Online Classification via Self-Organizing Space Partitioning

Huseyin Ozkan, N. Denizcan Vanli, and Suleyman S. Kozat, *Senior Member, IEEE*

***Abstract*—The authors study online supervised learning under the empirical zero-one loss and introduce a novel classification algorithm with strong theoretical guarantees. The proposed method is a highly dynamical self-organizing decision tree structure, which adaptively partitions the feature space into small regions and combines (takes the union of) the local simple classification models specialized in those regions. The authors' approach sequentially and directly minimizes the cumulative loss by jointly learning the optimal feature space partitioning and the corresponding individual partition-region classifiers. They mitigate overtraining issues by using basic linear classifiers at each region while providing a superior modeling power through hierarchical and data adaptive models. The computational complexity of the introduced algorithm scales linearly with the dimensionality of the feature space and the depth of the tree. Their algorithm can be applied to any streaming data without requiring a training phase or a priori information, hence processing data on-the-fly and then discarding it. Therefore, the introduced algorithm is especially suitable for the applications requiring sequential data processing at large scales/high rates. The authors present a comprehensive experimental study in stationary and nonstationary environments. In these experiments, their algorithm is compared with the state-of-the-art methods over the well-known benchmark datasets and shown to be computationally highly superior. The proposed algorithm significantly outperforms the competing methods in the stationary settings and demonstrates remarkable adaptation capabilities to nonstationarity in the presence of drifting concepts and abrupt/sudden concept changes.**

***Index Terms*—Online learning, sequential, classification, self-organizing, adaptive, tree, randomized algorithms.**

I. Introduction

IN the contemporary machine learning applications [1]–[4], algorithms are required to process data at an extremely fast rate, yet to learn complex models often in a non-stationary environment. In addressing this ambitious goal, one generally aims at maximally exploiting the information per instance—with only a single access—in the *online* setting and update the most recently learned hypothesis. To this end, we target such applications requiring sequential data processing at large scales/high rates and propose an online algorithm to learn arbitrarily complex and non-stationary structures with strong performance guarantees. In particular, we propose a novel, highly efficient and effective online classification algorithm, which can operate continuously—without an interrupt—on an infinite stream of possibly correlated (labeled) observations from a possibly non-stationary process. We study the problem under this wide generality without any statistical assumptions since we desire to draw conclusions about the worst case situations in the most realistic manner to effectively address the unknown environments, which might be non-stationary, chaotic and generate data even adversely [5]–[8].

To learn complex relations while exploiting local regularities, we consider completely adaptive piecewise linear models by partitioning the observation domain, i.e., the feature space, into different regions. Specifically, we use a binary partitioning tree, where a separator (e.g., a hyperplane split or partitioner) and an online linear classifier (a "simple model" such as the perceptron) are assigned to each node/region. The sequential losses of the regional classifiers (i.e., the simple models) are combined into a global loss that is parameterized over separator/split as well as the node/region classifier parameters. We minimize this global loss using the stochastic gradient descent method and obtain the updates for the complete set of tree parameters, i.e., the separators and the region classifiers, at each newly observed instance. The result is a highly dynamical self-organizing decision tree structure that jointly (and in a truly online manner) learns the region classifiers and the optimal feature space partitioning. In this respect, our strategy is highly novel and remarkably robust to drifting source statistics, i.e., non-stationarity. Since our approach is essentially based on a finite combination of linear models, it generalizes well and does not overfit or limitedly overfits [9] (as rigorously shown by our extensive set of experiments).

The introduced partitioning tree effectively defines a class of hierarchical partitions (of the feature space) and a corresponding class of a doubly exponential number ($\sim 1.5^{2^D}$, where D is the depth) of piecewise linear and online base classifiers, cf. Figs. 2 and 3. The proposed online classifier combines the outputs of these online base classifiers at each instance and generates its classification output. We prove that without any statistical assumptions, the proposed algorithm asymptotically, i.e., as $t \to \infty$, performs as well as the best base classifier (at time infinity or practically after processing sufficiently many data instances) that can only be chosen in hindsight. Here, the best base classifier is itself time varying and defined at time

Manuscript received October 07, 2015; revised February 15, 2016; accepted March 31, 2016. Date of publication April 21, 2016; date of current version June 22, 2016. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Gustau Camps-Valls. This work was supported in part by the Turkish Academy of Sciences Outstanding Researcher Program and in part by the Scientific and Technological Research Council of Turkey under Contract 113E517.

H. Ozkan is with the Department of Brain and Cognitive Sciences, Massachusetts Institute of Technology, Cambridge, MA 02139 USA (e-mail: hozkan@mit.edu).

N. D. Vanli is with the Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, Cambridge, MA 02139 USA (e-mail: denizcan@mit.edu).

S. S. Kozat is with the Department of Electrical and Electronics Engineering, Bilkent University, Bilkent, Ankara 06800, Turkey (e-mail: kozat@ee.bilkent.edu.tr).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSP.2016.2557307

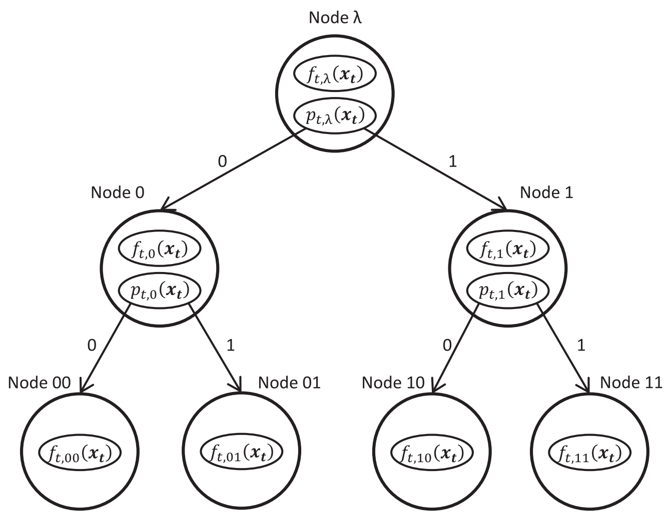

Node λ
$f_{t,\lambda}(\boldsymbol{x}_t)$
$p_{t,\lambda}(\boldsymbol{x}_t)$
0
1
Node 0
Node 1
$f_{t,0}(\boldsymbol{x}_t)$
$p_{t,0}(\boldsymbol{x}_t)$
$f_{t,1}(\boldsymbol{x}_t)$
$p_{t,1}(\boldsymbol{x}_t)$
Node 00
Node 01
Node 10
Node 11
$f_{t,00}(\boldsymbol{x}_t)$
$f_{t,01}(\boldsymbol{x}_t)$
$f_{t,10}(\boldsymbol{x}_t)$
$f_{t,11}(\boldsymbol{x}_t)$

Fig. 1. An illustration of the complete tree structure. $f_{t,n}(\cdot)$ and $p_{t,n}(\cdot)$ represent the classifier and the separator function of node n, respectively.

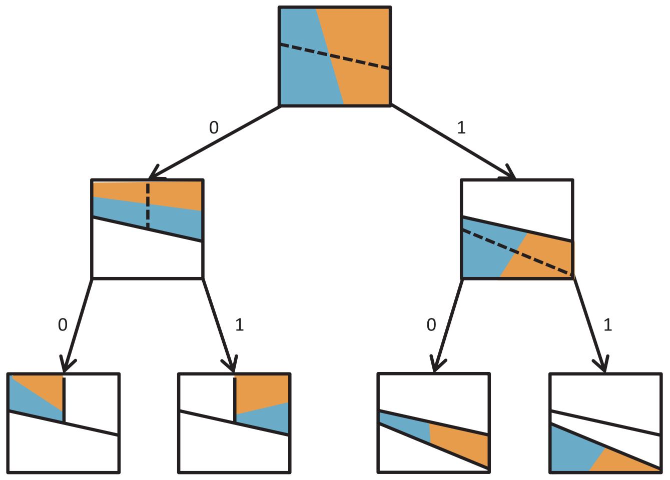


Fig. 2. The partitioning of a 2-dimensional feature space using a depth-2 tree. The whole feature space is first bisected by $p_{t,n=\lambda}$ and split into two regions $n = 0$ and $n = 1$: if an instance $\boldsymbol{\phi}_{t,n=\lambda}^T \boldsymbol{x}_t \geq 0$ (or if $p_{t,n=\lambda}(\boldsymbol{x}_t) \geq 0.5$ in (3)), then it follows the 1-branch; otherwise, it follows the 0-branch. The corresponding regions are similarly bisected by $p_{t,0}$ and $p_{t,1}$, respectively. The active region at a node resulting from the previous splits is illustrated colored, where the dashed line represents the separating hyperplane (whose normal vector is $\boldsymbol{\phi}_{t,n}$) at that node and the two differently colored subregions are the corresponding local classification by $f_{t,n}$.

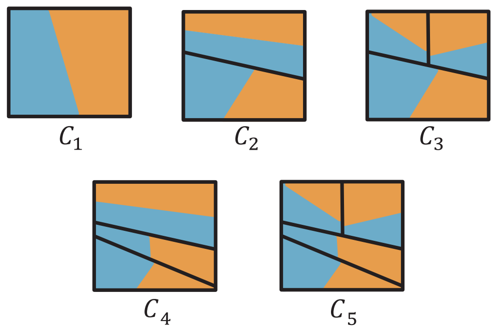


Fig. 3. The competition class of base classifiers defined by the depth-2 tree in Fig. 2. Each base classifier corresponds to a complete subtree in Fig. 2.

t as the base classifier, which achieves the smallest empirical loss, i.e., accumulated number of classification errors, at time t. For instance, in the case of a depth-2 tree partitioning, the best base classifier switches (in time from one classifier to another) in the class of base classifiers that is illustrated in Fig. 3. We point out that our algorithm also optimizes the structure of this partitioning, cf. Figs. 1 and 2; and each base classifier is a specific union of hierarchically arranged regional linear classifiers. All such possible unions generate the class of the base classifiers and the final combined classifier is in fact an optimal union classifier. Our results hold for every possible input stream regardless of the underlying data generation process. The computational complexity of the proposed algorithm is controllable over the depth of the tree and it linearly grows with the depth, the data dimensionality and the number of streamed instances. We perform an extensive set of experiments over both stationary and non-stationary real and synthetic data. In our stationary data experiments, our algorithm significantly outperforms the state-of-the-art as well as the most recently proposed techniques [10]–[14]. In our non-stationary data experiments, we experimentally analyze the proposed approach under the continuous concept drifts and abrupt concept changes [9], [15], [16], where we follow the comparison framework of [15]. We demonstrate that our algorithm also achieves a superior adaptation to non-stationarity, especially when it is not known what type expert is the best to combine with the competing ensemble methods or when the class separation is strongly non-linear. Furthermore, our algorithms (two versions) are computationally significantly more efficient compared to the proposals [9]–[12], [14]–[16].

A. *Related Work*

In the literature of classification and regression trees [17], [18], the split criteria are typically chosen a priori and fixed such as the dyadic partitioning [19]; and a specific loss, e.g., the gini index [20], is minimized separately for each node. For instance, multivariate trees are extended to allow the simultaneous use of functional inner and leaf nodes to draw a decision in [21]. Similarly, the node specific individual decisions are combined in [13] via the context tree weighting method [7] and a piecewise linear model for sequential classification is obtained. Since the tree structures in both of these methods are fixed and chosen even before the processing starts, the resulting modeling power is very limited and significantly deteriorates in case of high dimensionality [22]. In contrast, our algorithm provides a theoretically analyzed and computationally efficient fully adaptive tree for online classification, which learns the split criteria without any limitation or restriction. This generates a dynamical self-organizing tree that is sequentially tuned to the data and adaptive to high dimensionality. Moreover, we do not use any statistical assumptions regarding the data source unlike [23] and our analyses hold in a strong mathematical sense [24]. Self-organizing trees have been successfully applied to regression problems in a recent study [25]. However, the computational complexity of the algorithm [25] is exponential in the depth of the tree, whereas our approach operates with significantly smaller computational costs (i.e., linear in the depth of the tree). We also emphasize that both the problem as well as the algorithm in our study are completely different than [25]. Another successful application of the self-organizing trees is presented in [9] in the context of classification. The problem in [9] is formulated in the batch setting, thus it does not address our sequential requirements. Unlike [9], [26], where the final classifier is described by a single pruning of the tree (i.e., one of the partitions of the feature space defined by the decision tree), we consider the complete set of base classifiers constructed by all possible prunings.

Our approach combines simple models to obtain a strong online classifier, hence it is directly related to "boosting" [27], [28] and "predicting with expert advice" [5] methods of the online setting. We emphasize that the corresponding online algorithms [6], [10]–[12], [14]–[16], [24], [29]–[46] essentially consider a weighted linear combination of weak learners/experts that run independently in parallel with no structure. On the contrary, we consider the union of hierarchically structured simple models that yields a superior classification performance with minimal computational complexity. Several online boosting methods are proposed such as the Oza's online algorithm [10] and [29] that are asymptotically convergent to the batch solution of Adaboost [27]. The corresponding variants [30]–[33] are heuristically developed with no convergence results. These studies are collected in a single theoretical framework via the stochastic gradient descent in [34]; and their robustness is investigated under noisy labels in [11]. Further analytical justifications can be found in [12] (an online extension to [47]). Unlike these stochastic gradient descent methods, another ensemble method uses a Bayesian framework [14]. However, all these methods are appropriate only for stationary environments and based on weighted linear combinations of weak learners. In contrast, we consider unions of simple linear models that are adaptively and optimally (to minimize the final loss) located in the feature space with respect to the possible changes in the source statistics.

Our theoretical analyses are inline with the "predicting with expert advice" framework [5]. The weighted majority algorithm and similar aggregation strategies are presented in [24], [35], which are generalized in [36], [37], [39] to allow switching experts at a fixed and specified rate and modified for drifting concepts in [38]. The switching rate [36] is also incorporated into the learning algorithm in [6]. Generally, the algorithms proposed in this framework, such as [6], [24], [35]–[39], consider a fixed ensemble of expert algorithms and obtain an optimal weighted linear combination in terms of the regret bounds. Accordingly, we prove that the proposed algorithm asymptotically achieves the performance of the best union of the regional linear classifiers that can only be selected in hindsight. However, we do not restrict our algorithm to a fixed set of models since the non-stationarity has an unpredictable nature and cannot be confined to a fixed set of experts as in these studies [6], [24], [35]–[39]. A dynamically weighted majority algorithm that can enhance the ensemble by addition or removal of the experts to further adapt to non-stationarity, named as "concept drift" [40], is presented in [15]. Similar ensemble pruning/enhancing techniques are also considered in [16], [41]–[45] to avoid rigidity of the fixed set of experts/weak learners approach. In contrast, we follow a different approach: our algorithm learns the region classifiers and collectively organizes them at every instance. Neither the union structure (partitioning) nor the simple models in our approach are fixed; yet, the proposed algorithm achieves the performance of the optimal union classifier selected in hindsight. In [16], [41]–[46], concept changes are generally tracked in sliding windows of the stream, which results in incremental (not sequential since batch data is used) learning. In addition, a change detector in the windowed parts is used in [44], [46] to obtain increased adaptivity. On the other hand, the adaptation to concept changes in our work is due to the inherent joint learning process of the split criteria and the region classifiers in our tree. This brings the online processing capability that does not require windowed batch data. The algorithms in the literature of drifting concepts are generally heuristically developed. In this respect, in addition to the introduced error bounds in [15], the presented theoretical analysis in this study provides furthers insight into the non-stationary setting.

B. Summary of Contributions

1) We propose two novel online classification algorithms that are based on a highly dynamical self organizing decision tree, which sequentially learn both the optimal feature space partitioning and the optimal combination (union of) of the local linear models of the regions of the feature space partition. Our online algorithms are mathematically guaranteed—without any assumptions about the data source—to asymptotically perform as well the best classifier (obtained from the best combination of the local models) that can only be chosen in hindsight.
2) The proposed online algorithms generate a piece-wise linear model to learn complex relations while exploiting the local regularities in a completely data driven manner. Our approach generalizes well and (does not or) limitedly overfits with strong sequential adaptation to non-linearity even in the case of non-stationary data.
3) The computational complexity of our algorithms grow linearly with the data size, dimensionality and the depth of the partitioning tree. Hence, the proposed algorithms are computationally highly efficient and appropriate for sequential data processing at large scales/high rates.
4) The proposed algorithms significantly outperform the state-of-the-art competing techniques in our extensive stationary and non-stationary real data experiments.

After we provide the problem description in Section II, we introduce our sequential classifier and present the theoretical guarantees in Section III. We demonstrate the performance of our algorithms (two versions) via extensive experiments in Section IV and conclude with final remarks in Section V.

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] In particular, the aim is to learn a classification function $f_t(\boldsymbol{x}_t)$ with $\boldsymbol{x}_t \in \mathbb{R}^p$ and $y_t \in \{-1, 1\}$ such that when applied in an online manner to any streaming data, the empirical

[1]All vectors are column vectors and denoted by boldface lower case letters. Matrices are represented by boldface uppercase letters. For a vector $\boldsymbol{u}$, $\boldsymbol{u}^T$ is the ordinary transpose. Throughout the paper, the time index appears as a subscript.

loss of the classifier $f_t(\cdot)$, 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 (after averaging over T) the empirical loss of the best classifier $C(\boldsymbol{\phi})$ from a competition class $\mathcal{S}(\boldsymbol{\phi})$ of base classifiers for any sequence length T (where T is not a design parameter, i.e., our algorithms (two versions) are truly sequential). The set of classifiers $\mathcal{S}(\boldsymbol{\phi})$ is a parameter dependent class that can be optimized over $\boldsymbol{\phi}$, where $\boldsymbol{\phi}$ is not a specific algorithm dependent parameter such as the separation hyperplane of a linear classifier, but instead it determines the "shape" of the competition class.[2] Unlike the relevant works in the literature of "predicting with expert advice" [5], the goal, in this paper, is not only to achieve the performance of the best expert, i.e., the best base classifier, but also to optimize the competition class $\mathcal{S}(\boldsymbol{\phi})$ over the "shape" $\boldsymbol{\phi}$ to further and directly minimize the final error.

To be more precise, we measure the relative performance of f_t with respect to the performance of a base classifier[3] $f_t^{(C(\boldsymbol{\phi}))}$, where $C(\boldsymbol{\phi}) \in \mathcal{S}(\boldsymbol{\phi})$, using the following regret

$$R_T\left(f_t; f_t^{(C(\boldsymbol{\phi}))}\right) \triangleq \frac{L_T(f_t) - L_T\left(f_t^{(C(\boldsymbol{\phi}))}\right)}{T}, \tag{2}$$

for any arbitrary length T. Our aim is then *i)* to construct an online algorithm with guaranteed upper bounds on this regret for any base classifier and *ii)* to optimize over $\boldsymbol{\phi}$ in order to minimize the classification error. In this sense, the proposed algorithm f_t competes against the best base classifier that itself constantly improves. We emphasize that a classifier $C(\boldsymbol{\phi})$ in the competition class, i.e., $C(\boldsymbol{\phi}) \in \mathcal{S}(\boldsymbol{\phi})$, can be implemented in various ways. In this paper, we consider the union classifier, which can approximate any arbitrarily nonlinear class separation by piecewise linear curves with limited (or without) overfitting, cf. the finite VC dimension discussion in [9]. To efficiently construct this set of base classifiers that is dynamically adaptive to the data through the shape $\boldsymbol{\phi}$, we next introduce a self-organizing partitioning tree.

A. Adaptive Space Partitioning With Self-Organizing Trees

A tree—in our work—defines a nested partitioning of the feature space at each node of which, we have a simple region/local classifier (e.g., a linear and online classifier such as the perceptron) and a separator/partitioner (or a split) of the corresponding region. A generalized view of a depth-2 tree is given in Fig. 1, where $f_{t,n}$ represents the region classifier and $p_{t,n}$ represents the separator function of node n at time t. In Fig. 1, a depth-2 tree is used to partition the feature space as follows. The root node (or node λ) represents the entire feature space, where the separator function $p_{t,\lambda}$ bisects this region and creates node 0 and node 1. Similarly, each of these nodes are also split via $p_{t,0}$ and $p_{t,1}$ creating children nodes 00, 01 and 10, 11, respectively. The selection of the node classifiers and separator functions are completely up to preference and can be arbitrary. However, we use perceptrons as our node classifiers and hyperplanes as our node separators. The separator $p_{t,n}$ is a function of $\boldsymbol{\phi}_{t,n}$ such as the sigmoid

$$p_{t,n}(\boldsymbol{x}_t) = \frac{1}{1 + e^{\boldsymbol{\phi}_{t,n}^T \boldsymbol{x}_t}}, \tag{3}$$

where $\boldsymbol{\phi}_{t,n}$ is the angle of the normal line to the separating hyperplane for each node n. We consider these differentiable separator functions as randomized decisions such that $p_{t,n}(\boldsymbol{x}_t)$ is the probability of assigning $\boldsymbol{x}_t$ to the right child node of n.

As an example, the 2-dimensional feature space is partitioned using a depth-2 tree in Fig. 2. Operationally, each instance $\boldsymbol{x}_t$ is propagated from the root node to a leaf node through a certain branch such that if $\boldsymbol{\phi}_{t,n}^T \boldsymbol{x}_t \geq 0$ (cf. (3)) at 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 local node (region) classifier. In Fig. 2, the dashed lines represent the partitioning of the feature space corresponding to each inner node and the two different colored regions represent the node classifier outputs at the respective regions. Each complete subtree (or pruning) generates a certain partition with a certain piecewise linear classification structure and hence, yields a complete base classifier. According to the partitioning in Fig. 2, 5 different base classifiers producing the competition class can be defined and these classifiers are presented in Fig. 3. Note that for a base classifier $C(\boldsymbol{\phi}) \in \mathcal{S}(\boldsymbol{\phi})$, the output $f_t^{(C(\boldsymbol{\phi}))}(\boldsymbol{x}_t)$ makes its decision according to the decision 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 $C(\boldsymbol{\phi})$.

Based on this tree partitioning, for a depth-D tree, there exist approximately 1.5^{2^D} different base classifiers [48]. The proposed classifier f_t combines the outputs of all these base classifiers at each time and generates its final output with the desired competitive classification performance. Namely, we achieve a diminishing regret in (2) for any base classifier such that the performance of the best base classifier is matched. Note that each base classifier is an online union classifier as it operates on a union of the regions spanning the entire feature space with simple and online linear region classifiers at each region. In addition to the goal of obtaining the competitive performance, we also aim to constantly improve this competitive performance by directly improving the performances of the base classifiers (i.e., competitors) over time through jointly learning the optimal region classifiers as well as the optimal partitioning structure. Hence, the proposed algorithm f_t is competitive against a competition class that itself is designed to constantly improve over time. To this end, we parameterize the introduced tree over the collection of the separator function parameters $\boldsymbol{\phi} = \{\boldsymbol{\phi}_{t,n}\}$, which also defines the aforementioned "shape" of our competition class $\mathcal{S}(\boldsymbol{\phi})$. Then, we directly minimize the resulting final classification loss of f_t in (1) over the complete set of tree parameters. We refer this structure as self-organizing tree.

In this framework, we emphasize two points. First, there is a $1-1$ mapping between the partitions and the base classifiers, and we use these phrases interchangeably: $C(\boldsymbol{\phi})$ is referred as a partition or a base classifier conveniently for clarity. However, we use the notation $f_t^{(C(\boldsymbol{\phi}))}$ to precisely denote the actual operational and online base classifier. Second, these partitions, the

[2]The precise meaning of the parameter $\boldsymbol{\phi}$ will become clear shortly.

[3]We use the notation $f_t^{(C(\boldsymbol{\phi}))}$ to precisely denote the actual operational and online base classifier $C(\boldsymbol{\phi}) \in \mathcal{S}(\boldsymbol{\phi})$.

simple region classifiers and hence the resulting base classifiers are all time varying due to our online setting. In this setting, our goal is to find a computationally efficient sequential algorithm that achieves the performance of the optimal base classifier (cf. (2)) and also simultaneously improves that optimal base classifier.

III. Adaptive Tree-Based Non-Linear Classifier

In this section, we propose two variants of our online classifier f_t. 1) ATNC.Rnd: f_t randomly chooses a base classifier using a specific weighting scheme over the class $\mathcal{S}(\boldsymbol{\phi})$ and matches the output of the chosen base classifier; and 2) ATNC.Avg: f_t outputs the weighted average of the base classifier decisions. Here, ATNC stands for "Adaptive Tree-based Non-linear Classifier", where the non-linear classification capability is due to the introduced union structure.

In our theoretical analysis, we concentrate on the algorithm ATNC.Rnd; however, our results hold for ATNC.Avg as well in a straightforward manner. We prove that the performance of the proposed algorithm f_t, i.e., ATNC.Rnd, is asymptotically as well as the best base classifier, where the adaptation of $\boldsymbol{\phi}$ results significant performance gains as it directly improves the competitors, i.e., the best base classifier. In particular, we provide an upper bound on the regret $R_T(f_t; f_t^{(C(\boldsymbol{\phi}))})$ such that as $T \to \infty$, $R_T(f_t; f_t^{(C(\boldsymbol{\phi}))}) \to 0$ for this highly dynamical self-organizing tree. 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 universal randomized union classifier presented in Alg. 1, i.e., ATNC.Rnd, when applied to these data sequences, sequentially yields

$$\max_{C(\boldsymbol{\phi})\in\mathcal{S}(\boldsymbol{\phi})} E\left[R_T\left(f_t; f_t^{(C(\boldsymbol{\phi}))}\right)\right] \leq O\left(\sqrt{\frac{2^D}{T}}\right), \quad (4)$$

for all 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 Algorithm ATNC. Rnd: Note that by this theorem, we present performance guarantees for finite or infinite data since our results hold for all T without any limitation on the amount of the data that might be as small as one single data instance or even infinite, where T is not a design parameter (the horizon is thus unknown in this work as an important algorithmic capability), i.e., our algorithms (two versions) are truly sequential. However, we observe that the introduced upper bound (which is rate optimal [5], i.e., it cannot be improved in terms of its rate of convergence with respect to T) on the regret is convergent to 0 as $T \to \infty$ and therefore our algorithm asymptotically performs as well as the best base classifier. On the other hand, for finite T, our results hold and provide performance guarantees (again with respect to the best base classifier) in a rate-optimal manner. Secondly, note that the proposed algorithm ATNC.Rnd randomly chooses one of the base classifiers at each time t with respect to a certain set of weights and matches its output to declare the classification decision. The precise definition of this randomization will become clear shortly in the development. Therefore, the expectation in our theorem is with respect to this internal algorithmic randomization, i.e., weights over the base classifiers; and it is not in any way related to the data statistics. In fact, our results hold for every possible input stream regardless of its stationary or non-stationary unknown statistics. We start the proof by 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 optimizing $\boldsymbol{\phi}$ to minimize the classification of the final algorithm.

A. Preliminaries and the Competition Class $\mathcal{S}(\boldsymbol{\phi})$

Before proceeding, we first introduce the following notation. For ease of exposition in specifying the nodes, each node of the tree is labeled with a binary string $n = m_1 \ldots m_d$, 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 $n' = m'_1 \ldots m'_{d'}$ as the prefix of node $n = m_1 \ldots m_d$ if $d' \leq d$ and $m'_i = m_i$ for all $i = 1, \ldots, d'$. Using this definition, we denote n_i as the depth-i prefix to node n, where $i = \{0, \ldots, d\}$. This labeling operation can be observed for a depth-2 tree in Fig. 1.

According to the partitioning method described in Section II-A, the output of a base classifier $C(\boldsymbol{\phi}) \in \mathcal{S}(\boldsymbol{\phi})$ is softly constructed using the partitioning functions $p_{t,n}$ as follows. Without loss of generality, suppose that the instance $\boldsymbol{x}_t$ has fallen into the region represented by the leaf node n. Then, $\boldsymbol{x}_t$ is contained in the nodes $n_0, \ldots, n_D$, where $n_D = n$ and $n_0 = \lambda$. For example, if node n_d is a leaf node of the subtree generating the base classifier $C(\boldsymbol{\phi})$, then one can simply set $f_t^{(C(\boldsymbol{\phi}))}(\boldsymbol{x}_t) = f_{t,n_d}(\boldsymbol{x}_t)$. Instead of making a hard selection, we allow an error margin for the classification output $f_{t,n_d}(\boldsymbol{x}_t)$ in order to be able to update the region boundaries later in the proof. To achieve this, for each leaf node of $C(\boldsymbol{\phi})$, we define a parameter called "path probability" to measure the contribution of each leaf node to the classification task at time t. This parameter is equal to the multiplication of the partitioning functions of the nodes from the respective leaf node to the root node, which represents the probability that $\boldsymbol{x}_t$ should be classified using the region classifier of node n_d. This path probability is defined as

$$P_{t,n_d}(\boldsymbol{x}_t) \triangleq \prod_{i=0}^{d-1} p_{t,n_i,m_{i+1}}(\boldsymbol{x}_t), \quad (5)$$

where $p_{t,n_i,m_{i+1}}(\cdot)$ represents the value of the partitioning function corresponding to node n_i towards the m_{i+1} direction: $p_{t,n_i,m_{i+1}}(\boldsymbol{x}_t) \triangleq p_{t,n_i}(\boldsymbol{x}_t)$, if $m_{i+1} = 0$ and $p_{t,n_i,m_{i+1}}(\boldsymbol{x}_t) \triangleq 1 - p_{t,n_i}(\boldsymbol{x}_t)$, if $m_{i+1} = 1$ with $p_{t,n_i}(\boldsymbol{x}) = [1 + \exp(\boldsymbol{\phi}_{t,n_i}^T \boldsymbol{x})]^{-1}$ as in (3). We consider that the classification output of node n_d can be trusted with a probability of $P_{t,n_d}(\boldsymbol{x}_t)$. This and the other probabilities in our development are independently defined for ease of exposition and gaining intuition, i.e., these probabilities are not related to the unknown data statistics in any way and they definitely cannot be regarded as certain assumptions on the data. Indeed, we do not take any assumptions about the data source in this study.

Intuitively, the path probability is low 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). Using this path probabilities, we aim to update the region boundaries by learning whether an efficient node classifier is used to classify $\boldsymbol{x}_t$, instead of directly assigning $\boldsymbol{x}_t$ to node n_d and lose a significant degree of freedom. To this end, we define the final output of each node classifier according to a Bernoulli random variable with outcomes $\{-f_{t,n_d}(\boldsymbol{x}_t), f_{t,n_d}(\boldsymbol{x}_t)\}$ where the probability of the latter outcome is $P_{t,n_d}(\boldsymbol{x}_t)$. Although the final classification output of node n_d is generated according to this Bernoulli random variable, we continue to call $f_{t,n_d}(\boldsymbol{x}_t)$ the final classification output of node n_d, with an abuse of notation. Then, the classification output of the base classifier is set to $f_t^{(C(\boldsymbol{\phi}))}(\boldsymbol{x}_t) = f_{t,n_d}(\boldsymbol{x}_t)$.

After constructing 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 empirical loss of the proposed classifier f_t at time t be denoted by $\ell_t(f_t) \triangleq \mathbb{1}_{\{f_t(\boldsymbol{x}_t)\neq y_t\}}$. Then, the expected empirical loss of this classifier over a sequence of length T can be found by

$$L_T(f_t) = E\left[\sum_{t=1}^{T} \ell_t(f_t)\right], \tag{6}$$

with the expectation taken with respect to the randomization parameters of the classifier f_t. We also define the effective region of each node n_d at time t as follows $\mathcal{R}_{t,n_d} \triangleq \{\boldsymbol{x} : P_{t,n_d}(\boldsymbol{x}) \geq (0.5)^d\}$. Note that according to the aforementioned 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 empirical loss of any node n during the data stream is given by

$$L_{T,n} \triangleq \sum_{t\leq T:\{\boldsymbol{x}_t\}_{t\geq 1}\in\mathcal{R}_{t,n}} \ell_t(f_{t,n}). \tag{7}$$

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

Remark 1: For example, if one prunes our binary partitioning tree such that the deepest level is excluded, i.e., such that the resulting subtree includes only Node-λ, Node-0 and Node-1 (Fig. 1), then this subtree corresponds to the base classifier $C_2(\boldsymbol{\phi})$, cf. Figs. 2 and 3 (where the argument $\boldsymbol{\phi}$ is dropped for simplicity), and is said to generate $C_2(\boldsymbol{\phi})$ (as mentioned before). In this case, since $\mathcal{L}(C_2(\boldsymbol{\phi}))$ is the set of the leaf nodes of the subtree generating $C_2(\boldsymbol{\phi})$, we have $\mathcal{L}(C_2(\boldsymbol{\phi})) = \{\mathrm{Node}-0, \mathrm{Node}-1\}$. On the other hand, $J(C_2(\boldsymbol{\phi}))$ measures the "complexity" of the base classifier $C_2(\boldsymbol{\phi})$ based on the "number of bits required to represent the subtree generating the classifier $C_2(\boldsymbol{\phi})$" (this can also be seen as the size of a pruning [49]), for which we have $J(C_2(\boldsymbol{\phi})) \leq 2\mathcal{L}(C_2(\boldsymbol{\phi})) - 1$; and $\sum_{C(\boldsymbol{\phi})\in\mathcal{S}(\boldsymbol{\phi})} J(C(\boldsymbol{\phi})) = 1$ in general [7], [50]. Note that this is a popular prior in the coding literature cf. [7], [49], [50] and the references therein.

B. Definition of the Proposed Algorithm That Achieves the Performance Guarantees of Theorem 1:

Using these preliminaries, we define the proposed algorithm and first introduce a direct and inefficient implementation of our mixture-of-experts approach. We set the final classification output of our algorithm as $f_t(\boldsymbol{x}_t) = f_t^{(C(\boldsymbol{\phi}))}$ with probability $w_t^{(C(\boldsymbol{\phi}))}$, where $w_t^{(C(\boldsymbol{\phi}))} = 2^{-J(C(\boldsymbol{\phi}))}\exp(-b\,L_{t-1}^{(C(\boldsymbol{\phi}))})/Z_{t-1}$, and prove that we can achieve the upper bound in (4) with these weights. Here, $b \geq 0$ is a constant controlling the learning rate of the algorithm, $J(C(\boldsymbol{\phi})) \leq 2|\mathcal{L}(C(\boldsymbol{\phi}))| - 1$ represents the number of bits required to code the classifier $C(\boldsymbol{\phi})$ (which satisfies $\sum_{C(\boldsymbol{\phi})\in\mathcal{S}(\boldsymbol{\phi})} J(C(\boldsymbol{\phi})) = 1$), and $Z_t = \sum_{C(\boldsymbol{\phi})\in\mathcal{S}(\boldsymbol{\phi})} 2^{-J(C(\boldsymbol{\phi}))}\exp(-b\,L_t^{(C(\boldsymbol{\phi}))})$ is the normalization factor. Since Z_t is—by definition—a summation of terms that are all positive, we have $Z_t \geq 2^{-J(C(\boldsymbol{\phi}))}\exp(-b\,L_t^{(C(\boldsymbol{\phi}))})$ and, after taking the logarithm of both sides and arranging the terms,

$$-\frac{1}{b}\log Z_T \leq L_T^{(C(\boldsymbol{\phi}))} + \frac{J(C(\boldsymbol{\phi}))\log 2}{b} \tag{8}$$

$\forall C(\boldsymbol{\phi}) \in \mathcal{S}(\boldsymbol{\phi})$ at the (last) iteration at time T. We then make the following observation $Z_T = \prod_{t=1}^{T} \frac{Z_t}{Z_{t-1}}$ and

$$rZ_T = \prod_{t=1}^{T}\left\{\sum_{C(\boldsymbol{\phi})\in\mathcal{S}(\boldsymbol{\phi})} \frac{2^{-J(C(\boldsymbol{\phi}))}\exp\left(-b\,L_{t-1}^{(C(\boldsymbol{\phi}))}\right)}{Z_{t-1}} \times \exp\left(-b\,\ell_t\left(f_t^{(C(\boldsymbol{\phi}))}\right)\right)\right\}$$
$$\leq \exp\left(-b\,L_T(f_t) + \frac{Tb^2}{8}\right), \tag{9}$$

where the second line follows from the definition of Z_t and the last line follows from the Hoeffding's inequality [51] by treating the $w_t^{(C(\boldsymbol{\phi}))} \triangleq 2^{-J(C(\boldsymbol{\phi}))}\exp(-b\,L_{t-1}^{(C(\boldsymbol{\phi}))})/Z_{t-1}$ terms as the randomization probabilities. Note that $L_T(f_t)$ represents the expected loss of the final algorithm, cf. (6). Combining (8) and (9), we obtain

$$\frac{L_T(f_t)}{T} \leq \frac{L_T^{(C(\boldsymbol{\phi}))}}{T} + \frac{J(C(\boldsymbol{\phi}))\log 2}{Tb} + \frac{b}{8},$$

and choosing $b = \sqrt{2^D/T}$, we find the desired upper bound in (4) since $J(C(\boldsymbol{\phi})) \leq 2^{D+1} - 1, \forall C(\boldsymbol{\phi}) \in \mathcal{S}(\boldsymbol{\phi})$.

C. Efficient Implementation of the Proposed Algorithm and the Adaptive Feature Space Partitioning

Although we achieve the desired upper bound in (4) with this randomization method, the final algorithm f_t—in its current form—requires a computational complexity $O(1.5^{2^D}p)$ since the randomization $w_t^{(C(\boldsymbol{\phi}))}$ is performed over the set $\mathcal{S}(\boldsymbol{\phi})$ and $|\mathcal{S}(\boldsymbol{\phi})| \approx 1.5^{2^D}$. However, the set of all possible classification decisions has a cardinality as small as $D+1$ since $\boldsymbol{x}_t \in \mathcal{R}_{t,n_D}$ for the corresponding leaf node n_D (in which $\boldsymbol{x}_t$ is included) and $f_t^{(C(\boldsymbol{\phi}))} = f_{t,n_d}$ for some $d = 0,\ldots,D, \forall C(\boldsymbol{\phi}) \in \mathcal{S}(\boldsymbol{\phi})$. Hence, 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 $O(Dp)$ by performing the exact same randomization over f_{t,n_d}'s using the new set of weights w_{t,n_d}, which can be straightforwardly derived as

$$w_{t,n_d} = \sum_{C(\boldsymbol{\phi})\in\mathcal{S}(\boldsymbol{\phi}) : f_t^{(C(\boldsymbol{\phi}))}(\boldsymbol{x}_t)=f_{t,n_d}(\boldsymbol{x}_t)} w_t^{(C(\boldsymbol{\phi}))}. \qquad (10)$$

To efficiently calculate (10) with complexity $O(Dp)$, we consider the universal coding scheme and let

$$M_{t,n} \triangleq \begin{cases} \exp(-bL_{t,n}), & \text{if } n \text{ has depth } D \\ \frac{1}{2}\left[M_{t,n0}M_{t,n1} + \exp(-bL_{t,n})\right], & \text{otherwise} \end{cases} \qquad (11)$$

for any node n and observe that we have $M_{t,\lambda} = Z_t$ [50]. Therefore, we can use the recursion (11) to obtain the denominator of the randomization probabilities $w_t^{(C(\boldsymbol{\phi}))}$. To efficiently calculate the nominator of (10), we introduce another intermediate parameter as follows. Letting n'_d denote the sibling of node n_d, we recursively define

$$\kappa_{t,n_d} \triangleq \begin{cases} \frac{1}{2}, & \text{if } d = 0 \\ \frac{1}{2}M_{t-1,n'_d}\,\kappa_{t,n_{d-1}}, & \text{if } 0 < d < D \\ M_{t-1,n'_d}\,\kappa_{t,n_{d-1}}, & \text{if } d = D \end{cases}, \qquad (12)$$

$\forall d \in \{0,\ldots,D\}$, where $\boldsymbol{x}_t \in \mathcal{R}_{t,n_D}$. Using the intermediate parameters in (11) and (12), it can be shown that we have

$$w_{t,n_d} = \frac{\kappa_{t,n_d}\exp(-b\,L_{t,n_d})}{M_{t,\lambda}}. \qquad (13)$$

Hence, we can obtain the final output of the algorithm as $f_t(\boldsymbol{x}_t) = f_{t,n_d}(\boldsymbol{x}_t)$ with probability w_{t,n_d}, where $d \in \{0,\ldots,D\}$ (i.e., with a computational complexity $O(D)$).

We then use the final output of the introduced algorithm and update the region boundaries of the tree (i.e., organize the tree) to minimize the final classification error. To this end, we minimize the loss $E[\ell_t(f_t)] = E[\mathbb{1}_{\{f_t(\boldsymbol{x}_t)\neq y_t\}}] = \frac{1}{4}E[(y_t - f_t(\boldsymbol{x}_t))^2]$ with respect to the region boundary parameters, i.e., we use the stochastic gradient descent method, as follows

$$\begin{aligned}\boldsymbol{\phi}_{t+1,n_d} &= \boldsymbol{\phi}_{t,n_d} - \eta\nabla E[\ell_t(f_t)] \\ &= \boldsymbol{\phi}_{t,n_d} - (-1)^{m_{d+1}}\eta\,(y_t - f_t(\boldsymbol{x}_t))\,p_{t,n_d,m'_{d+1}}(\boldsymbol{x}_t) \\ &\quad \times \left[\sum_{i=d+1}^{D} f_{t,n_i}(\boldsymbol{x}_t)\right]\boldsymbol{x}_t, \end{aligned} \qquad (14)$$

$\forall d \in \{0,\ldots,D-1\}$, where η denotes the learning rate of the algorithm and m'_{d+1} represents the complementary letter to m_{d+1} from the binary alphabet $\{0,1\}$. Defining a new intermediate variable $\pi_{t,n_d} \triangleq f_{t,n_d}(\boldsymbol{x}_t)$, if $d = D-1$ and $\pi_{t,n_d} \triangleq \pi_{t,n_{d+1}} + f_{t,n_d}(\boldsymbol{x}_t)$, if $d < D-1$; one can perform the update in (14) with a computational complexity $O(p)$ for each node $n_d, d = 0,\ldots,D-1$, resulting in an overall computational complexity of $O(Dp)$ as follows

Algorithm 1: ATNC.Rnd.

```
1:  for $t = 1$ to $T$ do
2:      Propagate $\{\boldsymbol{x}_t\}_{t\geq 1}$ from the root to the leaf and
        obtain the visited nodes $n_0,\ldots,n_D$.
3:      Calculate $P_{t,n_d}(\boldsymbol{x}_t)$ for all $d \in 0,\ldots,D$ using (5).
4:      Calculate $w_{t,n_d}(\boldsymbol{x}_t)$ for all $d \in 0,\ldots,D$ using (13).
5:      Draw a node among $n_0,\ldots,n_D$ with probabilities
        $w_{t,n_0},\ldots,w_{t,n_D}$, respectively; suppose that $n_d$ is
        drawn.
6:      Draw a classification output $\{1,-1\}$ with
        probabilities $P_{t,n_d}(\boldsymbol{x}_t)$ and $1 - P_{t,n_d}(\boldsymbol{x}_t)$,
        respectively; $f_t(\boldsymbol{x}_t)$ is equated to the selected output.
7:      Update the region classifiers (perceptron) at the
        visited nodes [52].
8:      $\ell_t(f_t) \leftarrow \mathbb{1}_{\{f_t(\boldsymbol{x}_t)\neq y_t\}}$
9:      Update $L_{t,n_d}$ for all $d \in 0,\ldots,D$ using (7).
10:     Apply the recursion in (11) to update $M_{t+1,n_d}$ for all
        $d \in 0,\ldots,D$.
11:     Update the separator parameters $\boldsymbol{\phi}$ using (15).
12: end for
```

$$\begin{aligned}\boldsymbol{\phi}_{t+1,n_d} &= \boldsymbol{\phi}_{t,n_d} - (-1)^{m_{d+1}}\eta\,(y_t - f_t(\boldsymbol{x}_t))\,\pi_{t,n_d} \\ &\quad \times p_{t,n_d,m'_{d+1}}(\boldsymbol{x}_t)\;\boldsymbol{x}_t. \end{aligned} \qquad (15)$$

Note that in (15), both $p_{t,n_d}(\boldsymbol{x}_t)$ and $1 - p_{t,n_d}(\boldsymbol{x}_t)$ terms appear in the product, which can disturb the learning rate of the algorithm if $p_{t,n_d}(\boldsymbol{x}_t)$ is close to 0 or 1. Therefore, in order to avoid such a scenario, using a small positive constant $p_{\lim} > 0$, the partitioning function can be restricted to $[p_{\lim}, 1 - p_{\lim}]$, i.e., $0 < p_{\lim} \leq p_{t,n_d}(\boldsymbol{x}_t) \leq 1 - p_{\lim}$, by the following

$$p_t(\boldsymbol{x}_t) = p_{\lim} + (1 - 2p_{\lim})\frac{1}{1 + e^{\boldsymbol{\phi}_t^T\boldsymbol{x}_t}}.$$

This concludes the proof of Theorem 1 and the pseudocode of the ATNC.Rnd can be found in Algorithm 1. □

Remark 2: Instead of randomly selecting a base classifier and repeating its output to generate the final decision, the same randomization probabilities can be used as weighting factors. In this case, the outputs of all base classifiers are combined and our results hold in a similar expectation sense. We denote this classifier ATNC.Avg, cf. Section IV.

IV. Experiments

In this section, we demonstrate the performance of the proposed algorithms (two versions: ATNC.Rnd and ATNC.Avg) through several experiments in three separate parts. In the first part, we concentrate on stationary cases, where the source statistics are stationary over time. We show that in this case, our algorithm successfully combines simple classification models and significantly outperforms the most recent ensemble techniques [10]–[14]. We then study non-stationary cases and illustrate the adaptation power of our algorithm to concept changes/drifts with respect to the state-of-the-art approaches [9], [15], [16]. In the final part, we present the computational running times of the compared methods.

A. Stationary Data

In this part, we study our algorithms in the stationary environments when the data source statistics do not change over time; and in particular, we follow the comparison framework of [12] for this purpose. We compare our algorithms (ATNC.Rnd and ATNC.Avg) with the following state-of-the-art as well as the most recently proposed techniques: *Online AdaBoost*—"OZAB" [10]; *Online GradientBoost*—"OGB" [11]; *Online SmoothBoost*—"OSB" [12]; *Online SmoothBoost with Online Convex Programming*—"OSB.Ocp" [12]; and *Online Tree based Non-adaptive Competitive Classification*—"TNC.Rnd" [13]. The parameters for all of these compared methods are set as in their original proposals. For the method *Online GradientBoost*—"OGB" [11] which uses K weak learner per M selectors essentially resulting in MK weak learners in total, we use $K = 1$, as in [12], for a fair comparison along with the logit loss that has been shown to consistently outperform other choices in [11]. The method *Online Tree based Non-adaptive Competitive Classification*—"TNC.Rnd" [13] is non-adaptive, i.e., not self organizing, in terms of the space partitioning, which we use in our comparisons to illustrate the gain due the self organizing structure proposed in this paper (the depth of the tree is set to 4 for this method uniformly in all of our experiments).

We use the perceptron algorithm [52] as the weak learners in these compared methods and as the simple local models in our algorithms (ATNC.Rnd and ATNC.Avg) and in TNC.Rnd. We set $\eta = 0.05$ (learning rate) and $D = 4$ (tree depth) in our algorithms uniformly in all of our stationary as well as non-stationary data experiments. We use $N = 100$ weak learners for all other methods. Note that a depth-4 tree corresponds to $31 = 2^5 - 1$ local models. The proposed algorithms have linear complexity in the depth, whereas the compared methods have linear complexity in the number of weak learners.

In addition to the datasets of the processed format[4] as in [12], we also study the well-known "Banana" dataset and a binarized multi-class data "BMC" consisting of 6 identical Gaussian components that are located in two dimensions such that the separation between the two classes is highly nonlinear, cf. Fig. 4. 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 datasets of length longer than 10000) and the average error rates (along with the standard deviations) are reported in Table I. For a fair comparison, we truncate each data instance to unitary norm, i.e., $\boldsymbol{x}_t \leftarrow \frac{\boldsymbol{x}_t}{\max(\|\boldsymbol{x}_t\|,1)}$, as in [12] and our corresponding results are in the second row of Table I. In the first row, we present the results for the data normalized to the range $[-1, 1]$ without the norm truncation.

Our algorithms (including TNC.Rnd [13]) consistently outperform the other methods with only one exception in case of the "Mushrooms" dataset[5]. In particular, the compared

[4]http://www.csie.ntu.edu.tw/cjlin/libsvmtools/datasets/

[5]For instance, ATNC.Avg yields an average error rate of 4.65, whereas OSB yields 5.36 on the "Breast Cancer" dataset, both with a standard deviation of 0.5. Therefore, in this case, we say that the proposed technique ATNC.Avg outperforms OSB with 76% confidence, since $Pr(Er_{\mathrm{ATNC.Avg}} \leq \tau) \simeq 0.76$ and $Pr(Er_{OSB} \geq \tau) \simeq 0.76$, where we assume Gaussian distribution and $\tau = \frac{5.36+4.65}{2}$. In other cases except the "Mushrooms" dataset, our algorithms are observed to be superior with highly stronger confidence.

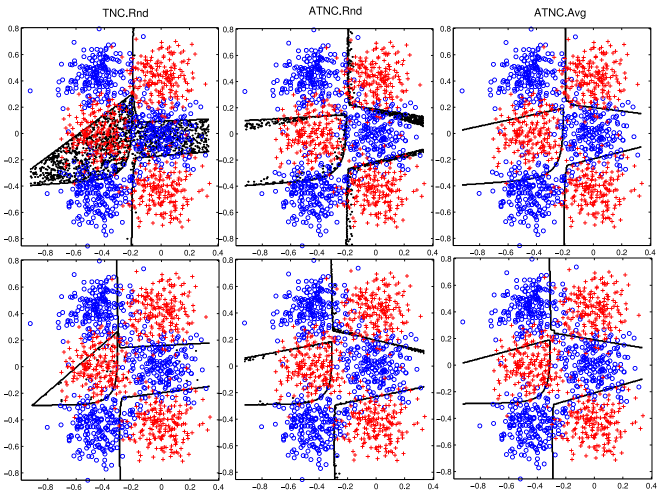


Fig. 4. Piecewise linear separation in the "BMC" dataset after one and two passes (first row: first pass, second row: second pass; first column: TNC.Rnd, second column: ATNC:Rnd, third column: ATNC:Avg). The randomized algorithms are unsure in the black dotted regions near the boundaries.

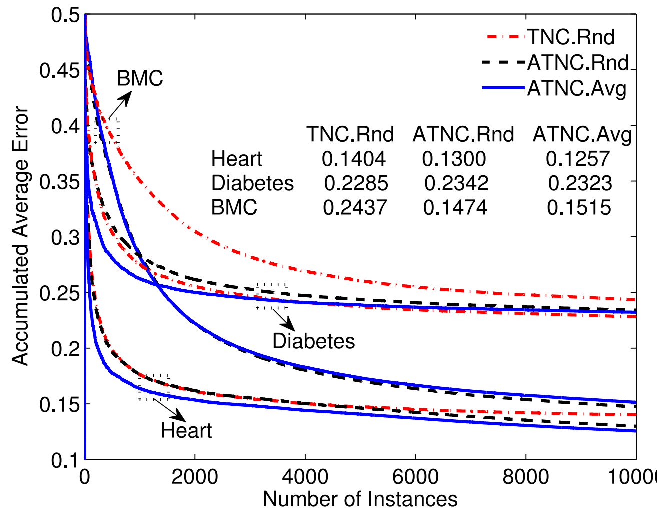
methods essentially fail at the "Banana" and "BMC" datasets, which indicates that other methods are not able to extend to complex nonlinear separations starting from linear weak learners. On the contrary, our method successfully models those complex nonlinear separations with piecewise linear curves (cf. Fig. 4) and therefore, provides a highly superior performance especially on the "Banana" and "BMC" datasets (cf. Table I). On these datasets, the gain due to the self adaptation capabilities (which are devised in this paper) is also clearly demonstrated, cf. ATNC.Rnd and ATNC.Avg vs TNC.Rnd on "Banana" and "BMC". We also observe that averaging incorporates better with our self organizing strategy as ATNC.Avg almost always outperforms ATNC.Rnd. We emphasize that ATNC.Avg has superior performance compared to ATNC.Rnd in the transient state, however, both of them are asymptotically convergent to the same performance level, cf. the comparable performance in the last three datasets of relatively long sequences in Table I. In general, the proposed methods ATNC.Avg and ATNC.Rnd generalize and asymptotically cover the solution of the method TNC.Rnd [13]. However, ATNC.Avg has a significantly better transient characteristics and TNC.Rnd occasionally performs poorly (such as "BMC", "Banana" and "Adult") depending on the complexity of the intrinsic optimal separation in the data. To validate this, we present the long term behavior of these three algorithms on concatenated datasets in Fig. 5, where the performances improve in favor of ATNC.Rnd and ATNC.Avg compared to TNC.Rnd with sufficiently many observations. For instance, ATNC.Rnd significantly outperforms TNC.Rnd on the "Heart" dataset in the long run, although initially performed comparable in Table I (on relatively short sequences). Note that ATNC.Rnd and ATNC.Avg converge to the same performance level but ATNC.Avg has better transient performance.

Remark 3: i) The proposed algorithms (ATNC.Rnd and ATNC.Avg) perform better with data normalized to the range $[-1, 1]$ given in the first row of Table I; however, we continue with the norm truncated data to be aligned with the results in the corresponding studies [12], [14]. *ii)* The small

TABLE I

AVERAGE ERROR RATES AS WELL AS THEIR STANDARD DEVIATIONS (THE DEVIATION IS WITHIN PARENTHESES NEXT TO THE ERROR RATE) ARE PRESENTED ON SEVERAL REAL DATASETS ("BMC" AND "BANANA" ARE THE ONLY SYNTHETIC ONES). THE PROPOSED ALGORITHMS SIGNIFICANTLY OUTPERFORM THE STATE-OF-THE-ART TECHNIQUES. NOTE ALSO THAT IN THE STARRED CASES (8/10 OF APPLICABLE CASES, I.E., "BMC" AND "BANANA" DO NOT APPLY), ATNC.AVG WITH PERCEPTRON OUTPERFORMS THE COMPARED METHODS USED WITH THE NAIVE BAYES ACCORDING TO THE RESULTS REPORTED IN [12]. ALL VALUES ARE PRESENTED IN PERCENTAGE, I.E., ERROR (STD) $\times 10^{-2}$

Data Sets (Size/Dimension)	Perceptron	OZAB	OGB	OSB	OSB.Ocp	TNC.Rnd	ATNC.Rnd	ATNC.Avg
First Row in each set: Our Results with Normalized Data, i.e., each attribute is linearly mapped to $[-1, 1]$								
Second Row in each set: Our Results with Truncated Data, i.e., $\vec{x}_t \leftarrow \frac{\vec{x}_t}{\max(\|\vec{x}_t\|, 1)}$								
Heart	24.66 (1.75)	23.96 (1.74)	23.28 (1.46)	23.63 (1.62)	24.37 (1.59)	21.75 (1.85)	21.86 (1.82)	**20.09*** (1.51)
(270/13)	24.52 (1.81)	24.00 (2.01)	23.14 (1.83)	23.17 (1.74)	23.43 (1.85)	23.95 (2.11)	23.72 (2.16)	**20.83** (1.70)
Breast Cancer	5.77 (0.47)	5.44 (0.53)	5.71 (0.68)	5.23 (0.51)	5.36 (0.51)	4.84 (0.57)	4.86 (0.55)	**4.65*** (0.50)
(683/10)	5.90 (0.50)	5.38 (0.56)	5.33 (0.57)	4.90 (0.50)	5.04 (0.53)	4.99 (0.54)	4.75 (0.61)	**4.58*** (0.53)
Australian	20.82 (1.04)	20.26 (1.10)	19.70 (1.00)	20.01 (1.00)	20.55 (1.10)	15.92 (1.06)	15.77 (0.87)	**14.86*** (0.82)
(693/14)	20.73 (1.04)	20.10 (1.14)	19.31 (0.99)	19.00 (1.05)	19.43 (1.03)	16.95 (0.98)	17.13 (1.22)	**15.39*** (0.81)
Diabetes	32.25 (1.22)	32.43 (1.35)	33.49 (1.44)	31.33 (1.24)	31.55 (1.27)	26.89 (1.07)	27.72 (1.31)	**25.75*** (1.15)
(768/8)	32.40 (1.25)	32.58 (1.15)	33.35 (1.31)	31.17 (1.12)	31.30 (1.17)	28.75 (1.46)	29.33 (1.50)	**27.28** (1.41)
German	32.45 (1.13)	31.86 (1.05)	32.72 (1.07)	31.86 (1.08)	32.21 (1.01)	28.13 (0.98)	27.90 (1.11)	**26.74*** (0.92)
(1000/24)	32.40 (1.29)	32.12 (1.25)	32.41 (1.16)	31.37 (1.15)	31.53 (1.12)	28.61 (0.93)	28.37 (1.13)	**27.15*** (0.84)
BMC	47.09 (1.53)	45.72 (1.54)	46.92 (1.62)	46.37 (1.44)	46.58 (1.54)	25.37 (1.43)	18.33 (1.80)	**17.03** (1.54)
(1200/2)	48.08 (1.40)	48.08 (1.53)	48.69 (1.48)	48.02 (1.41)	48.19 (1.45)	34.51 (1.62)	26.83 (4.57)	**25.07** (5.57)
Splice	33.42 (0.60)	32.59 (0.59)	32.79 (0.66)	32.81 (0.62)	32.93 (0.67)	18.88 (0.60)	18.86 (0.58)	**18.56** (0.53)
(3175/60)	27.28 (0.56)	26.86 (0.63)	26.43 (0.59)	25.67 (0.54)	25.65 (0.55)	21.98 (1.60)	21.11 (1.12)	**20.81** (1.06)
Banana	48.91 (0.63)	47.96 (0.64)	48.00 (0.69)	48.84 (0.70)	48.82 (0.70)	27.98 (0.93)	18.23 (1.80)	**17.60** (1.32)
(5300/2)	49.00 (0.74)	48.07 (0.63)	48.27 (0.66)	48.97 (0.67)	48.93 (0.66)	27.84 (0.90)	19.04 (2.50)	**18.31** (1.43)
Mushrooms	1.74 (0.12)	**0.89** (0.07)	1.80 (0.18)	1.60 (0.25)	1.40 (0.28)	1.04 (0.12)	1.08 (0.15)	1.01 (0.13)
(8124/112)	1.36 (0.08)	0.64 (0.06)	0.75 (0.06)	**0.63** (0.06)	0.65 (0.06)	1.78 (0.30)	1.38 (0.18)	1.29 (0.16)
Adult	20.98 (0.09)	20.79 (0.13)	20.61 (0.17)	20.62 (0.14)	20.56 (0.13)	**15.35** (0.06)	15.40 (0.07)	15.36* (0.07)
(48842/122)	20.89 (0.14)	20.49 (0.14)	20.79 (0.13)	19.86 (0.13)	19.88 (0.13)	22.34 (0.24)	**15.60** (0.19)	15.66* (0.17)
Cod-Rna	35.27 (0.05)	36.41 (0.04)	36.76 (0.05)	34.68 (0.05)	34.58 (0.05)	4.82 (0.03)	**4.81** (0.03)	4.83* (0.03)
(488565/8)	18.99 (0.03)	21.93 (0.05)	18.62 (0.04)	18.26 (0.03)	18.31 (0.03)	**12.63** (0.02)	12.65 (0.04)	12.65 (0.05)
Cover-Type	34.25 (0.08)	34.99 (0.05)	34.96 (0.05)	33.61 (0.05)	33.61 (0.04)	**24.47** (0.03)	24.51 (0.03)	24.50* (0.03)
(581012/54)	34.36 (0.07)	34.54 (0.05)	34.72 (0.05)	33.26 (0.07)	33.27 (0.07)	24.55 (0.05)	24.55 (0.06)	**24.55*** (0.05)



	TNC.Rnd	ATNC.Rnd	ATNC.Avg
Heart	0.1404	0.1300	0.1257
Diabetes	0.2285	0.2342	0.2323
BMC	0.2437	0.1474	0.1515

Fig. 5. Long term behavior over 100 trials based on concatenation of random permutations of datasets: norm-truncated "BMC", "Heart" and "Diabetes".

mismatches between our findings over 100 trials and originally reported error rates in [12] over 5 trials are due to the relatively larger standard deviations in the results of [12] and the randomization (random permutations) of the trials.

Remark 4: In our preliminary experiments, a depth-4 tree has been found to be appropriate for all of our stationary as well as non-stationary performance evaluations. Note that with deeper trees, the proposed algorithms (ATNC.Rnd and ATNC.Avg) gain stronger adaptation to non-linearity, however, at the cost of an increased parameter complexity and an increased demand for more data. We provide here two explanatory examples from our preliminary experiments. We run our algorithm ATNC.Avg on the highly non-linear dataset "BMC" and on the most (among our 14 datasets that we use in our experiments) sparse dataset "Adult" for various depths $D \in \{1, 2, 4, 8\}$ and obtain the averaged accumulated error rates (over 100 trials obtained by random permutations with normalized data) as $\{37.73, 17.37, 17.03, 16.50\} \times 10^{-2}$ for "BMC", respectively; and $\{15.42, 15.40, 15.36, 15.36\} \times 10^{-2}$ for "Adult", respectively. The improvement from 17.37 $(D = 2)$ to 16.50 $(D = 8)$ in the case of "BMC" and the improvement from 15.42 $(D = 1)$ to 15.36 $(D = 8)$ in the case of "Adult" are significant since the standard deviations of these average errors are approximately $\sim 1.5 \times 10^{-2}$ for "BMC", and $\sim 0.07 \times 10^{-2}$ for "Adult". Also note that the linear perceptron classifier (when run solely on the complete space) yields an error rate 47.09 for "BMC" and 20.98 for "Adult", which corresponds to our algorithm run with a depth-0, i.e., $D = 0$, tree (single node). In addition, the effect of sparsity in the case of "Adult" is remarkable: the error rate improves (to 15.06×10^{-2}) by the amount $(15.36 - 15.06)/0.07 = 4.28\times$ standard deviation in the case of $D = 8$, when we lift the sequence length from 48842 to $\sim 500 \times 10^3$ by the concatenations of random permutations. Therefore, based on our preliminary experiments, the choice of a depth-4 tree is sufficient for these as well as the other datasets (after similarly analyzing 14 datasets in total that we use in our performance evaluations) to successfully adapt to their classification complexity. However, the proposed algorithms are certainly capable of straightforwardly adapting to higher degree of non-linearities as well—when a more complex dataset is presented—by further increasing the depth (note that the computational complexity of the proposed algorithm scales linearly with the depth, hence it can be easily used with deeper trees). Such a dataset with higher degree of non-linearity with the binary classification task is readily found in the multi-class classification problems, where, for instance, one usually applies the one-vs-all binary classification successively to converge to a multi-class solution [53]. In fact, our "BMC" dataset is actually a multi-class problem (with 6 separate classes), where our presented binary classification problem on "BMC" is a three-vs-three instance.

B. *Non-Stationary Data: Concept Change/Drift*

In this part, we study the proposed algorithms (ATNC.Rnd and ATNC.Avg) with non-stationary data, where there might be continuous or sudden/abrupt changes in the source statistics, i.e., concept change. Since our algorithms process only one instance at a time without storing it, we choose the *Dynamically Weighted Majority Algorithm*—"DWM" [15] with perceptron or naive bayes experts for the comparison, which is also a truly online algorithm without storage or batch processing requirements. Hence, we obtain two algorithms: "DWM-P" and "DWM-N". Most of the other algorithms [16], [29], [41]–[44] specialized on concept drift have sliding window approaches. In these approaches, the window size must be chosen large enough to fully capture the statistics of the active concept; and if it is too large, then the desired adaptation to the concept change quickly degrades at the risk of wasting the computational resources spent on the window processing. Clearly, there is no optimal window size, which introduces another parameter that has to be tuned for each experiment, i.e., window size $= ws \in \{100, 200, 500, \ldots, 2000\}$ in [16]. We also emphasize that such sliding window approaches are essentially batch algorithms, i.e., not truly online, and—in general—ws times slower than the truly online counterparts such as DWM or ATNC.Avg. Therefore, such approaches do not truly fit into our framework. Nevertheless, we devise an online version of the batch classifier [9] using the sliding window approach (which also learns the space partitioning and the classifier using the coordinate ascent approach) and name it as *Sliding Window based Local Space Partitioning*—"WLSP". For the method *Dynamically Weighted Majority Algorithm*—"DWM" which allows the addition and removal of experts during the stream, we set the initial number of experts to 1, where the maximum number of experts is bounded by 100. For the method *Sliding Window based Local Space Partitioning*—"WLSP" [9], we provide the algorithm with the most recent $ws = 100$ instances at each time. The parameters for these compared methods are set as in their original proposals.

We run these methods on the "BMC" dataset (1200 instances, Fig. 4), where a sudden/abrupt concept change is obtained such that the instances are rotated (clock-wise around the origin) 180° after the 600th instance. This effectively means a label flip and the resulting dataset is denoted as "BMC-F". For a continuous concept drift, we rotate each instance 180°/1200 starting from the beginning; and the resulting dataset is denoted as "BMC-C". In Fig. 6, we present the error plots for the compared methods over 1000 trials. At each 10th instance, we test the algorithms with 1200 instances drawn from the active set of statistics (active concept).

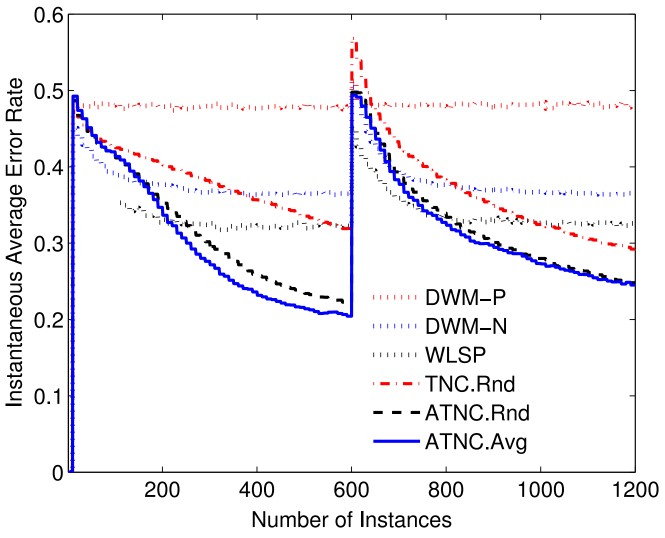


Fig. 6. Performance of the compared methods in case of the abrupt concept change in the "BMC-F" dataset. At the 600th instance, there is a 180° clock-wise rotation around the origin (derived from the "BMC dataset") that is effectively a label flip. In the first 100 instances, the sliding window based approach WLSP does not produce results.

Note that since the "BMC" data is strongly Non-Gaussian with strongly non-linear class separations, the method DWM with perceptron or naive bayes do not perform well on "BMC-F". For instance, DWM-P operates with an error rate fluctuating around 0.48–0.49 (random guess). This results since the performance of the method DWM is directly dependent on the expert success and observe that, both base learners (perceptron or the naive bayes) fail due to the high separation complexity in "BMC-F". On the other hand, the method WLSP quickly converges to its steady state, however, it is also asymptotically outperformed by our methods at both concepts with sufficient number of observations. Increasing the window size is clearly expected to boost the performance of WLSP, though, at the cost of increased computational complexity. It is already significantly slower than our techniques with even $ws = 100$, cf. Section IV-C. When the method WLSP is run on the "BMC-C" dataset in case of the continuous concept drift, cf. Fig. 7, its performance significantly degrades (compared to the one on "BMC-F") since, in this case, WLSP is trained with the batch data of a continuous mixture of concepts in the sliding windows. Under this continuous concept drift, ATNC always—not only asymptotically as in the case of "BMC-F"—performs better than WLSP. Hence, the sliding window approach is sensitive to the continuous drift. Our discussion about the DWM method on the concept change data "BMC-F" remains valid in the case of the concept drift of "BMC-C". In these experiments, the power of the proposed self-organizing strategy is obvious as ATNC (both .Rnd and .Avg) almost always outperforms TNC.Rnd [13].

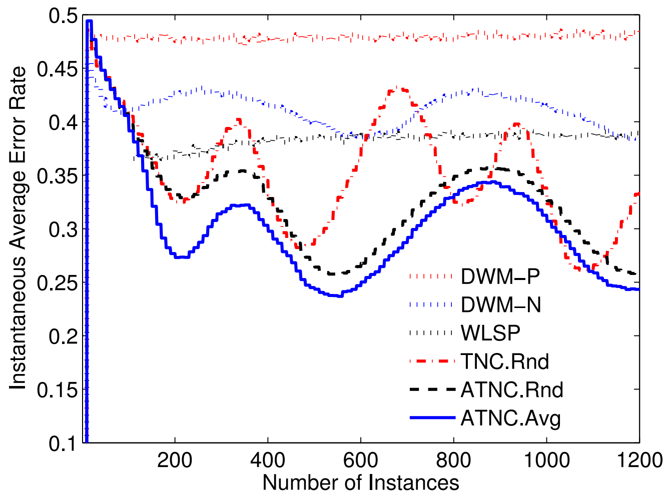


Fig. 7. Performance of the compared methods in case of the continuous concept change in the "BMC-C" dataset. At each instance, there is a 180°/1200 clock-wise rotation around the origin (derived from the "BMC dataset"). In the first 100 instances, the sliding window based approach WLSP does not produce results.

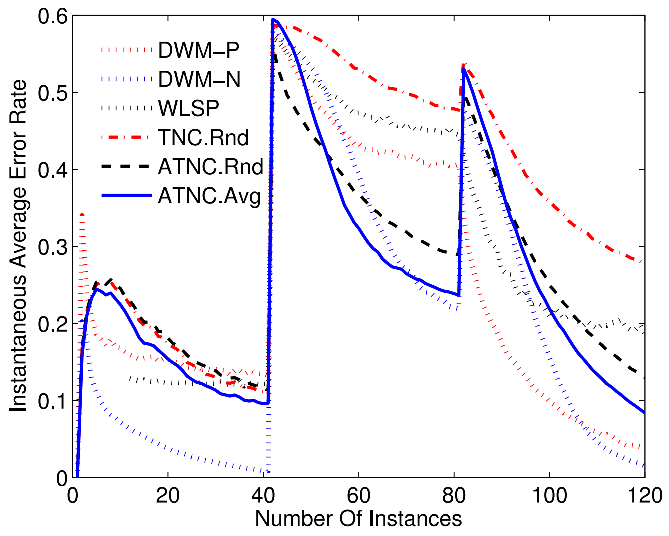


Fig. 8. Performance of the compared methods in case of the stagger concepts.

In the rest of the experiments involving concept changes, we follow the comparison framework of [15]. We conduct tests on the stagger concepts [15], where the data $\{1, 2, 3\}^{120}$ includes 2 concept switches among three concepts $1 \rightarrow 2 \rightarrow 3$ at the 41th and 81th instances. The concept definitions are as follows. Concept 1: $y = 1$, if $\mathbf{x}(1) = 3 \wedge \mathbf{x}(3) = 1$, Concept 2: $y = 1$, if $\mathbf{x}(1) = 1 \vee \mathbf{x}(2) = 2$ and Concept 3: $y = 1$, if $\mathbf{x}(3) = 2 \vee \mathbf{x}(3) = 3$ (otherwise, $y = -1$). In this case, we use the window length $ws = 10$ for the method WLSP. In Fig. 8, we present the error curves for the compared methods. Although the method DWM-P and DWM-N do not perform well on the "BMC-F" and "BMC-C" datasets, DWM-N and our method ATNC.Avg perform comparably on the second and third concepts, whereas DWM-N performs better on the first one. The method WLSP is outperformed by all other techniques. We observe that DWM-P is not able to adapt to the second concept (nonlinear class separation), whereas it outperforms DWM-N on the third concept (linear class separation): it is difficult to choose the right expert to be used with DWM. Note that the naive bayes expert can (limitedly) adapt to nonlinear separations at the cost of a slower convergence in the case of linear separations (compared to the percetpron), whereas the perceptron cannot adapt these nonlinear separations. On the other hand, the proposed methods ATNC.Avg and ATNC.Rnd can adapt to arbitrarily nonlinear separations (cf. Figs. 6 and 7) without sacrificing much from the transient state accuracy. In particular, the proposed methods either outperform (on the second concept) all the compared algorithms or perform comparably with the best competitor (on the first and third concept) in the steady state, cf. the long term results presented in Fig. 9.

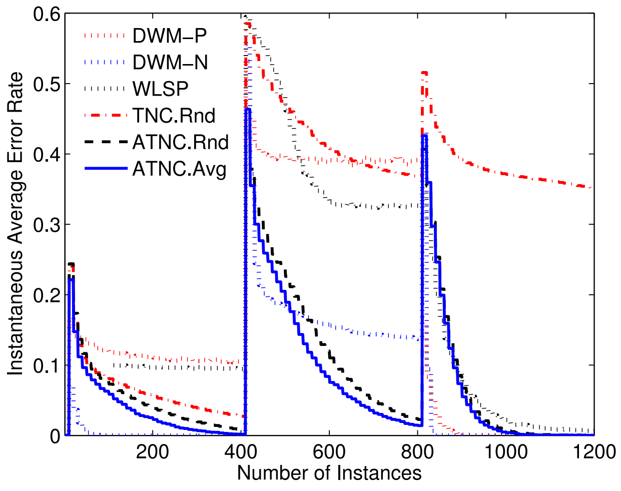


Fig. 9. Long term performance in case of the stagger concepts.

We finally run all algorithms on a larger concept change problem, where we use the drifting hyperplane (DH) dataset [15], which consists of 2000 instances of dimension 10, i.e., $\mathbf{x} \in [0, 1]^{10}$. The dataset includes 3 concept changes among 4 concepts: $1 \rightarrow 2 \rightarrow 3 \rightarrow 4$, where the concept change from i to $i+1$ happens at the $(500i + 1)$th instance during the stream. Concept definitions are as follows: If the concept i is active, then $y = 1$, if $\mathbf{x}(j) + \mathbf{x}(j+1) + \mathbf{x}(j+2) > 0.5$ for $(i, j) \in \{(1, 1), (2, 2), (3, 4), (4, 7)\}$ ($y = -1$, otherwise). On this dataset, the proposed algorithms perform as the second best with comparable performance to the best performing algorithm (DWM-N) on the second concept, Fig. 10.

We conclude that the DWM algorithm is significantly sensitive to the expert choice and its performance is upper-bounded by the chosen expert success. When the—within concept—target separations are relatively simple and linear, DWM-P demonstrates a quick concept adaptation; when the target separations are strongly non-linear, both the methods DWM-P and DWM-N do not perform well, e.g., BMC-F or BMC-C. Choosing more sophisticated experts is always a good option, though, at the cost of increased computational load, cf. IV-C. On the contrary, the proposed algorithms ATNC.Avg and ATNC.Rnd are able to learn the arbitrarily complex separations both in the stationary and non-stationary settings without mandating to choose the right local simple model/expert (as opposed to the method

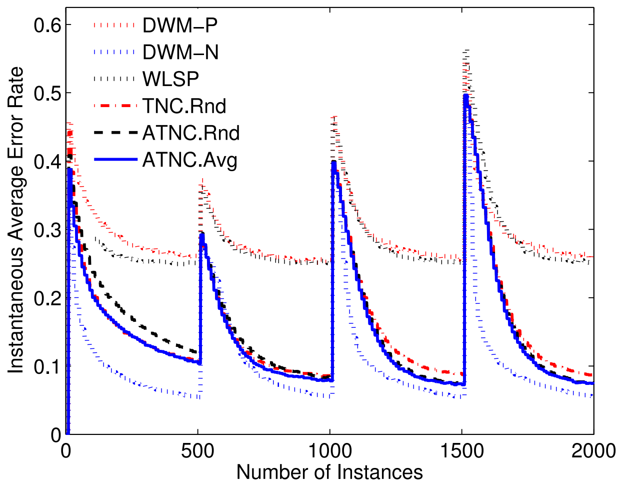


Fig. 10. Performance of the compared methods on the drifting hyper plane.

TABLE II
COMPUTATIONAL RUNNING TIMES (IN SECONDS) OF THE COMPARED METHODS WHEN PROCESSING THE "BMC" DATASET OF 1200 INSTANCES

Perceptron	OSB	OSB.Ocp	OGB	OZAB	DWM-P
0.06	3.91	8.01	3.57	12.90	2.06
	DWM-N	WLSP	ATNC.Rnd	TNC.Rnd	ATNC.Avg
	6.91	68.40	0.70	0.43	0.62

DWM) while outperforming the state-of-the-art techniques in the stationary setting with strong adaptation capabilities under the concept change/drift.

C. Computational Complexity and Running Times

We present the running times of the compared methods on the dataset "BMC" (processing 1200 instances over 1 trial) for the optimized MATLAB implementations run on a daily-use machine (Intel(R) Core(TM) i5-3317U CPU @ 1.70 GHz with 4 GB memory) in Table II. The presented times for our algorithms should roughly scale with $\frac{T}{1200}$ for any other dataset (in our experiments) of size T since we uniformly use a depth-4 tree in our experiments and the contribution of the dimensionality to the computational complexity of our algorithms is negligible (in these data sets, where the largest dimensionality is 122) compared to our tree operations. Hence, these running times can only realize in favor of the proposed algorithms when the dimension increases in case of the other datasets. The proposed methods are computationally the most efficient (except the base learner perceptron) among the competitors, e.g., ATNC.Avg $\sim$3.32$\times$ faster than DWM-P. In particular, the method DWM becomes significantly computationally demanding when used with a more sophisticated expert, e.g., DWM-N is $\sim$3.35$\times$ slower than DWM-P.

We emphasize that our online algorithms require $O(DpT + DcT)$ computational complexity ($O\left(Dp + Dc\right)$ per instance) and $O\left(2^D p\right)$ storage complexity for processing T data instances, where D is the tree depth and c is a constant accounting for the tree operations (p is negligible compared to c in our experiments). For this reason, we compare our algorithms only with the truly online methods of similar type in terms of the computational and storage complexity. In particular, we follow the comparison framework of [12] in the stationary part, and the comparison framework of [15] in the non-stationary part, i.e., we compare our algorithms with the compared techniques in those frameworks based on the datasets used in those frameworks.

On the other hand, the kernel machines such as [54]–[57] are extensively studied powerful techniques providing impressive performance results in modeling the complex non-linear data structures, however, they are not comparable to the proposed algorithms in this paper due to their heavy computational and storage demands. For instance, Support Vector Machines (SVM) have—as a batch algorithm—in general $O\left(T^3\right)$ (computational) complexity in the training and $O\left(T\right)$ in the test [54], which definitely cannot be used in our online large scale data processing framework. The solution of SVM is asymptotically obtained by an online algorithm "LASVM" in [55], which is relatively more efficient (since early stopping is possible due to sequential processing), however, with computational complexity at least $O\left(T^2\right)$ in the best scenario, and still $O\left(T^3\right)$ in general [55]. The online kernel method in [56] (which is not an SVM) requires one to save at least a window (of size ws) of the past stream in order to exploit the "reproducing" property, which is therefore not truly online and requires in general $O\left(T^2\right)$ or at least $O\left(wsT\right)$ (where the choice of the optimal window length ws is another issue) computations. We note that the kernel machines generally require (at each time) the pair-wise kernel evaluations with respect to the past data to fully exploit the "reproducing property" resulting $O\left(T^2\right)$ computational complexity; and this complexity further increases to $O\left(T^3\right)$ if one desires to preserve the SVM optimality. Remarkably, the study [57] does not enforce a strict SVM optimality but approximates it in a controllable manner to finitely bound the number of support vectors (which unboundedly increases in other studies such as [55]) and achieves a complexity $O\left(T^2\right)$ in an online manner with an approximate SVM optimality. Despite these efficient implementations, the state-of-the-art kernel machines do not fairly fit into our truly online framework since even the best achieved complexity $O\left(T^2\right)$ (in this line of kernel research) is computationally prohibitive in our consideration. In this study, we mainly target at processing large scale data at fast rates continuously without an interrupt up to infinity while achieving strong modeling capabilities for highly non-linear complex data structures even under strong non-stationarity.

To validate this discussion, we also and finally compare the proposed algorithm ATNC.Avg with the method OISVM[6] in [57] in terms of the classification error rate and computational running time and present our results (averaged over Monte Carlo trials) on certain explanatory examples in Table III. We observe that the proposed algorithm ATNC.Avg and OISVM perform similarly on the datasets "Australian" and "German", whereas OISVM is superior on the highly non-linear datasets "BMC" and "Splice". This is due to the (approximate) SVM optimality of OISVM, whereas the proposed algorithm ATNC.Avg optimizes (minimizes) its accumulated zero-one loss locally, i.e.,

[6]We use the MATLAB implementation [58] for Online Independent Support Vector Machines (OISVM) provided by the authors of [57], where we use the RBF kernel and optimize the parameters, i.e., the bandwidth g of the kernel and the cost C of the SVM, for each dataset separately. Other parameters are set as in [57].

TABLE III
COMPARISON OF ATNC.AVG AND OISVM [57] BASED ON NORMALIZED DATA. WE REPORT THE AVERAGE ERROR RATES AS WELL AS THEIR STANDARD DEVIATIONS (THE DEVIATION IS WITHIN PARENTHESES NEXT TO THE ERROR RATE) IN THE FIRST ROW FOR EACH ALGORITHM; AND THE COMPUTATIONAL RUNNING TIMES IN THE SECOND ROW. ERROR RATES AND STANDARD DEVIATIONS ARE PRESENTED IN PERCENTAGE, I.E., ERROR (STD) $\times 10^{-2}$; THE RUNNING TIMES ARE PRESENTED IN SECONDS

Datasets T/p	BMC 1200/2	Austrailan 693/14	German 1000/24	Splice 3175/60
ATNC.Avg	17.03 (1.54)	14.86 (0.82)	26.74 (0.92)	18.56 (0.53)
	0.62 s	0.36 s	0.60 s	2.43 s
OISVM	10.88 (0.37)	14.68 (0.43)	25.97 (0.72)	12.05 (0.23)
	1.25 s	0.95 s	15.47 s	788 s

not globally unlike OISVM, due its stochastic gradient based updates. However, this local optimality allows the proposed algorithm to operate computationally highly efficiently at a rate $O(T)$ (within a few seconds in these data sets), whereas OISVM rapidly becomes impractical at a rate $O\left(T_{sv}T^2\right)$ as the data size and dimensionality increases (where T_{sv} is the number of support vectors which is asymptotically finite), cf. running times of OISVM for the data sets "German" or "Splice". Our conclusion is that OISVM and in general kernel machines are appropriate only in the case of humble datasets that are small in size and dimension. On the contrary, in the case of large scale and complex (such as the high non-linearity) data applications under possibly non-stationary source statistics, which is the main topic in this paper, the proposed algorithm is superior due its computationally efficient, adaptive and high performance design.

V. CONCLUSION

We propose a highly efficient (in terms of the computational complexity) online algorithm for supervised learning that is appropriate for the applications requiring sequential data processing at large scales/high rates. The introduced approach combines simple linear classifiers specialized in local regions of the feature space to generate a decision such that both the feature space partitioning and the corresponding local linear models are jointly optimized. The result is a highly dynamical decision tree structure that adaptively organizes itself to minimize the accumulated loss over time for any given data stream regardless of the underlying stationary or non-stationary statistics. We provide strong theoretical performance guarantees without any statistical assumptions on the underlying data. We present a comprehensive experimental comparison of our algorithm with respect to the state-of-the-art techniques from the literature of ensemble methods and classification trees on the commonly used benchmark datasets. We experimentally show that our algorithm significantly outperforms the state-of-the-art techniques with remarkable adaptation capabilities to non-stationarity, i.e., concept change/drift.

We point out that since we employ stochastic gradient based updates, the learning of the feature space partitioning is convergent to a locally optimal solution and therefore, this creates sensitivity to the initial conditions in our framework. Although we draw conclusions about the worst case scenarios in this paper, i.e., our performance results hold for every possible input stream without any statistical assumptions about the data source, we consider finding a globally optimal, i.e., not only locally, feature space partitioning by reasonably assuming certain statistical conditions about the data source and convexly defining/relaxing loss functions as a future research direction. Secondly, the proposed algorithms can straightforwardly adapt to arbitrarily high degree of non-linearities with a sufficiently deep partitioning tree. This makes the proposed technique appealing especially for multi-class problems since, in such problems, one usually applies the one-vs-all binary classification successively to converge to a multi-class solution, where the one-vs-all (or one-vs-one) problem is in general highly non-linear. We also consider obtaining this extension to multi-class while maintaining computational efficiency as a future study.

REFERENCES

[1] J. Ziniel, P. Schniter, and P. Sederberg, "Binary linear classification and feature selection via generalized approximate message passing," *IEEE Trans. Signal Process.*, vol. 63, no. 8, pp. 2020–2032, Apr. 2015.

[2] K. Slavakis, S. Theodoridis, and I. Yamada, "Online kernel-based classification using adaptive projection algorithms," *IEEE Trans. Signal Process.*, vol. 56, no. 7, pp. 2781–2796, Jul. 2008.

[3] F. Desobry, M. Davy, and C. Doncarli, "An online kernel change detection algorithm," *IEEE Trans. Signal Process.*, vol. 53, no. 8, pp. 2961–2974, Aug. 2005.

[4] J. Kivinen, A. Smola, and R. Williamson, "Online learning with kernels," *IEEE Trans. Signal Process.*, vol. 52, no. 8, pp. 2165–2176, Aug. 2004.

[5] N. Cesa-Bianchi and G. Lugosi, *Prediction, Learning, and Games*. Cambridge, U.K.: Cambridge Univ. Press, vol. 1, no. 1.1, 2006.

[6] C. Monteleoni and T. Jaakkola, "Online learning of non-stationary sequences," in *Adv. Neural Inf. Process. Syst.*, 2003.

[7] 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, 2007.

[8] H. Ozkan, O. Pelvan, and S. Kozat, "Data imputation through the identification of local anomalies," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 10, pp. 2381–2395, Oct. 2015.

[9] J. Wang and V. Saligrama, "Local supervised learning through space partitioning," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2012, pp. 91–99.

[10] N. C. Oza and S. Russell, "Online bagging and boosting," in *Proc. Artif. Intell. Statist.*, 2001, pp. 105–112.

[11] C. Leistner, A. Saffari, P. M. Roth, and H. Bischof, "On robustness of on-line boosting—A competitive study," in *Proc. IEEE 12th Int. Conf. Comput. Vis. Workshops*, 2009, pp. 1362–1369.

[12] S.-T. Chen, H.-T. Lin, and C.-J. Lu, "An online boosting algorithm with theoretical justifications," in *Proc. Int. Conf. Mach. Learn.*, 2012.

[13] H. Ozkan, M. A. Donmez, O. S. Pelvan, A. Akman, and S. S. Kozat, "Competitive and online piecewise linear classification," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, May 2013, pp. 3452–3456.

[14] Q. Bai H. Lam, and S. Sclaroff, "A Bayesian framework for online classifier ensemble," in *Proc. Int. Conf. Mach. Learn., JMLR Workshop and Conference Proceedings*, 2014, pp. 1584–1592.

[15] J. Z. Kolter and M. Maloof, "Dynamic weighted majority—An ensemble method for drifting concepts," *J. Mach. Learn. Res.*, vol. 8, pp. 2755–2790, 2007.

[16] M. Grbovic and S. Vucetic, "Tracking concept change with incremental boosting by minimization of the evolving exponential loss," *Machine Learning and Knowledge Discovery in Databases*. New York, NY, USA: Springer, 2011, pp. 516–532.

[17] W.-Y. Loh, "Classification and regression trees," *Wiley Interdisciplinary Rev., Data Min. Knowl. Discov.*, vol. 1, no. 1, pp. 14–23, 2011.

[18] L. Brieman, J. Friedman, C. J. Stone, and R. A. Olsen, *Classification and Regression Trees*. London, U.K.: Chapman & Hall, 1984.

[19] C. Scott and R. D. Nowak, "Minimax-optimal classification with dyadic decision trees," *IEEE Trans. Inf. Theory*, vol. 52, no. 4, pp. 1335–1353, 2006.

[20] C. Strobl, A.-L. Boulesteix, and T. Augustin, "Unbiased split selection for classification trees based on the gini index," *Comput. Statist. Data Anal.*, vol. 52, no. 1, pp. 483–501, 2007.

[21] J. Gama, "Functional trees," *Mach. Learn.*, vol. 55, no. 3, pp. 219–250, 2004.

[22] E. Takimoto, A. Maruoka, and V. Vovk, “Predicting nearly as well as the best pruning of a decision tree through dyanamic programming scheme,” *Theoretic. Comput. Sci.*, vol. 261, pp. 179–209, 2001.
[23] Y. Yilmaz and S. S. Kozat, “Competitive randomized nonlinear prediction under additive noise,” *IEEE Signal Process. Lett.*, vol. 17, no. 4, pp. 335–339, 2010.
[24] V. G. Vovk, “Aggregating strategies,” in *Proc. 3rd Workshop Comput. Learn. Theory*, 1990, pp. 371–383.
[25] N. D. Vanli and S. S. Kozat, “A comprehensive approach to universal nonlinear regression based on trees,” *IEEE Trans. Signal Process.*, vol. 62, no. 20, pp. 5471–5486, 2013.
[26] S. Dasgupta and Y. Freund, “Random projection trees for vector quantization,” *IEEE Trans. Inf. Theory*, vol. 55, no. 7, pp. 3229–3242, 2009.
[27] Y. Freund and R. E. Schapire, “A decision-theoretic generalization of on-line learning and an application to boosting,” *J. Comput. Syst. Sci.*, vol. 55, no. 1, pp. 119–139, 1997.
[28] I. Mukherjee, C. Rudin, and R. E. Schapire, “The rate of convergence of adaboost,” *J. Mach. Learn. Res.*, vol. 14, no. 1, pp. 2315–2347, 2013.
[29] R. Pelossof, M. Jones, I. Vovsha, and C. Rudin, “Online coordinate boosting,” in *Proc. IEEE Int. Conf. Comput. Vis. Workshops*, 2009, pp. 1354–1361.
[30] X. Liu and T. Yu, “Gradient feature selection for online boosting,” in *Proc. IEEE Int. Conf. Comput. Vis.*, 2007, pp. 1–8.
[31] H. Grabner and H. Bischof, “On-line boosting and vision,” in *Proc. Comput. Vis. Pattern Recognit.*, 2006, vol. 1, pp. 260–267.
[32] B. Babenko, M.-H. Yang, and S. Belongie, “Robust object tracking with online multiple instance learning,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 8, pp. 1619–1632, Aug. 2011.
[33] A. Saffari, M. Godec, T. Pock, C. Leistner, and H. Bischof, “Online multiclass lpboost,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2010, pp. 3570–3577.
[34] B. Babenko, M.-H. Yang, and S. Belongie, “A family of online boosting algorithms,” in *Proc. IEEE Int. Conf. Comput. Vis. Workshops*, 2009, pp. 1346–1353.
[35] N. Littlestone and M. K. Warmuth, “The weighted majority algorithm,” *Inf. Comput.*, vol. 108, no. 2, pp. 212–261, 1994.
[36] M. Herbster and M. K. Warmuth, “Tracking the best expert,” *Mach. Learn.*, vol. 32, no. 2, pp. 151–178, 1998.
[37] A. Gyorgy, T. Linder, and G. Lugosi, “Efficient tracking of large classes of experts,” *IEEE Trans. Inf. Theory*, vol. 58, no. 11, pp. 6709–6725, 2012.
[38] A. Blum, “Empirical support for winnow and weighted-majority algorithms: Results on a calendar scheduling domain,” *Mach. Learn.*, vol. 26, no. 1, pp. 5–23, 1997.
[39] O. Bousquet and M. K. Warmuth, “Tracking a small set of experts by mixing past posteriors,” *J. Mach. Learn. Res.*, vol. 3, pp. 363–396, 2003.
[40] L. L. Minku, A. P. White, and X. Yao, “The impact of diversity on online ensemble learning in the presence of concept drift,” *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 5, pp. 730–742, 2010.
[41] H. Wang, W. Fan, P. S. Yu, and J. Han, “Mining concept-drifting data streams using ensemble classifiers,” in *Proc. Int. Conf. Knowl. Discov. Data Min.*, 2003, pp. 226–235.
[42] W. N. Street and Y. Kim, “A streaming ensemble algorithm (sea) for large-scale classification,” in *Proc. ACM SIGKDD Conf. Knowl. Discov. Data Min.*, 2001, pp. 377–382.
[43] F. Chu and C. Zaniolo, “Fast and light boosting for adaptive mining of data streams,” *Advances in Knowledge Discovery and Data Mining*. New York, NY, USA: Springer, 2004, pp. 282–292.
[44] A. Bifet, G. Holmes, B. Pfahringer, R. Kirkby, and R. Gavaldà, “New ensemble methods for evolving data streams,” in *Proc. ACM SIGKDD Conf. Knowl. Discov. Data Min.*, 2009, pp. 139–148.
[45] A. Pocock, P. Yiapanis, J. Singer, M. Luján, and G. Brown, “Online non-stationary boosting,” *Multiple Classifier Systems*. New York, NY, USA: Springer, 2010, pp. 205–214.
[46] V. Attar, P. Sinha, and K. Wankhade, “A fast and light classifier for data streams,” *Evolv. Syst.*, vol. 1, no. 3, pp. 199–207, 2010.
[47] R. A. Servedio, “Smooth boosting and learning with malicious noise,” *J. Mach. Learn. Res.*, vol. 4, pp. 633–648, 2003.
[48] A. V. Aho and N. J. A. Sloane, “Some doubly exponential sequences,” *Fibonacci Quart.*, vol. 11, pp. 429–437, 1970.
[49] D. P. Helmbold and R. E. Schapire, “Predicting nearly as well as the best pruning of a decision tree,” *Mach. Learn.*, vol. 27, no. 1, pp. 51–68, 1997.
[50] F. M. J. Willems, Y. M. Shtarkov, and T. J. Tjalkens, “The context-tree weighting method: Basic properties,” *IEEE Trans. Inf. Theory*, vol. 41, no. 3, pp. 653–664, May 1995.
[51] W. Hoeffding, “Probability inequalities for sums of bounded random variables,” *J. Amer. Statist. Assoc.*, vol. 58, no. 301, pp. 13–30, Mar. 1963.
[52] Y. Freund and R. E. Schapire, “Large margin classification using the perceptron algorithm,” *Mach. Learn.*, vol. 37, no. 3, pp. 277–296, 1999.
[53] S. Escalera, D. Tax, O. Pujol, P. Radeva, and R. Duin, “Subclass problem-dependent design for error-correcting output codes,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 6, pp. 1041–1054, Jun. 2008.
[54] O. Chapelle, “Training a support vector machine in the primal,” *Neural Comput.*, vol. 19, no. 5, pp. 1155–1178, 2007.
[55] A. Bordes, S. Ertekin, J. Weston, and L. Bottou, “Fast kernel classifiers with online and active learning,” *J. Mach. Learn. Res.*, vol. 6, pp. 1579–1619, 2005.
[56] L. C. S. Schuurmans and S. Caelli, “Implicit online learning with Kernels,” *Adv. Neural Inf. Process. Syst.*, vol. 19, pp. 249, 2007.
[57] F. Orabona, C. Castellini, B. Caputo, L. Jie, and G. Sandini, “On-line independent support vector machines,” *Pattern Recognit.*, vol. 43, no. 4, pp. 1402–1412, 2010.
[58] F. Orabona, *DOGMA: A MATLAB Toolbox for Online Learning*, 2009 [Online] Available: http://dogma.sourceforge.net

Huseyin Ozkan is currently with the Department of Brain and Cognitive Sciences at the Massachusetts Institute of Technology (MIT), Cambridge, MA, USA, as a postdoctoral scholar. He received his B.Sc. degrees in electrical and electronics engineering, and mathematics from Bogazici University, Istanbul, Turkey, in 2007; his M.Sc. degree in electrical engineering from Boston University, MA, USA, in 2010; and his Ph.D. degree in electrical and electronics engineering from Bilkent University, Ankara, Turkey in 2015. Before joining CSAIL at MIT, he worked as a research scientist at Aselsan Inc., Ankara, Turkey, where he conducted computer vision research for large are surveillance and also focused on anomaly detection and recommendation problems. He also worked as a research intern at Mitsubishi Electric Research Laboratories, Cambridge, MA, USA, where he developed efficient algorithms for vision based road sign detection.

He has been awarded the best paper award by the IEEE conference on Advanced Video and Signal based Surveillance (2011) and the best student paper award by the IEEE conference on Signal Processing Applications (2012). His research interests include statistical learning, pattern recognition, computer vision and statistical signal processing.

N. Denizcan Vanli received the B.S. and M.S. degrees in electrical and electronics engineering from Bilkent University, Ankara, Turkey, in 2013 and 2015, respectively. He is currently working toward the Ph.D. degree in electrical engineering and computer science at Massachusetts Institute of Technology (MIT), Cambridge. His research interests include convex optimization, online learning, and distributed optimization.

Suleyman S. Kozat (SM'12) received his B.S. degree in electrical and electronics engineering from Bilkent University, Ankara, Turkey. He received the M.S. and Ph.D. degrees in electrical engineering from University of Illinois at Urbana Champaign, IL, USA, in 2001 and 2004, respectively. After graduation, Dr. Kozat joined IBM Research, T. J. Watson Research Center, Yorktown, NY, USA, as a Research Staff Member in Pervasive Speech Technologies Group, where he focused on problems related to statistical signal processing and machine learning. He also worked as a Research Associate at Microsoft Research, Redmond, WA, USA, in Cryptography and Anti-Piracy Group. Currently, Dr. Kozat is an Associate Professor at the Electrical and Electronics Engineering Department, Bilkent University, Turkey. His research interests include intelligent systems, adaptive filtering for smart data analytics, online learning and machine learning algorithms for signal processing. Dr. Kozat has been awarded IBM Faculty Award by IBM Research in 2011, Outstanding Faculty Award by Koc University in 2011, Outstanding Young Researcher Award by the Turkish National Academy of Sciences in 2010, ODTU Prof. Dr. Mustafa N. Parlar Research Encouragement Award in 2011 and holds Career Award by the Scientific Research Council of Turkey, 2009.