1 = [1.4565,\ 1.0203]^T$, $\boldsymbol{\mu}_2 = [0.6203,\ -0.4565]^T$, $\boldsymbol{\mu}_3 = [-0.5013,\ 0.5903]^T$, and $\boldsymbol{\mu}_4 = [-1.0903,\ -1.0013]^T$ with covariance matrices $\boldsymbol{\Sigma}_i = 1.2 \times \boldsymbol{I}_2$ for $i = 1, \ldots, 4$.

Fig. 3 shows the normalized time accumulated regression error of the proposed algorithms for a sample function of the process in (2). We observe that the DAT algorithm significantly outperforms its competitors by learning the true partitioning of the regressor space, whereas the other algorithms yield unsatisfactory results. We emphasize that even without any prior information and assumption on the underlying sequence, the DAT algorithm adapts its region boundaries and can capture the salient characteristics of the underlying data perfectly.

5. CONCLUDING REMARKS

We study nonlinear regression of deterministic signals using trees, where the regressor space is partitioned using a nested tree structure and different regressors are assigned to each region. In this framework, we introduce a tree based algorithm that both adapts its regressors in each region as well as its tree structure to best match to the underlying data while asymptotically achieving the performance of the best linear combination of a doubly exponential number of adaptive nonlinear regressors represented on a tree with a computational complexity only polynomial in the number of nodes of the tree. Furthermore, the introduced algorithm does not require a priori information on the data such as upper bounds or the length of the signal. Since our algorithm directly minimize the final regression error and avoid using any artificial weighting coefficients, they readily outperform different tree based regressors in our examples.

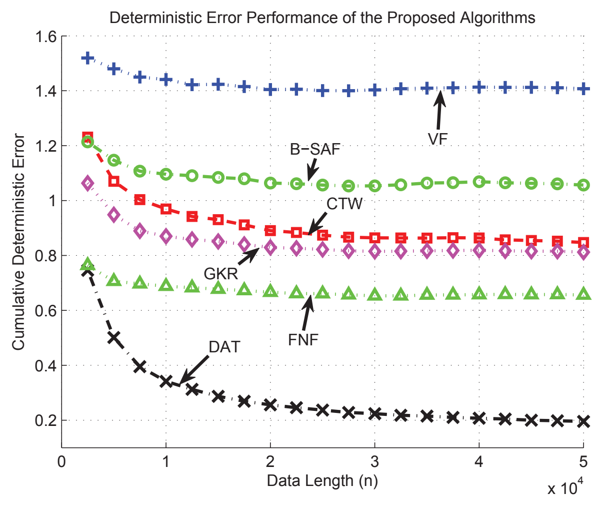

Fig. 3: Regression error performances for the second order piecewise linear model in (2). The time accumulated sequential regression error for the ANR and CTW algorithms (both using depth-2 tree structure), and the OGK regressor tuned to the underlying sequence.

REFERENCES

[1] O. J. J. Michel, A. O. Hero, and A.-E. Badel, "Tree-structured nonlinear signal modeling and prediction," *IEEE Transactions on Signal Processing*, vol. 47, no. 11, pp. 3027–3041, 1999.

[2] S. S. Kozat, A. C. Singer, and G. C. Zeitler, "Universal piecewise linear prediction via context trees," *IEEE Transactions on Signal Processing*, vol. 55, no. 7, pp. 3730–3745, 2007.

[3] D. P. Helmbold and R. E. Schapire, "Predicting nearly as well as the best pruning of a decision tree," *Machine Learning*, vol. 27, no. 1, pp. 51–68, 1997.

[4] A. H. Sayed, *Fundamentals of Adaptive Filtering*. NJ: John Wiley & Sons, 2003.

[5] S. Dasgupta and Y. Freund, "Random projection trees for vector quantization," *IEEE Transactions on Information Theory*, vol. 55, no. 7, pp. 3229–3242, 2009.

[6] F. M. J. Willems, Y. M. Shtarkov, and T. J. Tjalkens, "The context-tree weighting method: basic properties," *IEEE Transactions on Information Theory*, vol. 41, no. 3, pp. 653–664, 1995.

[7] E. Hazan, A. Agarwal, and S. Kale, "Logarithmic regret algorithms for online convex optimization," *Machine Learning*, vol. 69, no. 2-3, pp. 169–192, 2007.

[8] N. D. Vanli and S. S. Kozat, "A comprehensive approach to universal nonlinear regression based on trees," *CoRR*, vol. abs/1311.6392, 2013.

[9] A. Carini and G. L. Sicuranza, "Fourier nonlinear filters," *Signal Processing*, vol. 94, no. 0, pp. 183 – 194, 2014.

[10] M. Scarpiniti, D. Comminiello, R. Parisi, and A. Uncini, "Nonlinear spline adaptive filtering," *Signal Processing*, vol. 93, no. 4, pp. 772 – 783, 2013.