Universal Nonlinear Regression on High Dimensional Data Using Adaptive Hierarchical Trees

Farhan Khan, Dariush Kari, Ilyas Alper Karatepe, and Suleyman S. Kozat, *Senior Member, IEEE*

Abstract—We study online sequential regression with nonlinearity and time varying statistical distribution when the regressors lie in a high dimensional space. We escape the curse of dimensionality by tracking the subspace of the underlying manifold using a hierarchical tree structure. We use the projections of the original high dimensional regressor space onto the underlying manifold as the modified regressor vectors for modeling of the nonlinear system. By using the proposed algorithm, we reduce the computational complexity to the order of the depth of the tree and the memory requirement to only linear in the intrinsic dimension of the manifold. The proposed techniques are specifically applicable to high dimensional streaming data analysis in a time varying environment. We demonstrate the significant performance gains in terms of mean square error over the other state of the art techniques through simulated as well as real data.

Index Terms—Big data, regression on high dimensional manifolds, online learning, tree based methods

1 Introduction

Online learning is widely investigated in adaptive signal processing [1], [2], [3], [4], [5], neural networks [6], [7], [8], [9], [10], [11], [12], [13], data engineering [14], [15], [16], [17], and machine learning [18], [19], [20] literatures and is the core for several research themes. Nonlinear models are considered for applications where linear models are inadequate. However, non-linear models usually suffer from overfitting, stability and convergence issues [1], [6], [7], [8], [9], [10]. Furthermore, for applications involving big data [14], [15], [16], for instance, when the input vectors are high dimensional, the non-linear modeling offers substantial challenges. These challenges include computational complexity, which is usually beyond manageable, and time varying statistical distributions [11].

In this paper, we study non-linear regression using high dimensional data assuming that the data lies on a manifold. We partition the regressor space into several regions to construct a piecewise linear model as an approximation of the non-linearity between the observed and the desired data. However, instead of fixing the boundaries of the regions, we partition the space in a hierarchical manner [24]. We use the notion of context trees [25], [27] to represent a broad class of all possible partitions for the piecewise linear models. We specifically introduce an algorithm that incorporates context trees for online learning of the high dimensional manifolds and perform regression on the big data. In this approach, regression directly adapts to the intrinsic lower dimension of the data while operating in the original regressor space. The algorithm achieves the performance of the best partitioning of the regressor space, competing against a broader class of piecewise linear algorithms. This broader class consists of various partitioning methods, region boundaries and regression algorithms for each region as explained later in the paper.

In most modern applications, where high dimensional data is involved, learning and regression on the manifolds are widely investigated [32], [33]. For instance, in network traffic [29], large amounts of high dimensional, time varying data from various nodes are used to identify certain trends. Another example is video surveillance [30] (which involves high dimensional, time varying images from various cameras). The high dimensional data from security cameras is analyzed for any mischievous activity in a sensitive area [31]. However, online regression on high dimensional data suffers from performance degradation and computational complexity, known as Bellman's *curse of dimensionality*, and is statistically challenging [11], [40], [41], [42]. The problem of manifold learning and regression is rather easy when all the data is available in advance (batch), and lies around the same single static submanifold [37]. In online manifold learning, however, it is difficult to track the variation in data because of the high dimensionality and time varying statistical distributions [36], [37]. Therefore, we introduce a comprehensive solution that includes online learning and adaptive regression over high dimensional feature vectors in a dynamic setting. The algorithm is best suitable for distributed/parallel learning since a linear base estimator is used for each node, that can be implemented in parallel by using different cores and the final

- *F. Khan is with the Department of Electrical and Electronics Engineering, Bilkent University, Bilkent, Ankara 06800, Turkey and the Electrical Engineering Department, COMSATS Institute of Information Technology, Pakistan. E-mail: khan@ee.bilkent.edu.tr.*
- *D. Kari and S.S. Kozat are with the Department of Electrical and Electronics Engineering, Bilkent University, Bilkent, Ankara 06800, Turkey. E-mail: {kari, kozat}@ee.bilkent.edu.tr.*
- *I.A. Karatepe is with the AveaLabs, AVEA Iletisim Hizmetleri A.S. Istanbul, Turkey. E-mail: alper.karatepe@avea.com.tr.*

Manuscript received 11 Oct. 2015; revised 8 Mar. 2016; accepted 12 Apr. 2016. Date of publication 24 Apr. 2016; date of current version 29 July 2016. Recommended for acceptance by H. Xiong.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TBDATA.2016.2555323

estimator combines these weak learners. The algorithm's low computational complexity and faster learning results in high performance computing.

For applications involving high dimensional data, various approaches are studied to escape the curse of dimensionality and perform online learning [16], [17], [21], [22], [23], [41], [42], [43]. In [36], the authors performed online tracking of high dimensional data by approximating the underlying submanifolds as union of subsets of the high dimensional space. The low dimensional approximation is used as a pre-processing step for the change point detection [37] and logistic regression [38] for high dimensional time series. In our approach, however, we use context trees to perform non-linear regression, which adapts automatically to the intrinsic low dimensionality of the data by maintaining the "geodesic distance" [34], [35] while operating on the original regressor space. Note that context trees perform a hierarchical, nested partitioning of the regressor space for the piecewise linear models [25], [27]. However, unlike [36], [37], [38] where a single partitioning is used with varying depth, context trees construct a weighted average of all possible partitions defined on a tree [25], [27], and the partitioning in [36], [37], [38] is a subclass of our tree structure. We propose an algorithm where the weights assigned to each node as well as the partitions vary according to variations in the data. Furthermore, we use a piecewise linear model, i.e., different linear regression parameters in each region defined by the tree, whereas in [38], the same logistic regression model is used in each subset or region. In this manner, our algorithm inherently uses weighted combination of all possible partitions defined by trees of various depths, and compete well against a doubly exponential class yet with a complexity linear in the depth of the tree [27]. For instance, an adaptive hierarchical tree (AHT) of depth K also inherently incorporates trees with depths less than K, i.e., $0, 1, \ldots, K-1$. This makes the algorithm suitable for various ranges of complexity in the data structure.

In the domain of online non-linear regression, context trees have been used to partition the regressor space hierarchically, and to construct a competitive algorithm among a broader class of algorithms [27]. Although we also use the context tree weighting (CTW) of Willems [25] as [27], there are major differences between our method and the method in [27]. In contrast to non-linear regression using context trees, we use a hierarchical tree structure to track and learn the manifold in a high dimensional setting. We use the regions defined by the tree to learn the underlying low dimensional projection and perform piecewise linear regression whereas in [27], the tree is used to learn the actual piecewise regions where the data lie in a $D-$dimensional space. Furthermore, the regions defined by our algorithm are time varying ellipsoids and a parent region on the tree is not necessarily a union of its children. Unlike [27], the actual data does not belong to the regions defined by the tree and we use a quadratic distance measure to decide on the membership of a data instance to a certain region. Moreover, for the test data, where the desired labels are not available, the algorithm in [27] does not update the tree structure as well as the node estimators. However, in our algorithm, we update the node performance measure by the tracking performance of the submanifold structure in the observed data. We finally use the projection of the actual data on the low dimensional regions for the regression. In addition to solving the problem of high dimensionality by incorporating manifold learning, our algorithm also performs online piecewise regression.

We first introduce an algorithm that is guaranteed to asymptotically achieve the performance of the best combination of a doubly exponential number of different models that can be represented by a depth-K tree with computational complexity only linear in the depth of the tree. We use a tree structure to hierarchically partition the high dimensional regressor space. We then incorporate an approximate Mahalanobis distance as in [36], [37], [38] to adapt the regressor space to its intrinsic lower dimension.

The Mahalanobis distance is a measure of the distance between a point P and a distribution D, which is unit-less and scale-invariant (unlike the euclidean distance), and takes into account the correlations in the data set. Therefore, for general classification of data points, the Mahalanobis distance is shown to perform better than the Minkowski distances [44]. Furthermore, we can effectively track the true curvature of each submanifold by using the Mahalanobis distance instead of the euclidean distance (as a special case of the Minkowski distance [44]), since the Mahalanobis distance takes into account the spreading of data in each direction (resulting in a non-symmetric ellipsoid for each submanifold). Our algorithm also adapts to the corresponding regressors in each region to minimize the final regression error. We then prove that as the data length increases, the algorithm achieves the performance of the best partitioning by providing an upper bound on the performance of the algorithm. We show that the method used is truly sequential and generic in the sense that it is independent of the statistical distribution or structure of the data. Moreover, the algorithm does not presume the structure or variation of the manifolds and adapts to the underlying data sequentially. In this sense, the algorithm learns $i)$ the structure of the manifolds, $ii)$ the structure of the tree, $iii)$ the low dimensional projections in each region, $iv)$ the linear regressors in each region, and $v)$ the linear combination weights of all possible partitions, to minimize the final regression error. We also show that a perfect manifold tracking for the regression purpose is not only unnecessary but also increases the complexity of the algorithm and may even increase the final error due to overfitting. This is in contrast to [37], [38] where the ultimate goal is to reduce the tracking error as much as possible.

The paper is organized as follows. In Section 2, we formally describe the problem setting in detail. We model the non-linear regression on big data mathematically and propose piecewise models to approximate the non-linear model. We discuss piecewise linear regression and introduce the context tree algorithm for piecewise linear regression. In Section 3, we extend the context tree algorithm to the high dimensional case and describe the tools that we use such as approximate Mahalanobis distance, and define our parameters. Then, we formally propose our algorithm. In Section 4, we perform online regression on several synthetic high dimensional datasets using the proposed algorithm. We also provide the performance analysis of our proposed algorithm over benchmark real data sets, e.g., computer activity [46] and KDD CUP99 [47]. We analyze

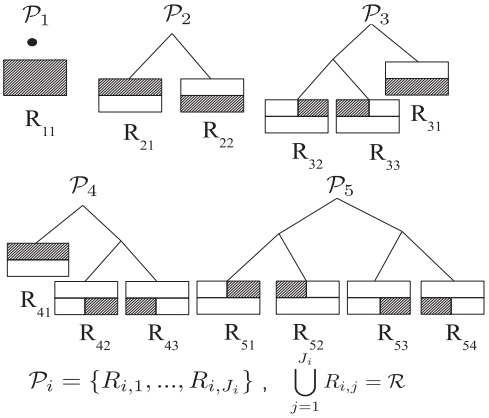

Fig. 1. A full tree of depth 2 that represents all possible partitions of the two dimensional space, $\mathcal{P} = \{\mathcal{P}_1, ..., \mathcal{P}_{N_K}\}$ and $N_K \approx (1.5)^{2^K}$, where K is the depth of the tree. Here $N_K = 5$.

the performance of our algorithm using computational complexity and mean square error (MSE) by comparing them with the previously proposed algorithms and demonstrating significant gains [27], [28].

2 Problem Description

All vectors used in this paper are column vectors, denoted by boldface lowercase letters. Matrices are denoted by boldface uppercase letters. For a vector $\mathbf{v}$, $\|\mathbf{v}\|^2 = \mathbf{v}^T\mathbf{v}$ is squared euclidean norm and $\mathbf{v}^T$ is the ordinary transpose. $\mathbf{I}_k$ represents a $k \times k$ identity matrix.

We investigate online non-linear regression using high dimensional data, i.e., when the dimension of data $D \gg 1$. We observe a desired sequence $\{y[n]\}_{n\geq 1}$, $y[n] \in \mathrm{IR}$, and regression vectors $\{\mathbf{x}[n]\}_{n\geq 1}$, $\mathbf{x}[n] \in \mathrm{IR}^D$, where D denotes the *ambient dimension*. The data $\mathbf{x}[n]$ are measurements of points lying on a submanifold $S_{m[n]}$, where the subscript $m[n]$ denotes the time varying manifold, i.e., $\mathbf{x}[n] \in S_{m[n]}$. The *intrinsic dimension* of the submanifolds $S_{m[n]}$ are d, where $d \ll D$. The submanifolds $S_{m[n]}$ can be time varying. At each time n, a vector $\mathbf{x}[n]$ is observed. The estimate of the desired sequence $y[n]$ is given by:

$$\hat{y}[n] = f_n(\mathbf{x}[n]), \tag{1}$$

where $f_n(.)$ is a non-linear, time varying function. The instantaneous regression error is given by: $e[n] = y[n] - \hat{y}[n]$.

The non-linear model of (1) could establish a perfect fit to the underlying relationship between the desired and observed data in certain situations. However, identifying this non-linear relationship could be challenging, and it may be unnecessary and computationally complex to use the perfect model [41]. Furthermore, the non-linear model of (1) may suffer from overfitting, stability and convergence issues [1]. Therefore, we use a piecewise linear model as an approximation of the non-linear relationship between the observed sequence and the desired data. We begin our discussion of piecewise linear modeling with a fixed partitioning of the regressor space, i.e., IR^D. We next use the context tree algorithm to include arbitrary partitions from a large class of possible partitions, as explained later in the Section 2.1. Finally, we extend our results to the case when the input data is high dimensional.

In piecewise linear modeling, we partition the regressor space into J regions, where a linear relationship is assumed between the desired data and the observed data within each region. Since the statistical distribution of data and the dimension of regressor space vary with time, our partitioning method as well as the linear model in each region should be dynamic. To this end, we use a tree structure to hierarchically partition the regressor space. One such tree structure to partition a two dimensional regressor space is shown in Fig. 1. Here, a depth-2 tree is used to partition the $\mathbb{R}^2$ regressor space, i.e., $D = 2$ for this figure. We define a "partition" of the D-dimensional regressor space as a specific partitioning $\mathcal{P}_i = \{R_{i,1}, \ldots, R_{i,J_i}\}$, where $\bigcup_{j=1}^{J_i} R_{i,j} = \mathcal{R}$, $R_{i,j}$ is a region in the D-dimensional regressor space and $\mathcal{R} \in \mathrm{IR}^D$ is the complete D-dimensional regressor space. Fig. 1 shows all possible partitionings of the two dimensional regressor space with a tree of depth-2. In general, for a tree of depth K, there are as many as 1.5^{2^K} possible partitions, $\mathcal{P}_i$, where $i \in \{1, \ldots, 1.5^{2^K}\}$. Each of these doubly exponential number of partitions can be used to construct a piecewise linear model.

To clarify the notation, as an example, we consider a sample partition $\mathcal{P}_3$ in Fig. 1, where the regressor space is divided into three regions. At jth region of a specific partition, e.g., $\mathcal{P}_3$, we generate the estimate:

$$\hat{y}_j[t] = \mathbf{x}^T[t]\mathbf{v}_j[t], \tag{2}$$

where $\mathbf{v}_j[t]$ is the regressor weight vector for the jth region, $t = \{n; \mathbf{x}[n] \in R_j\}$, $j \in \{1, \ldots, J\}$ and $J = 3$ is the number of regions the regressor space is divided into by the partition $\mathcal{P}_3$. The final estimate of $y[n]$ is given by:

$$\hat{y}_{\mathcal{P}_3}[n] = \mathbf{x}^T[n]\mathbf{v}_j[n], \tag{3}$$

for $j \in \{1, 2, 3\}$ when $\mathbf{x}[n] \in R_j$. For a fixed partition, e.g., $\mathcal{P}_3$, we try to achieve the performance of the best piecewise linear regressor. We then extend these results to all possible partitions.

We try to achieve the performance of the best piecewise linear model when there is a large class of possible partitions of the regressor space, i.e., over all $\mathcal{P}_i$. We specifically minimize the following regret over any n [27]:

$$\sum_{t=1}^{n}(y[t] - \hat{y}_q[t])^2 - \inf_{\mathcal{P}_i}\sum_{t=1}^{n}(y[t] - \hat{y}_{\mathcal{P}_i}[t])^2, \tag{4}$$

where $\hat{y}_{\mathcal{P}_i}[t]$ is the estimation of $y[t]$ from the partition $\mathcal{P}_i$, $i = 1, \ldots, 1.5^{2^K}$ and $\hat{y}_q[i]$ is the estimation from a sequential algorithm. We seek a sequential algorithm that can estimate the desired sequence $y[n]$ from $\mathbf{x}[n]$, as well as the best piecewise linear model. However, instead of brute-forcing over all possible partitionings, we seek an algorithm with linear complexity in the depth of the tree. For this purpose, we use a context tree approach [25], [27].

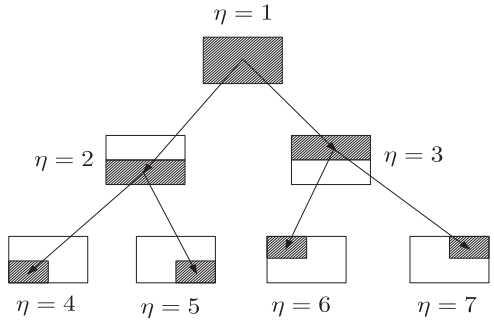

Fig. 2. A two dimensional context tree of depth 2.

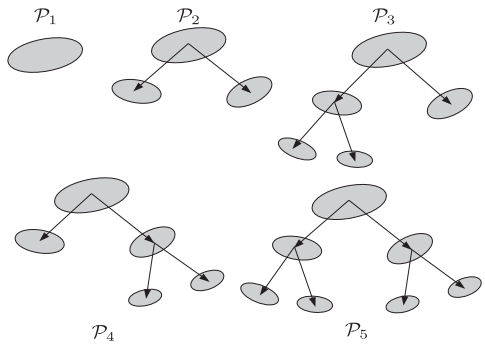

Fig. 3. A doubly exponential number of partitions defining the piecewise models on time varying submanifold, each leaf node represents a subset defined by (6).

2.1 Context Tree Algorithm for Piecewise Linear Regression

The context tree algorithm achieves the performance of the best partition among the doubly exponential class of partitions with a complexity linear in the depth of the tree [27]. We use a full tree of depth K with up to 2^K finest partition bins as shown in Fig. 2. Each node $\eta = 1, \ldots, 2^{K+1} - 1$ on the tree represents a certain region among the $2^{K+1} - 1$ regions. The 2^K nodes corresponding to the finest partitioning of the regressor space are called the leaf nodes. The union of two leaf nodes η_l and η_u is a node one level above these nodes and is called the parent node of η_l and η_u, i.e., $R_{\eta_p} = R_{\eta_l} \cup R_{\eta_u}$. If a data sample $\mathbf{x}[n] \in R_{\eta^*}$, where η^* is one of the leaf nodes, it also belongs to the ancestor nodes of η^*. In principle, $\mathbf{x}[n]$ is an element of a single node on each level of tree. These $K + 1$ nodes are called the "dark nodes". We define the set of dark nodes by $\mathcal{K} \triangleq \{\eta^*, \mathsf{fix}(\eta^*/2), \mathsf{fix}(\eta^*/4), \ldots, 1\}$, where we use the MATLAB notation \mcodefix(.) that rounds off the expression to the nearest integer towards zero. Linear regression is applied on each dark node and the final estimate is computed as a weighted combination of estimates from each of these $K + 1$ regressors,

$$\hat{y}[n] = \sum_k \omega_k[n-1]\hat{y}_k[n], \tag{5}$$

where $k \in \mathcal{K}$, the weights $\omega_k[n-1]$ correspond to the performance of each node in the past and the regression error is used as a performance measure [27]. Here, $\hat{y}_k[n]$ is the estimate of $y[n]$ by the regressor of node k. As an example, in the beginning of the adaptation when there is a small amount of input data available, the nodes on the upper level of the tree may perform better and are given more weights [25], [27]. The calculation and update of these weights is explained in Section 3.2.

However, if the input regressor vectors are high dimensional, i.e., $D \gg 1$, the regression process can be challenging due to the curse of dimensionality [11], [41]. Hence, we dynamically map the high dimensional input vectors to lower dimension for the regression. We introduce an algorithm that performs piecewise linear regression in the high dimensional setting while inherently exploiting the underlying manifold structure. Furthermore, in contrast to the context tree algorithm, where regions of each partition are fixed, we learn these regions dynamically according to the submanifold variation in the data. In the following sections, we propose an algorithm that uses adaptive hierarchical trees tuned to the dynamics of the data and performs piecewise linear regression.

3 Manifold Learning and Regression Using Adaptive Hierarchical Trees

To escape the curse of dimensionality, we perform regression on high dimensional data by mapping the regressor vectors to low dimensional projections. We assume that the observed data $\mathbf{x}[n] \in \mathbb{R}^D$ lies on time varying submanifolds $S_{m[n]}$. We can solve the problem of non-linear regression by using piecewise linear modeling as explained in Section 2.1, where the regressor space, i.e., $\mathbb{R}^D$ can be partitioned into several regions. However, in the new setting, since the data lies on submanifolds with a lower intrinsic dimension, we use the lower dimensional projections instead of the original $\mathbb{R}^D$ regressor space. We define the piecewise regions in $\mathbb{R}^d$ for each node that correspond to the low dimensional submanifolds. However, since the submanifolds are time varying, the regions are not fixed. We define these regions by the subsets [37], [38]:

$$\begin{aligned}\Re_j[n] = \{\mathbf{x}[n] \in \mathbb{R}^{\mathrm{D}} : \mathbf{x}[n] = \mathbf{Q}_j[n]\boldsymbol{\beta}_j[n] + \mathbf{c}_j[n],\\ \boldsymbol{\beta}_j^T[n]\boldsymbol{\Lambda}_j^{-1}[n]\boldsymbol{\beta}_j[n] \leq 1, \boldsymbol{\beta}_j[n] \in \mathbb{R}^d\},\end{aligned} \tag{6}$$

where each subset $\Re_j[n]$ is a $d-$dimensional ellipsoid assigned to each node of the tree. The matrix $\mathbf{Q}_j[n] \in I\!R^{D\times d}$ is the subspace basis in $d-$dimensional hyperplane and the vector $\mathbf{c}_j[n]$ is the offset of the ellipsoid from the origin. The matrix $\boldsymbol{\Lambda}_j[n] \triangleq \mathrm{diag}\{\lambda_j^{(1)}[n], \ldots, \lambda_j^{(d)}[n]\}$ with $\lambda_j^{(1)}[n] \geq \ldots \geq \lambda_j^{(d)}[n] \geq 0$, contains the eigen-values of the covariance matrix of the data $\mathbf{x}[n]$ projected onto each hyperplane. The subspace basis $\mathbf{Q}_j[n]$ specify the orientation or direction of the hyperplane and the eigen-values specify the spread of the data within each hyperplane [37], [38]. The projections of $\mathbf{x}[n]$ on the basis $\mathbf{Q}_j[n]$ are used as new regression vectors, $\tilde{\mathbf{x}}_j[n] = \mathbf{Q}_j^T[n]\mathbf{x}[n]$ and $\boldsymbol{\beta}_j[n] = \tilde{\mathbf{x}}_j[n] - \mathbf{c}_j[n]$. We represent these regions by a tree structure, where the leaf nodes correspond to these regions.

We use a tree structure to learn the underlying time varying manifold structure and the best piecewise linear model, however, instead of using Fig. 3, we use the notion of context trees given in Fig. 2 that represents the doubly exponential class in an efficient manner. We emphasize that each node on the tree is assigned a particular subset $\Re_\eta$, with

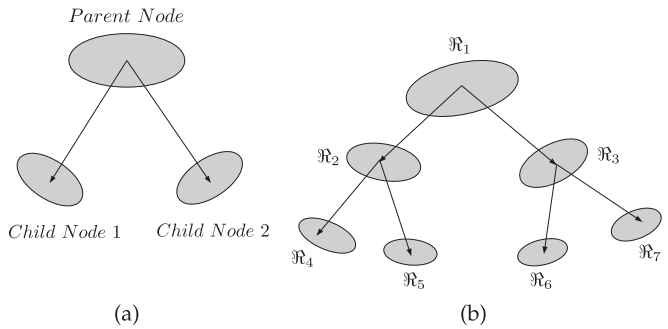

Fig. 4. (a) A parent node and its two children in an adaptive hierarchical tree. Each node (subset) is a two dimensional ellipsoid defined by its parameters $\{\mathbf{Q}_\eta[n], \mathbf{\Lambda}_\eta[n], \mathbf{c}_\eta[n]\}$. (b) A dynamic hierarchical tree of depth K where each η represents a subset defined by (6) and $\eta \in \{1, 2, ..., 2^{K+1} - 1\}$.

$\eta \in \{1, \ldots, 2^{K+1} - 1\}$, in a hierarchical manner. However, unlike a regular tree introduced in Section 2.1, in this framework, the subset belonging to the parent node is not the union of its children nodes. The subset belonging to the parent node does not cover the space spanned by its two children and may not be in the same space as shown in Fig. 4a. Moreover, the subsets defined by the nodes of the tree are low dimensional submanifolds while the actual data is of high dimension. In this sense, as shown in this paper, we can update the tree according to the variation in the observed data. We next use adaptive hierarchical trees for the partitioning of high dimensional regressor space and learning the submanifolds. We show that the proposed algorithm significantly improves the performance of piecewise linear regression operating in the high dimensional setting.

The adaptive hierarchical tree shown in Fig. 4b partitions the time varying manifold into $d-$dimensional subsets. However, the regions belonging to each subset are not fixed. Each node of the tree, $\eta \in \{1, \ldots, 2^{K+1} - 1\}$, corresponds to a $d-$dimensional ellipsoid, $\Re_\eta[n]$, with parameters $\{\mathbf{Q}_\eta[n], \mathbf{\Lambda}_\eta[n], \mathbf{c}_\eta[n]\}$ as defined in (6). These subsets are evolving in time according to the dynamics of the data, hence their parameters, $\{\mathbf{Q}_\eta[n], \mathbf{\Lambda}_\eta[n], \mathbf{c}_\eta[n]\}$, are adaptively updated with time. In the following, we explain that instead of updating all the subsets, we only update the recent $K+1$ subsets that are defined by the dark nodes defined in Section 2.1. Each subset partially contributes to approximate the underlying submanifold with the leaf nodes representing finer approximations. The levels of the tree are chosen dynamically according to the curvature of the submanifolds. However, at a certain time n, instead of choosing a single level, we use the context tree weighting method [25] to assign weights to the subsets on each level, i.e., we use all possible partitions. We assign more weight to the children node when there is more curvature in the submanifold. The nodes of the tree structure shown in Fig. 4b represent ellipsoids that may all be in different $d-$dimensional subspaces.

The actual data samples $\mathbf{x}[n] \in \mathbb{R}^D$ do not lie in the space of $\Re_\eta[n]$ as $\Re_\eta[n] \in \mathbb{R}^d$. Therefore, we seek to determine the nearest region among the subsets in terms of a certain distance measure. After selecting the nearest regions, we then use the projection of $\mathbf{x}[n]$ on the specific region as our regressor input vectors,

$$\tilde{\mathbf{x}}_\eta[n] = \mathbf{Q}_\eta^T[n]\mathbf{x}[n], \tag{7}$$

where $\mathbf{Q}_\eta[n]$ is the basis for the region $\Re_\eta[n]$. We proceed to use the context tree algorithm for piecewise linear regression, given in [25], [27] that is briefly explained in Section 2.1.

With the arrival of a new data sample $\mathbf{x}[n]$, we determine which region $\Re_\eta[n]$ among the leaf nodes $\eta \in \{2^K, \ldots, 2^{K+1} - 1\}$ it is nearest to by calculating the quadratic distance between $\mathbf{x}[n]$ and $\Re_\eta$, i.e.,

$$\eta^* = \arg\min_\eta D_M(\mathbf{x}[n], \Re_\eta[n]), \tag{8}$$

where $D_M(\mathbf{x}[n], \Re_\eta[n])$ is the distance between $\mathbf{x}[n]$ and the subset $\Re_\eta[n]$, $\eta = 2^K, \ldots, 2^{K+1} - 1$. We use the approximate Mahalanobis distance [36], [37], [38] as the distance measure [see Appendix A, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TBDATA.2016.2555323]. The approximate Mahalanobis distance is defined by,

$$D_M(\mathbf{x}, \Re) \triangleq \delta(\mathbf{x} - \mathbf{c})^T \mathbf{Q}_1 \Lambda_1^{-1} \mathbf{Q}_1^T (\mathbf{x} - \mathbf{c}) + \|\mathbf{Q}_2^T(\mathbf{x} - \mathbf{c})\|^2, \tag{9}$$

where $\Lambda_1 = \text{diag}\{\lambda^{(1)}, \ldots, \lambda^{(d)}\}$, $\lambda^{(1)}, \ldots, \lambda^{(d)}$ are the largest d eigenvalues of the covariance matrix of $\mathbf{x} \in \Re$ with mean $\mathbf{c}$, and $\mathbf{Q}_1$ is the corresponding eigenvector matrix and is the subspace basis for the $d-$dimensional hyperplane, $\delta > 0$ is the average of the remaining eigenvalues, typically a small number and $\mathbf{Q}_2$ is the corresponding eigenvector matrix representing the residual subspace basis [37]. We use the minimum distance node, η^*, as the $\{K+1\}$th dark node in the context tree algorithm [27] and the rest of K dark nodes are the ancestor nodes of η^* till the root node $\eta = 1$. For instance, in Fig. 4b, if $\Re_{\eta^*} = \Re_7$, then the remaining K dark node regions are $\Re_3$ and $\Re_1$. We then project the observed sample $\mathbf{x}[n] \in I\!R^D$ on the basis of each dark node k for $k \in \mathcal{K}$, the set of dark nodes defined in Section 2.1, i.e.,

$$\tilde{\mathbf{x}}_k[n] = \mathbf{Q}_k^T[n]\mathbf{x}[n], \tag{10}$$

where $\tilde{\mathbf{x}}_k[n] \in I\!R^d$. We train a linear regressor using each $\tilde{\mathbf{x}}_k[n]$ to learn the regressor weight vectors and estimate $y[n]$:

$$\mathbf{w}_k[n] = \mathbf{w}_k[n-1] + \nu\tilde{\mathbf{x}}_k[n](y[n] - \mathbf{w}_k^T[n-1]\tilde{\mathbf{x}}_k[n]), \tag{11}$$

where ν is the step-size of the Least Mean Square (LMS) algorithm and $\mathbf{w}_k[n] \in I\!R^d$ is the regressor weight vector for node k, $k \in \mathcal{K}$. The estimate of $y[n]$ from each dark node regressor is given by:

$$\hat{y}_k[n] = \mathbf{w}_k^T[n]\tilde{\mathbf{x}}_k[n]. \tag{12}$$

Finally, we use the context tree weighting method to estimate $y[n]$ as a weighted combination of the estimates of the dark nodes using (5). The complete algorithm is given in Algorithm 1. The construction of the proposed algorithm is based on the following theorem whereas the complexity of the algorithm is linear in the depth of the hierarchical tree structure.

Theorem 1. *Let $\{\mathbf{x}[n]\}_{n\geq 1}$ is the observed $D-$ dimensional sequence and $\{y[n]\}_{n\geq 1} \in I\!R$ is the desired sequence, where $|y[n]| \leq A_y$. Then the Algorithm 1, whose complexity is linear in the depth of the tree, yields*

$$\sum_{n=1}^{N}(y[n]-\hat{y}[n])^2 \le \min_{\mathcal{P}_i}\Bigg(\sum_{n=1}^{N}(y[n]-\hat{y}_{\mathcal{P}_i}[n])^2 + 8A_y^2\mathbb{C}(\mathcal{P}_i)\ln(2)\Bigg)+O(1), \tag{13}$$

for any N, where $\hat{y}[n]$ is the estimate of $y[n]$ as given by (5), $\mathcal{P}_i$ is the ith partition from the doubly exponential class of Fig. 3 with the cost of partition $\mathbb{C}(\mathcal{P}_i)$, and $\hat{y}_{\mathcal{P}_i}[n]$ is the estimate of $y[n]$ by the partition $\mathcal{P}_i$. The cost of partition $\mathcal{P}_i$ is given by, $\mathbb{C}(\mathcal{P}_i)=J_i+\eta_i-1$, where J_i is the number of regions in the partition $\mathcal{P}_i$ and η_i is the number of branches of the tree that are not fully grown [26], [27].

This theorem is a basic application of Theorem 1 of [27]. The weights $\omega_k[n-1]$ in (5) assigned to each dark node regressors are determined by the performance of these nodes until the current time. Therefore, we sequentially measure the performance of each node that is used for estimation and update them after each iteration. In the next section we explain how to measure the performance of each node in estimating the desired sequence $y[n]$. We then use this performance measure to assign combination weights to each node.

3.1 Node Performance Measure

In our method, instead of using fixed partitions for the piecewise linear regression and manifold learning, we use a tree structure that dynamically partitions the regressor space on each level of the tree. We then use the context tree weighting method to linearly combine the estimates of each node. The weights assigned to each node in (5) are determined by the node performance in the previous iterations. We assign C_η to each node as a measure of performance, which is an exponential function of the regret $\sum_{i=1}^{n_\eta-1}\big(y[i]-\hat{y}_\eta[i]\big)^2$. These C_η are used to calculate the weight or portion of each node regressor in the mixture of (5). The universal performance measure, C_u is a weighted combination of all C_η below the root node and $C_r=C_u$, where C_r represents the performance of the root node [27]. We represent the desired data $y[t]$ for $t=1,\ldots,n$ by y^n, i.e., $y^n=\{y[1],y[2],\ldots,y[n]\}$. For a specific partition, $\mathcal{P}_i$, among the doubly exponential class of possible partitions, the performance is measured by [27]:

$$C(y^n|\hat{y}^n,\mathcal{P}_i)\triangleq\exp\left\{-\frac{1}{2a}\sum_{t=1}^{n}(y[t]-\hat{y}[t])^2\right\}, \tag{14}$$

which is the performance of the partition $\mathcal{P}_i$ in estimating the desired sequence y^n. Here, a is a constant that depends on $A_y=\max\{|y[n]|\}$ and $a\triangleq 4A_y^2$ [27]. For a given $\mathcal{P}_i$, the best predictor is the one with the minimum loss function, $\sum_{t=1}^{n}(y[t]-\hat{y}[t])^2$, i.e.,

$$C^*(y^n|\mathcal{P}_i)\triangleq\exp\left(-\frac{1}{2a}\min_{\hat{y}^n}\sum_{t=1}^{n}(y[t]-\hat{y}[t])^2\right). \tag{15}$$

The best partition can be chosen among all $\mathcal{P}_i$ by maximizing $C^*(y^n|\mathcal{P}_i)$ over all $\mathcal{P}_i$, i.e., $C^*(y^n|\mathcal{P}_i^*)\triangleq\max_{\mathcal{P}_i}C^*(y^n|\mathcal{P}_i)$.

In the context tree algorithm, the performance measure of a leaf node is defined as [27]:

$$\tilde{C}_\eta(y^n)\triangleq\exp\left(-\frac{1}{2a}\sum_{t=1}^{n_\eta}(y[t]-\hat{y}_\eta[t])^2\right), \tag{16}$$

where n_η are the number of past input samples closest to the leaf node η. Here, $\hat{y}_\eta[t]$ is the estimate of $y[t]$ from the node η and is given by (12). The performance measure of an inner node is defined as [27],

$$\tilde{C}_\eta(y^n)\triangleq\frac{1}{2}\tilde{C}_{\eta_u}(y^n)\tilde{C}_{\eta_l}(y^n) +\frac{1}{2}\exp\left(-\frac{1}{2a}\sum_{t=1}^{n_\eta}(y[t]-\hat{y}_\eta[t])^2\right), \tag{17}$$

which is a weighted combination of the performance measures assigned to the node η and its two children nodes, η_u and η_l. This way we define the universal performance measure as a weighted combination of all the leaf and inner nodes. The universal performance measure [27] for $y[n]$, given past observations till $n-1$, is given by:

$$\tilde{C}_u(y[n]|y^{n-1})=\sum_k\mu_k[n-1]\exp\left(-\frac{1}{2a}l(y^{n-1},\tilde{y}_k^{n-1})\right), \tag{18}$$

where $l(y^{n-1},\tilde{y}_k^{n-1})=(y[n-1]-\tilde{y}_k[n-1])^2$. The weights $\mu_k[n-1]$ are defined as:

$$\mu_k[n-1]\triangleq\frac{\sigma_k[n-1]\exp\left(-\frac{1}{2a}\sum_{t=1}^{n_k-1}(y[t]-\tilde{y}_k[t])^2\right)}{\tilde{C}_u(y^{n-1})} \tag{19}$$

where $\sigma_1[n-1]=\frac{1}{2}$ and $\sigma_k[n-1]=\frac{1}{2}\tilde{C}_s[n-1]\sigma_{k-1}[n-1]$ for $k>1$. Here, $\tilde{C}_s$ denotes the performance measure of the sibling node of k. The estimate of $y[n]$ is given by the weighted combination of the regressor outputs from each dark node,

$$\tilde{y}[n]=\sum_k\mu_k[n-1]\hat{y}_k[n] \tag{20}$$

which is the same as (5) with $\omega_k[n]=\mu_k[n]$ and $\mu_k[n]$ are defined in (19).

Alternatively, we can use the approximate Mahalanobis distance (9) to define the node performance measure. For the leaf nodes,

$$\tilde{C}_\eta(y^n)\triangleq\exp\Big(-\sqrt{D_M(\mathbf{x}[n],\Re_\eta[n])}\Big), \tag{21}$$

and for the inner nodes,

$$\tilde{C}_\eta(y^n)=\frac{1}{2}\tilde{C}_{\eta_u}(y^n)\tilde{C}_{\eta_l}(y^n) +\frac{1}{2}\exp\Big(-\sqrt{D_M(\mathbf{x}[n],\Re_\eta[n])}\Big). \tag{22}$$

In the next section, we describe the update mechanism for all the parameters of the submanifolds and the nodes.

3.2 Update Node Parameters

There are two layers of parameters which we update before the next data sample $\mathbf{x}[n+1]$ arrives, i) the submanifold

shape parameters $\{\mathbf{Q}_\eta, \mathbf{c}_\eta, \mathbf{\Lambda}_\eta\}$ and $ii)$ the context tree and regressor parameters, i.e., C_η and $\mathbf{w}_\eta$. Since $\mathbf{x}[n]$ only affects the performance measure and regressors associated with the dark nodes of the context tree, we use a greedy approach instead of updating all $2^{K+1}-1$ nodes, and update the tree only for the dark nodes in $K+1$ operations [27], [36], [37]. For the subsets shape parameters associated with each node, i.e., $\mathbf{Q}_\eta, \mathbf{c}_\eta, \mathbf{\Lambda}_\eta$, we update $\mathbf{Q}_\eta$ while keeping $\mathbf{\Lambda}_\eta$ and mean vectors $\mathbf{c}_\eta$ fixed. Then we update $\mathbf{\Lambda}_\eta$ and $\mathbf{c}_\eta$ by considering $\mathbf{Q}_\eta$ fixed and using the data sample $\mathbf{x}[n]$ and the projections $\boldsymbol{\beta}_\eta$. These updates are detailed in the Algorithm 2. For instance, we update $\mathbf{c}_\eta[n]$ as follows:

$$\mathbf{c}_\eta[n] = \alpha \mathbf{c}_\eta[n-1] + (1-\alpha)\mathbf{x}[n], \tag{23}$$

where $0 < \alpha \leq 1$ is a small positive constant that determines the dependence of $\mathbf{c}_\eta$ on its past values. In general, α should be close to 1 as in the Recursive Least Squares (RLS) algorithm [4], [5]. We choose small α for the fast evolving manifold and large α for the slow evolving manifold [37].

We update the subspace basis $\mathbf{Q}_\eta$ for each dark node using Parallel Estimation and Tracking by REcursive Least Squares with fast orthogonalization (PETRELS-FO) [37], [45]. The details of using the subspace tracking algorithm can be found in [36], [37], [38], [45]. After updating the node parameters for the dark nodes, Algorithm 1 is repeated for the next data sample, $\mathbf{x}[n+1]$ to estimate $y[n+1]$. Our algorithm achieves the performance of the best partition among the doubly exponential class with a complexity linear in the depth of the hierarchical tree structure, as given by the Theorem 1.

Remark. Note that the actual regression value $y[n]$ is needed to update the performance measure C of each node. However, if we use the tree for classification purposes, then we can choose the closest value to $\hat{y}[n]$ among all possible values as the true value of $y[n]$ in the decision directed mode [39]. Nevertheless, even for the regression task, once the training phase is complete, our algorithm still adapts to the structure of the data $\mathbf{x}[t]$, i.e., the node shape parameters $\mathbf{Q}_\eta, \Lambda_\eta, \mathbf{c}_\eta$ are updated based on the new data sample $\mathbf{x}[t]$. To update the node performance measure C when $y[n]$ is not available, we use 21 and (22).

Remark. Note that if we use the euclidean distance to decide on the membership of the data to the regions, we may end up having two very similar nodes, since we update only the dark nodes at each iteration. However, since we use the Mahalanobis distance, we do not encounter this issue. This is because the Mahalanobis distance takes into acount the correlation among data points. Hence, if the means of the data assigned to two different nodes are very similar (i.e., the euclidean distance between the means is small), the Mahalanobis distance between them is still large enough. Hence, the nodes may not be very similar.

Algorithm 1. Main Algorithm

Variables:

$\eta = 1, \ldots, 2^{K+1}-1$: All node indices, complete tree of depth K.
$C_\eta[n-1] \triangleq \tilde{C}_\eta(y^{n-1})$: Total node confidence
$E_\eta[n-1] \triangleq \exp\left(-\frac{1}{2a}\sum_{t=1}^{n_\eta^{-1}}(y_\eta[t] - \mathbf{w}_\eta^T[t-1]\tilde{\mathbf{x}}_\eta[n])^2\right)$: Prediction performance of node η
$y_\eta[n-1] \triangleq \mathbf{w}_\eta^T[n-1]\boldsymbol{\beta}_\eta[n]$: Prediction of node η for $y[n]$
$\boldsymbol{\beta}_\eta[n] = \mathbf{Q}_\eta^T[n](\mathbf{x}[n] - \mathbf{c}_\eta[n])$
$\tilde{\mathbf{x}}_\eta[n] = \mathbf{Q}_\eta^T[n]\mathbf{x}[n]$: Projection of $\mathbf{x}$ on the basis of η
$\boldsymbol{\gamma}[n] = (\mathbf{I} - \mathbf{Q}_\eta[n]\mathbf{Q}_\eta^T[n])(\mathbf{x}[n] - \mathbf{c}_\eta[n])$: Projection residual
$\mathbf{w}_\eta[n] = \mathbf{w}_\eta[n-1] + \nu\tilde{\mathbf{x}}_\eta[n](y[n] - \mathbf{w}_\eta^T[n-1]\tilde{\mathbf{x}}_\eta[n])$: regressor weight vectors for each node η. $\mathbf{w}_\eta[n] \in R^d$
$\delta, \delta_1, \delta_2$: small positive real constants
$\mathbf{d}(k)$: kth component of vector $\mathbf{d}$
$\mathbf{Q}_\eta$: basis of the subspace with node index η
$\Re_\eta[n]$: Subsets of the regressor space in R^D
$D_M(\mathbf{x}[n], \Re_\eta[n]) = \boldsymbol{\beta}_\eta^T[n]\Lambda_\eta^{-1}[n]\boldsymbol{\beta}_\eta[n] + \delta_\eta^{-1}[n]\|\mathbf{x}_\perp[n]\|^2$
$\eta_l = 1, \ldots, 2^K$: Leaf nodes
$\eta^* = \arg\min_{\eta_l} D_M(\mathbf{x}[n], \Re_{\eta_l}[n])$

Initialization:

for $\eta = 1$ to $2^{K+1}-1$; **do**
  $C_\eta[0] = \delta_1^{-1}$
  $E_\eta[0] = \delta_2^{-1}$
  $\hat{y}_\eta[0] = 0$
  $\mathbf{w}_\eta[0] = \mathbf{0}$ (initial weight vector for node η)
end for
for $k = 1, \ldots, K+1$; **do**
  $\mu_k[0] = 0, \sigma_k[0] = 0$
  $\Lambda_\eta[0] = diag\{\lambda_1, \ldots, \lambda_d\}$: here $\lambda_1, \ldots, \lambda_d$ are the eigen-values of covariance matrix Σ of initial training samples.
  $\mathbf{c}_\eta[0] = \mathbb{E}\{\mathbf{x}\}$: $\mathbf{x}$ are the training samples and $\mathbf{x} \in R_\eta$
  $\mathbf{Q}_\eta[0]$ = eigen-vector matrix of covariance matrix Σ
end for

Algorithm:

for $n = 1, \ldots, N$, **do**
  $\mathbf{d} = [\,]$; vector containing indices of dark nodes.
  for $\eta = 2^K, 2^K+1, \ldots, 2^{K+1}-1$: i.e. 2^K leaf nodes **do**
    $D_M(\mathbf{x}[n], S_\eta[n-1]) = \boldsymbol{\beta}_\eta^T[n-1]\Lambda_\eta^{-1}[n-1]\boldsymbol{\beta}_\eta[n-1] + \delta_\eta^{-1}[n-1]\|\boldsymbol{\gamma}_\eta[n-1]\|^2$
  end for
  $\mathbf{d}(K+1) = \arg\min_\eta D_M(\mathbf{x}[n], S_\eta[n-1])$ for $\eta = 2^K, 2^K+1, \ldots, 2^{K+1}-1$
  The remaining K dark nodes are determined by climbing up the tree till the root node:
  $\mathbf{d}(K)$: parent node of $\mathbf{d}(K+1)$
  $\mathbf{d}(K-1)$: parent node of $\mathbf{d}(K)$, till
  $\mathbf{d}(1)$: root node
  Find weights for each dark node:
  $\sigma_1[n-1] = \frac{1}{2}$
  for $\eta = \mathbf{d}(2), \ldots, \mathbf{d}(K+1)$, OR $k = 2, \ldots, K+1$, **do**
    if $k = K+1$, **then**
      $\sigma_k[n-1] = C_s[n-1]\sigma_{k-1}[n-1]$: s is the sibling node of $\mathbf{d}(k)$
    else if $k < K+1$, **then**
      $\sigma_k[n-1] = \frac{1}{2}C_s[n-1]\sigma_{k-1}[n-1]$: s
    end if
    $\mu_k[n-1] = \frac{\sigma_k[n-1]E_k[n-1]}{C_1[n-1]}$
  end for
  Estimation of $y[n]$ from $\mathbf{x}[n]$:
  $\tilde{y}_k[n-1] = \mathbf{w}_k^T[n-1]\tilde{\mathbf{x}}_k[n]$: Prediction of each dark node k
  $\tilde{y}[n] = \sum_{k=1}^{K+1}\mu_k[n-1]\tilde{y}_k[n-1]$: Weighted combination of the individual nodes prediction based on their past performance.
  Update the parameters using Algorithm 2.
end for

Algorithm 2. Update Parameters

for $k = K+1, \ldots, 1$, (Dark nodes of the previous iteration), **do**
  $\mathbf{w}_k[n] = \mathbf{w}_k[n-1] + \nu\tilde{\mathbf{x}}_k[n](y[n] - \mathbf{w}_k^T[n-1]\tilde{\mathbf{x}}_k[n])$: Updated regressor weight vectors for kth node. ν is the step size, a small positive constant.
  $E_k[n] = E_k[n-1]\exp\left(-\frac{1}{2a}(y[n] - \tilde{y}_k[n-1])^2\right)$ Update node performance measure
  if $k = K+1$, **then**
    $C_k[n] = E_k[n]$
  else if $k \neq K+1$, **then**
    $C_k[n] = \frac{1}{2}C_{k_u}[n-1]C_{k_l}[n-1] + \frac{1}{2}E_k[n]$, k is inner node, and k_u and k_l are the two children nodes of k.
  end if
  Update Subspace and manifold parameters
  $\mathbf{c}_k[n] = \alpha\mathbf{c}_k[n-1] + (1-\alpha)\mathbf{x}[n]$
  $\lambda_k^{(m)}[n] = \alpha\lambda_k^{(m)}[n-1] + (1-\alpha)(\boldsymbol{\beta}_k[n])_m^2$ where $m = 1, \ldots, d$ and $(\boldsymbol{\beta}_k[n])_m$ is the mth component of vector $\boldsymbol{\beta}_k[n]$
  $\delta_k[n] = \alpha\delta_k[n-1] + (1-\alpha)\frac{\|\boldsymbol{\gamma}_k[n-1]\|^2}{(D-d)}$
  $\mathbf{Q}_k$ are updated using PETRELS-FO.
end for

3.3 Initialization and Choice of Parameter Values

The algorithm may be initialized by using a small training set to fix the initial values for $\mathbf{Q}_\eta, \mathbf{c}_\eta, \boldsymbol{\Lambda}_\eta$ and δ_η. However, for large data lengths, the effect of initialization is negligible and unnecessary, and the algorithm can be randomly initialized. For the first example used in this paper, we use a small training set of length $N = 1,000$, and use the k-means algorithm to bi-partition the data till the data is divided into 2^K regions. For the rest of the examples, the subspace and the regressor parameters are initialized either randomly or with zeros. We choose the parameter α for updating the subspace parameters that ranges from 0.8 to 0.95 based on the speed of variation in the submanifold. To update the regressor weight vectors, we use RLS [4], [5] with a forgetting factor between 0.5 and 0.75 in different examples.

Remark The choices of d and K are very crucial as they are directly related to the performance and complexity of the algorithm. In practice, one can use a few samples of the regressor vectors to compute the covariance matrix and obtain d. To this end, one can compute the Singular Value Decomposition (SVD) of the covariance matrix and put a threshold based on the values of the eigenvalues. Then, for the kth node, the number of eigenvalues greater than the threshold can be used as the dimensionality d_k. Moreover, in order to use the same dimensionality for all nodes, one can choose the largest d_k as the intrinsic dimensionality d. However, we do not determine d using the SVD, since it is computationally prohibitive in big data applications. Instead, we choose a fixed value for d (the same dimensionality for all nodes) and, as shown in the experiments, our algorithm is robust to mismatch in the true and our selected intrinsic dimension. In real life data, the intrinsic dimension as well as the curvature of the manifold is usually not known and can even be time varying. However, a sufficiently large K can also accommodate for the mismatch between the intrinsic dimension of the manifold and the chosen d. An adaptive hierarchical tree of depth K inherently incorporates all the possible trees of depths less than K, therefore in a time varying environment, our algorithm adapts to the fluctuations in data without changing K as shown by the real data tests in Section 4. By this, we achieve a significant regression performance over a wide range of time varying data without increasing the complexity of the algorithm.

4 Experiments

In this section, we illustrate the performance of the adaptive hierarchical tree algorithm with several real and synthetic examples. The first set of experiments involves a high dimensional sequence that lies on a time varying submanifold [37]. The dimension of the submanifold is $D = 10$ and the intrinsic dimension is $d_{\text{intrinsic}} = 1$. Let θ be a uniformly distributed random variable, i.e., $\theta \sim \mathcal{U}[-2, 2]$. We define $\{\mathbf{v}[n]\}_{n\geq 1}, \mathbf{v}[n] \in \mathbb{R}^D$ with it's pth element,

$$v_p[n] = \frac{1}{\sqrt{2\pi}} e^{-(z_p-\theta)^2/(2\kappa^2[n])}, \tag{24}$$

where $z_p = -2 + 4p/D, p = 1, \ldots, D$, corresponds to regularly spaced points between -2 and 2 and $\mathbf{v}[n] = [v_1[n], v_2[n], \ldots, v_D[n]]^T$. Here, $\{\kappa[n]\}_{n\geq 1}, \kappa[n] \in \mathbb{R}$ controls the variation in the submanifold since the curvature of the submanifold increases with increase in κ and decreases when κ decreases. We define $\kappa[n]$ by:

$$\kappa[n] = 100 - \kappa_o|(n \bmod 2s) - s|, \text{ for } n = 1, 2, \ldots, \tag{25}$$

where $s = 100/\kappa_o$ and κ_o determines the speed of variation of the submanifold. Here, $\kappa[n]$ corresponds to values between 0 and 100 following a triangular waveform. For the first s points, $\kappa[n]$ increases linearly till it reaches 100 and for the next s points it decreases till 0, where the period of the triangular wave is $2s$ and s is the number of points to reach from 0 to 100. The vector sequence $\mathbf{x}[n]$ is obtained by

$$\mathbf{x}[n] = \mathbf{v}[n] + \boldsymbol{\rho}[n],$$

where $\boldsymbol{\rho}[n] \in \mathbb{R}^D$ is white Gaussian noise with autocovariance $\boldsymbol{\Sigma}_{\boldsymbol{\rho}} = 4 \times 10^{-4}\mathbf{I}_D$. The resultant vector sequence $\mathbf{x}[n]$ lies on a time varying submanifold with *ambient dimension*, $D = 10$ while $d_{\text{intrinsic}} = 1$. We generate the desired scalar sequence $\{y[n]\}_{n\geq 1}, y[n] \in \mathbb{R}$ by the following piecewise model,

$$y[n] = \begin{cases} \mathbf{w}_1^T\mathbf{x}[n] + \varrho_1[n], & \text{if } \kappa[n] \leq 25 \\ \mathbf{w}_2^T\mathbf{x}[n] + \varrho_2[n], & \text{if } 25 < \kappa[n] \leq 50 \\ \mathbf{w}_3^T\mathbf{x}[n] + \varrho_3[n], & \text{if } 50 < \kappa[n] \leq 75 \\ \mathbf{w}_4^T\mathbf{x}[n] + \varrho_4[n], & \text{if } 75 < \kappa[n] \leq 100, \end{cases} \tag{26}$$

where $\mathbf{w}_j \in \mathbb{R}^D, j = 1, \ldots, 4$, and ϱ_j for $j = 1, \ldots, 4$, is zero mean white Gaussian noise with variance $\sigma^2 = 1 \times 10^{-4}$. Note that for this example, there is a piecewise linear structure that can be perfectly modeled by the leaves of the tree. Moreover, the algorithm can perfectly match the underlying time varying submanifold using the adaptive tree structure. The curvature of the submanifold increases and decreases

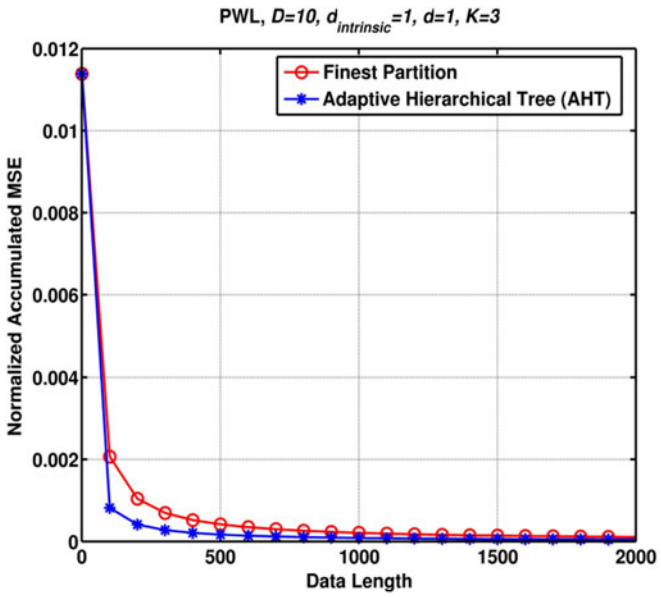

Fig. 5. Sequential piecewise linear prediction of the desired data $\hat{y}[n]$ of (26) with $d = 1$ and tree depth, $K = 3$, sequential piecewise linear predictor with finest partitions on the tree.

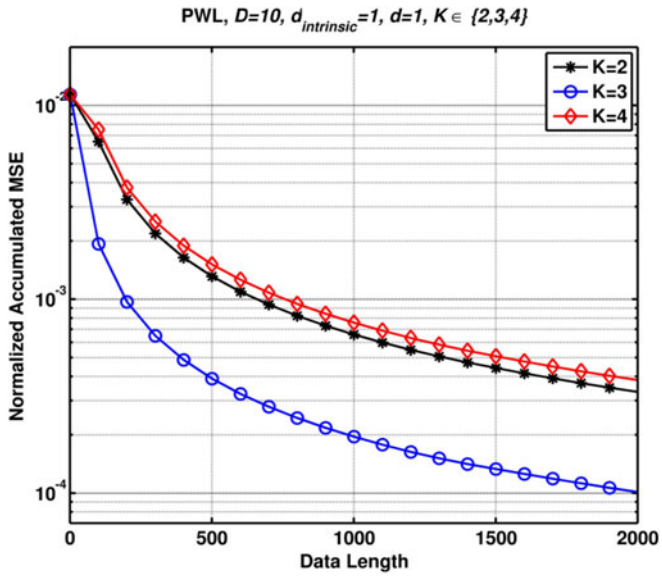

Fig. 6. Piecewise linear prediction on high dimensional, time varying submanifolds using adaptive hierarchical trees. $D = 10, d_{\text{intrinsic}} = 1, d = 1$, and $K = 2, 3, 4$.

with $\kappa[n]$ in a cyclic manner and yet the algorithm is able to track the variations in the submanifold.

We use $d = 1$ as the dimension of projection and use PETRELS algorithm for subspace tracking [45] with fast orthogonalization, and hence we denote it by PETRELS-FO [37], [38]. We apply piecewise linear regression using the context tree weighting method on $\mathbb{R}^d$ by projecting the observed vectors $\mathbf{x}[n]$ on the estimated subspace for each region of the tree with $K = 3$ as described in Algorithm 1 and Algorithm 2. We display the results in Fig. 5. Here, we use $\kappa_o = 0.04$ and plot the normalized accumulated mean square errors against $2,000$ samples of the data. We also calculate the mean square errors using the regression on the finest partitions. The adaptive hierarchical trees algorithm is particularly useful for short data records. As expected, the performance of the finest partition suffers when the data length is small, due to overfitting. Since, our algorithm adaptively combines predictors for each different partition based on their performance, it is able to favor the coarser models with a small number of parameters during the initial phase of the algorithm. This avoids the overfitting problems faced by the sequential algorithms using the finest partition. As the data length increases, both algorithms almost converge to the same minimum error rate. Hence, the tree based adaptation is attractive for adaptive processing in a time varying environments for which a windowed version of the most recent data is typically used. The performance of the algorithm improves with increasing the depth of the tree, however, it saturates at a specific K, i.e., when the piecewise model of (26) can be perfectly modeled by the proposed algorithm with a certain number of regions. In our current example, we observe that choosing $K = 2$ must be sufficient since it divides the regressor space into a maximum four regions. However, due to the time varying nature of the submanifold, choosing $K > 2$ improves the performance. The error rate reaches the saturation point at $K = 3$ and further increasing the depth of the tree results in overfitting. This behavior can be seen in Fig. 6.

We next compare the performance of our algorithm for various choices of the dimension of projection using the same dataset. We plot the normalized MSE for $d = 1, 2, 3$ as shown in Fig. 7. We observe that for the dimension of projection higher than the intrinsic dimension of the submanifold, i.e., $d > d_{\text{intrinsic}}$, there is an improvement in the performance of the algorithm for the same K. Since the manifold data is corrupted with $D-$dimensional noise, although the intrinsic dimension is $d_{\text{intrinsic}} = 1$, we record a significant performance improvement for the same values of K. The submanifold subspaces are better tracked when the dimension of projections is chosen higher than the intrinsic dimension of the submanifold, resulting in better performance overall. Hence, we can deduce that the prediction performance of our algorithm can be increased by either increasing d or K, yet at the cost of complexity and memory

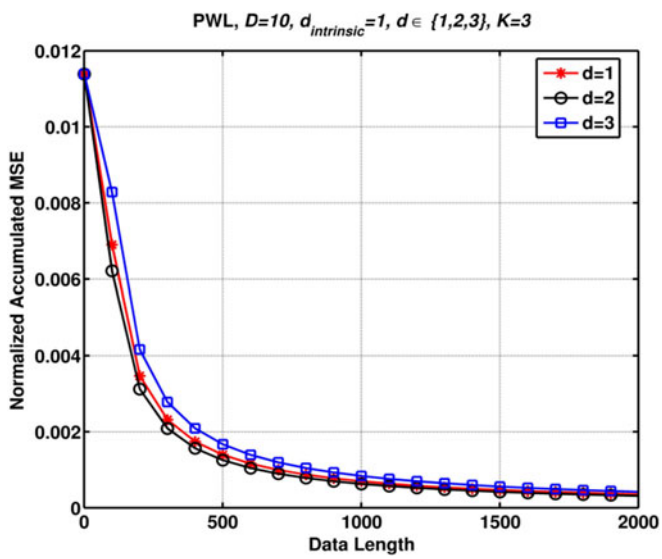

Fig. 7. Sequential piecewise linear prediction of the desired data $\hat{y}[n]$ of (26) with $d = 1, 2, 3$ and tree depth, $K = 3$. The intrinsic dimension is $d_{\text{intrinsic}} = 1$.

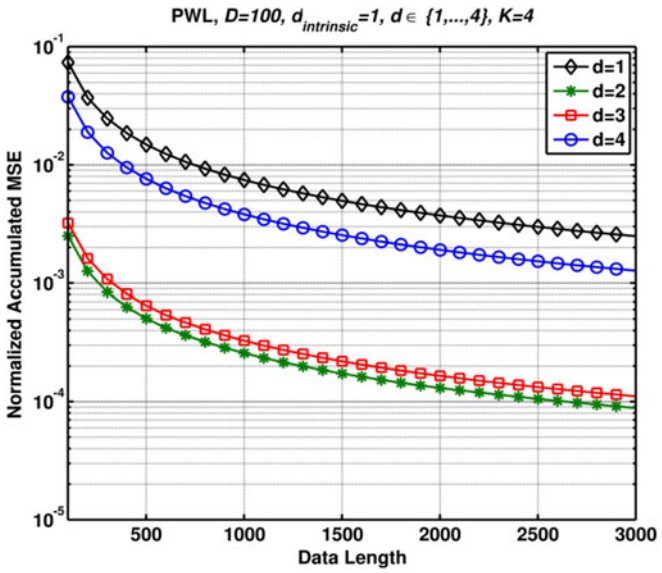

Fig. 8. Piecewise linear regression on $100-$ dimensional time varying submanifold with intrinsic dimension $d_{intrinsic} = 1$ using adaptive Hierarchical trees. Normalized MSE are plotted for $3,000$ samples of online data using $K = 4$ and $d = 1, ..., 4$.

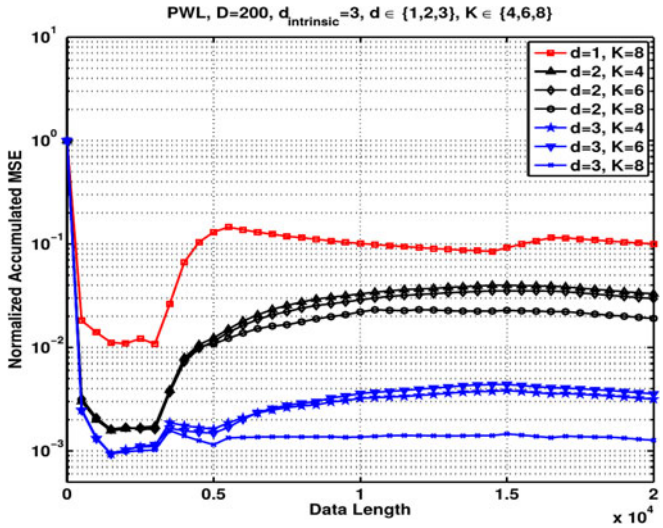

Fig. 9. Normalized MSE based performance analysis using AHT with $K \in \{4, 6, 8\}$ and $d = \{1, 2, 3\}$, to see the effect of these parameters on the time-varying submanifold tracking performance.

requirement. However, increasing d or K beyond an optimal value results in overfitting, and in turn degrades the performance. In the current example, the algorithm performs better when $d = 2$, yet worse when $d = 3$ as compared to $d = 1$. These phenomena can further be explained by the subsequent experiments.

In the next example, we use a $D = 100$ dimensional data that lies on a time varying submanifold with the intrinsic dimension $d_{\text{intrinsic}} = 1$. We generate the datasets in the same manner as 24 and (26). We choose the tree depth, $K = 4$ and plot the normalized mean square errors for various choices of d while $d \geq d_{\text{intrinsic}}$. Fig. 8 shows a significant decrease in MSE for $d = 2$ over $d = 1$. Furthermore, the algorithm attains the minimum error rate faster for larger d. This makes the choice of larger d attractive for adaptive processing in time varying environment for which a shorter length of the most recent data is used. However, even with $d = 1$, the algorithm attains almost the same minimum error rate but for a larger data length. Still choosing d as large as possible is not recommended and rather makes the algorithm inefficient since larger d results in overfitting and may even increase the error rate. This is evident from Fig. 8, where choosing $d > 3$ increases the process complexity and memory requirement yet the performance is degraded.

We next compare the performance of Adaptive Hierarchical Trees for a new dataset with dimensionality $D = 200$ and intrinsic dimension $d_{\text{intrinsic}} = 3$ [37], [38]. Let the two-dimensional parameters $[f_o, \phi]$ be uniformly and randomly generated with frequency $f_o \sim \mathcal{U}[1, 100]$ and phase $\phi \sim \mathcal{U}[0, 1]$. We define $\mathbf{v}[n] \in \mathbb{R}^D$, with its pth element

$$v_p[n] = \sin\left[2\pi(f_o \tilde{z}_p + \frac{\tilde{\kappa}[n]^2}{2}\tilde{z}_p^2 + \phi)\right], \tag{27}$$

where $\tilde{z}_p = 10^{-4}p$ for $p = 1, 2, \ldots, 100$, corresponds to regularly spaced points between 0 and 0.01. The parameter $\tilde{\kappa}[n]$ for $n \in \{1, \ldots, N\}, N = \text{length of the data}$, controls the variations in the submanifold and is set according to the following equation

$$\tilde{\kappa}[n] = 100|\cos(\theta[n])|, \text{ for } n = 1, 2, \ldots, \tag{28}$$

where $\theta[n] \in [0, 3\pi]$. Then, $\mathbf{v}[n] = [v_1[n], v_2[n], \ldots, v_D[n]]^T$. The observed $D-$dimensional data $\mathbf{x}[n]$ is given by

$$\mathbf{x}[n] = \mathbf{v}[n] + \boldsymbol{\rho}[n],$$

where $\boldsymbol{\rho}[n] \in \mathbb{R}^D$ is a white Gaussian noise with autocovariance $\boldsymbol{\Sigma}_{\boldsymbol{\rho}} = 4 \times 10^{-4}\mathbf{I}_D$. The desired data is generated by (26), with four piecewise linear regions.

In Fig. 9, we plot the normalized MSE for the adaptive hierarchical tree algorithm using trees of various depths $K \in \{4, 6, 8\}$ and different values of the dimensionality, $d \in \{1, 2, 3\}$. The results here show that a value of $d = 3$ is optimal in this example. It is interesting to note that our algorithm not only attains the minimum error rate faster but also keeps it stable when the data length increases. Since, our algorithm "always" dynamically updates and combines the weights of the partitions, it is capable of catching up with the sudden changes in the model. This makes the algorithm effective for applications with dynamic environment.

The choice of d affects the performance of the algorithm that can be seen in Fig. 9b, where $d > 1$ shows great improvement on the performance. In addition, in order to see the effect of the tree depth K on the performance, we have performed the simulations for $K \in \{4, 6, 8\}$, and for $d \in \{2, 3\}$. The results show that we can improve the performance by increasing the depth of the tree for both $d = 2$ and $d = 3$. However, since the depth K directly affects the computational complexity, it should be determined based on the complexity considerations. Moreover, as we observe from the simulations, when we choose $d = 3$, the algorithm can effectively match the underlying data model and reach a significantly low Normalized Accumulated MSE. Hence, if the intrinsic dimension of the submanifold is known, it gives a good initial guess to use for the value of d for d should be at least equal to the intrinsic dimension. It is interesting to note that for an optimal choice of K, a $d < d_{\text{intrinsic}}$ may even be sufficient for the

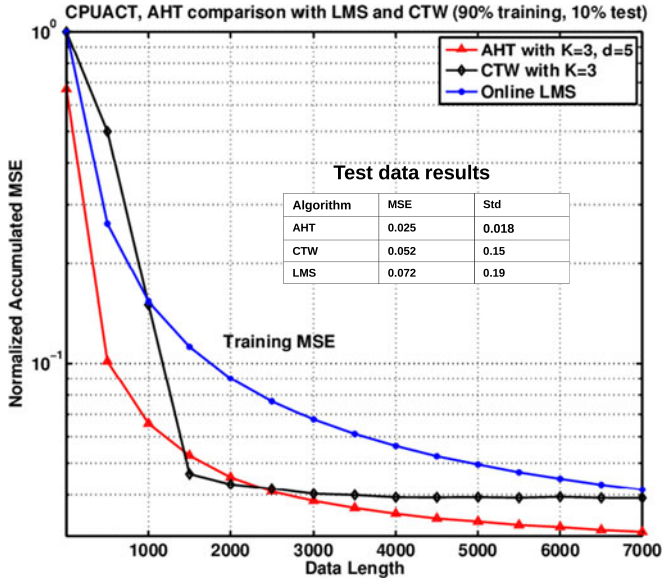

Fig. 10. Performance Analysis of AHT on real data (Computer Activity) with $d = 5, D = 21$ and $K = 3$. 90% of the data is used for training and the remaining data is used as the test data.

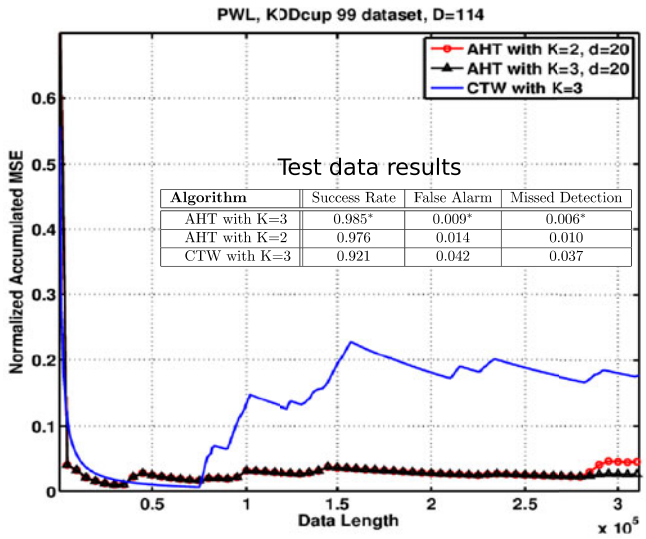

Fig. 11. Performance Analysis of AHT on real data (KDD CUP99 Database) with $d = 20, D = 114$ and $K \in \{2, 3\}$. 90 percent of the data is used for training and the remaining data is used for the test.

algorithm to track and predict the desired data sequence from the high dimensional time varying submanifold. An optimal K not only determines the modeling strength with respect to the nonlinearity of the model but also the approximation to the true manifold. This is specifically helpful for the real data where the intrinsic dimensionality may be unknown and even varying, then choosing K optimally and adaptively can be used to track the subspace for a short training data. In such case, the best practice would be to start from a simpler model, with small d and K, and adjust these parameters till a certain minimum error rate is achieved or the saturation point occurs. However, this is only required at the training stage. Furthermore, if real online data yet has more fluctuations than initially guessed, there is no need to change K since the weighting coefficients μ_k on each level of the tree already take care of the time varying curvature. For instance, if the curvature of the manifold increases, the algorithm increases the weights of finer nodes. This is in contrast to [37], [38] where the manifold is tracked on a single level at a certain time and if the curvature changes, the tree has to grow or prune. We show by examples that in most real world scenarios, simpler models in terms of d and K works well enough to achieve better results than previous state of the art techniques.

We next illustrate the performance of our algorithm on a number of real world data sets. The public data sets are obtained by measurements on specific parameters related to the underlying system. Then, these measurements are collected into vectors to be used as the regressor vectors in the regression task. However, these measurements are not necessarily independent of each other and there are correlations between the measurements of different parameters. Therefore, it is safe to assume that the observed data lies on a time varying submanifold of lower intrinsic dimension. The following simulations demonstrate the superior performance of our algorithm on such data sets, which is consistent with our assumptions.

In the following experiments, we use 90 percent of the data for training and the remaining 10 percent data for testing. In the first example, we use the computer activity dataset [46], where we predict the portion of time the CPU runs in user mode from the observed 21 attributes. The results are given in Fig. 10. Here we achieve a significantly small MSE and faster convergence as compared to context tree weighting [27] and online Least Mean Square [4] by using $d = 5$ whereas the actual dimension is $D = 21$. The results demonstrate the superior performance of our algorithm not only in the training phase, but also the in test phase.

In the next example, we evaluate the performance of our algorithm on the real world dataset KDD-CUP99 [47]. The task is to design a software to detect network intrusions in order to protect a computer network from unauthorized users, including (perhaps) the insiders. We build a predictive model (i.e., a classifier) capable of distinguishing between "bad" connections, called intrusions or attacks, and "good" connections. A connection is a sequence of TCP packets starting and ending at some well defined times, between which data flows to and from a source IP address to a target IP address under some well defined protocol. The objective is to classify the connections as "normal" or an "attack", based on the corresponding features of that connection. We use the corrected dataset of KDD-CUP99 [47], which consists of 42 features, a mix of categorical and numeric features. We convert the categorical features to numeric by introducing dummy variables and the final dataset has $D = 114$ dimensional data points (feature vectors).

We compare the performance of our algorithm with that of the conventional context tree weighting method. For the CTW method, we use a tree with depth $K = 3$, whereas we have done the experiment with $K = 2$ and $K = 3$, and $d = 20$ for our algorithm. As depicted in Fig. 11, our algorithm significantly outperforms the CTW method as our algorithm achieves a lower Normalized Accumulated MSE during the training phase. The results also reveal the superior performance of our algorithm during the test phase (see the table included in the Fig. 11). This is because our algorithm, unlike the CTW, continues to adapt itself to the structure of the data in the test phase. In addition, the results

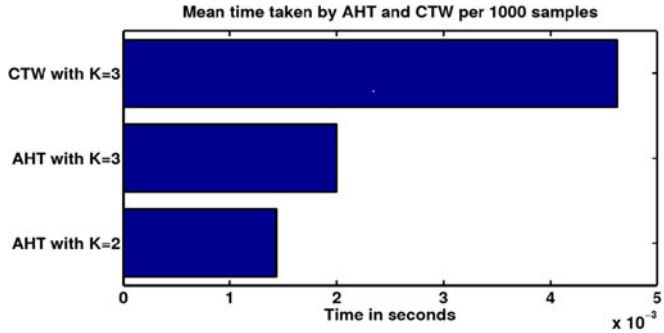

Fig. 12. The comparison between the time consumption of different algorithms in the KDD CUP99 experiment.

show that when we use $K = 3$, our algorithm can perform slightly better than when we use $K = 2$. Furthermore, Fig. 12 shows that our algorithm is remarkably faster than the CTW method.

Finally, we summarize the performance of AHT on other widely used real world datasets and compare with the previous state of the art algorithms in terms of MSE and computational complexity [28]. The results are shown in Table 1. We compare the performance and complexity of our algorithm with Decision Adaptive Trees (DAT) [28] and context tree weighting [27]. The Kinematics data ($D = 8$) [46] is the simulation of the dynamics of an $8-$ link all-revolute robotic arm where we predict the distance of the end-effector from the target. In the Elevators dataset with $D = 18$ [48], [49], the desired task is to take action on the elevators of an F16 aircraft. The Bank dataset ($D = 32$) [49] is a realistic simulation of the queues in a series of banks and the task is to predict the portion of customers who would leave the bank because of full queues. The Pumadyn dataset ($D = 32$) [48] is a realistic simulation of the dynamics of Unimation Puma 560 robot arm where the target task is to predict the angular acceleration of the robot arm's links. Here we use $K = 2$ and $d = 1$ in all the examples and achieve significantly good performance with relatively reduced complexity, i.e., $O(Kd)$ as compared to $O(4^K D)$ in case of DAT and $O(KD)$ in case of CTW, where $d \ll D$.

TABLE 1
Normalized Time Accumulated MSE of the Proposed Algorithm Compared with Results Found in the Literature

Data Sets / **Algorithms**	DAT $\{O(4^K D)\}$	CTW $\{O(KD)\}$	**AHT** $\{O(Kd)\}$
Kinematics	0.0639	0.0728	**0**.0600*
Elevators	0.0091	0.0210	**0**.0130*
Bank	0.0511	0.1100	**0**.0320*
Pumadyn	0.0781	0.0923	**0**.0505*

The order of computations is given for each algorithm.

5 Conclusion

We consider the problem of piecewise linear regression on high dimensional data from a competitive algorithm perspective. The data is lying on a time varying submanifold and we use a hierarchical tree structure to track the submanifold and escape the curse of dimensionality. Using the adaptive hierarchical tree structure together with the subspace tracking algorithms based on sequential performance measure, we introduce a regression algorithm whose total squared prediction error is within $O(dJ \ln(n/J)) + O(\mathbb{C}(\mathcal{P}_i))$ of that of the best batch piecewise model, where J is the number of regions in the best piecewise model. We introduce a method using the context tree notion for the piecewise modeling that competes well against a doubly exponential class of possible partitioning of the regressor space. The algorithm is shown to instantly adapt to the variations in the data and hence, is effective for high dimensional data with short data lengths in an online setting. The resulting algorithms are suitable for rather complex datasets since they have time complexity only linear in the depth of the tree and perform well for a variety of real and synthetic data. Our algorithm can be efficiently used in high performance computing due to the low computational complexity and faster learning.

Acknowledgments

This work has been supported in part by AveaLabs, TUBITAK TEYDEB 1501 no. 3130095 project and TUBITAK project 113E517.

References

[1] A. C. Singer, G. W. Wornell, and A. V. Oppenheim, "Nonlinear autoregressive modeling and estimation in the presence of noise," *Digit. Signal Process.*, vol. 4, no. 4, pp. 207–221, 1994.
[2] A. C. Singer, S. S. Kozat, and M. Feder, "Universal linear least squares prediction: Upper and lower bounds," *IEEE Trans. Inf. Theory*, vol. 48, no. 8, pp. 2354–2362, Aug. 2002.
[3] A. C. Singer and M. Feder, "Universal linear prediction by model order weighting," *IEEE Trans. Signal Process.*, vol. 47, no. 10, pp. 2685–2699, Oct. 1999.
[4] A. H. Sayed, *Fundamentals of Adaptive Filtering*. Hoboken, NJ, USA: Wiley, 2003.
[5] S. O. Haykin, *Adaptive Filter Theory*. Englewood Cliffs, NJ, USA: Prentice-Hall, 2002.
[6] I. G. Ali and Y.-T. Chen, "Design quality and robustness with neural networks," *IEEE Trans. Neural Netw.*, vol. 10, no. 6, pp. 1518–1527, Nov. 1999.
[7] L. Rutkowski, "Generalized regression neural networks in time-varying environment," *IEEE Trans. Neural Netw.*, vol. 15, no. 3, pp. 576–596, May. 2004.
[8] R. Setiono, W. K. Leow, and J. M. Zurada, "Extraction of rules from artificial neural networks for nonlinear regression," *IEEE Trans. Neural Netw.*, vol. 13, no. 3, pp. 564–577, May. 2002.
[9] A. Krzyzak and T. Linder, "Radial basis function networks and complexity regularization in function learning," *IEEE Trans. Neural Netw.*, vol. 9, no. 2, pp. 247–256, Mar. 1998.
[10] R. Gribonval, "From projection pursuit and CART to adaptive discriminant analysis?" *IEEE Trans. Neural Netw.*, vol. 16, no. 3, pp. 522–532, May. 2005.
[11] S. Bengio and Y. Bengio, "Taking on the curse of dimensionality in joint distributions using neural networks," *IEEE Trans. Neural Netw.*, vol. 11, no. 3, pp. 550–557, May. 2000.
[12] N. C. Bianchi, P. M. Long, and M. K. Warmuth, "Worst-case quadratic loss bounds for prediction using linear functions and gradient descent," *IEEE Trans. Neural Netw.*, vol. 7, no. 3, pp. 604–619, May. 1996.
[13] K. Muller, S. Mika, G. Ratsch, K. Tsuda, and B. Scholkopf, "An introduction to kernel-based learning algorithms," *IEEE Trans. Neural Netw.*, vol. 12, no. 2, pp. 181–201, Mar. 2002.
[14] X. Wu, X. Zhu, G.-Q. Wu, and W. Ding, "Data mining with big data," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 1, pp. 97–107, Jan. 2014.
[15] H. Liu and L. Yu, "Toward integrating feature selection algorithms for classification and clustering," *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 4, pp. 491–502, Apr. 2005.
[16] Q. Song, J. Ni, and G. Wang, "A fast clustering-based feature subset selection algorithm for high-dimensional data," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 1, pp. 1–14, Jan. 2012.

[17] O. Egecioglu, H. Ferhatosmanoglu, and U. Ogras, "Dimensionality reduction and similarity computation by inner-product approximations," *IEEE Trans. Knowl. Data Eng.*, vol. 16, no. 6, pp. 714–726, Jun. 2004.
[18] V. Vovk, "Aggregating strategies," in *Proc. 3rd Annu. Workshop Comput. Learn. Theory*, 1990, pp. 371–383.
[19] V. Vovk, "Competitive on-line linear regression," in *Proc. Adv. Neural Inform. Process. Syst.*, 1998, pp. 364–370.
[20] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York, NY, USA: Springer, 2006.
[21] I. T. Jolliffe, *Principal Component Analysis*. New York, NY, USA: Springer, 2002.
[22] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Sci., New Series*, vol. 290, no. 5500, pp. 2323–2326, 2000.
[23] J. B. Tenenbaum, V. de Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, pp. 2319–2323, 2000.
[24] O. J. J. Michel, A. O. Hero, and A.-E. Badel, "Tree-structured nonlinear signal modeling and prediction," *IEEE Trans. Signal Process.*, vol. 47, no. 11, pp. 3027–3041, Nov. 1999.
[25] F. M. J. Willems, "The context-tree weighting method: Extensions," *IEEE Trans. Inform. Theory*, vol. 44, no. 2, pp. 792–798, Mar. 1998.
[26] F. M. J. Willems, "Coding for a binary independent piecewise-identically-distributed source," *IEEE Trans. Inf. Theory*, vol. 42, no. 6, pp. 2210–2217, Nov. 1996.
[27] S. S. Kozat, A. C. Singer, and G. C. Zeitler, "Universal piecewise linear prediction via context trees," *IEEE Trans. Signal Process.*, vol. 55, no. 7, pp. 3730–3745, Jul. 2007.
[28] N. D. Vanli and S. S Kozat, "A comprehensive approach to universal piecewise nonlinear regression based on trees," *IEEE Trans. Signal Process.*, vol. 62, no. 20, pp. 5471–5486, Oct. 2014.
[29] A. Lakhina, M. Crovella, and C. Diot, "Diagnosing network-wide traffic anomalies," in *Proc. Conf. Appl., Technol., Archit., Protocols Comput. Commun.*, 2004, pp. 219–230.
[30] K.-C. Lee and D. Kriegman, "Online learning of probabilistic appearance manifolds for video-based recognition and tracking," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recog.*, vol. 1, pp. 852–859, Jun. 2005.
[31] R. T. Collins, A. J. Lipton, and T. Kanade, "Introduction to the special section on video surveillance," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 8, pp. 745–746, Aug. 2000.
[32] L. Shen and E. C. Tan, "Dimension reduction-based penalized logistic regression for cancer classification using microarray data," *IEEE/ACM Trans. Comput. Biol. Bioinf.*, vol. 2, no. 2, pp. 166–175, Apr. 2005.
[33] J. Nilsson, F. Sha, and M. Jordan, "Regression on manifolds using kernel dimension reduction," in *Proc. 24th Int. Conf. Mach. Learn.*, 2007, pp. 697–704.
[34] J. M. Lee, *Riemannian Manifolds: Introduction Curvature*. New York, NY, USA: Springer, 1997.
[35] J. Jost, *Riemannian Geometry Geometric Analysis*. New York, NY, USA: Springer, 2008.
[36] Y. Xie, J. Huang, and R. Willett, "Multiscale online tracking of manifolds," in *Proc. IEEE Statist. Signal Process. Workshop*, 2012, pp. 620–623.
[37] Y. Xie, J. Huang, and R. Willett, "Change-point detection for high-dimensional time series with missing data," *IEEE J. Sel. Topics Signal Process.*, vol. 7, no. 1, pp. 12–27, Feb. 2013.
[38] Y. Xie and R. Willett, "Online logistic regression on manifolds," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2013, pp. 3367–3371.
[39] W. C. Stirling and M. Morf, "A new decision-directed algorithm for nonstationary priors," in *Proc. 21st IEEE Conf. Decision Control*, Dec. 1982, pp. 706–707.
[40] R. E. Bellman, *Adaptive Control Processes: A Guided Tour*. Princeton, NJ, USA: Princeton Univ. Press, 1961.
[41] S. Dasgupta and Y. Freund, "Random projection trees for vector quantization," *IEEE Trans. Inf. Theory*, vol. 55, no. 7, pp. 3229–3242, Jul. 2009.
[42] S. Kpotufe and S. Dasgupta, "A tree-based regressor that adapts to intrinsic dimension," *J. Comput. Syst. Sci.*, vol. 78, no. 5, pp. 1496–1515, 2011.
[43] N. Cristianini and J. Shawe-Taylor, *Support Vector Machines and Other Kernel-Based Learning Methods*. Cambridge, U.K.: Cambridge Univ. Press, 2000.
[44] S. L. France, J. Douglas Carroll, and H. Xiong, "Distance metrics for high dimensional nearest neighborhood recovery: Compression and normalization," *Inform. Sci.*, vol. 184, no. 1, pp. 92–110, Feb. 2012.
[45] Y. Chi, Y. C. Eldar, and R. Calderbank, "PETReLS: Subspace estimation and tracking from partial observations," in *Proc. Int. Conf. Acoust., Speech, Signal Process.*, Mar. 2012, pp. 3301–3304.
[46] C. E. Rasmussen, R. M. Neal, G. Hinton, D. Camp, M. Revow, Z. Ghahramani, R. Kustra, and R. Tibshirani, "Delve data sets," (1996). [Online]. Available: http://www.cs.toronto.edu/~delve/data/datasets.html
[47] S. D. Bay, D. Kibler, M. J. Pazzani, and P. Smyth, "The UCI KDD archive of large data sets for data mining research and experimentation," *ACM SIGKDD Explorations Newslett.*, vol. 2, no. 2, pp. 81–85, 2000.
[48] J. Alcala-Fdez, A. Fernandez, J. Luengo, J. Derrac, S. Garca, L. Snchez, and F. Herrera, "KEEL data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework," *J. Multiple-Valued Logic Soft Comput.*, vol. 17, no. 2-3, pp. 255–287, 2011.
[49] L. Torgo, "Regression data sets," (2001). [Online]. Available: http://www.dcc.fc.up.pt/~ltorgo/Regression/DataSets.html

Farhan Khan received the BSc degree with honors in electrical engineering from the University of Engineering and Technology, Peshawar, Pakistan, in 2007, and the MSc degree in rf communication systems from the University of Southampton, United Kingdom, in 2009. He is currently working toward the PhD degree in the Department of Electrical and Electronics Engineering, Bilkent University, Ankara, Turkey. After graduation, he joined the Department of Electrical Engineering at the COMSATS Institute of Information Technology, Pakistan. His research interests include adaptive signal processing, big data, machine learning, and neural networks.

Dariush Kari received the BS degree with high honor in two majors, electrical engineering and computer science, from the Amirkabir University of Technology (Tehran Polytechnic), Tehran, Iran, July 2014. He is currently working toward the MS degree at the Department of Electrical and Electronics Engineering, Bilkent University, Ankara, Turkey. His current research interests include machine learning, big data analytics, optimization, statistical signal processing, adaptive filtering, and control theory.

Ilyas Alper Karatepe received the BS degree in electronics engineering from Hacettepe University, Ankara, Turkey, in 2004. He is currently working toward the MSc degree in Bogazici University Software Engineering program since September 2013. From January 2005 to August 2006, he was with Vodafone Turkey working as a software developer in the Special Projects Department. In September 2007, he joined to Alcatel-Lucent Turkey as a solution architect for femtocell management systems project. In April 2010, he joined the Middleware Department of AVEA as a senior software engineer. From August 2013, he has been at AveaLabs as a senior R&D engineer. His current research interests include big data analytics, social media analysis, service oriented architectures.

Suleyman S. Kozat (A'10–M'11–SM'11) received the BS degree with full scholarship and high honors from Bilkent University, Turkey. He received the MS and PhD degrees in electrical and computer engineering from the University of Illinois at Urbana Champaign, Urbana, IL. He is a graduate of Ankara Fen Lisesi. After graduation, he joined IBM Research, T. J. Watson Research Lab, Yorktown, New York as a research staff member in the Pervasive Speech Technologies Group. While doing his PhD, he was a research associate at Microsoft Research, Redmond, WA in the Cryptography and Anti-Piracy Group. He holds several patent inventions due to his research accomplishments at IBM Research and Microsoft Research. He is currently an associate professor at the Electrical And Electronics Engineering Department of Bilkent University. He is the president of the IEEE Signal Processing Society, Turkey Chapter. He has been elected to the IEEE Signal Processing Theory and Methods Technical Committee and IEEE Machine Learning for Signal Processing Technical Committee, 2013. He received the IBM Faculty Award by IBM Research in 2011, Outstanding Faculty Award by Koc University in 2011 (granted the first time in 16 years), Outstanding Young Researcher Award by the Turkish National Academy of Sciences in 2010, ODTU Prof. Dr. Mustafa N. Parlar Research Encouragement Award in 2011, Outstanding Faculty Award by Bilim Kahramanlari, 2013 and holds Career Award by the Scientific Research Council of Turkey, 2009. He is a senior member of the IEEE.