Digital Signal Processing 81 (2018) 61–78

Contents lists available at ScienceDirect

Digital Signal Processing

www.elsevier.com/locate/dsp

Boosted adaptive filters

Dariush Kari [a,*], Ali H. Mirza [c], Farhan Khan [c], Huseyin Ozkan [b], Suleyman S. Kozat [c]

[a] *Coordinated Science Laboratory, University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA*
[b] *Faculty of Engineering and Natural Sciences, Sabanci University, Istanbul 34956, Turkey*
[c] *Department of Electrical and Electronics Engineering, Bilkent University, Ankara 06800, Turkey*

ARTICLE INFO

Article history:
Available online 25 July 2018

Keywords:
Online boosting
Adaptive filtering
Boosted filter
Ensemble learning
Smooth boost
Mixture methods

ABSTRACT

We introduce the boosting notion of machine learning to the adaptive signal processing literature. In our framework, we have several adaptive filtering algorithms, i.e., the weak learners, that run in parallel on a common task such as equalization, classification, regression or filtering. We specifically provide theoretical bounds for the performance improvement of our proposed algorithms over the conventional adaptive filtering methods under some widely used statistical assumptions. We demonstrate an intrinsic relationship, in terms of boosting, between the adaptive mixture-of-experts and data reuse algorithms. Additionally, we introduce a boosting algorithm based on random updates that is significantly faster than the conventional boosting methods and other variants of our proposed algorithms while achieving an enhanced performance gain. Hence, the random updates method is specifically applicable to the fast and high dimensional streaming data. Specifically, we investigate Recursive Least Square-based and Least Mean Square-based linear and piecewise-linear regression algorithms in a mixture-of-experts setting and provide several variants of these well-known adaptation methods. Furthermore, we provide theoretical bounds for the computational complexity of our proposed algorithms. We demonstrate substantial performance gains in terms of mean squared error over the base learners through an extensive set of benchmark real data sets and simulated examples.

1. Introduction

Boosting is considered as one of the most important ensemble learning methods in the machine learning literature and it is extensively used in several different real-life applications from classification to regression [1,2]. As an ensemble learning method [3,4], boosting combines several parallel running "weakly" performing algorithms to build a final "strongly" performing algorithm [2,5]. This is accomplished by finding a linear combination of weak learning algorithms in order to minimize the total loss over a set of training data commonly using a functional gradient descent [6,7]. Boosting is successfully applied to several different problems in the machine learning literature including classification [7], regression [6,8], and prediction [9,10]. However, significantly less attention is given to the idea of boosting in adaptive signal processing [11,12] framework. To this end, our goal is (a) to introduce a new boosting approach for adaptive filtering, (b) derive several different adaptive filtering algorithms based on the boosting approach, (c) provide mathematical guarantees for the performance improvements of our algorithms, and (d) demonstrate the intrinsic connections of boosting with the adaptive mixture-of-experts algorithms [13,14] and data reuse algorithms [15]. Even though in [16] and [17], boosting has been used in limited scenarios of adaptive filtering, this paper provides mathematical guarantees along with an extensive set of experiments to illustrate the details of boosting approach in different adaptive filtering methods.

Although boosting is initially introduced in the batch setting [7], where algorithms boost themselves over a fixed set of training data, it is later extended to the online setting [18]. In the online setting, however, we neither need nor have access to a fixed set of training data, since the data samples arrive one by one as a stream [3,19]. Each newly arriving data sample is processed and then discarded without any storing. The online setting is naturally motivated by many real-life applications especially for the ones involving big data, where there may not be enough storage space available or the constraints of the problem require instant processing [20].

For adaptive filtering purposes [12], the online setting is especially important, where the sequentially arriving data is used to adjust the internal parameters of the filter, either to dynami-

* Corresponding author.
E-mail addresses: dkari2@illinois.edu (D. Kari), mirza@ee.bilkent.edu.tr (A.H. Mirza), khan@ee.bilkent.edu.tr (F. Khan), hozkan@sabanciuniv.edu (H. Ozkan), kozat@ee.bilkent.edu.tr (S.S. Kozat).

https://doi.org/10.1016/j.dsp.2018.07.012
1051-2004/© 2018 Elsevier Inc. All rights reserved.

cally learn the underlying model or to track the nonstationary data statistics [13,21].

Specifically, we have m parallel running constituent filters (CF) [2] that receive the input vectors sequentially. Each CF uses an update method, such as Recursive Least Squares (RLS) or Least Mean Squares (LMS), depending on the target of the applications or problem constraints [21]. After receiving the input vector, each algorithm produces its output and then calculates its instantaneous error after the observation is revealed. In the most generic setting, this estimation/prediction error and the corresponding input vector are then used to update the internal parameters of the algorithm to minimize a priori defined loss function, e.g., instantaneous error for the LMS algorithm. These updates are performed for all of the m CFs in the mixture. However, in the online boosting approaches, these adaptations at each time proceed in rounds from top to bottom, starting from the first CF to the last one to achieve the "boosting" effect [22]. Furthermore, unlike the usual mixture approaches [13,14], the update of each CF depends on the previous CFs in the mixture. In particular, at each time t, after the kth CF calculates its error over $(\boldsymbol{x}_t, d_t)$ pair, it passes a certain weight to the next CF, the $(k+1)$th CF, quantifying how much error the constituent CFs from 1st to kth made on the current $(\boldsymbol{x}_t, d_t)$ pair. Based on the performance of the CFs from 1 to k on the current $(\boldsymbol{x}_t, d_t)$ pair, the $(k+1)$th CF may give a different emphasis (importance weight) to $(\boldsymbol{x}_t, d_t)$ pair in its adaptation in order to rectify the mistake of the previous CFs.

The proposed idea for online boosting is clearly related to the adaptive mixture-of-experts algorithms widely used in the machine learning literature, where several parallel running adaptive algorithms are combined to improve the performance. In the mixture methods, the performance improvement is achieved due to the diversity provided by using several different adaptive algorithms each having a different view or advantage [14]. This diversity is exploited to yield a final combined algorithm, which achieves a performance better than any of the algorithms in the mixture. Although the online boosting approach is similar to mixture approaches [14], there are significant differences. In the online boosting notion, the parallel running algorithms are not independent, i.e., one deliberately introduces the diversity by updating the CFs one by one from the first CF to the kth CF for each new sample based on the performance of all the previous CFs on this sample. In this sense, each adaptive algorithm, say the $(k+1)$th CF, receives feedback from the previous CFs, i.e., 1st to kth, and updates its inner parameters accordingly. As an example, if the current $(\boldsymbol{x}_t, d_t)$ is well modeled by the previous CFs, then the $(k+1)$th CF performs minor update using $(\boldsymbol{x}_t, d_t)$ and may give more emphasis (importance weight) to the later arriving samples that may be worse modeled by the previous CFs. Thus, by boosting, each adaptive algorithm in the mixture can concentrate on different parts of the input and output pairs achieving diversity and significantly improve the gain.

The linear adaptive filters, such as LMS or RLS, are among the simplest as well as the most widely used regression algorithms in the real-life applications [21]. Therefore, we use such algorithms as the CFs in our boosting algorithms. To this end, we first apply the boosting notion to several parallel running linear RLS-based CFs and introduce three different approaches to use the importance weights [22], namely "weighted updates", "data reuse", and "random updates". In the first approach, we use the importance weights directly to produce certain weighted RLS algorithms. In the second approach, we use the importance weights to construct data reuse adaptive algorithms [18]. However, data reuse in boosting, such as [18], is significantly different from the usual data reusing approaches in adaptive filtering [15]. As an example, in boosting, the importance weight coming from the kth CF determines the data reuse amount in the $(k+1)$th CF, i.e., it is not used for the kth filter, hence, achieving the diversity. The third approach uses the importance weights to decide whether to update the constituent CFs or not, based on a random number generated from a Bernoulli distribution with the parameter equal to the weight. The latter method can be effectively used for big data processing [23] due to the reduced complexity. The output of the constituent CFs is also combined using a linear mixture algorithm to construct the final output. We then update the final combination algorithm using the LMS algorithm [14]. Furthermore, we extend the boosting idea to parallel running linear LMS-based algorithms similar to the RLS case.

We start our discussion by investigating the related work in Section 2 and continue by introducing the problem setup and background in Section 3. We introduce our generic boosted adaptive filter algorithm in Section 4 and provide the mathematical justifications for its performance. Then, in Section 5, three different variants of the proposed boosting algorithm are derived, using the RLS and LMS, which are extended to piecewise linear filters in Section 6. Then, in Section 7 we provide the mathematical analysis for the computational complexity of the proposed algorithms. The paper concludes with an extensive set of experiments over the well-known benchmark data sets and simulation models widely used in the machine learning literature to demonstrate the significant gains achieved by the boosting notion.

2. Related work

AdaBoost is one of the earliest and most popular boosting methods, which has been used for binary and multiclass classifications as well as regression [7]. This algorithm has been well studied and has clear theoretical guarantees, and its excellent performance is explained rigorously [24]. However, AdaBoost cannot perform well on noisy data sets [25], therefore, other boosting methods have been suggested that are more robust against noise.

In order to reduce the effect of noise, SmoothBoost was introduced in [25] in a batch setting, which avoids overemphasizing the noisy samples, hence, provides robustness against noise. In [18], the authors extend bagging and boosting methods to an online setting, where they use a Poisson sampling process to approximate the reweighting algorithm. However, the online boosting method in [18] corresponds to AdaBoost, which is susceptible to noise. In [26], the authors use a greedy optimization approach to develop the boosting notion to the online setting and introduce stochastic boosting. Nevertheless, while most of the online boosting algorithms in the literature seek to approximate AdaBoost, [22] investigates the inherent difference between batch and online learning, extends the SmoothBoost algorithm to an online setting, and provides the mathematical guarantees for their algorithm. [22] points out that the online weak learners do not need to perform well on all possible distributions of data, instead, they have to perform well only with respect to smoother distributions. Recently, in [27], the authors have developed two online boosting algorithms for classification, an optimal algorithm in terms of the number of weak learners, and also an adaptive algorithm using the potential functions and boost-by-majority [28].

In addition to the classification task, the boosting approach has also been developed for the regression [6]. In [29], a boosting algorithm for regression is proposed, which is an extension of Adaboost.R [29]. Moreover, in [6], several gradient descent algorithms are presented, and some bounds on their performances are provided. In [26], the authors present a family of boosting algorithms for online regression through greedy minimization of a loss function. Also, in [30], the authors propose an online gradient boosting algorithm for regression.

In this paper, we propose a novel family of boosted adaptive filters (as an ensemble or combination technique) using the "online boosting" in [22], and investigate three different variants of the introduced algorithm. Furthermore, we show that our algorithm achieves a desired mean squared error (MSE) regret bound, given sufficient amount of data and a sufficient number of CFs.

The presented novel approach relates to "adaptive filters" [21] and "their combinations" [13] (since boosting can be seen as a combination). Hence, literature on adaptive filters [21], and in particular, combination techniques [13], are of special interest.

Linear filters (e.g., LMS, RLS) [21], and their nonlinear extensions, e.g., kernel RLS [31], in addition their widely-studied step-size-optimized instances, e.g., Normalized-LMS (NLMS) [21] or Armijo-rule-learning-rate-LMS (ALR-LMS) [11], are among the well-known examples of adaptive filters. For instance, in NLMS [21] and NLMS with adaptive regularization [32], the step size are variable leading to a guaranteed convergence with no assumption on the source statistics, whereas in [11], the step size is chosen from a set of learning rates. Both and other examples [33,34] essentially address the unknown and possibly varying source statistics. In these examples, using a single algorithm for the problem in hand, e.g., regression/prediction [21], or classification (i.e. perceptron) [4], turns out sub-optimal when the constituent algorithm does not sufficiently address the complexity of the problem. The reason is that the overall performance is obviously bounded by the one of the employed algorithm, e.g., LMS or RLS, which might underperform due to insufficient/improper modeling of the unknown complexity of the problem in hand, even when trained with the best step-sizes/parameters for addressing non-stationarity. Using a sophisticated technique – for instance, kernel RLS [31] – to address the complexity might be a solution, where – however – it is hard to optimize the parameters, regularize and control the complexity, leaving it susceptible to "overfitting" [4].

Hence, to gradually increase the modeling power to any desired degree without being bounded by the performance of a "single algorithm" and to explicitly control the complexity to mitigate overfitting [35], combination or ensemble techniques are largely exploited under boosting [7,18,22,36], mixture of experts [37–39] and combination of adaptive filters [13]. In addition to decent modeling and generalization, ensemble techniques are also well-known to be highly capable of adapting to non-stationarity, cf. the use of ensemble approach in "concept drift" studies, e.g., [40]. Adaboost [7] generalizes the weighted majority algorithm [36] for batch processing, and [18] presents the convergence to the result of Adaboost in the online setting under certain assumptions. On the other hand, this paper strongly contributes by introducing the boosting notion of machine learning to adaptive filtering literature, and the goal is to exploit these well-established and powerful boosting properties for signal processing by combining adaptive filters via the importance weights in [22].

There are two prominent approaches in the literature on combination of adaptive filters [13]. First approach trains the constituent algorithms independently by their own updates with no feedbacks, whereas the second approach allows inter-filter communications of feedbacks for an improved combination performance. In [41] (an example of the first approach), a convex combination of two adaptive filters (weighting is through sigmoid) is proposed, which is guaranteed to perform as well as the best constituent algorithm (in steady state under stationarity in the sense of minimum mean squared error) and also shown to outperform the constituents under further statistical assumptions. For this algorithm [41], the tracking performance with non-stationary data is presented in [42], where improvements are possible in steady state when different kind of filters (i.e., not of the same kind with different parameters/step sizes) are combined [43]; and a deterministic worst-case analysis is also presented in [44].

In these examples [41–44], the constituent algorithms are run independently. In our work, on the contrary, updates of the constituent algorithms depend on each other sequentially from top to bottom through the flow of the importance weights [22]. In this regard, the above-mentioned second approach of combination of adaptive filters are perhaps related, where feedbacks are allowed inter filters [13]. For example, when a sudden change occurs in the data source statistics [13], coefficients of the fast adapting constituent filter can be transferred in one-way to the other ones gradually [45] or simply copied over them [46], resulting in a more robust performance. Unlike the unilateral sharing [45,46], the method in [47] exploits a certain feedback configuration based on a cyclic and periodic coefficient sharing among all filters. In [48], a second order Taylor approximation for the non-linearity of the convex combination [41] of filters together with the coefficient sharing idea is observed to improve the convergence behavior of the combination.

We emphasize that our proposed technique for boosting adaptive filters radically differs in comparison to the approaches of coefficients sharing among constituent filters. Those examples [13, 45–48] generally sense the change in the environment, and the coefficients of the significantly outperforming filter(s) (filter whose weight is close to 1 in the convex combination) gradually or suddenly overwrites the ones of the others. In contrast, we exploit the boosting notion in our combination: the constituent filters are ordered to best model the instantaneous desired outcome in a constantly improving manner from the first filter to the last by using the importance weights [22] in three different paradigms: "weighted updates", "data reuse", "random updates". The resulting combination naturally boosts the overall performance. Unlike [13,45–48], our approach utilizes all of the filters collaboratively towards a boosted decision instead of letting one of the filters dominate. We present this novel paradigm to the literature of combination of adaptive filters.

In another example, a weighting-level combination is used [12]. The weights from a proportionate type adaptive filtering [49] (to exploit sparse impulse responses) and weights from an original adaptive filtering such as NLMS [21] (to exploit dispersive impulse responses) are combined through sigmoid for a faster convergence, where the combination is at the weight-mixing level in contrast to our decision level combination. Sparse impulse response can also appear due to the unknown and time-varying low-SNR conditions as in the case of acoustic echo cancellation, where the learning of the less significant adaptive impulse response coefficients is prone to estimation errors due to the gradient noise when – for instance – NLMS is used [50]. A block-wise combination of adaptive filters is proposed to handle this in [50], where the adaptive impulse response is split into non-overlapping blocks and a linear combination of NLMSs is obtained for the final output (the weight of each NLMS is learned by [41] using a virtual zero block in parallel). Improvement in the mean squared error performance in achieved under low-SNR. Similarly, a block-wise combination is also used in [51] that, in contrast, uses zero-attraction (i.e., terms in the update forcing coefficients to get closer to 0) instead of a complete zero block. Hence, unlike [50], a satisfactory performance is obtained under low-SNR as well as high-SNR.

Predicting with expert advice [52] is another popular framework for combining adaptive filters to obtain mixture of experts based algorithms. In [53], a piecewise linear prediction algorithm is proposed as a combination of experts each of which is defined based on a tree-partition of the observation space with RLS filters in partition regions. Learning the optimal partition in addition to the optimal combination is shown to yield a universal piecewise nonlinear regression algorithm in [54] whose computationally efficient $O(n)$ (n being the number of processed instances) extension to classification is presented in [55] with perceptrons [4] as the basis for constructing the experts. Such algorithms are generally shown to asymptotically perform at least as well as the best constituent filter for every possible input stream on an individual sequence basis. Hence, the general approach in this framework is competitive against a large class in the worst case without any statistical assumptions. In this regard, the prediction algorithm (as another example) in [56] is competitive against the best linear predictor for which the non-stationarity is also considered in [57]. Generally, a regret analysis [58] is used in these studies and relatively older other studies [37–39,59,60] to prove that the combination asymptotically performs at least as well as the best constituent filter. However, we emphasize that in this presented work, our goal is to achieve beyond and obtain a combination that outperforms the best constituent filter through the boosting notion of machine learning.

3. Problem description and background

We sequentially receive r-dimensional[1] input (regressor) vectors $\{\boldsymbol{x}_t\}_{t\geq 1}$, $\boldsymbol{x}_t \in \mathbb{R}^r$, and desired data $\{d_t\}_{t\geq 1}$, and estimate d_t by $\hat{d}_t = f_t(\boldsymbol{x}_t)$, where $f_t(.)$ is an adaptive filtering (online regression) algorithm that competes against the class $\mathcal{F}$ of predictors. At each time t the estimation error is given by $e_t = d_t - f_t(\boldsymbol{x}_t)$ and is used to update the parameters of the CF. We also define squared loss function as $\ell_t(f_t(\boldsymbol{x}_t)) = (d_t - f_t(\boldsymbol{x}_t))^2$ where $\ell_t : \mathbb{R}^r \to \mathbb{R}$. For presentation purposes, we assume that $d_t \in [-1, 1]$, however, our derivations hold for any bounded but arbitrary desired data sequences. In our framework, we do not use any statistical assumptions on the input feature vectors or on the desired data such that our results are guaranteed to hold in an individual sequence manner [61].

The linear methods are considered as the simplest filters, which estimate the desired data d_t by a linear model as $\hat{d}_t = \boldsymbol{x}_t^T \boldsymbol{w}_t$, where $\boldsymbol{w}_t$ is the linear algorithm's coefficients at time t. Note that the previous expression also covers the affine model if one includes a constant term in $\boldsymbol{x}_t$, hence, we use the purely linear form for notational simplicity. When the true d_t is revealed, the algorithm updates its coefficients $\boldsymbol{w}_t$ based on the error e_t. As an example, in the basic implementation of the RLS algorithm, the coefficients are selected to minimize the accumulated squared regression error, i.e., $\ell_t(\boldsymbol{x}_t^T \boldsymbol{w}) = (d_t - \boldsymbol{x}_t^T \boldsymbol{w})^2$, up to time $t-1$. We emphasize that in the basic application of the RLS algorithm, all data pairs $(\boldsymbol{x}_i, d_i)$, $i = 1, \ldots, t$, receive the same "importance" or weight. Although there exists exponentially weighted or windowed versions of the basic RLS algorithm [21], these methods weight (or concentrate on) the most recent samples for better modeling of the nonstationarity [21]. However, in the boosting framework [7], each sample pair receives a different weight based on not only those weighting schemes, but also the performance of the boosted algorithms on this pair. As an example, if a CF performs worse on a sample, the next CF concentrates more on this example to better rectify this mistake. In the following sections, we use this notion to derive different boosted adaptive filtering algorithms.

4. New boosted adaptive filtering algorithm

In this section, we present the generic form of our proposed algorithms and provide the guaranteed performance bounds for that. Regarding the notion of "online boosting" introduced in [22], the online constituent filters need to perform well only over smooth distributions of data points. We first present the generic algorithm in Algorithm 1 and provide its theoretical justifications, then discuss about its structure and the intuition behind it.

In this algorithm, each constituent filter receives a sequence of data points $(\boldsymbol{x}_t, d_t)$ and a corresponding weight $0 \leq \lambda_t \leq 1$ for each point. Since $d_t \in [-1, 1]$, we define the Weighted MSE (WMSE) [62] of a learning algorithm as $\frac{\sum_{t=1}^T \lambda_t (e_t)^2}{4\sum_{t=1}^T \lambda_t}$, where $e_t = d_t - \hat{d}_t \in [-2, 2]$. Before proceeding to the algorithm, we provide the definitions that help the readers to better understand the results.

Definition 1 *(Regret bound).* Suppose that an online algorithm A (the adaptive filter) is designed to compete against a function class $\mathcal{F}$. The regret bound of the algorithm A is defined as

$$R(T) \triangleq \sum_{t=1}^{T} \ell_t(A_t(\boldsymbol{x}_t)) - \inf_{f\in\mathcal{F}} \sum_{t=1}^{T} \ell_t(f(\boldsymbol{x}_t)).$$

Definition 2 *(β-smoothness of $\ell(x)$).* A function $\ell(x)$ is said to be β-smooth with respect to norm $||\cdot||$ if and only if for any pair x and y we have

$$\ell(y) \leq \ell(x) + \nabla\ell(x)(y-x) + \frac{\beta}{2}||x-y||^2,$$

where $\nabla\ell(x)$ is the gradient of the function ℓ with respect to x.

Definition 3 *(Online constituent filter edge).* Given the sequence of losses $\{||d_t - \hat{d}_t||^2\}_{t\geq 1}$, the online constituent filter generates a sequence of predictions $\{\hat{d}_t\}_{t\geq 1}$ that has an edge $\gamma \in (0, 1]$, with respect to the trivial predictor $\hat{d}_t = 0$, such that with high probability [63]:

$$\sum_{t=1}^{T} ||d_t - \hat{d}_t||^2 \leq (1-\gamma)\sum_{t=1}^{T} ||d_t||^2 + R(T),$$

where $R(T)$ is known as the excess loss that includes in itself the regret of the online algorithm.

Definition 4 *(ξ-strong convexity).* A function $\ell(x)$ is said to be ξ-strong convex with respect to norm $||\cdot||$ if and only if for any pair x and y, we have

$$\ell(y) \geq \ell(x) + \nabla\ell(x)(y-x) + \frac{\xi}{2}||y-x||^2.$$

For $y = x^* = \text{argmin}_x \ell(x)$, as shown in the Appendix B, we have [63]

$$||\nabla\ell(x)||^2 \geq 2\xi\big(\ell(x) - \ell(x^*)\big). \tag{1}$$

Note that β-smoothness and ξ-strong convexity provide upper and lower bounds on the variation $\ell(y) - \ell(x)$, respectively.

In the following theorem, we show that if $\sum_{t=1}^T \lambda_t$ is large enough, there exists an online constituent filter that achieves a specific (better than the trivial solution) WMSE.

Theorem 1. *Suppose for any sequence of data points and corresponding weights λ_t, where $\lambda_1 = 1$, there is an offline algorithm with a WMSE of σ^2_{off}, i.e.,*

$$\frac{\sum_{t=1}^T \lambda_t (e_t^{\text{off}})^2}{4\sum_{t=1}^T \lambda_t} = \sigma^2_{\text{off}}$$

Moreover, assume that $\ell_t = e_t^2$ is ξ-strongly convex and β-smooth, and

[1] All vectors are column vectors and represented by bold lower case letters. Matrices are represented by bold upper case letters. For a vector $\boldsymbol{a}$ (or a matrix $\boldsymbol{A}$), $\boldsymbol{a}^T$ (or $\boldsymbol{A}^T$) is the transpose and Tr($\boldsymbol{A}$) is the trace of the matrix $\boldsymbol{A}$. Here, $\boldsymbol{I}_m$, $\boldsymbol{0}_m$ and $\boldsymbol{1}_m$ represent the identity matrix of dimension $m \times m$ and the all zeros and ones vector of length m, respectively. Except $\boldsymbol{I}_m$ and $\boldsymbol{0}_m$, the time index is given in the subscript, i.e., x_t is the sample at time t. We work with real data for notational simplicity. We denote the mean of a random variable x as $E[x]$. Also, we show the cardinality of a set S by $|S|$.

$$\sum_{t=1}^{T} \lambda_t \geq \frac{\beta^2}{4\epsilon\xi\sigma_{\text{off}}^2}, \tag{2}$$

where ϵ is a positive number. Under the stated conditions, there exists an online algorithm with a WMSE of at most $\sigma^2 = (1+\epsilon)\sigma_{\text{off}}^2$.

Proof. Refer to the Appendix A □

Algorithm 1 Boosted Adaptive Filter algorithm.

```
1: Input: $(\boldsymbol{x}_t, d_t)$ (data stream), $m$ (number of constituent filters running in parallel), $\sigma_m^2$ (the modified desired MSE), and $\sigma^2$ (the guaranteed achievable WMSE).
2: Initialize the filter coefficients $\boldsymbol{w}_1^{(k)}$ for each CF; and the combination coefficients as $\boldsymbol{z}_1 = \frac{1}{m}\mathbf{1}_m$; $\lambda_1^{(k)} = 1$;
3: for $t = 1$ to $T$ do
4:     Receive the input data instance $\boldsymbol{x}_t$;
5:     Compute the CFs outputs $\hat{d}_t^{(k)}$;
6:     Produce the final estimate $\hat{d}_t = \boldsymbol{z}_t^T \boldsymbol{y}_t = \boldsymbol{z}_t^T [\hat{d}_t^{(1)}, \ldots, \hat{d}_t^{(m)}]^T$;
7:     Receive the true output $d_t$ (desired data);
8:     $\lambda_t^{(1)} = 1$; $l_t^{(1)} = 0$;
9:     for $k = 1$ to $m$ do
10:        $\lambda_t^{(k)} = \min\left\{1, (\sigma^2)^{l_t^{(k)}/2}\right\}$ (for $t \geq 2$);
11:        Update the CF$^{(k)}$, such that it has a WMSE $\leq \sigma^2$;
12:        $e_t^{(k)} = d_t - \hat{d}_t^{(k)}$;
13:        $l_t^{(k+1)} = l_t^{(k)} + \left[\sigma_m^2 - \left(e_t^{(k)}\right)^2\right]$;
14:    end for
15:    Update $\boldsymbol{z}_t$ based on $e_t = d_t - \boldsymbol{z}_t^T \boldsymbol{y}_t$;
16: end for
```

Before we state the Theorem 2 and provide its proof, we make three important assumptions: (1) the loss function is strongly convex, (2) the loss function is smooth and (3) the constituent filters have edges, i.e., $\gamma \in (0, 1]$. One of the most typical choices in real life applications is the squared loss which satisfies the assumptions 1 and 2 here. On the other hand, assumption 3 is common in the boosting literature [7,63], which holds under mild conditions (for instance, cf. Section "Why weak learner edge is reasonable" in [63]). Based on the preceding assumptions, we derive the regret bound of the boosted adaptive filter algorithm in Theorem 2.

Theorem 2. *Suppose that ℓ_t is a ξ-strongly convex and β-smooth loss function. For the mth constituent filter, where the CFs have an online learning edge $\gamma \in (0, 1]$, and $f^* = \arg\min_{f\in\mathcal{F}} \sum_t \ell_t(f_t(\boldsymbol{x}_t))$, we have the following regret bound.*

$$\sum_{t=1}^{T} \ell_t(\hat{d}_t^{(m)}) - \sum_{t=1}^{T} \ell_t(f^*) \leq D_P^m \Delta_0 + \left(R'(T) + D_P D_Q T\right)\frac{1}{1-D_P}, \tag{3}$$

where $R'(T) \triangleq 2D_P\sqrt{D_Q T}\sqrt{(1-\gamma)\sum_{t=1}^{T}||d_t||^2 + R(T)}$ (see Definition 3), D_P and D_Q are constants defined in the proof, and Δ_0 denotes the regret of the first filter.

Proof. Refer to the Appendix C. □

In Theorem 2, we derived the regret bound of the algorithm. We proved that for a particular step size μ, the regret bound of the LMS-based adaptive filter decays exponentially in terms of the number of weak learners. We show that as the number of data samples increases, the total regret bound of the algorithm decreases. In addition to the decay of regret bound with respect to the number of data samples, we also prove that the total regret of the online boosting algorithm decreases as we parse through the constituent filters.

Note that, although we use copies of a base learner as the constituent filters and seek to improve its performance, the CFs can be different. However, by using the boosting approach, we can improve the MSE performance of the overall system as long as the CFs can provide a WMSE of at most σ^2. For example, we can improve the performance of mixture-of-experts algorithms [13] by leveraging the boosting approach introduced in this paper.

As shown in Fig. 1, at each iteration t, we have m parallel running CFs with estimating functions $f_t^{(k)}$, producing estimates $\hat{d}_t^{(k)} = f_t^{(k)}(\boldsymbol{x}_t)$ of d_t, $k = 1, \ldots, m$. As an example, if we use m "linear" algorithms, $\hat{d}_t^{(k)} = \boldsymbol{x}_t^T \boldsymbol{w}_t^{(k)}$ is the estimate generated by the kth CF. The outputs of these m CFs are then combined using the linear weights $\boldsymbol{z}_t$ to produce the final estimate as $\hat{d}_t = \boldsymbol{z}_t^T \boldsymbol{y}_t$ [14], where $\boldsymbol{y}_t \triangleq [\hat{d}_t^{(1)}, \ldots, \hat{d}_t^{(m)}]^T$ is the vector of outputs. After the desired output d_t is revealed, the m parallel running CFs will be updated for the next iteration. Moreover, the linear combination coefficients $\boldsymbol{z}_t$ are also updated using the normalized LMS [21], as detailed later in Section 4.1.

After d_t is revealed, the constituent CFs, $f_t^{(k)}$, $k = 1, \ldots, m$, are consecutively updated, as shown in Fig. 1, from top to bottom, i.e., first $k = 1$ is updated, then, $k = 2$ and finally $k = m$ is updated. However, to enhance the performance, we use a boosted updating approach [7], such that the $(k+1)$th CF receives a "total loss" parameter, $l_t^{(k+1)}$, from the kth CF, as

$$l_t^{(k+1)} = l_t^{(k)} + \left[\sigma_m^2 - \left(d_t - f_t^{(k)}(\boldsymbol{x}_t)\right)^2\right], \tag{4}$$

to compute a weight $\lambda_t^{(k)}$. The total loss parameter $l_t^{(k)}$, indicates the sum of the differences between the modified desired MSE (σ_m^2) and the squared error of the first $k-1$ CFs at time t. Then, we add the difference $\sigma_m^2 - (e_t^{(k)})^2$ to $l_t^{(k)}$, to generate $l_t^{(k+1)}$, and pass $l_t^{(k+1)}$ to the next CF, as shown in Fig. 1. Here, $\left[\sigma_m^2 - \left(d_t - f_t^{(k)}(\boldsymbol{x}_t)\right)^2\right]$ measures how much the kth CF is off with respect to the final MSE performance goal. For example, in a stationary environment, if $d_t = f(\boldsymbol{x}_t) + \nu_t$, where $f(\cdot)$ is a deterministic function and ν_t is the observation noise, one can select the desired MSE σ_m^2 as an upper bound on the variance of the noise process ν_t. In this sense, $l_t^{(k)}$ measures how the CFs $j = 1, \ldots, k$ are cumulatively performing on $(\boldsymbol{x}_t, d_t)$ pair with respect to the final performance goal.

We then use the weight $\lambda_t^{(k)}$ to update the kth CF with the "weighted updates", "data reuse", or "random updates" method, which we explain later in Section 5. Our aim is to make $\lambda_t^{(k)}$ large if the first $k-1$ CFs made large errors on d_t, so that the kth CF gives more importance to $(\boldsymbol{x}_t, d_t)$ in order to rectify the performance of the overall system. We now explain how to construct these weights, such that $0 < \lambda_t^{(k)} \leq 1$. To this end, we set $\lambda_t^{(1)} = 1$, for all t, and introduce a weighting similar to [22,25]. We define the weights as

$$\lambda_t^{(k)} = \min\left\{1, \left(\sigma^2\right)^{l_t^{(k)}/2}\right\}, \tag{5}$$

where σ^2 is the guaranteed upper bound on the WMSE of the constituent filters. However, since there is no prior information about the exact MSE performance of the constituent filters, we use the following weighting scheme

$$\lambda_t^{(k)} = \min\left\{1, \left(\delta_{t-1}^{(k)}\right)^{c\, l_t^{(k)}}\right\}, \tag{6}$$

where $\delta_{t-1}^{(k)}$ indicates an estimate of the kth constituent filter's MSE, and $c \geq 0$ is a design parameter, which determines the "dependence" of each CF update on the performance of the previous

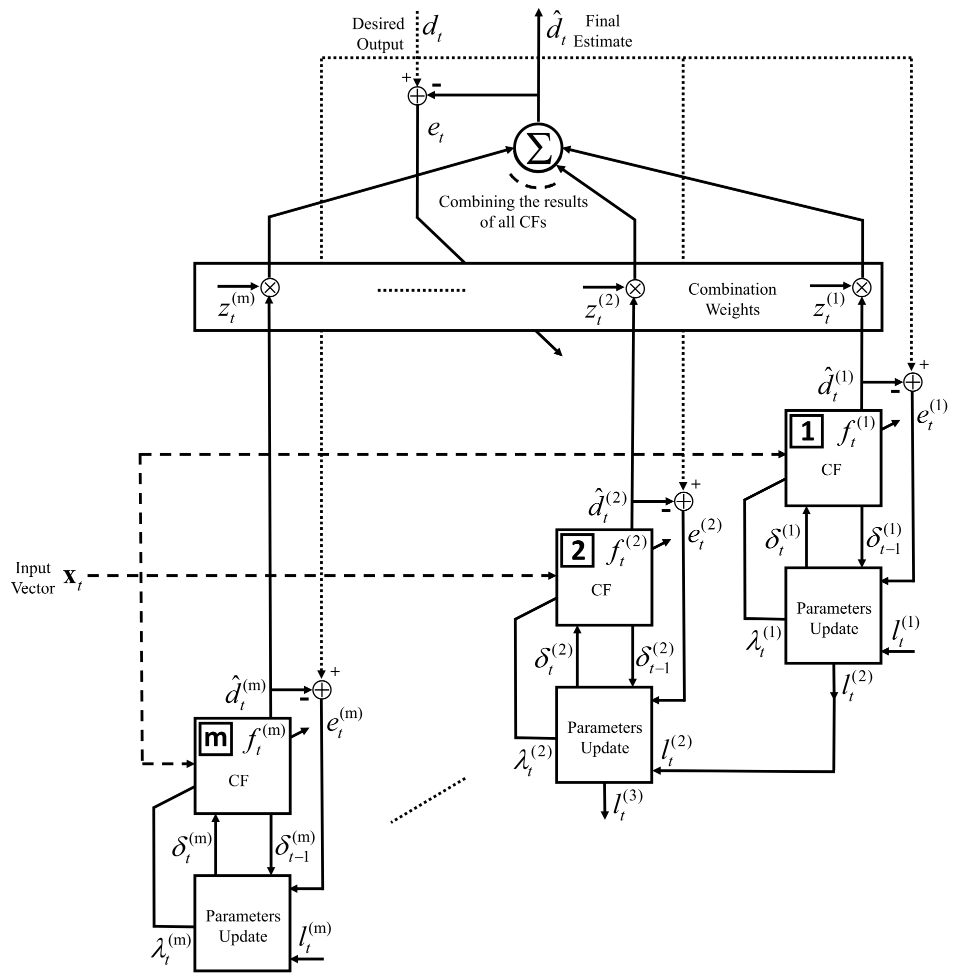


Fig. 1. The block diagram of a boosted adaptive filtering system that uses the input vector $\boldsymbol{x}_t$ to produce the final estimate $\hat{d}_t$. There are m constituent CFs $f_t^{(1)}, \ldots, f_t^{(m)}$, each of which is an online linear algorithm that generates its own estimate $\hat{d}_t^{(k)}$. The final estimate $\hat{d}_t$ is a linear combination of the estimates generated by all these constituent CFs, with the combination weights $z_t^{(k)}$'s corresponding to $\hat{d}_t^{(k)}$'s. The combination weights are stored in a vector that is updated after each iteration t. At time t the kth CF is updated based on the values of $\lambda_t^{(k)}$ and $e_t^{(k)}$, and provides the $(k+1)$th filter with $l_t^{(k+1)}$ that is used to compute $\lambda_t^{(k+1)}$. The parameter $\delta_t^{(k)}$ indicates the WMSE of the kth CF over the first t estimations, and is used in computing $\lambda_t^{(k)}$. Note that the parameters update block is described in Fig. 2.

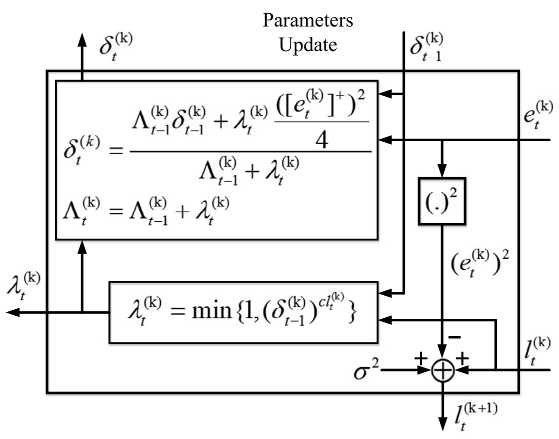

Parameters Update
$\delta_t^{(k)}$
$\delta_{t-1}^{(k)}$
$e_t^{(k)}$
$\delta_t^{(k)} = \dfrac{\Lambda_{t-1}^{(k)}\delta_{t-1}^{(k)} + \lambda_t^{(k)}\dfrac{([e_t^{(k)}]^+)^2}{4}}{\Lambda_{t-1}^{(k)} + \lambda_t^{(k)}}$
$\Lambda_t^{(k)} = \Lambda_{t-1}^{(k)} + \lambda_t^{(k)}$
$(.)^2$
$(e_t^{(k)})^2$
$\lambda_t^{(k)}$
$\lambda_t^{(k)} = \min\{1, (\delta_{t-1}^{(k)})^{c l_t^{(k)}}\}$
σ^2
$l_t^{(k)}$
$l_t^{(k+1)}$

Fig. 2. Parameters update block of the kth constituent filter, which is embedded in the kth filter block as depicted in Fig. 1. This block receives the parameter $l_t^{(k)}$ provided by the $(k-1)$th filter, and uses that in computing $\lambda_t^{(k)}$. It also computes $l_t^{(k+1)}$ and passes it to the $(k+1)$th filter. The parameter $[e_t^{(k)}]^+$ represents the error of the thresholded estimate as explained in (7), and $\Lambda_t^{(k)}$ shows the sum of the weights $\lambda_1^{(k)}, \ldots, \lambda_t^{(k)}$. The WMSE parameter $\delta_{t-1}^{(k)}$ represents the time averaged weighted square error made by the kth filter up to time $t-1$.

CFs, i.e., $c = 0$ corresponds to "independent" updates, like the ordinary combination of the CFs in adaptive filtering [13,14], while a greater c indicates the greater effect of the previous CFs performance on the weight $\lambda_t^{(k)}$ of the current CF. Note that including the parameter c does not change the validity of our proofs, since one can take $\left(\delta_{t-1}^{(k)}\right)^c$ as the new guaranteed WMSE. Here, $\delta_{t-1}^{(k)}$ is an estimate of the WMSE of the kth CF over $\{\boldsymbol{x}_t\}_{t\geq 1}$ and $\{d_t\}_{t\geq 1}$. In the basic implementation of the online boosting [22,25], $\left(1 - \delta_{t-1}^{(k)}\right)$ is set to the classification advantage of the constituent filters [25], where this advantage is assumed to be the same for all the constituent filters. In this paper, to avoid using any a priori knowledge and to be completely adaptive, we choose $\delta_{t-1}^{(k)}$ as the weighted and thresholded MSE of the kth CF up to time $t-1$ as

$$\delta_t^{(k)} = \frac{\sum_{\tau=1}^{t} \frac{\lambda_\tau^{(k)}}{4}\left(d_\tau - \left[f_\tau^{(k)}(\boldsymbol{x}_\tau)\right]^+\right)^2}{\sum_{\tau=1}^{t} \lambda_\tau^{(k)}}$$

$$= \frac{\Lambda_{t-1}^{(k)}\delta_{t-1}^{(k)} + \frac{\lambda_t^{(k)}}{4}\left(d_t - \left[f_t^{(k)}(\boldsymbol{x}_t)\right]^+\right)^2}{\Lambda_{t-1}^{(k)} + \lambda_t^{(k)}}, \tag{7}$$

where $\Lambda_t^{(k)} \triangleq \sum_{\tau=1}^{t} \lambda_\tau^{(k)}$, and $\left[f_\tau^{(k)}(\boldsymbol{x}_\tau)\right]^+$ thresholds $f_\tau^{(k)}(\boldsymbol{x}_\tau)$ into the range $[-1, 1]$. This thresholding is necessary to assure that $0 < \delta_t^{(k)} \leq 1$, which guarantees $0 < \lambda_t^{(k)} \leq 1$ for all $k = 1, \ldots, m$ and t. We point out that (7) can be recursively calculated.

Regarding the definition of $\lambda_t^{(k)}$, if the first k CFs are "good", we will pass less weight to the next CFs, such that those CFs can concentrate more on the other samples. Hence, the CFs can increase the diversity by concentrating on different parts of the data [14]. Note that according to its definition, the parameter $\delta_{t-1}^{(k)}$ is always between 0 and 1. Therefore, if $l_t^{(k)} > 0$, we have $\lambda_t^{(k)} < 1$, i.e., the larger $l_t^{(k)}$ is, the smaller $\lambda_t^{(k)}$ will be. However, $l_t^{(k)} > 0$ implies that the previous constituent filters have had a "good" performance on this data point (i.e., $\boldsymbol{x}_t$) and also a small $\lambda_t^{(k)}$ means a small importance weight for this data point. On the other hand, when $l_t^{(k)} \leq 0$, we have $\lambda_t^{(k)} = 1$, which is the largest possible importance weight. As a result, we observe that when the first $k-1$ CFs perform poorly/well with respect to the desired MSE goal, a higher/lower weight for the current data point is passed to the k-th CF. Nevertheless, note that the maximum possible weight for each data point is 1 in accordance to the definition of smooth weight distributions in [22]. Briefly, the smooth weight distributions avoid overemphasizing the noisy data points.

4.1. The combination algorithm

Although in the proof of our algorithm, we assume a constant combination vector $\boldsymbol{z}$ over time, we use a time varying combination vector in practice, since there is no knowledge about the exact number of the required constituent filters for each problem. Hence, after d_t is revealed, we also update the final combination weights $\boldsymbol{z}_t$ based on the final output $\hat{d}_t = \boldsymbol{z}_t^T \boldsymbol{y}_t$, where $\hat{d}_t = \boldsymbol{z}_t^T \boldsymbol{y}_t$, $\boldsymbol{y}_t = [\hat{d}_t^{(1)}, \ldots, \hat{d}_t^{(m)}]^T$. To update the final combination weights, we use the normalized LMS algorithm [21] yielding

$$\boldsymbol{z}_{t+1} = \boldsymbol{z}_t + \mu_z e_t \frac{\boldsymbol{y}_t}{\iota + \|\boldsymbol{y}_t\|^2}, \tag{8}$$

where ι is a small positive constant used for numerical stabilization when $\|\boldsymbol{y}_t\|^2$ is very small.

4.2. Choice of parameter values

The choice of σ_m^2 is a crucial task, i.e., one cannot reach any desired MSE for any data sequence unconditionally. Intuitively, there is a guaranteed upper bound (i.e., σ^2) on the worst case performance, since in the weighted MSE, the samples with a higher error have a more important effect. On the other hand, if one chooses a σ_m^2 smaller than the noise power, $l_t^{(k)}$ will be negative for almost every k, turning most of the weights into 1, and as a result the weak learners fail to reach a WMSE smaller than σ^2. Nevertheless, in practice we have to choose the parameter σ_m^2 reasonably and precisely such that the conditions of Theorem 2 are satisfied. For instance, we set σ_m^2 to be an upper bound on the noise power [64].

In addition, the number of constituent filters, m, as well as the parameter K in data reuse mode, are chosen regarding the computational complexity constraints. However, in our experiments we choose a moderate number of constituent filters, $m = 20$, which successfully improves the performance. Moreover, according to the results in Section 8.3, the optimum (in the sense of minimum MSE) value (using a brute force method) for c is around 1, hence, we set the parameter $c = 1$ in our simulations.

5. Boosted linear adaptive filters

At each time t, all of the CFs (shown in Fig. 1) estimate the desired data d_t in parallel, and the final estimate is a linear combination of the results generated by the CFs. When the kth CF receives the weight $\lambda_t^{(k)}$, it updates the linear coefficients $\boldsymbol{w}_t^{(k)}$ using one of the following methods.

5.1. Directly using λ's as sample weights

Here, we consider $\lambda_t^{(k)}$ as the weight for the observation pair $(\boldsymbol{x}_t, d_t)$ and apply a weighted RLS update to $\boldsymbol{w}_t^{(k)}$. For this particular weighted RLS algorithm, we define the Hessian matrix and the gradient vector as

$$\boldsymbol{R}_{t+1}^{(k)} \triangleq \beta \boldsymbol{R}_t^{(k)} + \lambda_t^{(k)} \boldsymbol{x}_t \boldsymbol{x}_t^T, \tag{9}$$

$$\boldsymbol{p}_{t+1}^{(k)} \triangleq \beta \boldsymbol{p}_t^{(k)} + \lambda_t^{(k)} \boldsymbol{x}_t d_t, \tag{10}$$

where β is the forgetting factor [21] and $\boldsymbol{w}_{t+1}^{(k)} = \left(\boldsymbol{R}_{t+1}^{(k)}\right)^{-1} \boldsymbol{p}_{t+1}^{(k)}$ can be calculated in a recursive manner as

$$\begin{aligned}
e_t^{(k)} &= d_t - \boldsymbol{x}_t^T \boldsymbol{w}_t^{(k)}, \\
\boldsymbol{g}_t^{(k)} &= \frac{\lambda_t^{(k)} \boldsymbol{P}_t^{(k)} \boldsymbol{x}_t}{\beta + \lambda_t^{(k)} \boldsymbol{x}_t^T \boldsymbol{P}_t^{(k)} \boldsymbol{x}_t}, \\
\boldsymbol{w}_{t+1}^{(k)} &= \boldsymbol{w}_t^{(k)} + e_t^{(k)} \boldsymbol{g}_t^{(k)}, \\
\boldsymbol{P}_{t+1}^{(k)} &= \beta^{-1} \left(\boldsymbol{P}_t^{(k)} - \boldsymbol{g}_t^{(k)} \boldsymbol{x}_t^T \boldsymbol{P}_t^{(k)}\right),
\end{aligned} \tag{11}$$

where $\boldsymbol{P}_t^{(k)} \triangleq \left(\boldsymbol{R}_t^{(k)}\right)^{-1}$, and $\boldsymbol{P}_0^{(k)} = v^{-1} \boldsymbol{I}$, and $0 < v \ll 1$.

We can straightforwardly write the update equation for the boosted LMS filters as follows. Obviously by construction method in (6), $0 < \lambda_t^{(k)} \leq 1$, thus, these weights can be directly used to scale the learning rates for the LMS updates. When the kth CF receives the weight $\lambda_t^{(k)}$, it updates its coefficients $\boldsymbol{w}_t^{(k)}$, as

$$\boldsymbol{w}_{t+1}^{(k)} = \left(\boldsymbol{I} - \mu^{(k)} \lambda_t^{(k)} \boldsymbol{x}_t \boldsymbol{x}_t^T\right) \boldsymbol{w}_t^{(k)} + \mu^{(k)} \lambda_t^{(k)} \boldsymbol{x}_t d_t, \tag{12}$$

where $0 < \mu^{(k)} \lambda_t^{(k)} \leq \mu^{(k)}$. Note that we can choose $\mu^{(k)} = \mu$ for all k, since the online algorithms work consecutively from top to bottom, and the kth CF will have a different learning rate $\mu^{(k)} \lambda_t^{(k)}$.

5.2. Data reuse approaches based on the weights

Another approach follows Ozaboost [18]. In this approach, from $\lambda_t^{(k)}$, we generate an integer, say $n_t^{(k)} = \text{ceil}(K \lambda_t^{(k)})$, where K is a design parameter that takes on positive integer values. We then apply the RLS (or LMS) update on the $(\boldsymbol{x}_t, d_t)$ pair repeatedly $n_t^{(k)}$ times, i.e., run the RLS (or LMS) update on the same $(\boldsymbol{x}_t, d_t)$ pair $n_t^{(k)}$ times consecutively. Note that K should be determined according to the computational complexity constraints. However, increasing K does not necessarily result in a better performance, therefore, we use moderate values for K, e.g., we use $K = 5$ in our simulations. The final $\boldsymbol{w}_{t+1}^{(k)}$ is calculated after $n_t^{(k)}$ RLS (or LMS) updates. As a major advantage, clearly, this reusing approach can be readily generalized to other adaptive algorithms in a straightforward manner.

We point out that Ozaboost [18] uses a different data reuse strategy. In this approach, $\lambda_t^{(k)}$ is used as the parameter of a Poisson distribution and an integer $n_t^{(k)}$ is randomly generated from this Poisson distribution. One then applies the RLS (or LMS) update $n_t^{(k)}$ times.

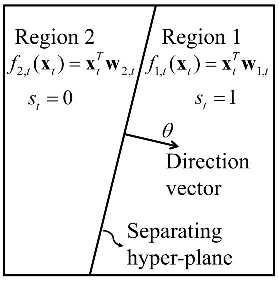

Region 2
$f_{2,t}(\mathbf{x}_t)=\mathbf{x}_t^T\mathbf{w}_{2,t}$
$s_t=0$
Region 1
$f_{1,t}(\mathbf{x}_t)=\mathbf{x}_t^T\mathbf{w}_{1,t}$
$s_t=1$
θ
Direction vector
Separating hyper-plane

Fig. 3. A sample 2-region partition of the input vector (i.e., $\boldsymbol{x}_t$) space, which is 2-dimensional in this example. s_t determines whether $\boldsymbol{x}_t$ is in Region 1 or not, hence, can be used as the indicator function for this region. Similarly, $1-s_t$ serves as the indicator function of Region 2.

5.3. Random updates approach based on the weights

In this approach, we simply use the weight $\lambda_t^{(k)}$ as a probability of updating the kth CF at time t. To this end, we generate a Bernoulli random variable, which is 1 with probability $\lambda_t^{(k)}$ and is 0 with probability $1-\lambda_t^{(k)}$. Then, we update the kth CF, only if the Bernoulli random variable equals 1. With this method, we significantly reduce the computational complexity of the algorithm. Moreover, due to the dependence of this Bernoulli random variable on the performance of the previous constituent CFs, this method does not degrade the MSE performance, while offering a considerably lower complexity, i.e., when the MSE is low, there is no need for further updates, hence, the probability of an update is low, while this probability is larger when the MSE is high.

6. Boosted piecewise linear adaptive filters

We use a piecewise linear adaptive filtering method, such that the desired signal is predicted as

$$\hat{d}_t=\sum_{i=1}^{N} s_{i,t}\boldsymbol{w}_{i,t}^T\boldsymbol{x}_t, \tag{13}$$

where $s_{i,t}$ is the indicator function of the ith region, i.e.,

$$s_{i,t}=\begin{cases}1 & \text{if } \boldsymbol{x}_t\in\mathcal{R}_i\\ 0 & \text{if } \boldsymbol{x}_t\notin\mathcal{R}_i.\end{cases} \tag{14}$$

Note that at each time t, only one of the $s_{i,t}$'s is nonzero, which indicates the region in which $\boldsymbol{x}_t$ lies. Thus, if $x_t\in\mathcal{R}_i$, we update only the ith linear filter. As an example, consider 2-dimensional input vectors $\boldsymbol{x}_t$, as depicted in Fig. 3. Here, we construct the piecewise linear filter f_t such that

$$\begin{aligned}\hat{d}_t=f_t(\boldsymbol{x}_t)&=s_{1,t}\boldsymbol{w}_{1,t}^T\boldsymbol{x}_t+s_{2,t}\boldsymbol{w}_{2,t}^T\boldsymbol{x}_t\\ &=s_t\boldsymbol{w}_{1,t}^T\boldsymbol{x}_t+(1-s_t)\boldsymbol{w}_{2,t}^T\boldsymbol{x}_t.\end{aligned} \tag{15}$$

Then, if $s_t=1$ we shall update $\boldsymbol{w}_{1,t}$, otherwise we shall update $\boldsymbol{w}_{2,t}$, based on the amount of the error, e_t. Note that one can use either LMS or RLS algorithm to update the linear filters in each region of a piecewise linear constituent filter.

As depicted in Fig. 4, each constituent filter is a piecewise linear filter consisting of N linear filters. At each time t, all of the constituent filters (shown in Fig. 1) estimate the desired data d_t in parallel, and the final estimate is a linear combination of the results generated by the constituent filters. However, at each time t, exactly one of the N linear filters in each constituent filter is used for estimating d_t. Correspondingly, when we update the constituent filters, only the filter that has been used for the estimation will be updated. To this end, we use the indicator function $s_{i,t}^{(k)}$ for the ith linear filter embedded in the kth constituent filter, as was explained before. Therefore, at each time t, only the filters whose indicator functions equal 1, will be updated. When the kth constituent filter receives the weigh $\lambda_t^{(k)}$, it updates the linear coefficients $\boldsymbol{w}_{i,t}^{(k)}$, assuming that $\boldsymbol{x}_t$ lies in the ith region of the kth constituent filter. We consider $\lambda_t^{(k)}$ as the weight for the observation pair $(\boldsymbol{x}_t, d_t)$ and apply a weighted RLS or LMS update to $\boldsymbol{w}_{i,t}^{(k)}$.

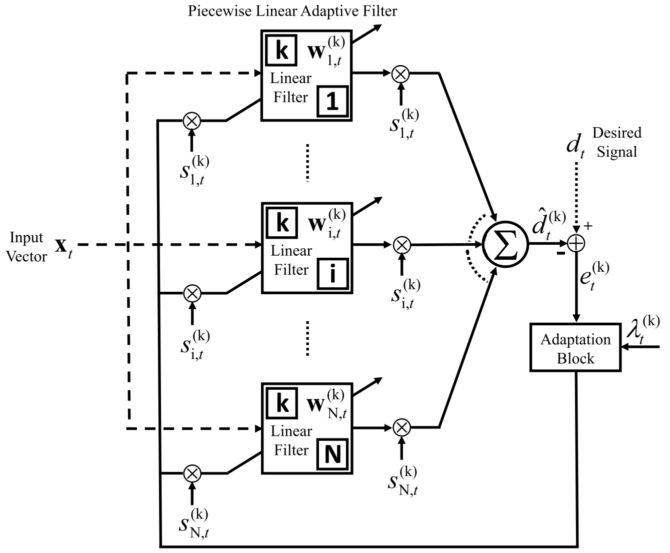


Fig. 4. A sample piecewise linear adaptive filter, used as the kth constituent filter in the system depicted in Fig. 1. This filter consists of N linear filters, one of which produces the estimate at each iteration t. Based on where the input vector at time t, $\boldsymbol{x}_t$, lies in the input vector space, one of the $s_{i,t}^{(k)}$'s is 1 and all others are 0. Hence, at each iteration only one of the linear filters is used for estimation and updated correspondingly.

Remark 1. We supposed that each constituent filter is built up based upon a fixed partition, which means that the partition cannot be updated during the algorithm. However, one can use a method similar to that in [65] to make the partitioning adaptive (i.e., soft boundaries). Refer to the Appendix D for a detailed explanation of this variant.

7. Analysis of the proposed algorithms

In this section, we provide the complexity analysis for the proposed algorithms. We prove an upper bound for the weights $\lambda_t^{(k)}$, which is significantly less than 1. This bound shows that the complexity of the "random updates" algorithm is significantly less than the other proposed algorithms and slightly greater than that of a single CF. Hence, it shows the considerable advantage of "boosting with random updates" in the processing of high dimensional data.

7.1. Complexity analysis

Here we compare the complexity of the proposed algorithms and find an upper bound for the computational complexity of random updates scenario, which shows its significantly lower computational burden with respect to two other approaches. For $\boldsymbol{x}_t\in\mathbb{R}^r$, each CF performs $O(r)$ computations to generates its estimate, and if updated using the RLS algorithm, requires $O(r^2)$ computations due to updating the matrix $\mathbf{R}_t^{(k)}$, while it needs $O(r)$ computations when updated using the LMS method (in their most basic implementation).

We first derive the computational complexity of using the RLS updates in different boosting scenarios. Since there are a total of m

CFs, all of which are updated in the "weighted updates" method, this method has a computational cost of order $O(mr^2)$ per each iteration t. However, in the "random updates", at iteration t, the kth CF may or may not be updated with probabilities $\lambda_t^{(k)}$ and $1-\lambda_t^{(k)}$ respectively, yielding

$$C_t^{(k)} = \begin{cases} O(r^2) & \text{with probability } \lambda_t^{(k)} \\ O(r) & \text{with probability } 1-\lambda_t^{(k)}, \end{cases} \tag{16}$$

where $C_t^{(k)}$ indicates the complexity of running the kth CF at iteration t. Therefore, the total computational complexity C_t at iteration t will be $C_t = \sum_{k=1}^m C_t^{(k)}$, which yields

$$E[C_t] = E\left[\sum_{k=1}^m C_t^{(k)}\right] = \sum_{k=1}^m E[\lambda_t^{(k)}] O(r^2) \tag{17}$$

Hence, if $E\big[\lambda_t^{(k)}\big]$ is upper bounded by $\tilde{\lambda}^{(k)} < 1$, the average computational complexity of the random updates method, will be

$$E[C_t] < \sum_{k=1}^m \tilde{\lambda}^{(k)} O(r^2). \tag{18}$$

In Theorem 3, we provide sufficient constraints to have such an upper bound.

Furthermore, we can use such a bound for the "data reuse" mode as well. In this case, for each CF $f_t^{(k)}$, we perform the RLS update $\lambda_t^{(k)} K$ times, resulting a computational complexity of order $E[C_t] < \sum_{k=1}^m K\ \tilde{\lambda}^{(k)}(O(r^2))$. For the LMS updates, we similarly obtain the computational complexities $O(mr)$, $\sum_{k=1}^m O\big(\tilde{\lambda}^{(k)} r\big)$, and $\sum_{k=1}^m O\big(K\tilde{\lambda}^{(k)} r\big)$, for the "weighted updates", "random updates", and "data reuse" scenarios respectively.

The following theorem determines the upper bound $\tilde{\lambda}^{(k)}$ for $E\big[\lambda_t^{(k)}\big]$.

Theorem 3. *If the CFs converge and achieve a sufficiently small MSE (according to the proof following this Theorem), the following upper bound is obtained for $\lambda_t^{(k)}$, given that σ_m^2 is chosen properly,*

$$E\left[\lambda_t^{(k)}\right] \le \tilde{\lambda}^{(k)} = \left(\gamma^{-2\sigma_m^2}(1 + 2\zeta^2 \ln \gamma)\right)^{\frac{1-k}{2}}, \tag{19}$$

where $\gamma \triangleq E\left[\delta_{t-1}^{(k)}\right]$ *and* $\zeta^2 \triangleq E\left[\left(e_t^{(k)}\right)^2\right]$.

It can be straightforwardly shown that, this bound is less than 1 for appropriate choices of σ_m^2, and reasonable values for the MSE according to the proof. This theorem states that if we adjust σ_m^2 such that it is achievable, i.e., the CFs can provide a slightly lower MSE than σ_m^2, the probability of updating the CFs in the random updates scenario will decrease. This is of course our desired result, since if the CFs are performing sufficiently well, there is no need for additional updates. Moreover, if σ_m^2 is opted such that the CFs cannot achieve a MSE equal to σ_m^2, the CFs have to be updated at each iteration, which increases the complexity.

Proof. Refer to Appendix E. □

The Table 1 summarizes the computational complexities of different approaches, where $\bar{\lambda} = 1/m \sum_{k=1}^m \tilde{\lambda}^{(k)}$.

Table 1
Computational complexity of different methods. BLMS/BRLS indicates boosted LMS/RLS filters.

Algorithms	WU	RU	DR
BLMS	$O(mr)$	$O(mr\bar{\lambda})$	$O(mr\bar{\lambda}K)$
BRLS	$O(mr^2)$	$O(mr^2\bar{\lambda})$	$O(mr^2\bar{\lambda}K)$

8. Experiments

In this section, we demonstrate the efficacy of the proposed boosting algorithms for the RLS and LMS linear, as well as piecewise linear, CFs under different scenarios. To this end, we first consider the "online regression" of data generated with a stationary linear model. Then, we illustrate the performance of our algorithms under nonstationary conditions, to thoroughly test the adaptation capabilities of the proposed boosting framework. Furthermore, since the most important parameters in the proposed methods are σ_m^2, c, and m, we investigate their effects on the final MSE performance. Finally, we provide the results of the experiments over several real and synthetic benchmark datasets.

Throughout this section, "LMS" represents the linear LMS-based CF, "RLS" represents the linear RLS-based CF, and a prefix "B" indicates the boosting algorithms. In addition, we use the suffixes "-WU", "-RU", or "-DR" to denote the "weighted updates", "random updates", or "data reuse" modes, respectively, e.g., the "BLMS-RU" represents the "Boosted LMS-based algorithm using Random Updates". Also, a prefix "P" before the "LMS" or "RLS" indicates a piecewise linear filter with two regions, based on the corresponding update method, and "SPLMS" denotes an LMS-based piecewise linear filter with "Soft" (flexible) boundaries.

In order to observe the boosting effect, in all experiments, we set the step size of LMS and the forgetting factor of the RLS to their optimal values (obtained by an exhaustive search), and use those parameters for the CFs, too. In addition, the initial values of all of the constituent filters in all of the experiments are set to zero. However, in all experiments, since we use $K = 5$ in Data Reuse variants, we set the step size of the CFs in BLMS-DR method to $\mu/K = \mu/5$, where, μ is the step size of the LMS. To compare the performance, we have provided the MSE results.

8.1. Stationary and non-stationary data

In this experiment, we consider the case where the desired data is generated by a stationary linear model. The input vectors $\boldsymbol{x}_t = [x_1\ x_2\ 1]$ are 3-dimensional, where $[x_1\ x_2]$ is drawn from a jointly Gaussian random process and then scaled such that $[x_1\ x_2]^T \in [0\ 1]^2$. We include 1 as the third entry of $\boldsymbol{x}_t$ to consider affine filters. Specifically the desired data is generated by $d_t = [1\ 1\ 0]\ \boldsymbol{x}_t + \nu_t$, where ν_t is a random Gaussian noise with a variance of 0.01. Moreover, in the non-stationary scenario, we have

$$d_t = \begin{cases} [1\ 1\ 0]\,\boldsymbol{x}_t + \nu_t, & t \le T/2 \\ [1\ {-1}\ 0]\,\boldsymbol{x}_t + \nu_t, & \text{O.W.} \end{cases}$$

In our simulations, we use $m = 20$ CFs and $\mu = 0.1$ for all LMS learners. In addition, for RLS-based boosting algorithms, we set the forgetting factor $\beta = 0.9999$ for all algorithms. Moreover, we choose $\sigma_m^2 = 0.02$ for LMS-based algorithms and $\sigma_m^2 = 0.004$ for RLS-based algorithms, $K = 5$ for data reuse approaches, and $c = 1$ for all boosting algorithms. To achieve robustness, we average the results over 100 trials.

As depicted in Fig. 5, our proposed methods boost the performance of a single linear LMS-based CF. Nevertheless, we cannot further improve the performance of a linear RLS-based CF in such

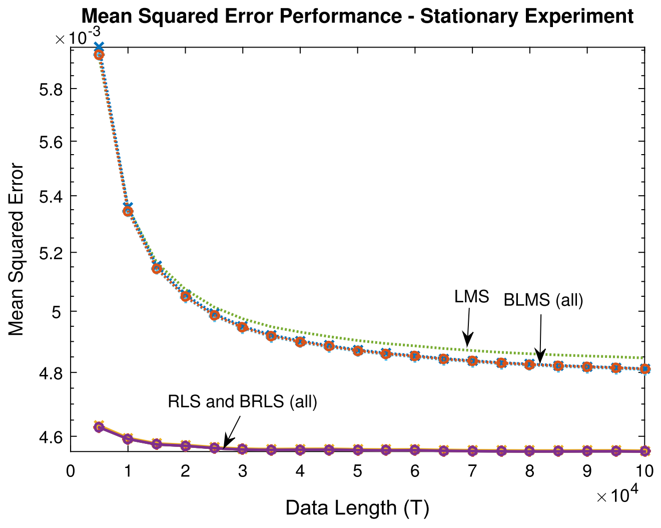


Fig. 5. The MSE performance of the proposed algorithms in the stationary data experiment.

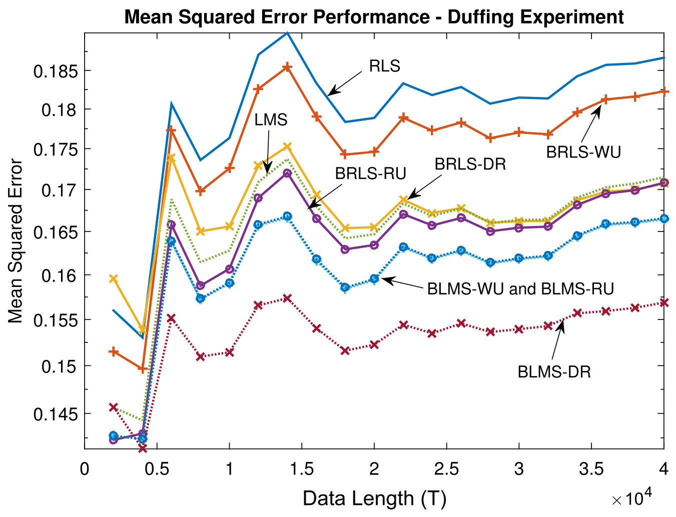


Fig. 7. MSE performance of the proposed linear methods on a Duffing data set.

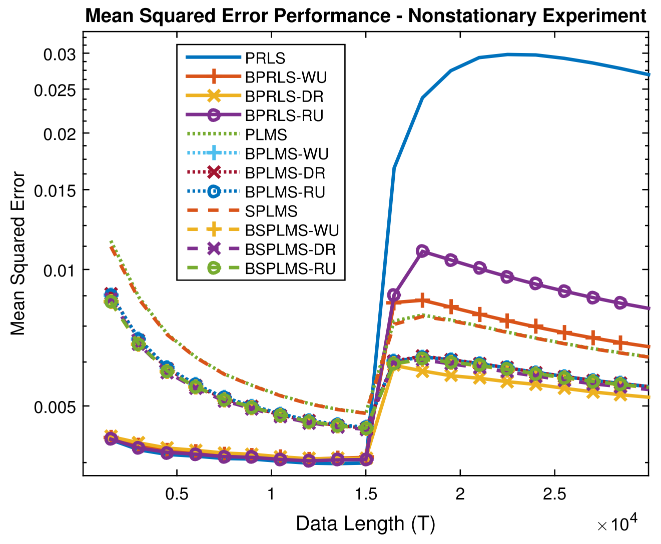


Fig. 6. The MSE performance of the piecewise linear filters in the non-stationary data experiment. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

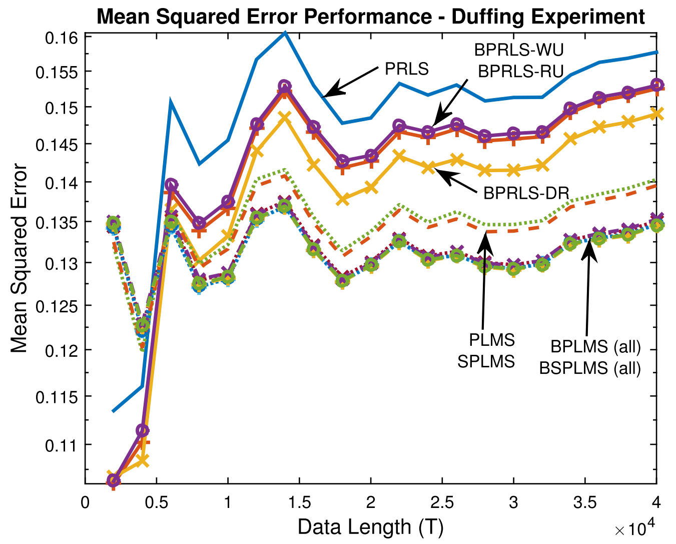


Fig. 8. MSE performance of the proposed piecewise linear methods on a Duffing data set.

a stationary experiment since the RLS achieves the lowest MSE. We point out that the random updates method achieves the performance of the weighted updates method and the data reuse method with a much lower complexity. In addition, we observe that by increasing the data length, the performance improvement increases. (Note that the distance between the MSE curves is slightly increasing.) In addition, according to Fig. 6, our piecewise linear methods significantly outperform a single constituent filter, even in RLS-based filters. As indicated in Fig. 6, the boosted approaches are relatively faster than the constituent filters. In fact, the constituent filters use fixed parameters, while our boosting method seeks to adapt these parameters in an online manner, which results in a faster reaction to the non-stationarity. Note that although it is possible to get a faster reaction by changing the forgetting factor of the RLS algorithm, it yields a higher MSE.

8.2. Chaotic data

Here, in order to show the tracking capability of our algorithms in nonstationary environments, we consider the case where the desired data is generated by the Duffing map [66] as a chaotic model. Specifically, the data is generated by the following equation $x_{t+1} = 2.75x_t - x_t^3 - 0.2x_{t-1}$, where we set $x_{-1} = 0.9279$ and $x_0 = 0.1727$. We consider $d_t = x_{t+1}$ as the desired data and $[x_{t-1}\ x_t\ 1]$ as the input vector. In this experiment, each boosting algorithm uses 20 CFs. The step sizes for the LMS-based algorithms are set to 0.1, the forgetting factor β for the RLS-based algorithms are set to 0.999, and the modified desired MSE parameter σ_m^2 is set to 0.25 for BLMS methods, and 0.17 for the BRLS methods. Note that although the value of σ_m^2 is higher than the achieved MSE, it can improve the performance significantly. This is because of the boosting effect, i.e., emphasizing on the harder data patterns. The figures show the superior performance of our algorithms over a single CF (whose step size is chosen to be the best), in this highly nonstationary environment. Moreover, as shown in Figs. 7 and 8, in the LMS-based boosted algorithms, the data reuse method shows a better performance relative to the other boosting methods. However, the random updates method has a significantly lower time consumption, which makes it desirable for larger data lengths. From Figs. 7 and 8, one can see that our method is truly boosting the performance of the conventional linear CFs in this chaotic environment.

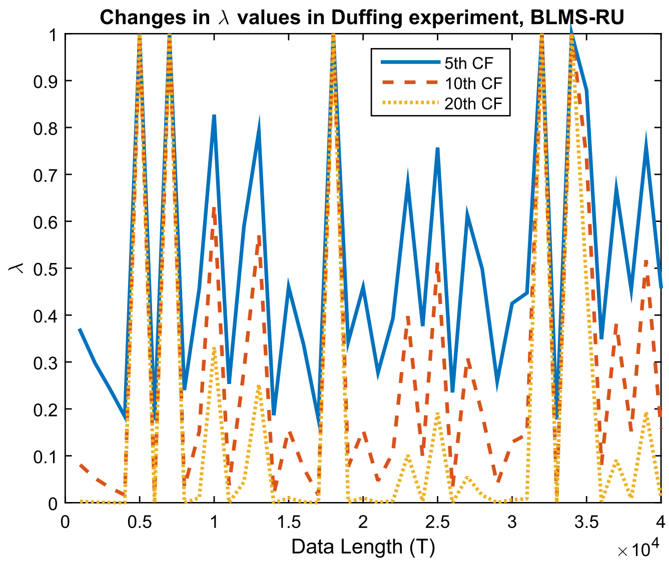


Fig. 9. The changing of the weights in BLMS-RU algorithm in the Duffing data experiment.

Table 2
The MSE of the LMS-based methods on real data sets.

Data sets \ Algorithms	LMS	BLMS-WU	BLMS-DR	BLMS-RU
MV	0.2711	0.2707	0.2706	0.2707
Puma8NH	0.1340	0.1334	0.1332	0.1334
Kinematics	0.0835	0.0831	0.0830	0.0831
Compactiv	0.0606	0.0599	0.0608	0.0598
Protein Tertiary	0.2554	0.2550	0.2549	0.2550
ONP	0.0015	0.0009	0.0009	0.0009
California Housing	0.0446	0.0450	0.0452	0.0448
YPMSD	0.0237	0.0237	0.0233	0.0237

Table 3
The MSE of the RLS-based methods on real data sets.

Data sets \ Algorithms	RLS	BRLS-WU	BRLS-DR	BRLS-RU
MV	0.2592	0.2645	0.2587	0.2584
Puma8NH	0.1296	0.1269	0.1295	0.1284
Kinematics	0.0804	0.0801	0.0803	0.0801
Compactiv	0.0137	0.0086	0.0304	0.0078
Protein Tertiary	0.2370	0.2334	0.2385	0.2373
ONP	0.0009	0.0009	0.0009	0.0009
California Housing	0.0685	0.0671	0.0579	0.0683
YPMSD	0.0454	0.0337	0.0302	0.0292

From Fig. 9, we observe the approximate changes of the weights, in the BLMS-RU algorithm running over the Duffing data. As shown in this figure, the weights do not change monotonically, and this shows the capability of our algorithm in terms of effective tracking of the nonstationary data. Furthermore, since we update the CFs in an ordered manner, i.e., we update the $(k+1)$th CF after the kth CF is updated, the weights assigned to the last CFs are generally smaller than the weights assigned to the previous CFs. As an example, in Fig. 9, we see that the weights assigned to the 5th CF are larger than those of the 10th and 20th CFs. Furthermore, note that in this experiment, the dependency parameter c is set to 1. We should mention that increasing the value of this parameter, in general, causes the lower weights, hence, it can considerably reduce the complexity of the random updates and data reuse methods.

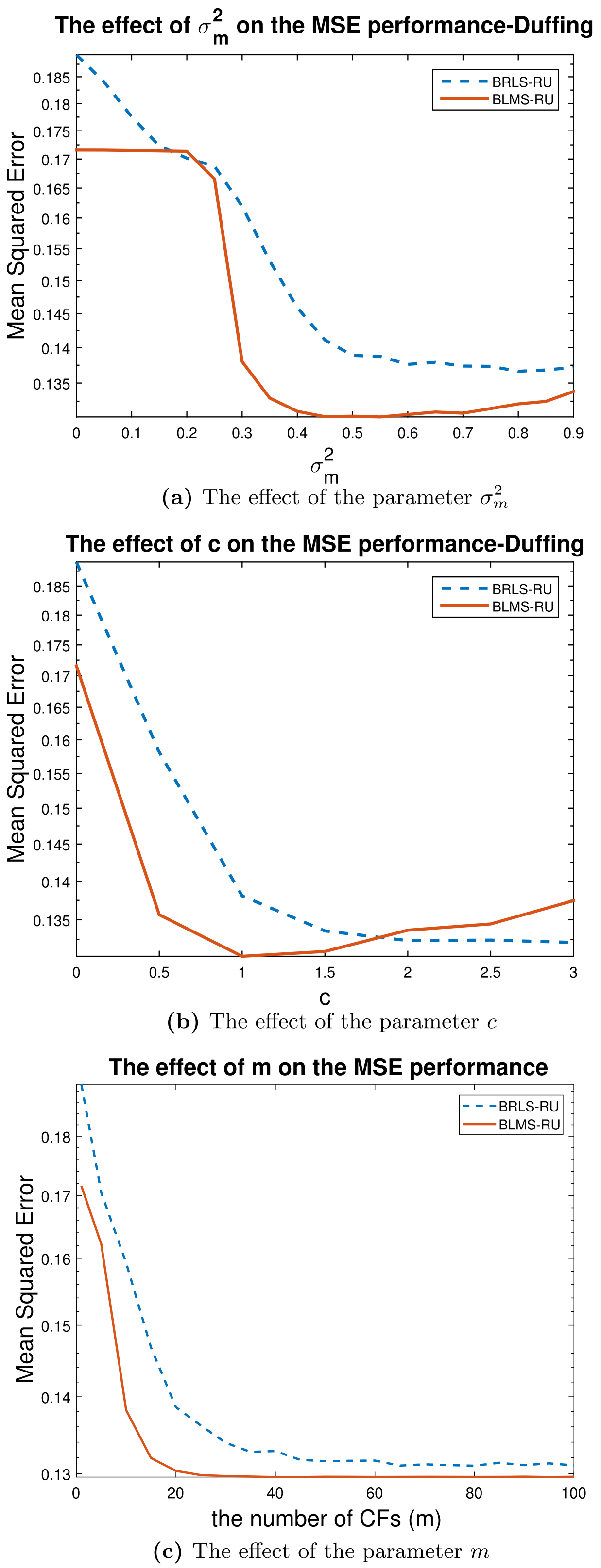
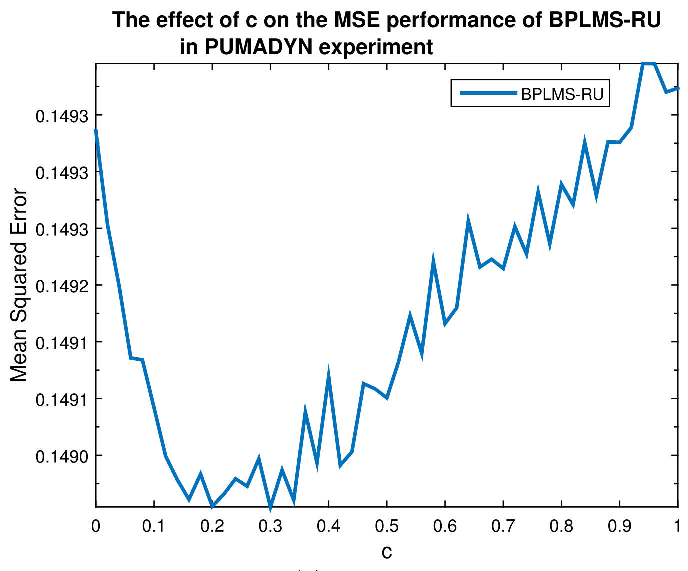
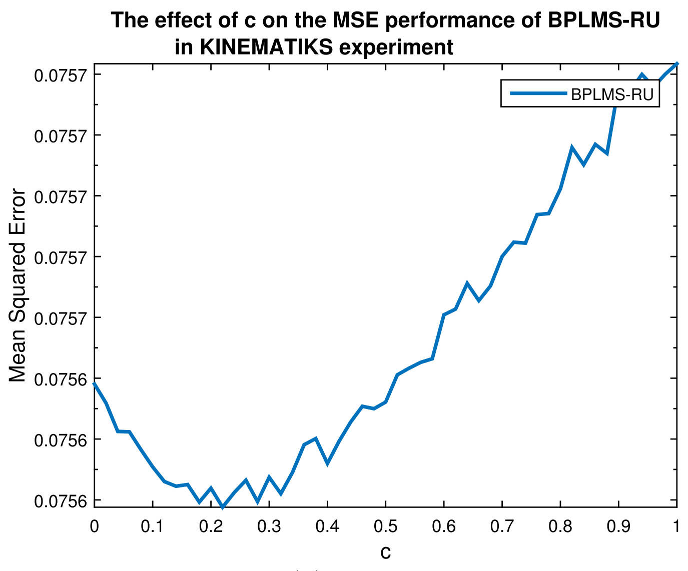


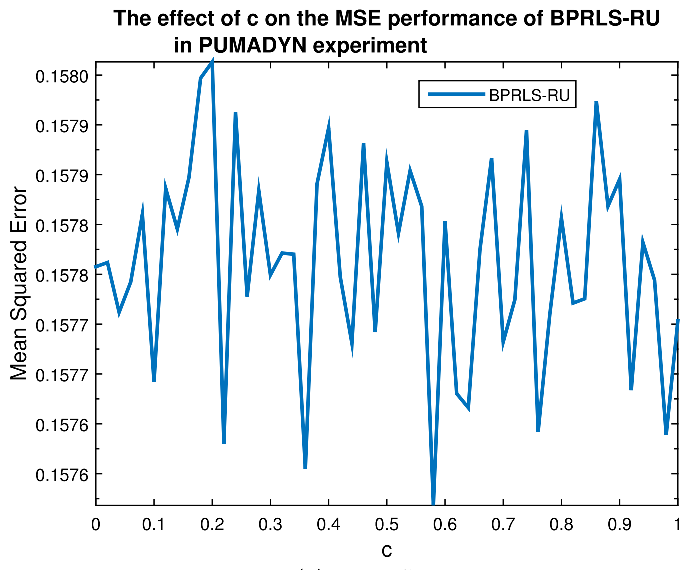
Fig. 10. The effect of the parameters σ_m^2, c, and m, on the MSE performance of the BRLS-RU and BLMS-RU algorithms in the Duffing data experiment.



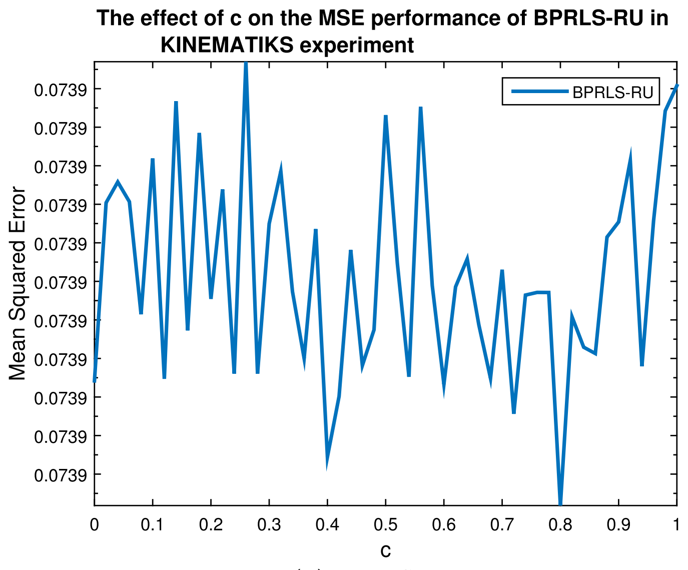
(a) BPLMS-RU



(b) BPLMS-RU



(c) BPRLS-RU



(d) BPRLS-RU

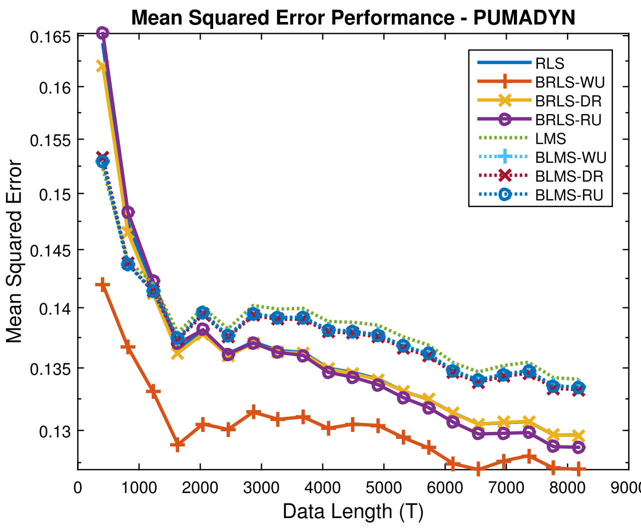
Fig. 11. The effect of the dependency parameter on the performance of BPLMS-RU and BPRLS-RU in the Puma8NH and Kinematics experiments.

8.3. The effect of parameters

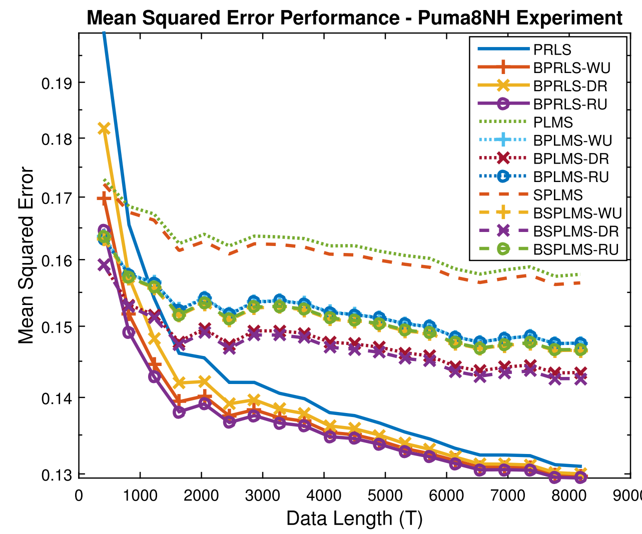
In this section, we investigate the effects of the dependence parameter c and the modified desired MSE σ_m^2 as well as the number of CFs, m, on the boosting performance of our methods in the Duffing data experiment, explained in Section 8.2. From the results in Fig. 10c, we observe that increasing the number of CFs up to 30 can improve the performance significantly, while further increasing of m only increases the computational complexity without improving the performance. In addition, as shown in Fig. 10b, in this experiment, the dependency parameter c has an optimum value around 1. We note that choosing small values for c reduces the boosting effect, and causes the weights to be larger, which in turn increases the computational complexity of random updates and data reuse approaches. On the other hand, choosing very large values for c increases the dependency, i.e., in this case, the generated weights are very close to 1 or 0, hence, the boosting effect is decreased.

According to Figs. 10b, 11b, 11d, 11a, and 11c, we observe that the MSE curve has a minimum at a nonzero point. We emphasize that the performance improvement due to increasing c from 0, shows that our method truly boosts the adaptive filters. However, one may note the difference between the boosting effect in these scenarios. Since the ensemble of the piecewise linear filters has a slight diversity among them, the diversity risen from the boosting (that finally yields an improvement in the MSE performance) is less in these cases.

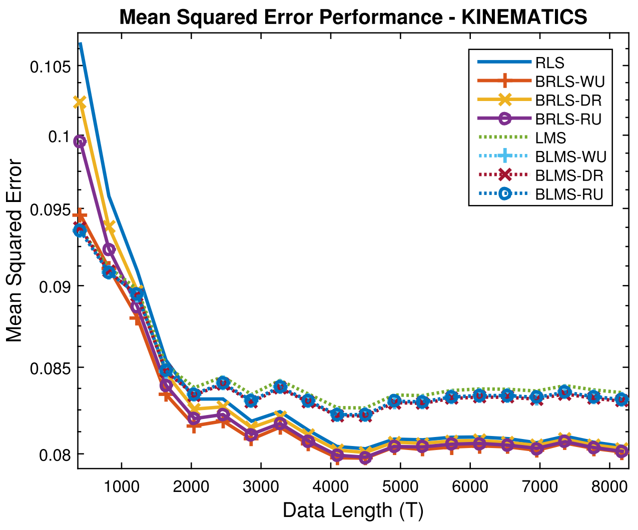
Furthermore, as depicted in Fig. 10a, there is an optimum value around 0.5 for σ_m^2 in this experiment. Note that, choosing small values for σ_m^2 results in large weights, thus, increases the complexity and reduces the diversity. However, choosing higher values for σ_m^2 results in smaller weights, and in turn, reduces the complexity. Nevertheless, we note that increasing the value of σ_m^2 does not necessarily enhance the performance. Through the experiments, we find out that σ_m^2 must be in the order of the MSE amount to obtain the best performance.



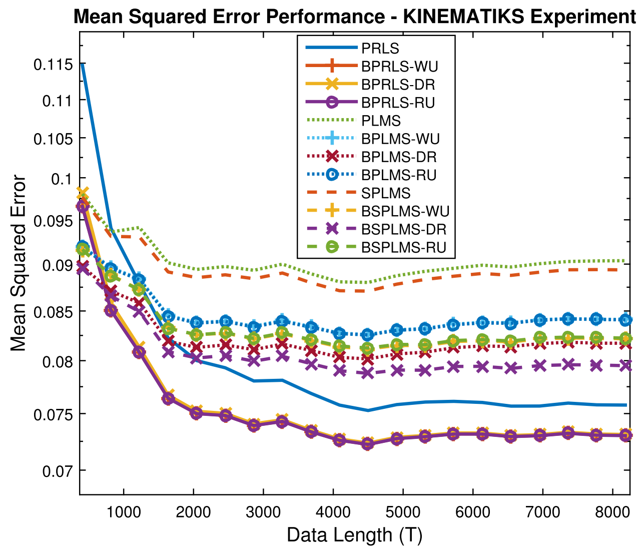
(a) Puma8NH



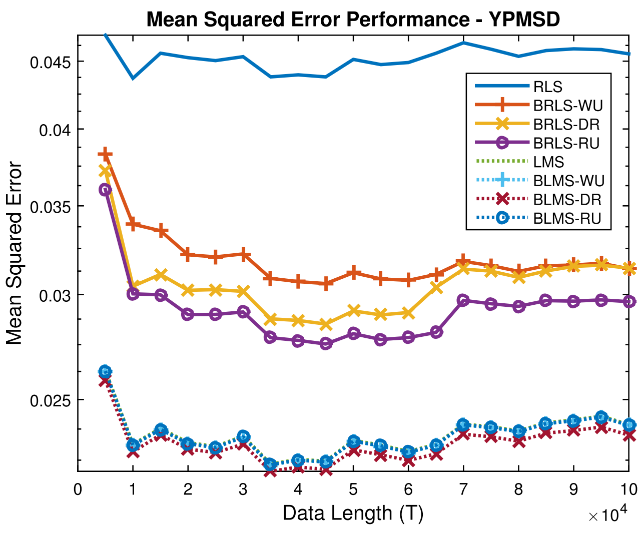
(b) Puma8NH



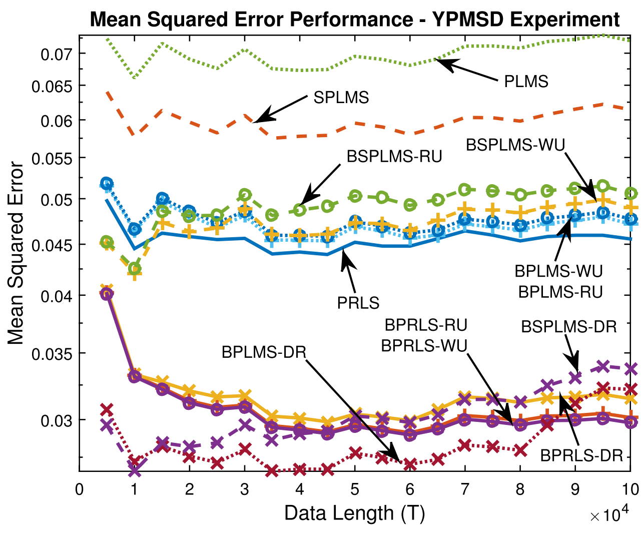
(c) Kinematics.



(d) Kinematics.



(e) YPMSD



(f) YPMSD

Fig. 12. The performance of the boosting methods on three real life data sets.

8.4. Benchmark real and synthetic data sets

In this section, we demonstrate the efficiency of the introduced methods over some widely used real-life machine learning regression data sets. We have normalized each dimension of the data to the interval $[-1, 1]$ in all algorithms. We present the MSE performance of the algorithms in Tables 2 and 3. These experiments show that our algorithms can successfully improve the performance of linear and piecewise linear CFs. The experiment descriptions and results are provided as follows.

Description of the datasets:

1. MV: This is an artificial dataset with dependencies between the attribute values. One can refer to [67] for further details. There are 10 attributes and one target value. In this dataset, we can slightly improve the performance of a single linear CF by using any of the proposed methods.
2. Puma Dynamics (Puma8NH): This dataset describes a realistic modeling of the dynamics of a robot arm, called Puma 560 [67], where we seek to estimate the angular acceleration of one of the robot arm's links. To this end, we use the input features consisting of angular positions, velocities and torques of the robot arm. According to the MSE results in Fig. 12a, the BRLS-WU has the best boosting performance in this experiment. Nonetheless, the LMS-based methods also improve the performance. In addition, as shown in Fig. 12b, we have also successfully boosted the piecewise methods.
3. Kinematics: This dataset is concerned with the forward kinematics of an 8 link robot arm [67]. We use the variant 8RLS, which is highly non-linear and noisy. As shown in Fig. 12c, our proposed algorithms slightly improve the performance in this experiment. Nevertheless, there is a significant improvement in the piecewise linear case, as depicted in Fig. 12d.
4. Computer Activity (Compactiv): This real dataset is a collection of computer systems activity measures [67]. The task is to predict USR, the portion of time that CPUs run in user mode from all attributes [67]. The RLS-based boosting algorithms deliver a significant performance improvement in this experiment, as shown by the results in Tables 2 and 3.
5. Protein Tertiary [68]: Having been collected from the Critical Assessment of protein Structure Prediction (CASP) experiments 5–9, the 45730 data samples of this dataset are used to estimate the residue size using the 9 given attributes.
6. Online News Popularity (ONP) [68,69]: This dataset consists of a heterogeneous features set regarding some articles that were published by Mashable in two consecutive years. We seek to estimate the total number of shares in social networks, which in turn shows the popularity of the articles.
7. California Housing: This dataset has been obtained from StatLib repository. They have collected information on the variables using all the block groups in California from the 1990 Census. Here, we seek to find the house median values, based on the given attributes. For further description, one can refer to [67].
8. Year Prediction Million Song Dataset (YPMSD) [70]: The aim is predicting the year when a song has been released, using its given audio features. This dataset mainly includes western commercial song tracks released between 1922 and 2011. We use a subset of the Million Song Dataset [70]. As shown in Tables 2 and 3 and Figs. 12e and 12f, our algorithms can significantly improve the performance of the linear and piecewise linear CFs in this experiment.

9. Conclusion

We introduced a novel family of boosted adaptive filtering algorithms and proposed three different boosting approaches, i.e., weighted updates, data reuse, and random updates, which can be applied to different adaptive filters. We provide theoretical bounds for the MSE performance of our proposed methods in a strong mathematical sense. While using the proposed techniques, we consider only mild widely used assumptions about the statistics of the desired data or feature vectors. As shown in the experiments, by the proposed boosting approaches, we can significantly improve the MSE performance of the conventional LMS and RLS algorithms over a wide variety of synthetic as well as real data. Moreover, we provide an upper bound for the weights generated by the algorithm that leads us to a thorough analysis of the computational complexity of these methods. The computational complexity of the random updates method is remarkably lower than that of the conventional mixture-of-experts and other variants of the proposed boosting approaches, without degrading the performance. Therefore, the boosting using random updates approach is an elegant alternative to the conventional mixture-of-experts method when dealing with real life large scale problems.

Acknowledgments

This work is supported in part by Turkish Academy of Sciences Outstanding Researcher Programme, TUBITAK Contract No. 113E517, and Turk Telekom Communications Services Incorporated.

Appendix A. Proof of Theorem 1

According to [58], if we use online gradient descent algorithm with the step sizes η_t, we reach the following upper bound on the regret of the online algorithm with respect to the best offline algorithm (which uses the constant vector $\boldsymbol{w}^*$).

$$\sum_{t=1}^{T} \lambda_t \left(e_t^2(\boldsymbol{w}_t) - e_t^2(\boldsymbol{w}^*) \right)$$
$$\leq \sum_{t=1}^{T} \lambda_t \|\boldsymbol{w}_t - \boldsymbol{w}^*\|^2 \left(\frac{1}{\eta_{t+1}} - \frac{1}{\eta_t} - \xi \right) + \beta^2 \sum_{t=1}^{T} \lambda_t \eta_{t+1}. \quad \text{(A.1)}$$

Also, by mathematical induction, it can be shown that if $0 \leq \lambda_t \leq 1$ and $\lambda_1 = 1$, we have

$$\sum_{t=1}^{T} \frac{\lambda_t}{\sum_{i=1}^{t} \lambda_i} \leq 1 + \ln \sum_{t=1}^{T} \lambda_t.$$

Hence, by choosing $\eta_{t+1} \triangleq \frac{1}{\xi \sum_{i=1}^{t} \lambda_i}$, it is straightforward to show that

$$\sum_{t=1}^{T} \lambda_t \left(e_t^2(\boldsymbol{w}_t) - e_t^2(\boldsymbol{w}^*) \right) \leq \frac{\beta^2}{\xi} \left(1 + \ln \sum_{t=1}^{T} \lambda_t \right). \quad \text{(A.2)}$$

Now, by dividing both sides by $4 \sum_{t=1}^{T} \lambda_t$, and taking into account the Assumption in (2), we observe that

$$1 + \ln \sum_{t=1}^{T} \lambda_t \leq \left(\frac{4\epsilon\xi\sigma_{\text{off}}^2}{\beta^2}\right) \sum_{t=1}^{T} \lambda_t,$$

or equivalently,

$$\frac{\beta^2}{4\xi \sum_{t=1}^{T} \lambda_t} \left(1 + \ln \sum_{t=1}^{T} \lambda_t \right) \leq \epsilon \sigma_{\text{off}}^2$$

This concludes the proof of Theorem 1. □

Appendix B. Proof of (1)

Based on the ξ-strong convexity of $\ell_t(x)$ and for any pair x and y, we have

$$\ell_t(y) \geq \ell_t(x) + \nabla\ell_t(x)(y-x) + \frac{\xi}{2}||y-x||^2.$$

For $y = x^* = \text{argmin}_x \ell_t(x)$, the above definition can be written as

$$\begin{aligned}
&\ell_t(x^*) \geq \ell_t(x) + \nabla\ell_t(x)(x^*-x) + \frac{\xi}{2}||x^*-x||^2\\
&2\xi\ell_t(x^*) \geq 2\xi\ell_t(x) + 2\xi\nabla\ell_t(x)(x^*-x) + \xi^2||x^*-x||^2\\
&2\xi\left(\ell_t(x) - \ell_t(x^*)\right) \leq -2\xi\nabla\ell_t(x)(x^*-x) - \xi^2||x^*-x||^2\\
&2\xi\left(\ell_t(x) - \ell_t(x^*)\right)\\
&\quad \leq ||\nabla\ell_t(x)||^2 - ||\nabla\ell_t(x)||^2 - 2\xi\nabla\ell_t(x)(x^*-x) - \xi^2||x^*-x||^2\\
&2\xi\left(\ell_t(x) - \ell_t(x^*)\right) \leq ||\nabla\ell_t(x)||^2 - ||\nabla\ell_t(x) + \xi(x^*-x)||^2\\
&2\xi\left(\ell_t(x) - \ell_t(x^*)\right) \leq ||\nabla\ell_t(x)||^2 \quad \square
\end{aligned}$$

Appendix C. Proof of Theorem 2

We relate the total regret of the ensemble of the first k $(1 \leq k \leq m)$ constituent filters, Δ_k, with the regret of the ensemble of the first $k+1$ constituent filters, Δ_{k+1}, and show that Δ_{k+1} shrinks Δ_k by a constant fraction while only adding a small regret. We have

$$\Delta_k = \sum_{t=1}^{T} \ell_t(\hat{d}_t^{(k)}) - \sum_{t=1}^{T} \ell_t(f^*)$$

where $\hat{d}_t^{(k)} = \boldsymbol{x}_t^T \boldsymbol{w}_t^{(k)}$. By using the LMS algorithm to update $\boldsymbol{w}_t^{(k)}$, the above equation becomes

$$\hat{d}_t^{(k)} = \boldsymbol{x}_t^T\left(\boldsymbol{w}_{t-1}^{(k)} - \mu\lambda_t^{(k)}\boldsymbol{x}_t\boldsymbol{x}_t^T\boldsymbol{w}_{t-1}^{(k)} + \mu\lambda_t^{(k)}\boldsymbol{x}_t d_t\right).$$

We define $\boldsymbol{\epsilon}_t^{(k)}$ as

$$\boldsymbol{\epsilon}_t^{(k)} = \boldsymbol{w}_{t-1}^{(k)} - \boldsymbol{w}_t^{(k-1)}.$$

Therefore,

$$\begin{aligned}
\hat{d}_t^{(k)} &= \boldsymbol{x}_t^T\left(\boldsymbol{\epsilon}_t^{(k)} + \boldsymbol{w}_t^{(k-1)} - \mu\lambda_t^{(k)}\boldsymbol{x}_t\boldsymbol{x}_t^T(\boldsymbol{\epsilon}_t^{(k)} + \boldsymbol{w}_t^{(k-1)}) + \mu\lambda_t^{(k)}\boldsymbol{x}_t d_t\right)\\
&= \hat{d}_t^{(k-1)} + Q_t^{(k)}(1-\mu\lambda_t^{(k)}P_t) - \mu\lambda_t^{(k)}P_t(\hat{d}_t^{(k-1)} - d_t),
\end{aligned}$$

where $P_t = \boldsymbol{x}_t^T\boldsymbol{x}_t$ and $Q_t^{(k)} = \boldsymbol{x}_t^T\boldsymbol{\epsilon}_t^{(k)}$, are assumed to have bounded norms. Now, we have

$$\begin{aligned}
\Delta_k = \sum_{t=1}^{T}\Big[&\ell_t\Big(\hat{d}_t^{(k-1)} + Q_t^{(k)}(1-\mu\lambda_t^{(k)}P_t)\\
&-\mu\lambda_t^{(k)}P_t(\hat{d}_t^{(k-1)} - d_t)\Big) - \ell_t(f^*)\Big].
\end{aligned}$$

By using β-smoothness of $\ell_t(\cdot)$, we obtain

$$\begin{aligned}
\Delta_k \leq \sum_{t=1}^{T}\Big[&\ell_t\Big(\hat{d}_t^{(k-1)} + Q_t^{(k)}(1-\mu\lambda_t^{(k)}P_t)\Big)\\
&-\mu\lambda_t^{(k)}P_t\nabla\ell_t\Big(\hat{d}_t^{(k-1)} + Q_t^{(k)}(1-\mu\lambda_t^{(k)}P_t)\Big)\Big(\hat{d}_t^{(k-1)} - d_t\Big)\\
&+(\mu\lambda_t^{(k)}P_t)^2\frac{\beta}{2}||\hat{d}_t^{(k-1)} - d_t||^2 - \ell_t(f^*)\Big]. \qquad \text{(C.1)}
\end{aligned}$$

Considering that $\ell_t(\hat{d}_t) = \|d_t - \hat{d}_t\|^2$, we have

$$\begin{aligned}
\Delta_k \leq \sum_{t=1}^{T}\Big[&\|\hat{d}_t^{(k-1)} - d_t\|^2 + \|Q_t^{(k)}(1-\mu\lambda_t^{(k)}P_t)\|^2\\
&+2\Big((\hat{d}_t^{(k-1)} - d_t)Q_t^{(k)}(1-\mu\lambda_t^{(k)}P_t)\Big)\\
&-2\mu\lambda_t^{(k)}P_t\Big(\hat{d}_t^{(k-1)} - d_t + Q_t^{(k)}(1-\mu\lambda_t^{(k)}P_t)\Big)\\
&\times(\hat{d}_t^{(k-1)} - d_t) + (\mu\lambda_t^{(k)}P_t)^2\frac{\beta}{2}\|\hat{d}_t^{(k-1)} - d_t\|^2 - \ell_t(f^*)\Big].
\end{aligned}$$

By simplifying the right hand side of the previous inequality and noting that $\beta = 2$ for our special choice of the loss function, we obtain

$$\begin{aligned}
\Delta_k \leq \sum_{t=1}^{T}\Big[&(1-\mu\lambda_t^{(k)}P_t)^2\|\hat{d}_t^{(k-1)} - d_t\|^2 + \|Q_t^{(k)}(1-\mu\lambda_t^{(k)}P_t)\|^2\\
&+2\Big((\hat{d}_t^{(k-1)} - d_t)Q_t^{(k)}(1-\mu\lambda_t^{(k)}P_t)^2\Big) - \ell_t(f^*)\Big].
\end{aligned}$$

We further assume that $(1-\mu\lambda_t^{(k)}P_t)^2 \leq D_P \leq 1$ (which is valid when the data are bounded) and also $\|Q_t^{(k)}\|^2 \leq D_Q$. Therefore, it can be shown that

$$\begin{aligned}
\Delta_k \leq\ & D_P\Delta_{k-1} + D_P D_Q T\\
&+\sum_{t=1}^{T}\Big[2\Big((\hat{d}_t^{(k-1)} - d_t)Q_t^{(k)}(1-\mu\lambda_t^{(k)}P_t)^2\Big)\Big].
\end{aligned}$$

Note that $\sum_{t=1}^{T}(\hat{d}_t^{(k-1)} - d_t) \leq \sum_{t=1}^{T}\|\hat{d}_t^{(k-1)} - d_t\| \leq \sqrt{T\sum_{t=1}^{T}\|\hat{d}_t^{(k-1)} - d_t\|^2}$. Hence, according to Definition 3, we achieve

$$\Delta_k \leq D_P\Delta_{k-1} + D_P D_Q T + R'(T), \qquad \text{(C.2)}$$

where $R'(T) \triangleq 2D_P\sqrt{D_Q T}\sqrt{(1-\gamma)\sum_{t=1}^{T}||d_t||^2 + R(T)}$ (see Definition 3). Based on (C.2), we obtain

$$\begin{aligned}
\Delta_m &\leq D_P^m\Delta_0 + \Big(R'(T) + D_P D_Q T\Big)\sum_{i=1}^{m}D_P^{i-1}\\
&= D_P^m\Delta_0 + \Big(R'(T) + D_P D_Q T\Big)\frac{1-D_P^m}{1-D_P}\\
&\leq D_P^m\Delta_0 + \Big(R'(T) + D_P D_Q T\Big)\frac{1}{1-D_P}, \qquad \text{(C.3)}
\end{aligned}$$

which completes the proof of the theorem. $\square$

Appendix D. Soft (adaptive boundaries) version of the boosted piecewise linear adaptive filters

As an example, suppose that each constituent filter is defined on a 2-region partition, as shown in Fig. 3, the regions of which are separated using a hyperplane with the direction vector $\boldsymbol{\theta}_t^{(k)}$, which is going to be updated at each time t. In order to boost the performance of a system made up of N such piecewise linear filters, we not only apply the weights effects to update the linear filters updates in each region of each constituent filter but also update the direction vectors $\boldsymbol{\theta}_t^{(k)}$ in a boosted manner. In order to indicate the region in which $\boldsymbol{x}_t$ lies, we use an indicator function $s_t^{(k)}$ defined as follows

$$s_t^{(k)} = \frac{1}{1+\exp(-\boldsymbol{\theta}_t^T\boldsymbol{x}_t)}, \qquad \text{(D.1)}$$

and the estimate made by the kth filter is represented by

$$\hat{d}_t^{(k)} = s_t^{(k)}\hat{d}_{1,t}^{(k)} + \left(1 - s_t^{(k)}\right)\hat{d}_{2,t}^{(k)}, \quad \text{(D.2)}$$

which, yields the following ordinary LMS update for $\boldsymbol{\theta}_t^{(k)}$ [65]

$$\begin{aligned}\boldsymbol{\theta}_{t+1}^{(k)} &= \boldsymbol{\theta}_t^{(k)} + \mu_{\boldsymbol{\theta}} e_t^{(k)}\left(\hat{d}_{1,t}^{(k)} - \hat{d}_{2,t}^{(k)}\right)\nabla_{\boldsymbol{\theta}_t}\left(s_t^{(k)}\right)\\ &= \boldsymbol{\theta}_t^{(k)} + \mu_{\boldsymbol{\theta}} e_t^{(k)}\left(\hat{d}_{1,t}^{(k)} - \hat{d}_{2,t}^{(k)}\right)s_t^{(k)}\left(1 - s_t^{(k)}\right)\boldsymbol{x}_t. \quad \text{(D.3)}\end{aligned}$$

Then, in “random updates” scenario we either will or will not perform this update with probabilities $\lambda_t^{(k)}$ and $1 - \lambda_t^{(k)}$, respectively, and for “weighted updates” scenario we achieve the following update for $\boldsymbol{\theta}_t^{(k)}$

$$\boldsymbol{\theta}_{t+1}^{(k)} = \boldsymbol{\theta}_t^{(k)} + \mu_{\boldsymbol{\theta}}\lambda_t^{(k)} e_t^{(k)}\left(\hat{d}_{2,t}^{(k)} - \hat{d}_{1,t}^{(k)}\right)s_t^{(k)}\left(1 - s_t^{(k)}\right)\boldsymbol{x}_t. \quad \text{(D.4)}$$

However, for the “data reuse” scenario, we perform this update $n_t^{(k)} = \text{ceil}(\lambda_t^{(k)}K)$ times, along with updating the linear filters coefficients, which results in

$$\begin{aligned}\boldsymbol{\vartheta}^{(a+1)} &= \boldsymbol{\vartheta}^{(a)} + \mu_{\boldsymbol{\theta}}\epsilon^{(a)}\boldsymbol{x}_t\boldsymbol{x}_t^T\left(\boldsymbol{q}_1^{(a)} - \boldsymbol{q}_2^{(a)}\right)\psi^{(a)}\left(1 - \psi^{(a)}\right),\\ \boldsymbol{q}_1^{(a+1)} &= \boldsymbol{q}_1^{(a)} + \mu_i^{(k)}\psi^{(a)}\epsilon^{(a)}\boldsymbol{x}_t,\\ \boldsymbol{q}_2^{(a+1)} &= \boldsymbol{q}_2^{(a)} + \mu_i^{(k)}(1 - \psi^{(a)})\epsilon^{(a)}\boldsymbol{x}_t,\\ \psi^{(a+1)} &= \frac{1}{1 + \exp(-\boldsymbol{\vartheta}_t^T\boldsymbol{x}_t)},\\ \epsilon^{(a+1)} &= d_t - \left(\psi^{(a+1)}\boldsymbol{q}_1^{(a+1)} + (1 - \psi^{(a+1)})\boldsymbol{q}_2^{(a+1)}\right)\boldsymbol{x}_t, \quad \text{(D.5)}\end{aligned}$$

where $a = 0, \ldots, (n_t^{(k)} - 1)$, $\boldsymbol{\vartheta}^{(0)} = \boldsymbol{\theta}_t^{(k)}$, $\epsilon^{(0)} = e_t^{(k)}$, $\psi^{(0)} = s_t^{(k)}$, and $\boldsymbol{q}_i^{(0)} = \boldsymbol{w}_{i,t}^{(k)}$ for $i = 1, 2$. Also, the updated values are $\boldsymbol{\theta}_{t+1}^{(k)} = \boldsymbol{\vartheta}^{(n_t^{(k)})}$, and $\boldsymbol{w}_{i,t+1}^{(k)} = \boldsymbol{q}_i^{(n_t^{(k)})}$ for $i = 1, 2$.

Appendix E. Proof of Theorem 3

For simplicity, in this proof, we have assumed that $c = 1$, however, the results are readily extended to the general values of c. We construct our proof based on the following assumption:

Assumption. Assume that $e_t^{(k)}$'s are independent and identically distributed (i.i.d) zero-mean Gaussian random variables with variance ζ^2.

We have

$$\begin{aligned}E\left[\lambda_t^{(k)}\right] &= E\left[\min\left\{1, \left(\delta_{t-1}^{(k)}\right)^{l_t^{(k)}}\right\}\right]\\ &\leq \min\left\{1, E\left[\left(\delta_{t-1}^{(k)}\right)^{l_t^{(k)}}\right]\right\}. \quad \text{(E.1)}\end{aligned}$$

Now, we show that under certain conditions, $E\left[\left(\delta_{t-1}^{(k)}\right)^{l_t^{(k)}}\right]$ will be less than 1, hence, we obtain an upper bound for $E\left[\lambda_t^{(k)}\right]$. We define $s \triangleq \ln(\delta_{t-1}^{(k)})$, yielding

$$E\left[\left(\delta_{t-1}^{(k)}\right)^{l_t^{(k)}}\right] = E\left[E\left[\exp\left(s\, l_t^{(k)}\right)\Big|s\right]\right] = E\left[M_{l_t^{(k)}}(s)\Big|s\right], \quad \text{(E.2)}$$

where $M_{l_t^{(k)}}(.)$ is the moment generating function of the random variable $l_t^{(k)}$. From Algorithm 1, $l_t^{(k)} = (k-1)\sigma_m^2 - \sum_{j=1}^{k-1}\left(e_t^{(j)}\right)^2$. According to the Assumption, $\frac{e_t^{(j)}}{\zeta}$ is a standard normal random variable. Therefore, $\sum_{j=1}^{k-1}\left(e_t^{(j)}\right)^2$ has a Gamma distribution as $\Gamma\left(\frac{k-1}{2}, 2\zeta^2\right)$ [71], which results in the following moment generating function for $l_t^{(k)}$

$$\begin{aligned}M_{l_t^{(k)}}(s) &= \exp\left(s(k-1)\sigma_m^2\right)\left(1 + 2\zeta^2 s\right)^{\frac{1-k}{2}}\\ &= \left(\delta_{t-1}^{(k)}\right)^{(k-1)\sigma_m^2}\left(1 + 2\zeta^2\ln\left(\delta_{t-1}^{(k)}\right)\right)^{\frac{1-k}{2}}. \quad \text{(E.3)}\end{aligned}$$

In the above equality $\delta_{t-1}^{(k)}$ is a random variable, the mean of which is denoted by γ. We point out that γ will approach to ζ^2 in convergence. We define a function $\varphi(.)$ such that $E\left[\lambda_t^{(k)}\right] = E\left[\varphi\left(\delta_{t-1}^{(k)}\right)\right]$, and seek to find a condition for $\varphi(.)$ to be a concave function. Then, by using the Jenssen's inequality [72] for concave functions, we have

$$E\left[\lambda_t^{(k)}\right] \leq \varphi(\gamma). \quad \text{(E.4)}$$

Inspired by (E.3), we define $A\left(\delta_{t-1}^{(k)}\right) \triangleq \delta_{t-1}^{(k)\;-2\sigma_m^2}\left(1 + 2\zeta^2\ln\left(\delta_{t-1}^{(k)}\right)\right)$ and also $\varphi\left(\delta_{t-1}^{(k)}\right) \triangleq \left(A\left(\delta_{t-1}^{(k)}\right)\right)^{\frac{1-k}{2}}$. By these definitions we obtain

$$\begin{aligned}\varphi''\left(\delta_{t-1}^{(k)}\right) = \frac{1-k}{2}\left(A\left(\delta_{t-1}^{(k)}\right)\right)^{\frac{-k-3}{2}}\Bigg[&\left(\frac{-k-1}{2}\right)\left(A'\left(\delta_{t-1}^{(k)}\right)\right)^2\\ &+ \left(A\left(\delta_{t-1}^{(k)}\right)\right)^2 A''\left(\delta_{t-1}^{(k)}\right)\Bigg]. \quad \text{(E.5)}\end{aligned}$$

Considering that $k > 1$, in order for $\varphi(.)$ to be concave, it suffices to have

$$\left(A\left(\delta_{t-1}^{(k)}\right)\right)^2 A''\left(\delta_{t-1}^{(k)}\right) > \left(\frac{k+1}{2}\right)\left(A'\left(\delta_{t-1}^{(k)}\right)\right)^2, \quad \text{(E.6)}$$

which reduces to the following necessary and sufficient conditions:

$$\frac{\left(\delta_{t-1}^{(k)}\right)^{2\sigma_m^2}}{\left(1 + 2\zeta^2\ln\left(\delta_{t-1}^{(k)}\right)\right)^2} < \frac{(1 + 2\sigma_m^2)^2}{4(k+1)}, \quad \text{(E.7)}$$

and

$$\frac{(1 - \xi_1)\sigma_m^2}{1 - 2\sigma_m^2\ln\left(\delta_{t-1}^{(k)}\right)} < \zeta^2 < \frac{(1 - \xi_2)\sigma_m^2}{1 - 2\sigma_m^2\ln\left(\delta_{t-1}^{(k)}\right)}, \quad \text{(E.8)}$$

where

$$\xi_1 = \frac{\alpha^2(1 + 2\sigma_m^2) + \alpha\sqrt{(1 + 2\sigma_m^2)^2\alpha^2 - 4(k+1)(\delta_{t-1}^{(k)})^{2\sigma_m^2}}}{2(k+1)(\delta_{t-1}^{(k)})^{2\sigma_m^2}},$$

$$\xi_2 = \frac{\alpha^2(1 + 2\sigma_m^2) - \alpha\sqrt{(1 + 2\sigma_m^2)^2\alpha^2 - 4(k+1)(\delta_{t-1}^{(k)})^{2\sigma_m^2}}}{2(k+1)(\delta_{t-1}^{(k)})^{2\sigma_m^2}},$$

and

$$\alpha \triangleq 1 + 2\zeta^2\ln\left(\delta_{t-1}^{(k)}\right).$$

Under these conditions, $\varphi(.)$ is concave, therefore, by substituting $\varphi(.)$ in (E.4) we achieve (19). This concludes the proof of the Theorem 3. □

References

[1] M. Du, X. Wu, W. Chen, Z. Li, Supervised training and contextually guided salient object detection, Digit. Signal Process. 63 (2017) 44–55.
[2] R.E. Schapire, Y. Freund, Boosting: Foundations and Algorithms, MIT Press, 2012.
[3] A. Fern, R. Givan, Online ensemble learning: an empirical study, Mach. Learn. 53 (1) (2003) 71–109.
[4] R.O. Duda, P.E. Hart, D.G. Stork, Pattern Classification, John Willey and Sons, 2001.
[5] Y. Freund, An adaptive version of the boost by majority algorithm, Mach. Learn. 43 (3) (2001) 293–318.
[6] N. Duffy, D. Helmbold, Boosting methods for regression, Mach. Learn. 47 (2) (2002) 153–200.
[7] Y. Freund, R.E. Schapire, A decision-theoretic generalization of on-line learning and an application to boosting, J. Comput. Syst. Sci. 55 (1997) 119–139.
[8] D.L. Shrestha, D.P. Solomatine, Experiments with AdaBoost.RT, an improved boosting scheme for regression, Neural Comput. 18 (7) (2006) 1678–1710.
[9] S.B. Taieb, R.J. Hyndman, et al., Boosting multi-step autoregressive forecasts, in: ICML, 2014, pp. 109–117.
[10] B. Taieb, R.J. Hyndman, A gradient boosting approach to the Kaggle load forecasting competition, Int. J. Forecast. (2013) 1–19.
[11] C. Boukis, D. Mandic, A. Constantinides, L. Polymenakos, A modified Armijo rule for the online selection of learning rate of the LMS algorithm, Digit. Signal Process. 20 (3) (2010) 630–639.
[12] F. Huang, J. Zhang, A family of gain-combined proportionate adaptive filtering algorithms for sparse system identification, Digit. Signal Process. 70 (2017) 49–58.
[13] J. Arenas-Garcia, L.A. Azpicueta-Ruiz, M.T.M. Silva, V.H. Nascimento, A.H. Sayed, Combinations of adaptive filters: performance and convergence properties, IEEE Signal Process. Mag. 33 (1) (2016) 120–140.
[14] S.S. Kozat, A.T. Erdogan, A.C. Singer, A.H. Sayed, Steady-state MSE performance analysis of mixture approaches to adaptive filtering, IEEE Trans. Signal Process. 58 (8) (2010) 4050–4063.
[15] S. Shaffer, C.S. Williams, Comparison of LMS, alpha-LMS, and data reusing LMS algorithms, in: Conference Record of the Seventeenth Asilomar Conference on Circuits, Systems and Computers, 1983.
[16] D. Kari, I. Marivani, I. Delibalta, S.S. Kozat, Boosted LMS-based piecewise linear adaptive filters, in: 2016 24th European Signal Processing Conference, EUSIPCO, 2016, pp. 1593–1597.
[17] B.C. Civek, D. Kari, I Delibalta, S.S. Kozat, Big data signal processing using boosted RLS algorithm, in: 2016 24th Signal Processing and Communication Application Conference, SIU, 2016, pp. 1089–1092.
[18] N.C. Oza, Online bagging and boosting, in: 2005 IEEE International Conference on Systems, Man and Cybernetics, vol. 3, IEEE, 2005, pp. 2340–2345.
[19] S. Ben-David, E. Kushilevitz, Y. Mansour, Online learning versus offline learning, Mach. Learn. 29 (1) (1997) 45–63.
[20] L. Bottou, O. Bousquet, The tradeoffs of large scale learning, in: NIPS, 2008.
[21] A.H. Sayed, Fundamentals of Adaptive Filtering, John Wiley and Sons, 2003.
[22] S.-T. Chen, H.-T. Lin, C.-J. Lu, An online boosting algorithm with theoretical justifications, in: ICML, 2012.
[23] P. Malik, Governing big data: principles and practices, IBM J. Res. Dev. 57 (3–4) (2013).
[24] L. Breiman, Prediction games and arcing algorithms, Neural Comput. 11 (7) (1999) 1493–1517.
[25] R.A. Servedio, Smooth boosting and learning with malicious noise, J. Mach. Learn. Res. 4 (2003) 633–648.
[26] B. Babenko, M.H. Yang, S. Belongie, A family of online boosting algorithms, in: 2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops, 2009, pp. 1346–1353.
[27] A. Beygelzimer, S. Kale, H. Luo, Optimal and adaptive algorithms for online boosting, in: International Conference on Machine Learning, ICML, 2015, pp. 2323–2331.
[28] Y. Freund, Boosting a weak learning algorithm by majority, Inf. Comput. 121 (2) (1995) 256–285.
[29] A. Bertoni, P. Campadelli, M. Parodi, A Boosting Algorithm for Regression, vol. 1327, Springer, 1997, pp. 343–348.
[30] A. Beygelzimer, E. Hazan, S. Kale, H. Luo, Online gradient boosting, in: Advances in Neural Information Processing Systems, NIPS, 2015, pp. 2458–2466.
[31] Y. Engel, S. Mannor, R. Meir, The kernel recursive least-squares algorithm, IEEE Trans. Signal Process. 52 (8) (2004) 2275–2285.
[32] D.P. Mandic, A generalized normalized gradient descent algorithm, IEEE Signal Process. Lett. 11 (2) (2004) 115–118.
[33] W.-P. Ang, B. Farhang-Boroujeny, A new class of gradient adaptive step-size LMS algorithms, IEEE Trans. Signal Process. 49 (4) (2001) 805–810.
[34] V.J. Mathews, Z. Xie, Stochastic gradient adaptive filters with gradient adaptive step sizes, in: International Conference on Acoustics, Speech, and Signal Processing, vol. 3, 1990, pp. 1385–1388.
[35] R.E. Schapire, Y. Freund, P. Bartlett, W.S. Lee, Boosting the margin: a new explanation for the effectiveness of voting methods, Ann. Stat. (1998) 1651–1686.
[36] N. Littlestone, M.K. Warmuth, The weighted majority algorithm, Inf. Comput. 108 (2) (1994) 212–261.
[37] N. Cesa-Bianchi, G. Lugosi, On prediction of individual sequences relative to a set of experts, in: IEEE International Symposium on Information Theor., 1998, pp. 16–21.
[38] V. Vovk, A game of prediction with expert advice, J. Comput. Syst. Sci. 56 (1998) 153–173.
[39] N. Cesa-Bianchi, Y. Freund, D. Haussler, D.P. Helmbold, R.E. Schapire, M.K. Warmuth, How to use expert advice, J. ACM 44 (3) (1997) 427–485.
[40] J.Z. Kolter, M.A. Maloof, Dynamic weighted majority: a new ensemble method for tracking concept drift, in: Third IEEE International Conference on Data Mining, 2003, ICDM 2003, IEEE, 2003, pp. 123–130.
[41] J. Arenas-Garcia, A.R. Figueiras-Vidal, A.H. Sayed, Mean-square performance of a convex combination of two adaptive filters, IEEE Trans. Signal Process. 54 (2006) 1078–1090.
[42] J. Arenas-Garcia, A.R. Figueiras-Vidal, A.H. Sayed, Tracking properties of a convex combination of two adaptive filters, in: IEEE/SP 13th Workshop on Statistical Signal Processing 2005, 2005, pp. 109–114.
[43] M.T.M. Silva, V.H. Nascimento, Improving the tracking capability of adaptive filters via convex combination, IEEE Trans. Signal Process. 56 (7) (2008) 3137–3149.
[44] H. Ozkan, M.A. Donmez, S. Tunc, S.S. Kozat, A deterministic analysis of an online convex mixture of experts algorithm, IEEE Trans. Neural Netw. Learn. Syst. 26 (7) (2015) 1575–1580.
[45] J. Arenas-García, M. Martínez-Ramón, Á. Navia-Vázquez, A.R. Figueiras-Vidal, Plant identification via adaptive combination of transversal filters, Signal Process. 86 (9) (2006) 2430–2438.
[46] V.H. Nascimento, R.C. de Lamare, A low-complexity strategy for speeding up the convergence of convex combinations of adaptive filters, in: 2012 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP, IEEE, 2012, pp. 3553–3556.
[47] L.F.O. Chamon, W.B. Lopes, C.G. Lopes, Combination of adaptive filters with coefficients feedback, in: 2012 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP, 2012, pp. 3785–3788.
[48] M.T.M. Silva, V.H. Nascimento, J. Arenas-García, A transient analysis for the convex combination of two adaptive filters with transfer of coefficients, in: 2010 IEEE International Conference on Acoustics, Speech and Signal Processing, 2010, pp. 3842–3845.
[49] D.L. Duttweiler, Proportionate normalized least-mean-squares adaptation in echo cancelers, IEEE Trans. Speech Audio Process. 8 (5) (2000) 508–518.
[50] L.A. Azpicueta-Ruiz, M. Zeller, A.R. Figueiras-Vidal, J. Arenas-García, W. Kellermann, Improved acoustic echo cancellation for low SNR based on blockwise combination of filters, in: 20th Int. Congr. Acoustics, 2010.
[51] B.K. Das, M. Chakraborty, A block-based convex combination of NLMS and ZA-NLMS for identifying sparse systems with variable sparsity, in: 2017 IEEE International Symposium on Circuits and Systems, ISCAS, 2017, pp. 1–4.
[52] N. Cesa-Bianchi, G. Lugosi, Prediction, Learning, and Games, Cambridge University Press, 2006.
[53] S.S. Kozat, A.C. Singer, G.C. Zeitler, Universal piecewise linear prediction via context trees, IEEE Trans. Signal Process. 55 (7) (2007) 3730–3745.
[54] N.D. Vanli, S.S. Kozat, A comprehensive approach to universal piecewise nonlinear regression based on trees, IEEE Trans. Signal Process. 62 (20) (2014) 5471–5486.
[55] H. Ozkan, N.D. Vanli, S.S. Kozat, Online classification via self-organizing space partitioning, IEEE Trans. Signal Process. 64 (15) (2016) 3895–3908.
[56] A.C. Singer, S.S. Kozat, M. Feder, Universal linear least squares prediction: upper and lower bounds, IEEE Trans. Inf. Theory 48 (8) (2002) 2354–2362.
[57] S.S. Kozat, A.C. Singer, Universal switching linear least squares prediction, IEEE Trans. Signal Process. 56 (1) (2008) 189–204.
[58] E. Hazan, A. Agarwal, S. Kale, Logarithmic regret algorithms for online convex optimization, Mach. Learn. 69 (2–3) (2007) 169–192.
[59] M. Herbster, M.K. Warmuth, Tracking the best expert, in: Proceedings of International Conference on Machine Learning, 1995, pp. 286–294.
[60] D. Haussler, J. Kivinen, M. Warmuth, Tight worst-case loss bounds for predicting with expert advice, in: P. Vitanyi (Ed.), Computational Learning Theory 2nd Euro. Conf., 1995, pp. 69–83.
[61] S.S. Kozat, A.C. Singer, Universal switching linear least squares prediction, IEEE Trans. Signal Process. 56 (Jan. 2008) 189–204.
[62] Y. Sai, R. Jinxia, L. Zhongxia, Learning of neural networks based on weighted mean squares error function, in: 2009 Second International Symposium on Computational Intelligence and Design, vol. 1, 2009, pp. 241–244.
[63] H. Hu, W. Sun, A. Venkatraman, M. Hebert, J.A. Bagnell, Gradient boosting on stochastic data streams, arXiv preprint, arXiv:1703.00377.
[64] M.H. Al-Badrawi, N.J. Kirsch, B.Z. Al-Jewad, Intrinsic mode function based noise power estimation with applications to semiblind spectrum sensing methods, IEEE Signal Process. Lett. 24 (7) (2017) 1088–1092, https://doi.org/10.1109/LSP.2017.2710883.
[65] N.D. Vanli, S.S. Kozat, A comprehensive approach to universal piecewise nonlinear regression based on trees, IEEE Trans. Signal Process. 62 (20) (2014) 5471–5486.

[66] S. Wiggins, Introduction to Applied Nonlinear Dynamical Systems and Chaos, Springer, New York, 2003.
[67] L. Torgo, Regression data sets, http://www.liaad.up.pt/~ltorgo/Regression/DataSets.html, 2003.
[68] D. Dheeru, E. Karra Taniskidou, UCI machine learning repository, http://archive.ics.uci.edu/ml, 2017.
[69] F. Pereira, P. Machado, E. Costa, A. Cardoso, Progress in artificial intelligence, in: 17th Portuguese Conference on Artificial Intelligence, EPIA 2015, Coimbra, Portugal, in: Lecture Notes in Computer Science, Springer International Publishing, 2015.
[70] T. Bertin-Mahieux, D.P. Ellis, B. Whitman, P. Lamere, The million song dataset, in: Proceedings of the 12th International Conference on Music Information Retrieval, ISMIR 2011, 2011.
[71] A. Papoulis, S.U. Pillai, Probability, Random Variables, and Stochastic Processes, 4th edition, McGraw-Hill Higher Education, 2002.
[72] H.V. Poor, An Introduction to Signal Detection and Estimation, Springer Science & Business Media, 2013.

Dariush Kari received the B.S. degree with high honor in two majors, Electrical Engineering and Computer Science, from Amirkabir University of Technology (Tehran Polytechnic), Iran, 2014, and his M.S. degree in Electrical and Electronics Engineering from Bilkent University, Ankara, Turkey. He is currently a Ph.D. student at the Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, Urbana, IL, USA. His current research interests include probabilistic machine learning, online learning, big data analytics, and optimization.

Ali H. Mirza received the B.Sc. degree with high honors in Electrical Engineering from University of Engineering and Technology, Lahore, Pakistan, in 2014. He is currently a Ph.D. student in the Department of Electrical and Electronics Engineering at Bilkent University, Ankara, Turkey. His research interests include machine learning, big data signal processing, online learning and deep neural networks.

Farhan Khan was born in Swabi, Pakistan, in 1983. He received the B.Sc. degree with honors in Electrical Engineering from University of Engineering and Technology, Peshawar, Pakistan, in 2007, and the M.Sc. degree in RF Communication Systems from University of Southampton, UK, in 2009. After graduation, he joined the Department of Electrical Engineering at the COMSATS Institute of Information Technology, Pakistan. He is currently working towards the Ph.D. degree in the Department of Electrical and Electronics Engineering at Bilkent University, Ankara, Turkey. His research interests include adaptive signal processing, big data, machine learning, and neural networks.

Huseyin Ozkan is currently a faculty member of Electronics Engineering and with the Faculty of Engineering and Natural Sciences at Sabanc University. Dr. Ozkan received his B.Sc. degrees in Electrical Engineering and Mathematics from Bogazici University; and his M.Sc. and Ph.D. degrees in Electrical Engineering from Boston University and Bilkent University, respectively. Previously, he had been working as a postdoctoral research associate in Vision and Computational Neuroscience at Massachusetts Institute of Technology. His research interests are in machine learning, signal processing, computer vision and computational neuroscience.

Suleyman S. Kozat (A′10–M′11–SM′11) received the B.S. degree with full scholarship and high honors from Bilkent University, Turkey. He received the M.S. and Ph.D. degrees in electrical and computer engineering from University of Illinois at Urbana-Champaign, Urbana, IL. Dr. Kozat is a graduate of Ankara Fen Lisesi. After graduation, Dr. Kozat joined IBM Research, T. J. Watson Research Lab, Yorktown, New York, US as a Research Staff Member in the Pervasive Speech Technologies Group. While doing his Ph.D., he was also working as a Research Associate at Microsoft Research, Redmond, Washington, US in the Cryptography and Anti-Piracy Group. He holds several patent inventions due to his research accomplishments at IBM Research and Microsoft Research. Dr. Kozat is currently an Associate Professor at the Electrical And Electronics Engineering Department of Bilkent University. Dr. Kozat is the President of the IEEE Signal Processing Society, Turkey Chapter. He has been elected to the IEEE Signal Processing Theory and Methods Technical Committee and IEEE Machine Learning for Signal Processing Technical Committee, 2013. He has been awarded IBM Faculty Award by IBM Research in 2011, Outstanding Faculty Award by Koc University in 2011 (granted the first time in 16 years), Outstanding Young Researcher Award by the Turkish National Academy of Sciences in 2010, ODTU Prof. Dr. Mustafa N. Parlar Research Encouragement Award in 2011, Outstanding Faculty Award by Bilim Kahramanlari, 2013 and holds Career Award by the Scientific Research Council of Turkey, 2009.