Piecewise Linear Regression Based On Adaptive Tree Structure Using Second Order Methods

Burak C. Civek
Department of Electrical and Electronics Engineering,
Bilkent University,
Ankara 06800, Turkey
Email: civek@ee.bilkent.edu.tr

Ibrahim Delibalta
Turk Telekom Labs
Istanbul, Turkey
Email: ibrahim.delibalta@turktelekom.com.tr

Suleyman S. Kozat
Department of Electrical and Electronics Engineering,
Bilkent University,
Ankara 06800, Turkey
Email: kozat@ee.bilkent.edu.tr

***Abstract*—We introduce a highly efficient online nonlinear regression algorithm. We process the data in a truly online manner such that no storage is needed, i.e., the data is discarded after used. For nonlinear modeling we use a hierarchical piecewise linear approach based on the notion of decision trees, where the regressor space is adaptively partitioned based directly on the performance. As the first time in the literature, we learn both the piecewise linear partitioning of the regressor space as well as the linear models in each region using highly effective second order methods, i.e., Newton-Raphson Methods. Hence, we avoid the well known over fitting issues and achieve substantial performance compared to the state of the art. We demonstrate our gains over the well known benchmark data sets and provide performance results in an individual sequence manner guaranteed to hold without any statistical assumptions.**

***Index Terms*—Hierarchical tree, big data, online learning, piecewise linear regression, Newton method.**

I. Introduction

Nonlinear regression problem is one of the most important topics in the machine learning and signal processing literatures and arises in several different applications such as signal modeling [1], [2], financial market [3] and trend analyses [4], intrusion detection [5] and recommendation [6]. However, the traditional regression techniques show less than adequate performance in real-life applications having big data since (1) data acquired from diverse sources are too large in size to be efficiently processed or stored by conventional signal processing and machine learning methods [7]; (2) the performance of the conventional methods is further impaired by the highly variable properties, structure and quality of data acquired at high speeds [7].

In this context, to accommodate these problems, we introduce online regression algorithms that process the data in an online manner, i.e., instantly, without any storage, and then discard the data after using and learning [8]. Hence our methods can constantly adapt to the changing statistics or quality of the data so that they can be robust and prone to variations and uncertainties [8]. From a unified point of view, in such problems, we sequentially observe a real valued vector sequence $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots$ and produce a decision (or an action) y_t at each time t based on the past $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_t$. After the desired output y_t is revealed, we suffer a loss and our goal is to minimize the accumulated (and possibly weighted) loss as much as possible while using a limited amount of information from the past.

To this end, for nonlinear regression we use a hierarchical piecewise linear model based on the notion of decision trees, where the space of the regressor vectors, $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots$, is adaptively partitioned and continuously optimized in order to enhance the performance [2], [9]. We note that the piecewise linear models are extensively used in the signal processing literature to mitigate the overtraining issues that arise due to using nonlinear models [2]. However their performance in real life applications are less than adequate since their successful application highly depends on the accurate selection of the piecewise regions that correctly model the underlying data [10]. Clearly, such a goal is impossible in an online setting since either the best partition is not known, i.e., the data arrives sequentially, or in real life applications the statistics of the data and the best selection of the regions change in time. To this end, as the first time in the literature, we learn both the piecewise linear partitioning of the regressor space as well as the linear models in each region using highly effective second order methods, i.e., Newton-Raphson Methods [11]. Hence, we avoid the well known over fitting issues by using piecewise linear models, however, since both the region boundaries as well as the linear models in each region are trained using the second order methods we achieve substantial performance compared to the state of the art [11]. We demonstrate our gains over the well known benchmark data sets extensively used in the machine learning literature. We also provide theoretical performance results in an individual sequence manner that are guaranteed to hold without any statistical assumptions [12]. In this sense, the introduced algorithm addresses computational complexity issues widely encountered in big data application while providing superior guaranteed performance in a strong deterministic sense.

II. Problem Description

In this paper, all vectors are column vectors and represented by lower case boldface letters. For matrices, we use upper case boldface letters. The ℓ^2-norm of a vector $\boldsymbol{x}$ is given by

978-0-9928-6265-7/16/$31.00 ©2016 IEEE

$\|\boldsymbol{x}\| = \sqrt{\boldsymbol{x}^T\boldsymbol{x}}$ where $\boldsymbol{x}^T$ denotes the ordinary transpose. The identity matrix with $n \times n$ dimension is represented by $\boldsymbol{I}_n$.

We work in an online setting, where we estimate a data sequence $y_t \in \mathbb{R}$ at time $t \geq 1$ using the corresponding observed feature vector $\boldsymbol{x}_t \in \mathbb{R}^m$ and then discard $\boldsymbol{x}_t$ without any storage. Our goal is to sequentially estimate y_t using $\boldsymbol{x}_t$ as

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

where $f_t(\cdot)$ is a function of past observations. In this work, we use piecewise linear functions, due to their capability of approximating most nonlinear models. In order to construct a piecewise linear model, we partition the space of regressor vectors into, say K, distinct regions S_k^m, where $\bigcup_{k=1}^{K} S_k^m = \mathbb{R}^m$ and $S_i^m \cap S_j^m = \emptyset$ when $i \neq j$. In each region, we use a linear regressor, i.e., $\hat{y}_{t,i} = \boldsymbol{w}_{t,i}^T\boldsymbol{x}_t + c_{t,i}$, where $\boldsymbol{w}_{t,i}$ is the linear regression vector, $c_{t,i}$ is the offset and $\hat{y}_{t,i}$ is the estimate corresponding to the i^{th} region. We represent $\hat{y}_{t,i}$ in a more compact form as $\hat{y}_{t,i} = \boldsymbol{w}_{t,i}^T\boldsymbol{x}_t$, by denoting $\boldsymbol{x}_t = [\boldsymbol{x}_t^T : 1]^T$ and $\boldsymbol{w}_{t,i} = [\boldsymbol{w}_{t,i}^T : c_{t,i}]^T$.

There are two major problems when using piecewise linear models: *i*) determining the piecewise regions such that the partitions do not cause the well known overfitting and underfitting issues [9] and *ii*) to find out the linear model that best fits the data in each distinct region in a sequential manner [10]. In this paper, we solve both of these problems using highly effective and completely adaptive second order piecewise linear regressors. In order to have a measure on how well the determined piecewise linear model fits the data, we use instantaneous squared loss, i.e., $e_t^2 = (y_t - \hat{y}_t)^2$ as our cost function. Our goal is to specify the partitions and the corresponding linear regressor weights at each iteration such that the total regressor error is minimized.

III. Highly Efficient Tree Based Sequential Piecewise Linear Predictor

In this section, we introduce a highly effective algorithm constructed by piecewise linear models. The presented algorithm provide efficient learning even for highly nonlinear data models. Moreover, continuous updating based on the upcoming data ensures our algorithms to achieve outstanding performance for online frameworks. We use a tree notion to partition the regression space, which provides a systematic way to determine the regions [2], [9]. We illustrate this method in Fig. 1 for 2-dimensional case to clarify the technique. In general, for the m-dimensional space partitioning, we define the differentiable separator functions represented by $p_{t,k}$,

$$p_{t,k} = \frac{1}{1 + e^{-\boldsymbol{x}_t^T\boldsymbol{n}_{t,k}}} \tag{1}$$

where $\boldsymbol{n}_{t,k} \in \mathbb{R}^m$ is the normal vector, k denotes the region label and $c_{t,k}$ is embedded in the last entry of $\boldsymbol{n}_{t,k}$. To get a highly versatile and data adaptive partitioning, we train the region boundaries by updating corresponding normal vectors. Specific to the case shown in Fig. 1, suppose, the observed feature vectors $\boldsymbol{x}_t = [x_{t,1} : x_{t,2}]^T$ come from a bounded set $\{\Omega\}$ such that $-A \leq x_{t,1}, x_{t,2} \leq A$ for some $A > 0$.

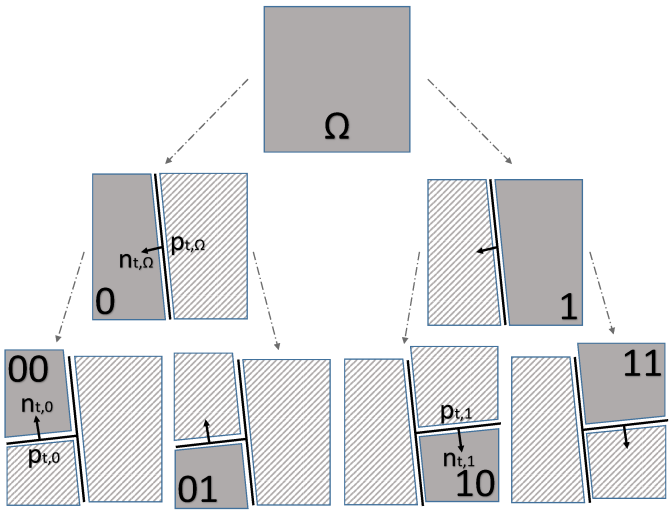

Fig. 1: Tree Based Partitioning of The Regression Space

At first, we have the whole space as a single set $\{\Omega\}$. Then we use a single separation function, which is a line in this case, to partition this space into subspaces $\{0\}$ and $\{1\}$ such that $\{0\} \cup \{1\} = \{\Omega\}$. In the following steps, the partition technique is quite different. Since we have two distinct subspaces after the first step, we work on them separately, i.e., the partition process continues recursively in each subspace independent of the others. When we add two more hyperplanes separating the sets $\{0\}$ and $\{1\}$, we get four distinct subspaces $\{00\}, \{01\}, \{10\}$ and $\{11\}$ where their union forms the initial regression space. Therefore, adding one more hyperplane has an effect on just a single region, not on the whole space. We use an identifier called the depth, which determines how deep the partition is, e.g. depth of the model shown in Fig. 1 is 2. In particular, the number of different regions generated by the depth-d models are given by 2^d. Hence, the number of distinct regions increases in the order of $O(2^d)$. For the tree based partitioning, we use the finest model of a depth-d tree. The finest partition consists of the regions generated at the deepest level, e.g. regions $\{00\}, \{01\}, \{10\}$ and $\{11\}$ shown in Fig. 1.

Following the model given in Fig. 1, say, we have three different separator functions, $p_{t,\Omega}, p_{t,0}, p_{t,1} \in \mathbb{R}$, which are defined by $\boldsymbol{n}_{t,\Omega}, \boldsymbol{n}_{t,0}, \boldsymbol{n}_{t,1} \in \mathbb{R}^2$ respectively. For the region $\{00\}$, the corresponding estimate is given by

$$\hat{y}_{t,00} = \boldsymbol{w}_{t,00}^T\boldsymbol{x}_t$$

where $\boldsymbol{w}_{t,00} \in \mathbb{R}^2$ is the regression vector of the region $\{00\}$. Since we have the estimates of all regions, the final estimate is given by

$$\begin{aligned}\hat{y}_t = {} & p_{t,\Omega}p_{t,0}\hat{y}_{t,00} + p_{t,\Omega}(1-p_{t,0})\hat{y}_{t,01} \\ & + (1-p_{t,\Omega})p_{t,1}\hat{y}_{t,10} + (1-p_{t,\Omega})(1-p_{t,1})\hat{y}_{t,11}\end{aligned} \tag{2}$$

when we observe the feature vector $\boldsymbol{x}_t$. This result can be extended to the depth-d models with $d > 2$.

We adaptively update the weights associated with each partition based on the overall performance. Boundaries of the

regions are also updated to reach the best partitioning. We use the second order algorithms, e.g. Online Newton Step [13], to update both separator functions and region weights. To accomplish this, the weight vector assigned to the region $\{00\}$ is updated as

$$\begin{aligned}\boldsymbol{w}_{t+1,00} &= \boldsymbol{w}_{t,00} - \frac{1}{\beta}\boldsymbol{A}_t^{-1}\nabla e_t^2 \\ &= \boldsymbol{w}_{t,00} + \frac{2}{\beta}e_t p_{t,\Omega} p_{t,0}\boldsymbol{A}_t^{-1}\boldsymbol{x}_t\end{aligned} \tag{3}$$

where β is the step size, ∇ is the gradient operator w.r.t. $\boldsymbol{w}_{t,00}$ and $\boldsymbol{A}_t$ is an $m \times m$ matrix defined as

$$\boldsymbol{A}_t = \sum_{i=1}^{t} \nabla_i \nabla_i^T + \epsilon \boldsymbol{I}_m \tag{4}$$

where $\epsilon > 0$ is used to ensure that $\boldsymbol{A}_t$ is positive definite, i.e., $\boldsymbol{A}_t > 0$, and invertible. Right selection of ϵ is discussed in [13]. Here, the matrix $\boldsymbol{A}_t$ is related to the Hessian of the error function, implying that the update rule uses the second order information [13].

Region boundaries are also updated in the same manner. For example, the direction vector specifying the separation function $p_{t,\Omega}$ in Fig. 1, is updated as

$$\begin{aligned}\boldsymbol{n}_{t+1,\Omega} &= \boldsymbol{n}_{t,\Omega} - \frac{1}{\eta}\boldsymbol{A}_t^{-1}\nabla e_t^2 \\ &= \boldsymbol{n}_{t,\Omega} + \frac{2}{\eta}e_t[p_{t,0}\hat{y}_{t,00} + (1-p_{t,0})\hat{y}_{t,01} \\ &\quad - p_{t,1}\hat{y}_{t,10} - (1-p_{t,1})\hat{y}_{t,11}]\boldsymbol{A}_t^{-1}\frac{\partial p_{t,\Omega}}{\partial \boldsymbol{n}_{t,\Omega}}\end{aligned} \tag{5}$$

where η is the step size to be determined, ∇ is the gradient operator w.r.t. $\boldsymbol{n}_{t,\Omega}$ and $\boldsymbol{A}_t$ is given in (4). Partial derivative of the separation function $p_{t,\Omega}$ w.r.t. $\boldsymbol{n}_{t,\Omega}$ is given by

$$\frac{\partial p_{t,\Omega}}{\partial \boldsymbol{n}_{t,\Omega}} = \frac{\boldsymbol{x}_t e^{-\boldsymbol{x}_t^T \boldsymbol{n}_{t,\Omega}}}{(1+e^{-\boldsymbol{x}_t^T \boldsymbol{n}_{t,\Omega}})^2}. \tag{6}$$

All separation functions are updated in the same manner. The final estimate of this algorithm is given by the following generic formula

$$\hat{y}_t = \sum_{j=1}^{2^d} \hat{\psi}_{t,R_d(j)} \tag{7}$$

where R_d is the set of all region labels with length d in the increasing order, i.e., $R_1 = \{0,1\}$ or $R_2 = \{00,01,10,11\}$, and $R_d(j)$ represents the j^{th} entry of the set R_d. Weighted estimate of each region is found as

$$\hat{\psi}_{t,r} = \hat{y}_{t,r} \prod_{i=1}^{d} \hat{p}_{t,r_i} \tag{8}$$

where r_i denotes the first $i-1$ character of label r as a string, i.e., $r = \{0101\}, r_3 = \{01\}$ and $r_1 = \{\Omega\}$, which is the empty string $\{\Omega\}$. Here, $\hat{p}_{t,r_i}$ is defined as

$$\hat{p}_{t,r_i} = \begin{cases} p_{t,r_i} & , r(i) = 0 \\ 1 - p_{t,r_i} & , r(i) = 1 \end{cases}. \tag{9}$$

We reformulate the update rules defined in (3) and (5) and present generic expressions for both regression weights and region boundaries. The generic update rule for the regression weights are given by

$$\begin{aligned}\boldsymbol{w}_{t+1,r} &= \boldsymbol{w}_{t,r} - \frac{1}{\beta}\boldsymbol{A}_t^{-1}\nabla e_t^2 \\ &= \boldsymbol{w}_{t,r} + \frac{2}{\beta}e_t \boldsymbol{A}_t^{-1}\frac{\partial \hat{y}_t}{\partial \boldsymbol{w}_{t,r}} \\ &= \boldsymbol{w}_{t,r} + \frac{2}{\beta}e_t \boldsymbol{A}_t^{-1}\sum_{j=1}^{2^d}\frac{\partial\left(\hat{y}_{t,R_d(j)}\prod_{i=1}^{d}\hat{p}_{t,R_d(j)_i}\right)}{\partial \boldsymbol{w}_{t,r}} \\ &= \boldsymbol{w}_{t,r} + \frac{2}{\beta}e_t \boldsymbol{A}_t^{-1}\boldsymbol{x}_t\prod_{i=1}^{d}\hat{p}_{t,r_i}\end{aligned} \tag{10}$$

and the region boundaries are updated as

$$\begin{aligned}\boldsymbol{n}_{t+1,k} &= \boldsymbol{n}_{t,k} - \frac{1}{\eta}\boldsymbol{A}_t^{-1}\nabla e_t^2 \\ &= \boldsymbol{n}_{t,k} + \frac{2}{\eta}e_t \boldsymbol{A}_t^{-1}\frac{\partial \hat{y}_t}{\partial p_{t,k}}\frac{\partial p_{t,k}}{\partial \boldsymbol{n}_{t,k}} \\ &= \boldsymbol{n}_{t,k} + \frac{2}{\eta}e_t \boldsymbol{A}_t^{-1}\left[\sum_{j=1}^{2^d}\frac{\partial \hat{\psi}_{t,R_d(j)}}{\partial p_{t,k}}\right]\frac{\partial p_{t,k}}{\partial \boldsymbol{n}_{t,k}} \\ &= \boldsymbol{n}_{t,k} \\ &\quad + \frac{2}{\eta}e_t \boldsymbol{A}_t^{-1}\left[\sum_{j=1}^{2^d}\hat{y}_{t,R_d(j)}\frac{\partial\left(\prod_{i=1}^{d}\hat{p}_{t,R_d(j)_i}\right)}{\partial p_{t,k}}\right]\frac{\partial p_{t,k}}{\partial \boldsymbol{n}_{t,k}} \\ &= \boldsymbol{n}_{t,k} \\ &\quad + \frac{2}{\eta}e_t \boldsymbol{A}_t^{-1}\left[\sum_{j=1}^{2^{d-\ell(k)}}\hat{y}_{t,\acute{r}}(-1)^{\acute{r}(\ell(k)+1)}\prod_{\substack{i=1 \\ \acute{r}_i \neq k}}^{d}\hat{p}_{t,\acute{r}_i}\right]\frac{\partial p_{t,k}}{\partial \boldsymbol{n}_{t,k}}\end{aligned} \tag{11}$$

where $\acute{r}$ is the label string generated by concatenating separation function identifier k and the label kept in j^{th} entry of the set $R_{(d-\ell(k))}$, i.e., $\acute{r} = [k; R_{(d-\ell(k))}(j)]$ and $\ell(k)$ represents the length of binary string k, e.g. $\ell(01) = 2$. Since we use the logistic regression function, we can use the following equality to calculate the partial derivative of $p_{t,k}$ w.r.t. $\boldsymbol{n}_{t,k}$,

$$\frac{\partial p_{t,k}}{\partial \boldsymbol{n}_{t,k}} = p_{t,k}(1-p_{t,k})\boldsymbol{x}_t. \tag{12}$$

In order to avoid taking the inverse of an $m \times m$ matrix, $\boldsymbol{A}_t$, at each iteration in (10) and (11), we generate a recursive formula using matrix inversion lemma for $\boldsymbol{A}_t^{-1}$ given as

$$\boldsymbol{A}_t^{-1} = \boldsymbol{A}_{t-1}^{-1} - \frac{\boldsymbol{A}_{t-1}^{-1}\nabla_t \nabla_t^T \boldsymbol{A}_{t-1}^{-1}}{1 + \nabla_t^T \boldsymbol{A}_{t-1}^{-1}\nabla_t} \tag{13}$$

where $\nabla_t \triangleq \nabla e_t^2$ w.r.t. the corresponding variable. The complete algorithm is given in Algorithm 1 with all updates and initializations.

Algorithm 1: Finest Model Partitioning

$A_0^{-1} \leftarrow \frac{1}{\epsilon}\boldsymbol{I}_m$;
for $t \leftarrow 1$ **to** n **do**
  $\hat{y}_t \leftarrow 0$;
  for $j \leftarrow 1$ **to** 2^d **do**
    $r \leftarrow R_d(j)$;
    $\hat{y}_{t,r} \leftarrow \boldsymbol{w}_{t,r}^T \boldsymbol{x}_t$;
    $\hat{\psi}_{t,r} \leftarrow \hat{y}_{t,r}$;
    $\gamma_{t,r} \leftarrow 1$;
    for $i \leftarrow 1$ **to** d **do**
      if $r(i) \leftarrow 0$ **then**
        $\hat{p}_{t,r_i} \leftarrow p_{t,r_i}$;
      else
        $\hat{p}_{t,r_i} \leftarrow 1 - p_{t,r_i}$;
      $\hat{\psi}_{t,r} \leftarrow \hat{\psi}_{t,r}\hat{p}_{t,r_i}$;
      $\gamma_{t,r} \leftarrow \gamma_{t,r}\hat{p}_{t,r_i}$;
    $\hat{y}_t \leftarrow \hat{y}_t + \hat{\psi}_{t,r}$;
  for $i \leftarrow 1$ **to** $2^d - 1$ **do**
    $k \leftarrow P(i)$;
    for $j \leftarrow 1$ **to** $2^{d-\ell(k)}$ **do**
      $r \leftarrow concat[k : R_{d-\ell(k)}(j)]$;
      $\alpha_{t,k} \leftarrow (-1)^{r(\ell(k)+1)}(\hat{\psi}_{t,r}/\hat{p}_{t,k})$;
  $e_t \leftarrow y_t - \hat{y}_t$;
  for $j \leftarrow 1$ **to** 2^d **do**
    $r \leftarrow R_d(j)$;
    $\nabla_{t,r} \leftarrow -2e_t\gamma_{t,r}\boldsymbol{x_t}$;
    $\boldsymbol{A}_{t,r}^{-1} \leftarrow \boldsymbol{A}_{t-1,r}^{-1} - \dfrac{\boldsymbol{A}_{t-1}^{-1}\nabla_{t,r}\nabla_{t,r}^T\boldsymbol{A}_{t-1,r}^{-1}}{1+\nabla_{t,r}^T\boldsymbol{A}_{t-1,r}^{-1}\nabla_{t,r}}$;
    $\boldsymbol{w}_{t+1,r} \leftarrow \boldsymbol{w}_{t,r} - \dfrac{1}{\beta}\boldsymbol{A}_{t,r}^{-1}\nabla_{t,r}$;
  for $i \leftarrow 1$ **to** $2^d - 1$ **do**
    $k \leftarrow P(i)$;
    $\nabla_{t,k} \leftarrow -2e_t\alpha_{t,k}p_{t,k}(1-p_{t,k})\boldsymbol{x_t}$;
    $\boldsymbol{A}_{t,k}^{-1} \leftarrow \boldsymbol{A}_{t-1,k}^{-1} - \dfrac{\boldsymbol{A}_{t-1,k}^{-1}\nabla_{t,k}\nabla_{t,k}^T\boldsymbol{A}_{t-1,k}^{-1}}{1+\nabla_{t,k}^T\boldsymbol{A}_{t-1,k}^{-1}\nabla_{t,k}}$;
    $\boldsymbol{n}_{t+1,k} \leftarrow \boldsymbol{n}_{t,k} - \dfrac{1}{\eta}\boldsymbol{A}_{t,k}^{-1}\nabla_{t,k}$;

The constructed algorithm partitions the regressor space into 2^d regions for the depth-d tree model. Hence, we perform $O(2^d)$ weight update at each iteration. Suppose that the regressor space is m-dimensional, i.e., $\boldsymbol{x}_t \in \mathbb{R}^m$. For each update, the proposed algorithm requires $O(m^2)$ multiplication and addition resulting from a matrix-vector product, since we apply second order update methods. Therefore, the resulting complexity is given by $O(m^2 2^d)$.

Theorem 1. *Let $\{y_t\}_{t\geq 1}$ and $\{\boldsymbol{x}_t\}_{t\geq 1}$ denote the randomly chosen real-valued data sequences. If $\|\nabla(y_t - \hat{y}_{t,r})^2\| \leq G$ and $\|\boldsymbol{w}_{t,r} - \boldsymbol{w}_r\|^2 \leq A^2$ conditions hold for some $G, A > 0$ and $\exp(-\alpha(y_t - \hat{y}_{t,r})^2)$ is concave for $\alpha > 0$, then the estimate $\hat{y}_t$ generated by following Algorithm 1, satisfies the following logarithmic bound:*

$$\sum_{t=1}^{n}(y_t - \hat{y}_{t,r})^2 - \min_{\boldsymbol{w}_r \in \mathbb{R}^m}\sum_{t=1}^{n}(y_t - \boldsymbol{w}_r^T\boldsymbol{x}_t)^2 \leq 5\left(GA + \frac{1}{\alpha}\right)m\log(n)$$

In Theorem 1, we emphasize that for the each region estimate, the regret at iteration n has a logarithmic upper bound. The proof of this theorem is accomplished by following the similar steps given in [13].

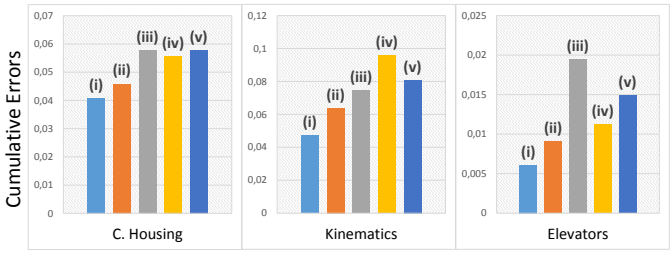

Fig. 2: Time accumulated error rates of the algorithms i) FMP, ii) DAT, iii) VF, iv) FNF, v) EMFNF for the real benchmark data sets.

IV. Simulations

In this section, we evaluate the performance of the proposed algorithm. The first set of simulations involves the well known real and synthetic benchmark data sets extensively used in the machine learning literature. We then consider the regression of a signal generated by a piecewise linear model whose partitions do not match the initial partitioning of the algorithms. Throughout this section, "FMP" represents Finest Model Partitioning algorithm, "DAT" stands for Decision Adaptive Tree [14], "CTW" is used for Context Tree Weighting [10], "GKR" represents Gaussian-Kernel regressor [15], "VF" represents Volterra Filter [16], "FNF" and "EMFNF" stand for the Fourier and Even Mirror Fourier Nonlinear Filter [17] respectively.

We first consider the regression of a benchmark real-life problem that can be found in many data set repositories such as: California Housing and Kinematics with 8-dimensional regressor spaces and Elevators with 18-dimensional regressor space [18]. For the California Housing problem, we set the learning rates to 0.004 for FMP, 0.01 the DAT, 0.05 for the VF, 0.005 for the FNF and the EMFNF. For the Kinematics and Elevators data sets, the learning rates are set to 0.01 for the DAT, 0.01 for the VF, the FNF and the EMFNF algorithms. For the FMP algorithm, it is set to 0.0625 for the Kinematics and 0.03 for the Elevators data sets. Fig. 2 illustrates the normalized time accumulated error rates of the stated algorithms. We emphasize that the proposed FMP algorithm significantly outperforms the state of the art for all the real life data sets given here.

We now consider the case where the desired data is generated by a piecewise linear model that mismatches with the initial partitioning of the proposed algorithms. Specifically, we

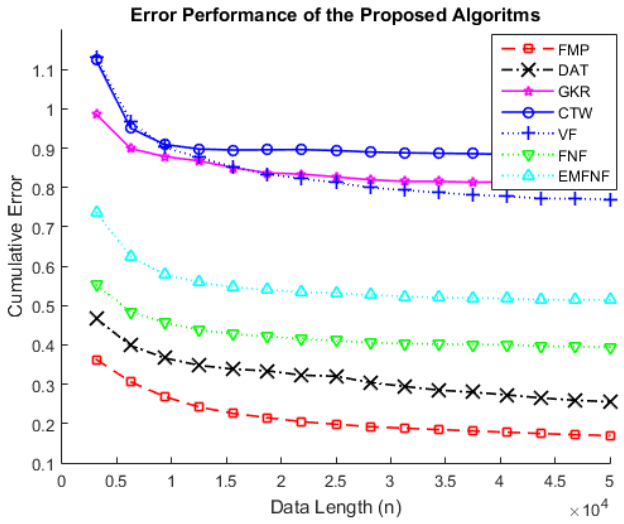

Fig. 3: Regression error performances for the mismatched partitioning case using piecewise linear model given by (14).

use the following piecewise linear model to generate the data sequence,

$$\hat{y}_t = \begin{cases} \boldsymbol{w}_1^T \boldsymbol{x}_t + \upsilon_t & , \boldsymbol{x}_t^T \boldsymbol{n}_0 \geq 0.5 \text{ and } \boldsymbol{x}_t^T \boldsymbol{n}_1 \geq -0.5 \\ \boldsymbol{w}_2^T \boldsymbol{x}_t + \upsilon_t & , \boldsymbol{x}_t^T \boldsymbol{n}_0 \geq 0.5 \text{ and } \boldsymbol{x}_t^T \boldsymbol{n}_1 < -0.5 \\ \boldsymbol{w}_2^T \boldsymbol{x}_t + \upsilon_t & , \boldsymbol{x}_t^T \boldsymbol{n}_0 < 0.5 \text{ and } \boldsymbol{x}_t^T \boldsymbol{n}_2 \geq -0.5 \\ \boldsymbol{w}_1^T \boldsymbol{x}_t + \upsilon_t & , \boldsymbol{x}_t^T \boldsymbol{n}_0 < 0.5 \text{ and } \boldsymbol{x}_t^T \boldsymbol{n}_2 < -0.5 \end{cases} \quad (14)$$

where $\boldsymbol{w}_1 = [1,1]^T$, $\boldsymbol{w}_2 = [1,-1]^T$, $\boldsymbol{n}_0 = [2,-1]^T$, $\boldsymbol{n}_1 = [-1,1]^T$ and $\boldsymbol{n}_2 = [2,1]^T$. The feature vector $\boldsymbol{x}_t = [x_{t,1}, x_{t,2}]^T$ is composed of two jointly Gaussian processes with $[0,0]^T$ mean and $\boldsymbol{I}_2$ variance. υ_t is a sample taken from a Gaussian process with zero mean and 0.1 variance. The generated data sequence is represented by $\hat{y}_t$. The learning rates maximizing the performance of each algorithm are determined as 0.04 for the FMP, 0.005 for the CTW and the FNF, 0.025 for the EMFNF and the VF, 0.5 for the GKR.

In Fig. 3, we demonstrate the normalized time accumulated error performance of the proposed algorithms. We emphasize that the CTW algorithm performs significantly worse, since the partitions do not match. Besides, the adaptive algorithms, FMP and DAT achieve considerably better performance, since these algorithms update their partitions in accordance with the data distribution. Fig. 3 exhibits that the FMP notably outperforms its competitors and even the DAT algorithm, since this algorithm exactly matches its partitioning to the partitions of the piecewise linear model given in (14) using second order update methods.

V. Concluding Remarks

In this paper, we introduce a highly efficient and effective nonlinear regression algorithm for online learning problems suitable for big data applications. We process only the currently available data for regression and then discard it, i.e., there is no need for storage. For nonlinear modeling, we use piecewise linear models, where we partition the regressor space using linear separators and fit linear regressors to each partition. As the first time in the literature, we adaptively update both the region boundaries and the linear regressors in each region using the second order methods, i.e., Newton-Raphson Methods. We illustrate that the proposed algorithm attains outstanding performance compared to the state of art even for the highly nonlinear data models. We also provide the individual sequence results demonstrating the guaranteed regret performance of the introduced algorithms without any statistical assumptions.

Acknowledgment

This work is in part supported by Turkish Academy of Sciences Outstanding Young Researcher Program and TUBITAK Contract No. 113E517.

References

[1] A. C. Singer, G. W. Wornell, and A. V. Oppenheim, "Nonlinear autoregressive modeling and estimation in the presence of noise," *Digital Signal Processing*, vol. 4, no. 4, pp. 207–221, 1994.

[2] 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.

[3] W. Cao, L. Cao, and Y. Song, "Coupled market behavior based financial crisis detection," in *The 2013 International Joint Conference on Neural Networks (IJCNN)*, Aug 2013, pp. 1–8.

[4] L. Deng, "Long-term trend in non-stationary time series with nonlinear analysis techniques," in *2013 6th International Congress on Image and Signal Processing (CISP)*, vol. 2, Dec 2013, pp. 1160–1163.

[5] K. mei Zheng, X. Qian, and N. An, "Supervised non-linear dimensionality reduction techniques for classification in intrusion detection," in *2010 International Conference on Artificial Intelligence and Computational Intelligence (AICI)*, vol. 1, Oct 2010, pp. 438–442.

[6] S. Kabbur and G. Karypis, "Nlmf: Nonlinear matrix factorization methods for top-n recommender systems," in *2014 IEEE International Conference on Data Mining Workshop (ICDMW)*, Dec 2014, pp. 167–174.

[7] L. Bottou and O. Bousquet, "The tradeoffs of large scale learning," in *Advances in Neural Information Processing (NISP)*, 2007, pp. 1–8.

[8] A. C. Singer, S. S. Kozat, and M. Feder, "Universal linear least squares prediction: upper and lower bounds," *IEEE Transactions on Information Theory*, vol. 48, no. 8, pp. 2354–2362, 2002.

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

[10] 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.

[11] D. Bertsimas and J. N. Tsitsiklis, *Introduction to linear optimization*, ser. Athena scientific series in optimization and neural computation. Belmont (Mass.): Athena Scientific, 1997. [Online]. Available: http://opac.inria.fr/record=b1094316

[12] N. Cesa-Bianchi and G. Lugosi, *Prediction, Learning, and Games*. Cambridge: Cambridge University Press, 2006.

[13] 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.

[14] N. Vanli and S. Kozat, "A comprehensive approach to universal piecewise nonlinear regression based on trees," *IEEE Transactions on Signal Processing*, vol. 62, no. 20, pp. 5471–5486, Oct 2014.

[15] R. Rosipal and L. J. Trejo, "Kernel partial least squares regression in reproducing kernel hilbert space," *J. Mach. Learn. Res.*, vol. 2, pp. 97–123, Mar. 2002. [Online]. Available: http://dl.acm.org/citation.cfm?id=944790.944806

[16] M. Schetzen, *The Volterra and Wiener Theories of Nonlinear Systems*. NJ: John Wiley & Sons, 1980.

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

[18] L. Torgo, "Regression data sets." [Online]. Available: http://www.dcc.fc.up.pt/ ltorgo/Regression/DataSets.html