Adaptive Hierarchical Space Partitioning for Online Classification

O. Fatih Kilic*, N. Denizcan Vanli†, Huseyin Ozkan‡ Ibrahim Delibalta§ and Suleyman S. Kozat*
*Department of Electrical and Electronics Engineering, Bilkent University, Ankara 06800, Turkey
{kilic,huseyin,kozat}@ee.bilkent.edu.tr
†School of Electrical and Computer Engineering, Massachusetts Institute of Technology, Cambridge, MA
denizcan@mit.edu
‡Department of Brain and Cognitive Sciences, Massachusetts Institute of Technology, Cambridge, MA
hozkan@mit.edu
§Turk Telekom Labs, Istanbul, Turkey
ibrahim.delibalta@turktelekom.com.tr

***Abstract*—We propose an online algorithm for supervised learning with strong performance guarantees under the empirical zero-one loss. The proposed method adaptively partitions the feature space in a hierarchical manner and generates a powerful finite combination of basic models. This provides algorithm to obtain a strong classification method which enables it to create a linear piecewise classifier model that can work well under highly non-linear complex data. The introduced algorithm also have scalable computational complexity that scales linearly with dimension of the feature space, depth of the partitioning and number of processed data. Through experiments we show that the introduced algorithm outperforms the state-of-the-art ensemble techniques over various well-known machine learning data sets.**

I. Introduction

Due to the recent advances in information technologies, we need to process data that is streamed at extremely fast rates and, usually, presented in unstructured complex forms [1], [2]. In particular, we propose a novel and highly efficient online classification algorithm for an arbitrary stream of possibly correlated observations.

Our algorithm uses piecewise linear functions to approximate complex (i.e., strongly nonlinear) classification boundaries and exploit the local regularities to mitigate convergence issues. In particular, we use a hierarchical model to generate a set of different feature space partitions, where we sequentially train a simple linear classifier at each region of every partition. Hence, each partition yields a different nonlinear classification model (which we call a base classifier) and all such models constitute a competition class of base classifiers in our framework. We parameterize this competition class over the partitioning parameters (i.e., the region separators) and then sequentially optimize our competition class over these parameters using the stochastic gradient descent method. By this optimization, our competition class sequentially and continuously improves itself -in the course of the data stream- by adjusting the partition structure.

The proposed online classifier combines the outputs of all base classifiers at each instance and generates its classification output. We prove that by this combination, the proposed algorithm asymptotically achieves the performance of the best base classifier without any statistical assumptions on the data. Our results hold for every possible data stream of arbitrary length regardless of the underlying data generation process. The computational complexity of the proposed algorithm scales linearly with the dimensionality of the data and the depth of the hierarchical models uniformly for all data instances. Since we use a finite combination of linear models, our algorithm generalizes well and does not overfit (or limitedly overfits) [3], [4].

II. Problem Description

We study online binary classification, where we observe feature vectors $\{\boldsymbol{x}_t\}_{t\geq 1}$ and determine their labels $\{y_t\}_{t\geq 1}$ in an online manner.[1] Here, we aim to construct an online classifier $f_t(\boldsymbol{x}_t)$, where $\boldsymbol{x}_t \in \mathbb{R}^p$ and $y_t \in \{-1, 1\}$, such that the empirical loss of this classifier, i.e.,

$$L_T(f_t) \triangleq \sum_{t=1}^{T} \mathbb{1}_{\{f_t(\boldsymbol{x}_t)\neq y_t\}}, \tag{1}$$

is asymptotically as small as the empirical loss of the best classifier $C(\boldsymbol{\phi})$ from a competition class $\mathcal{S}(\boldsymbol{\phi})$ of base classifiers for any unknown sequence length T. The set of base classifiers $\mathcal{S}(\boldsymbol{\phi})$ is a parameter dependent competition class that can be optimized over $\boldsymbol{\phi}$, where $\boldsymbol{\phi}$ is not a specific parameter for a base classifier, but instead it directly optimizes the competition class. In this manner, the classifier f_t competes against the best competitor that itself constantly improves.

In order to measure the relative performance of f_t with respect to the performance of a base classifier $f_t^{(C)}$, where $C \in \mathcal{S}(\boldsymbol{\phi})$ (we drop the $\boldsymbol{\phi}$-dependency of the base classifiers for notational simplicity), we use the following regret

$$R_T(f_t; f_t^{(C)}) \triangleq \frac{1}{T}\left[L_T(f_t) - L_T(f_t^{(C)})\right] \tag{2}$$

for any arbitrary stream length of T. Our aim is then to minimize this regret in a twofold optimization framework in

[1] All vectors are column vectors and denoted by boldface lower case letters. Throughout the paper, the time index appears as a subscript.

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

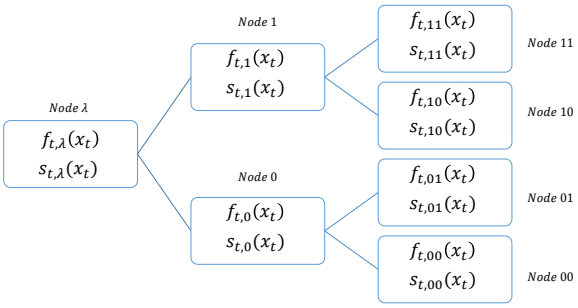

Fig. 1: The generalized view of the complete tree structure. $f_{t,n}(\cdot)$ represents the classifier of node n and $s_{t,n}(\cdot)$ represents the separator function corresponding to node n.

the sense that both the classifier selection weighting over $\mathcal{S}(\boldsymbol{\phi})$ and the optimization parameter $\boldsymbol{\phi}$ are adaptively learned.

III. Construction of the Competition Class

To efficiently construct the set of base classifiers that can be optimized via $\boldsymbol{\phi}$, we hierarchically partition the feature space according to a parameter vector $\boldsymbol{\phi}$. In particular, we bisect the feature space using a separator function (which is a function of $\boldsymbol{\phi}$). Then, we continue to bisect the resulting regions using different separator functions and construct a complete hierarchical model (i.e., a partitioning tree). In this manner, for each inner node of the tree, there exists a corresponding separator function, which bisects the region represented by that node. We also assign a simple region classifier (e.g., a linear and online classifier such as the perceptron) to each node of the tree. As an example, a depth-2 tree is depicted in Figure 1, where $f_{t,n}$ represents the region classifier and $s_{t,n}$ represents the separator function of node n at time t. In this figure, the root node (or node λ) represents the entire feature space, where the separator function $s_{t,\lambda}$ bisects this region and creates node 0 and node 1. Similarly, each of these nodes are also bisected via $s_{t,0}$ and $s_{t,1}$ creating the children nodes 00, 01 and 10, 11, respectively.

We emphasize that the selection of the region classifiers and separator functions are completely up to preference and can be arbitrary. However, throughout the paper, we use the perceptrons as our node classifiers and the hyperplanes as our node separators. In particular, the separator $s_{t,n}$ is a function of $\boldsymbol{\phi}_{t,n}$ such as the sigmoid function $s_{t,n}(\boldsymbol{x}_t) = (1+\exp(\boldsymbol{\phi}_{t,n}^T \boldsymbol{x}_t))^{-1}$, where $\boldsymbol{\phi}_{t,n}$ represents the angle of the normal line to the separating hyperplane for each node n. In this manner, the parametrization of the set of base classifiers is performed via the parameter vector $\boldsymbol{\phi} = \{\boldsymbol{\phi}_{t,n}\}$. According to the definition of the separator functions, each instance $\boldsymbol{x}_t$ follows a path starting from the root node to a leaf node through a certain branch such that if $\boldsymbol{\phi}_{t,n}^T \boldsymbol{x}_t \leq 0$ at a node n, then $\boldsymbol{x}_t$ follows the 1-branch; otherwise, it follows the 0-branch. Meanwhile, at each visited node, it is classified by the region classifier $f_{t,n}$.

By taking the union of non-overlapping regions, one can construct different base classifiers. As an example, in Figure 1, nodes 0 and 1 can define a base classifier. Similarly, nodes 00, 01, and 1 can be used to construct a base classifier. In this manner, a base classifier $C \in \mathcal{S}(\boldsymbol{\phi})$ classifies the instance $\boldsymbol{x}_t$ using the output of the region classifier $f_{t,n}(\boldsymbol{x}_t)$, where n is the leaf node (containing $\boldsymbol{x}_t$) of the subtree that generates C. Since for a depth-D tree, there exist approximately 1.5^{2^D} different subtrees [5] and any subtree (pruning) on a complete tree of depth-D can be used to classify the instance $\boldsymbol{x}_t$, we consider each subtree as a base classifier and construct our set of base classifiers $\mathcal{S}(\boldsymbol{\phi})$. We emphasize that since the separator functions elegantly partition the feature space, the resulting base classifiers are of highly nonlinear models.

IV. Online Adaptive Hierarchical Space Partitioning Classifier (AHSP)

Based on the aforementioned partitioning of the feature space, we construct the final classifier f_t by combining the outputs of all base classifiers in $\mathcal{S}(\boldsymbol{\phi})$. In this manner, as the data length T goes to infinity, the regret in (2) goes to zero, hence f_t achieves the performance of the best base classifier. While taking a weighted combination of base classifiers, our algorithm also adapts the partitioning of the feature space (by updating $\boldsymbol{\phi}$) to minimize its classification error. We provide the construction of the algorithm (and also the detailed construction of the base classifiers) in the proof of the following theorem, where we also present our theoretical results.

Theorem 1: *Let $\{\boldsymbol{x}_t\}_{t\geq 1}$ and $\{y_t\}_{t\geq 1}$ be arbitrary and real-valued sequence of feature vectors and their labels, respectively. The online classifier in Alg. 1, when applied to these data sequences, sequentially yields*

$$\max_{C\in\mathcal{S}(\boldsymbol{\phi})} E\left[R_T(f_t; f_t^{(C)})\right] \leq O\left(\frac{2^D}{T}\right), \tag{3}$$

for any T with a computational complexity $O(Dp)$, where p represents the dimensionality of the feature vectors and the expectation is with respect to the randomization parameters.

Proof of Theorem 1 and Construction of the Algorithm

Notation: We introduce the following notation to efficiently specify the nodes. Each node of the tree is labeled with a binary string $n = m_1 \dots m_d$, where $m_i = \{0, 1\}$ is a binary letter and d represents the depth of the node. For any inner node n, we label its left and right children as $n0$ and $n1$, respectively. We denote the empty string by λ. Moreover, we call a node $n' = m'_1 \dots m'_{d'}$ as the prefix of node $n = m_1 \dots m_d$ if $d' \leq d$ and $m'_i = m_i$ for all $i = 1, \dots, d'$. Using this definition, we denote n_i as the depth-i prefix to node n, where $i = \{0, \dots, d\}$. This labeling operation can be observed for a depth-2 tree in Figure 1.

We start the proof by explicitly constructing the base classifiers. We next introduce a low complexity method to achieve the best classifier among doubly exponential number

of different base classifiers. Then, we incorporate an adaptive method to optimize $\boldsymbol{\phi}$ in order to minimize the classification of the final algorithm.

Construction of the Base Classifiers: Suppose that the instance $\boldsymbol{x}_t$ has fallen into the region represented by some leaf node n. Then, $\boldsymbol{x}_t$ has also fallen to the nodes $n_0, \ldots, n_D$, where $n_D = n$ and $n_0 = \lambda$. Without loss of generality, assume that the node n_d is a leaf node of the subtree generating the base classifier C, then one can simply set $f_t^{(C)}(\boldsymbol{x}_t) = f_{t,n_d}(\boldsymbol{x}_t)$ as done in many prior work, cf. [6], [7]. In such conventional works, each instance is directly assigned to a node assuming that the base classifier will be able to classify that instance accurately.

However, in this paper, we acknowledge that a node classifier may not be able to classify each instance accurately since the partitioning of the feature space is set before the processing starts. Therefore, we assign each instance to a node with a certain weight (or probability) in order to be able to adaptively reconstruct the feature space partitioning. To this end, we define a parameter called "confidence rate" to measure the heaviness of the path between nodes n_d and λ. This parameter is defined as the multiplication of the separator functions of the nodes from the respective leaf node to the root node, which represents the confidence that $\boldsymbol{x}_t$ should be classified using the region classifier of node n_d. In particular, this confidence rate is defined as follows

$$c_{t,n_d}(\boldsymbol{x}_t) \triangleq \prod_{i=0}^{d-1} s_{t,n_i,m_{i+1}}(\boldsymbol{x}_t), \tag{4}$$

where $s_{t,n_i,m_{i+1}}(\cdot)$ represents the value of the partitioning function corresponding to node n_i towards the m_{i+1} direction, i.e.,

$$s_{t,n_i,m_{i+1}}(\boldsymbol{x}_t) \triangleq \begin{cases} s_{t,n_i}(\boldsymbol{x}_t) & \text{, if } m_{i+1} = 0 \\ 1 - s_{t,n_i}(\boldsymbol{x}_t) & \text{, if } m_{i+1} = 1 \end{cases}.$$

Intuitively, this confidence rate is low (i.e., close to $(0.5)^d$) when the feature vector is close to the region boundaries, hence we may consider to classify that feature vector by another node classifier (e.g., the classifier of the sibling node). Therefore, we consider that the classification output of node n_d can be trusted with a probability of $c_{t,n_d}(\boldsymbol{x}_t)$. Providing an error margin to the node classifier f_{t,n_d}, we consider that the complementary label $-f_{t,n_d}(\boldsymbol{x}_t)$ has a probability of $1 - c_{t,n_d}(\boldsymbol{x}_t)$. Then, the final classification output of node n_d is set to $\{f_{t,n_d}(\boldsymbol{x}_t), -f_{t,n_d}(\boldsymbol{x}_t)\}$ with probabilities $\{c_{t,n_d}(\boldsymbol{x}_t), 1 - c_{t,n_d}(\boldsymbol{x}_t)\}$, respectively. With abuse of notation, we continue to denote the node classifier by $f_{t,n_d}(\boldsymbol{x}_t)$. Finally, we set the output of the base classifier as follows $f_t^{(C)}(\boldsymbol{x}_t) = f_{t,n_d}(\boldsymbol{x}_t)$. By this procedure, we significantly increase the degree of freedom of the base classifiers, which helps us efficiently learn the feature space partitioning.

Direct Combination of Base Classifiers: Having constructed all base classifiers, we use a mixture-of-experts approach to achieve the performance of the best base classifier that minimizes the accumulated classification error. Before presenting this method, we first introduce certain definitions. Let the instantaneous expected empirical loss of the proposed classifier f_t at time t be denoted by $\ell_t(f_t) \triangleq E\left[\mathbb{1}_{\{f_t(\boldsymbol{x}_t) \neq y_t\}}\right]$, with the expectation taken with respect to the randomization parameters of the classifier f_t. Then, the expected empirical loss of this classifier over a sequence of length T can be found by $L_T(f_t) = \sum_{t=1}^{T} \ell_t(f_t)$.

We also define the effective region of each node n_d at time t as $\mathcal{R}_{t,n_d} \triangleq \left\{\boldsymbol{x} : P_{t,n_d}(\boldsymbol{x}) \geq (0.5)^d\right\}$. Then, according to the introduced structure of base classifiers, node n_d classifies an instance $\boldsymbol{x}_t$ only if $\boldsymbol{x}_t \in \mathcal{R}_{t,n_d}$. Therefore, the time accumulated expected empirical loss of any node n during the data stream is given by

$$L_{T,n} \triangleq \sum_{t \leq T : x_t \in \mathcal{R}_{t,n}} \ell_t(f_{t,n}). \tag{5}$$

Similarly, the time accumulated expected empirical loss of a base classifier $C \in \mathcal{S}(\boldsymbol{\phi})$ is found as $L_T^{(C)} \triangleq \sum_{n \in \mathcal{L}(C)} L_{T,n}$, where $\mathcal{L}(C)$ is the set of the leaf nodes of the subtree generating C.

Using these definitions, we introduce a direct implementation of the mixture-of-experts approach as follows. We set the final classification output of our algorithm as $f_t(\boldsymbol{x}_t) = \sum_{C \in \mathcal{S}(\boldsymbol{\phi})} w_t^{(C)} f_t^{(C)}$, where $w_t^{(C)} = 2^{-J(C)} \exp(-b\, L_{t-1}^{(C)}) / Z_{t-1}$, and prove that we can achieve the upper bound in (3) with these weights. Here, $b \geq 0$ is a constant controlling the learning rate of the algorithm, $J(C) \leq 2|\mathcal{L}(C)| - 1$ represents the number of bits required to code the classifier C (which satisfies $\sum_{C \in \mathcal{S}(\boldsymbol{\phi})} J(C) = 1$), and $Z_t = \sum_{C \in \mathcal{S}(\boldsymbol{\phi})} 2^{-J(C)} \exp(-b\, L_t^{(C)})$ is the normalization factor. We emphasize that although $f_t(\boldsymbol{x}_t) \in [-1, 1]$, the final output of the classifier can be set to $\{1, -1\}$ with probabilities $\{(1 + f_t(\boldsymbol{x}_t))/2, 1 - f_t(\boldsymbol{x}_t))/2\}$, yielding the desired expectation.

According to the definition of Z_t, the normalization parameter at the last iteration (i.e., the iteration at time T) satisfies

$$-\frac{1}{b} \log Z_T \leq L_T^{(C)} + \frac{J(C) \log 2}{b}, \tag{6}$$

$\forall C \in \mathcal{S}(\boldsymbol{\phi})$. We then make the following observation

$$Z_T = \prod_{t=1}^{T} \frac{Z_t}{Z_{t-1}} = \prod_{t=1}^{T} \left\{ \sum_{C \in \mathcal{S}(\boldsymbol{\phi})} w_t^{(C)} h_t(f_t^{(C)}) \right\}, \tag{7}$$

where the second equation follows from the definition of Z_t, $w_t^{(C)} \triangleq 2^{-J(C)} \exp(-b\, L_{t-1}^{(C)}) / Z_{t-1}$, and $h_t(f_t^{(C)}) \triangleq \exp(-b\, \ell_t(f_t^{(C)}))$. Here, we note that one can write

$$\ell_t(f_t^{(C)}) = E\left[\mathbb{1}_{\{f_t(\boldsymbol{x}_t) \neq y_t\}}\right] = \frac{1}{4} E\left[\left(y_t - f_t^{(C)}(\boldsymbol{x}_t)\right)^2\right].$$

Then, taking the second derivative of $h_t(f_t^{(C)})$ with respect to $f_t^{(C)}$, we obtain

$$h_t''(f_t^{(C)}) = \frac{b}{4} h_t(f_t^{(C)}) \left(b E\left[\left(y_t - f_t^{(C)}(\boldsymbol{x}_t)\right)^2\right] - 2 \right).$$

Algorithm 1 Online Adaptive Hierarchical Space Partitioning Classifier (AHSP)

for $t \geq 1$ **do**
  Propagate x_t from the root to the leaf and obtain the visited nodes $n_0, \ldots, n_D$.
  Calculate $c_{t,n_d}(\boldsymbol{x}_t)$ for all $d \in 0, ..., D$ using (4).
  Calculate $w_{t,n_d}(\boldsymbol{x}_t)$ for all $d \in 0, ..., D$ using (12).
  Draw a classification output $\{1, -1\}$ with probabilities $c_{t,n_d}(\boldsymbol{x}_t)$ and $1 - c_{t,n_d}(\boldsymbol{x}_t)$, respectively, to find $f_{t,n_d}(\boldsymbol{x}_t)$.
  Combine the node outputs $f_{t,n_d}(\boldsymbol{x}_t)$ with weights $w_{t,n_0}, \ldots, w_{t,n_D}$, and choose the final output randomly according to the combination.
  Update the region classifiers (perceptron) at the visited nodes [8].
  $\ell_t(f_t) \leftarrow \mathbb{1}_{\{f_t(\boldsymbol{x}_t) \neq y_t\}}$
  Update L_{t,n_d} for all $d \in 0, ..., D$ using (5).
  Apply the recursion in (10) to update M_{t+1,n_d} for all $d \in 0, ..., D$.
  Update the separator parameters $\boldsymbol{\phi}$ using (13).
end for

Note that we have $\frac{b}{4} h_t(f_t^{(C)}) \geq 0$, hence $h_t''(f_t^{(C)}) \leq 0$ if $b \leq 2/E[(y_t - f_t^{(C)}(\boldsymbol{x}_t))^2]$. Since $E[(y_t - f_t^{(C)}(\boldsymbol{x}_t))^2] \leq 4$, we have $h_t''(f_t^{(C)}) \leq 0$ for $b \leq 0.5$. Then, considering (7), we point out that $\sum_{C \in \mathcal{S}(\boldsymbol{\phi})} w_t^{(C)} = 1$, hence we have

$$\sum_{C \in \mathcal{S}(\boldsymbol{\phi})} w_t^{(C)} h_t(f_t^{(C)}) \leq h_t \left(\sum_{C \in \mathcal{S}(\boldsymbol{\phi})} w_t^{(C)} f_t^{(C)} \right), \qquad (8)$$

from the Jensen's inequality. Therefore, combining (6), (7), and (8), we obtain

$$\frac{L_T(f_t)}{T} \leq \frac{L_T^{(C)}}{T} + \frac{J(C) \log 2}{Tb},$$

which is the desired upper bound in (3) since $J(C) \leq 2^{D+1} - 1$, $\forall C \in \mathcal{S}(\boldsymbol{\phi})$.

An Efficient Combination Method: Although we achieve the desired upper bound in (3) with this combination method, the final algorithm f_t -in its current form- requires a computational complexity $O(1.5^{2^D} p)$ since $|\mathcal{S}(\boldsymbol{\phi})| \approx 1.5^{2^D}$. However, the set $\{f_t^{(C)}(\boldsymbol{x}_t)\}_{C \in \mathcal{S}(\boldsymbol{\phi})} = \{f_{t,n_d}(\boldsymbol{x}_t)\}_{0 \leq d \leq D}$ of all possible classification decisions for $\boldsymbol{x}_t \in \mathcal{R}_{t,n_D}$ has cardinality as small as $O(D)$. Namely, evaluating all the base classifiers in $\mathcal{S}(\boldsymbol{\phi})$ at the instance $\boldsymbol{x}_t$ to produce $f_t(\boldsymbol{x}_t)$ is unnecessary. In fact, the computational complexity for producing $f_t(\boldsymbol{x}_t)$ can be reduced from $O(1.5^{2^D} p)$ to $O(Dp)$ with the same exact combination over f_{t,n_d}'s using the new set of weights w_{t,n_d}, which can be straightforwardly derived as

$$w_{t,n_d} = \sum_{C \in \mathcal{S}(\boldsymbol{\phi}) \,:\, f_t^{(C)}(\boldsymbol{x}_t) = f_{t,n_d}(\boldsymbol{x}_t)} w_t^{(C)}. \qquad (9)$$

To efficiently calculate (9) with complexity $O(Dp)$, we consider the universal coding scheme and let

$$M_{t,n} \triangleq \begin{cases} \exp\left(-bL_{t,n}\right) & \text{, if } n \text{ has depth } D \\ \frac{1}{2}\left[M_{t,n0} M_{t,n1} + \exp\left(-bL_{t,n}\right)\right] & \text{, otherwise} \end{cases} \qquad (10)$$

for any node n and observe that we have $M_{t,\lambda} = Z_t$ [9]. Therefore, we can use the recursion (10) to obtain the denominator of the combination weights $w_t^{(C)}$. To efficiently calculate the nominator of (9), we introduce another intermediate parameter as follows. Letting n'_d denote the sibling of node n_d, we recursively define

$$\kappa_{t,n_d} \triangleq \begin{cases} \frac{1}{2} & \text{, if } d = 0 \\ \frac{1}{2} M_{t-1,n'_d} \, \kappa_{t,n_{d-1}} & \text{, if } 0 < d < D \\ M_{t-1,n'_d} \, \kappa_{t,n_{d-1}} & \text{, if } d = D \end{cases}, \qquad (11)$$

$\forall d \in \{0, \ldots, D\}$, where $\boldsymbol{x}_t \in \mathcal{R}_{t,n_D}$. Using the intermediate parameters in (10) and (11), it can be shown that we have

$$w_{t,n_d} = \frac{\kappa_{t,n_d} \exp\left(-b\, L_{t,n_d}\right)}{M_{t,\lambda}}. \qquad (12)$$

Hence, we can obtain the final output of the algorithm as $f_t(\boldsymbol{x}_t) = \sum_{d=0}^{D} w_{t,n_d} f_{t,n_d}(\boldsymbol{x}_t)$ with computational complexity $O(D)$.

Learning the Space Partitioning: We use the final output of the introduced algorithm and update the region boundaries of the tree to minimize the final classification error. To this end, we use the stochastic gradient descent method to update $\boldsymbol{\phi}$ as follows

$$\boldsymbol{\phi}_{t+1,n_d} = \boldsymbol{\phi}_{t,n_d} - (-1)^{m_{d+1}} \, \eta \, (x_t - f_t(\boldsymbol{x}_t)) \, \pi_{t,n_d} s_{t,n_d,m'_{d+1}}(\boldsymbol{x}_t) \; \boldsymbol{x}_t. \qquad (13)$$

$\forall d \in \{0, \ldots, D-1\}$, where η denotes the learning rate of the algorithm, m'_{d+1} represents the complementary letter to m_{d+1} from the binary alphabet $\{0, 1\}$, and

$$\pi_{t,n_d} \triangleq \begin{cases} f_{t,n_d}(\boldsymbol{x}_t) & \text{, if } d = D-1 \\ \pi_{t,n_{d+1}} + f_{t,n_d}(\boldsymbol{x}_t) & \text{, if } d < D-1 \end{cases}$$

is an intermediate parameter to perform the update in (13) with a computational complexity $O(p)$ for each node n_d, $d = 0, \ldots, D-1$, which results in a overall computational complexity of $O(Dp)$.

This concludes the proof of Theorem 1 and the pseudocode of the introduced algorithm (AHSP) can be found in Algorithm 1. □

V. Experiments

In this section, we compare the empirical performance of our method (AHSP) with some well known the state-of-the-art ensemble techniques which are AdaBoost algorithm (ABA) and GradientBoost algorithm (GBA) [10]. For our algorithm (AHSP), the learning rate is set to $\eta = 0.05$ and a depth-4 tree is used in all of the experiments. The perceptron algorithm [8] is used as the weak learners in the compared methods and as the region classifiers in our method. Note that the compared methods have linear complexity in the number

TABLE I: Average error rates on benchmark data sets. The first row in each set represents results with normalized data, i.e., each attribute is linearly mapped to $[-1, 1]$ and the second row represents the results with truncated data, i.e., $\boldsymbol{x}_t \leftarrow \frac{\boldsymbol{x}_t}{\max(\|\boldsymbol{x}_t\|,1)}$.

Veriler (Size/Dimension)	ABA	GBA	AHSP
Heart (270/13)	0.2396	0.2328	**0.2009**
	0.2400	0.2314	**0.2083**
Breast Cancer (683/10)	0.0544	0.0571	**0.0465**
	0.0538	0.0533	**0.0458**
Diabetes (768/8)	0.3243	0.3349	**0.2575**
	0.3258	0.3335	**0.2728**

of weak learners. In contrast, although our algorithm uses $2^{D+1} - 1$ local models, it has linear complexity in the tree depth D.

We have tested these algorithms on some of the data sets presented in [10]. Each method is sequentially presented with the same data sequence and we calculate the error rate for the complete stream. This process is repeated for 100 random permutations (10 for the data sets of length larger than 10000) and the average error rates are reported in Table I. As we can see here that the algorithm we presented here (AHSP) outperforms the state-of-the-art ensemble algorithms that is because AHSP algorithm designed to work well over complex data sets with strong non-linearities.

VI. Conclusion

We proposed an online supervised learning algorithm that is highly efficient in terms of computational scalability and appropriate for big data applications. In the proposed method, we combined the outputs of the basic linear classifiers defined in local regions to generate the decision. We showed that our approach jointly optimizes the partitioning structure and the corresponding local linear models. Using the resulting highly dynamic hierarchical structure, we proved an upper bound for the regret of the system for any given data stream of arbitrary length. We present a comprehensive experimental comparison and illustrate that our algorithm significantly outperforms the state-of-the-art techniques on various benchmark data sets.

VII. Acknowledgment

This work is in part supported by Turkish Academy of Science Outstanding Researcher Programme and Tubitak Contract No: 113E517.

References

[1] O. Bousquet and L. Bottou, "The tradeoffs of large scale learning," in *Advances in Neural Information Processing Systems*, 2008, pp. 161–168.

[2] T. Mohamadpoor and B. Pfister, "A boosting framework on grounds of online learning," in *Advances in Neural Information Processing Systems*, 2014, pp. 2267–2275.

[3] Joseph Wang and Venkatesh Saligrama, "Local supervised learning through space partitioning," in *Advances in Neural Information Processing Systems (NIPS)*, pp. 91–99. 2012.

[4] N. D. Vanli and S. 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.

[5] A. V. Aho and N. J. A. Sloane, "Some doubly exponential sequences," *Fibonacci Quarterly*, vol. 11, pp. 429–437, 1970.

[6] 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.

[7] Wei-Yin Loh, "Classification and regression trees," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 1, no. 1, pp. 14–23, 2011.

[8] Yoav Freund and Robert E Schapire, "Large margin classification using the perceptron algorithm," *Machine learning*, vol. 37, no. 3, pp. 277–296, 1999.

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

[10] Shang-Tse Chen, Hsuan-Tien Lin, and Chi-Jen Lu, "An online boosting algorithm with theoretical justifications," *International Conference on Machine Learning*, 2012.