Sequential Prediction Over Hierarchical Structures

N. Denizcan Vanli, Kaan Gokcesu, Muhammed O. Sayin, Hikmet Yildiz, and Suleyman Serdar Kozat, *Senior Member, IEEE*

***Abstract*—We study sequential compound decision problems in the context of sequential prediction of real valued sequences. In particular, we consider finite state (FS) predictors that are constructed based on a hierarchical structure, such as the order preserving patterns of the sequence history. We define hierarchical equivalence classes by tying certain models at a hierarchy level in a recursive manner in order to mitigate undertraining problems. These equivalence classes defined on a hierarchical structure are then used to construct a super exponential number of sequential FS predictors based on their combinations and permutations. We then introduce truly sequential algorithms with computational complexity only linear in the pattern length that $1)$ asymptotically achieve the performance of the best FS predictor or the best linear combination of all the FS predictors in an individual sequence manner without any stochastic assumptions over any data length n under a wide range of loss functions; $2)$ achieve the mean square error of the best linear combination of all FS filters or predictors in the steady-state for certain nonstationary models. We illustrate the superior convergence and tracking capabilities of our algorithm with respect to several state-of-the-art methods in the literature through simulations over synthetic and real benchmark data.**

***Index Terms*—Sequential prediction, online learning, finite state machine, hierarchical modeling, big data.**

I. Introduction

A. Preliminaries

We investigate sequential compound decision problems that arise in several different signal processing [1]–[5], information theory [6], [7] and machine learning applications [8]–[11]. In particular, we sequentially observe a real valued sequence $x_1, x_2, \ldots$ and produce a decision (or an action) $\hat{d}_t$ at each time t as our output based on the past $x_1, x_2, \ldots, x_t$. We then suffer a loss based on this output when the true d_t is revealed and our goal is to minimize the (weighted) accumulated or expected loss as much as possible while using a limited amount of information from the past. As an example, in the well-known sequential prediction problem under the square error loss, the output $\hat{d}_t$ at time t corresponds to an estimate of the next data point x_{t+1}, where the algorithm suffers the loss $(x_{t+1} - \hat{d}_t)^2$ after x_{t+1}, i.e., $d_t = x_{t+1}$, is revealed. The algorithm can then adjust itself in order to reduce the future losses. This generic setup models a wide range problems in various different applications ranging from adaptive filtering [12], channel equalization [13], repeated game playing [9] to online compression by sequential probability assignment [14], [15].

Manuscript received April 14, 2015; revised May 30, 2016; accepted August 2, 2016. Date of publication September 8, 2016; date of current version October 6, 2016. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Gesualdo Scutari. This work was supported in part by the Outstanding Researcher Program of Turkish Academy of Sciences and in part by the TUBITAK under Contract 113E517.

The authors are with the Department of Electrical and Electronics Engineering, Bilkent University, Ankara 06800, Turkey (e-mail: vanli@ee.bilkent.edu.tr; gokcesu@ee.bilkent.edu.tr; sayin@ee.bilkent.edu.tr; hyildiz@ee.bilkent.edu.tr; kozat@ee.bilkent.edu.tr).

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

Digital Object Identifier 10.1109/TSP.2016.2607141

In this paper, we investigate an important version of sequential compound decision problems, where we consider finite state (FS) machines. State (or side information) dependent data processing in the context of filtering or prediction is extensively studied both in signal processing [16]–[19] and information theory [7], [14], [15], [20] literatures since this structure naturally arises in different real life applications and is suitable for big data applications due to its compact representation [21].

As an example application, we specifically study predicting the trend of an observed sequence, i.e., whether x_{t+1} will be greater/smaller than x_t at each time t, e.g., whether an increase $d_t = 1$ or a decrease $d_t = -1$ happens in the future. Since we are interested in the relative value of the next sample, we use the relative ordering patterns of the sequence history to define states. In particular, at each time t, we use the last l samples of the underlying sequence, i.e., $(x_{t-l+1}, \ldots, x_t)$, to construct relative ordering patterns and define states using them. However, we emphasize that our results and derivations are generic and independent of the particular application or the desired sequence d_t, i.e., our results apply to many state definitions under a wide range of loss functions (both statistical and deterministic) as clarified later in the paper. In this sense, we cover both sequential prediction [4] and adaptive filtering problems [12] as demonstrated in our simulations. For this particular example application, we set the relative ordering patterns as our states since an uphill trend in product usage or a downhill trend in a stock price is expected to continue in the future [22]. As a real life application, we demonstrate that we can accurately predict the relative electric consumption of actual customers in Turkey using their past consumption patterns in Section IV.

In state dependent compound decision problems that we study in this paper, there is one issue that naturally arises, which is the selection of the state space. In our example application of trend prediction problem, this corresponds to selecting the length of the relative ordering patterns, i.e., choosing the value of l (sequence history length). We may choose a large l, and thus, select a large number of states for accurate and fine modeling. However, selecting too many states may result in undertraining due to the sparsity of data, since some states may not occur frequently enough for sufficient training. We point out that for the trend prediction scenario, i.e., when the states are defined to be the relative ordering patterns, a sequence history of length l, i.e., $(x_{t-l+1}, \ldots, x_l)$, can have $l! \approx (l/e)^l$ different ordering patterns [23]. Hence, even for a moderate l value that defines meaningful patterns in real life applications [23]–[25], say for $l = 10$, the number of patterns grows as $10! \approx (10/e)^{10} = 4.54 \times 10^5$. Therefore, training a sequential FS machine using this many patterns directly as states is impractical since we require a substantial amount of past observations (which may not be available in most real life applications even for stationary data). Therefore, we may choose a fewer number of states that still covers the whole observation space. As an example, in learning with decision trees, pruning [26] is a widely used technique for overcoming undertraining problems by decreasing the number of states to be trained. However, note that, selecting states too coarsely may not be sufficient for accurate representation of the data. Fixed selection of states is also a problem when the "correct" number of states changes over time, e.g., when the data is nonstationary, as in most real life applications [27].

Therefore, we use hierarchical structures, which are a powerful tool for state selections. Instead of simply choosing a suitable set of sufficiently informative states and training them, we can model the data via equivalence classes and train all the states (both fine and coarse) simultaneously using hierarchical structures. Hierarchical structures are encountered in every contextual signal processing application when the observation data can be clustered in a certain way. The clustering is done via common features, which are repeatedly occurring. Since certain features can be tied together, these qualities can be classified via a hierarchical structure. Therefore, hierarchical structures can accurately model wide range of diverse state information and are suitable for many applications.

For example, in sequential source coding problem studied in [15], Willems proposes two nearly optimal source code methods. For the first method, the state of the FS machines are given by the number of changes encountered so far and the time index of the last change, while for the second method only the time index of the last change is used. The states of these two can be combined in a hierarchical manner such the finest states constitute both of these parameters as in their first method and the coarsest states are given by only the last change time index as in the second method. Sequential coding for a binary tree source has been studied in [14], where context tree weighting is used for sequential probability assignment. The states of the FS predictors correspond to the past symbols. Each observable sequence constitutes a state and these states are connected in a hierarchical manner as in our hierarchical structure. With each observed symbol, we move to the lower levels in the hierarchy. In [28], Feder *et al.* provides universal codes for hierarchical structure of classes. In [29], Gyorgy *et al.* uses a hierarchical implementation of switching strategies to compete against the class of switching experts. Different experts with different switching times constitute a hierarchical structure if they made some of their switches at the same time. In [30], Helmbold and Schapire propose an algorithm that predicts nearly as well as the best pruning of an unpruned decision tree, where the states correspond to the nodes of that tree. In [31], Takimoto *et al.* improves upon the results of [30] and provides dynamical programming schemes for prediction as well as the best pruning of the decision tree. Decision trees constitute a hierarchical structure since the mother nodes are created from hierarchical combination of child nodes. In the linear and nonlinear prediction problem studied in [16], [17], rate optimal prediction algorithms using context trees are provided, where the states correspond to the partitions of the observation space for linear predictors, which creates piecewise linear models. The partitions are combined in a hierarchical manner in the context tree to create different partitionings of the observation space. For channel equalization problem in [13],

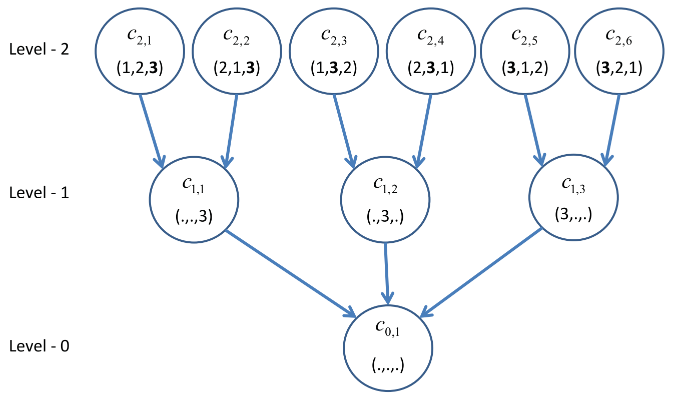


Fig. 1. All equivalence classes for the FS diagram with $l = 3$ and $h = 2$.

context trees are used for the classification of side information, where the states correspond to the partitioned regions of the variance vector of the input signal. The hierarchical combination of partitioned regions creates different partitionings of the feature space.

Note that, all these applications either directly uses hierarchical structures or can be straightforwardly converted into our framework, they are merely specialized examples of the general class of hierarchical approaches that we propose. Any context tree structure used in the state dependent data processing problems falls into our model. We generalize all the applications of the hierarchical structures in our comprehensive algorithm. Instead of deciding on a particular structure beforehand, we propose an algorithm that works with any kind of hierarchical structure. Our general model covers all tree structures (uniform and nonuniform) and all of the splitting model classes considered in [32]. The hierarchical structure used in our algorithm can be any kind of oriented directed graph [33] satisfying two properties:

- the connections have to be from the classes in the lower levels of the hierarchy to the classes in the higher levels,
- the connections to a node in the hierarchical graph needs to correspond to disjoint regions in the state space whose union gives the state space region of the connected node.

In the trend prediction problem, we create the hierarchical structure by defining "super set" equivalence classes in relative ordering patterns. Defining super sets is a widely used approach in speech recognition applications when there are not enough data to adequately train all the phoneme states [34]. Even though, a sequence of length l can have $l!$ different ordering patterns, most of these share similar characteristics that can be exploited to group (or tie) them together to form intermediate states each representing a collection of the original patterns as shown in Fig. 1. For the scenario in Fig. 1, we consider the location of the largest element as the main characteristics in order to group the patterns in a recursive hierarchical manner. With such a hierarchical equivalence class definition, we can construct a huge number of different sequential FS predictors corresponding to different permutations and combinations of these equivalence classes, e.g., the sequential FS predictor using all the order preserving patterns directly as states and the sequential FS predictor using none of the patterns as states. Note that one of the FS predictors defined by the hierarchical structure is optimal for the underlying task at hand. The optimal predictor may change in time for nonstationary data, as is generally

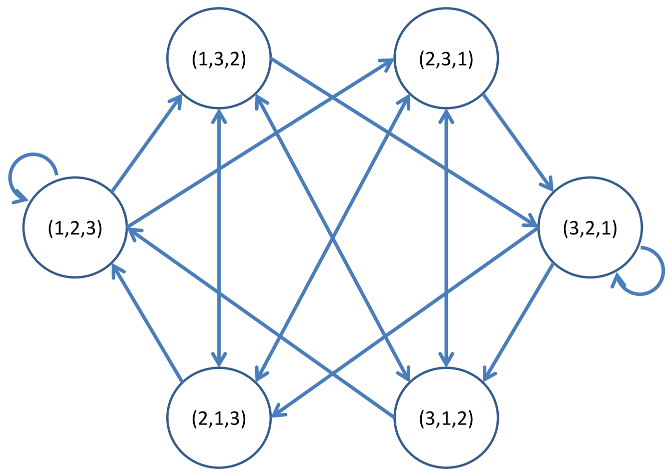


Fig. 2. FS Diagram for $l = 3$ and $h = 2$, where all allowable transitions are drawn.

the case in real life applications, but the hierarchical structure can react quickly to these changes, since the coarser states are trained faster. One may think of training the coarser states as tracking the general behavior of the data and training the finer states as more accurate modeling of the data.

B. Relevant Work

State dependent prediction under a particular hierarchical model, i.e., using context trees, is extensively investigated in the computational learning theory [30], [31], [35] and in the signal processing literature [16], [17]. However, these methods mainly and merely use the weighting ideas [14], [15] from universal source coding or derandomization methods [36] in order to efficiently construct either the Aggregated Algorithm (AA) [37] (or its variants) or the universal mixture weights [38]. Hence, the resulting specific mixture algorithm can only achieve the accumulated loss performance of the best algorithm in the mixture for this specific tree structure. Note that the accumulated loss as a performance measure may not be useful or relevant in most adaptive signal processing applications where the tracking and the transient performances are more critical [12]. To resolve this issue, switching competition frameworks are incorporated into the AA weighting and its variants [4], [29], [39]. However, the assumptions required by these algorithms (such as the desired data to be bounded) do not hold for most real life adaptive signal processing applications such as Gaussian data [27]. Furthermore, most of these algorithms require a priori information on the underlying data sequence to be optimized in an individual sequence manner, such as the data length or the number of switches [39].

On the other hand, there exist universal binary prediction algorithms [21] that achieve the performance of any finite state predictor in the long run. In this sense, the choice of the states, such as defining "patterns" as the states or other equivalence classes as the states, or selection of the transition functions between the states are not relevant in the long run since the algorithms of [21] achieve the performance of any state dependent predictor. However, note that, the results in [21] are asymptotic, i.e., the data length goes to infinity, and states of the finite state machines or transitions between them are fixed. We emphasize that the state definition, i.e., the equivalence class defined by a state, or the relevance of a state information can change in time since statistics of the data may change in time. Hence a successful prediction algorithm for finite data lengths should also learn or update the best state assignment, i.e., what we define as a state or side information from the data, and quickly adapt to changes in the data statistics. In this sense, although such results apply in the long run, they are not applicable over finite length data sequences, which is the main contribution of this paper since the best choice of the underlying FS predictor can change in time.

C. Contributions

For the first time in the literature, we have introduced a truly sequential comprehensive algorithm to solve the online compound decision problem given a generic hierarchical model. We emphasize that our approach is universal and generic such that our algorithm can be applied to a wide range of hierarchical equivalence class definitions. Although the problem of sequential prediction in order preserving patterns is considered in the paper, our results and derivations are independent of the particular application and apply to many state definitions under a wide range of loss functions (both statistical and deterministic), e.g., general convex loss functions with Lipschitz gradients. We cover both sequential prediction [4] and adaptive filtering problems [12] as demonstrated in our simulations.

We propose a self working, computationally efficient algorithm that works with any kind of hierarchical structure. Our algorithm is universal over all possible hierarchical structures, such that it is not dependent on the specific hierarchical model. We introduce a truly sequential algorithm with computational complexity only linear in the hierarchy depth h (e.g., we can set $h = l - 1$ for the trend prediction scenario, cf. Fig. 1) for a generic hierarchical structure that *i)* asymptotically achieves the performance of the best FS predictor among the doubly exponential number of possible FS predictors in an individual sequence manner without any stochastic assumptions over any n under a wide range of loss functions; *ii)* asymptotically achieves the performance of the best "linear combination" of all FS predictors defined on the hierarchy in an individual sequence manner over any n under a wide range of loss functions; *iii)* achieve the MSE of the best linear combination of all FS filters or predictors in the steady-state [27] for certain nonstationary models [12], [40]. We emphasize that our algorithms are truly sequential such that they do not need any a priori information on the underlying data sequence such as the sequence length, bounds on the sequence values or the statistical distribution of the data. In this sense, the introduced algorithm is suitable for big data and real life applications under both stationary and nonstationary settings. We also show that the weights of our algorithm converge to the minimum MSE (MMSE) optimal linear combination weights.

We emphasize that we propose a general operational algorithm and not a weighting model. Our algorithm is applicable for use with numerous weightings in literature [9], [37], [38], [41]. Our algorithm can also use other potential functions in a straightforward manner. Probability assignment ideas in universal coding [21], [28] can be directly implemented. To this end, the universal weighting methods used in sequential coding literature can be converted to prediction algorithms through the methods used in this paper. There are only two requirements for the weighting scheme to be used:

- the weight updates for the FS predictors needs to be multiplicative for our telescoping rule to work,
- the weight update of the connected states in the hierarchy needs to be same if the observed sample falls within their respective state space regions.

D. Organization of the Paper

In Section II, we describe the use of the hierarchical structures for FS prediction and provide the problem framework. In Section III-A, we introduce a sequential FS predictor that solves the underlying problem using a brute force approach with a computational complexity doubly exponential in the hierarchy depth. We then show in Section III-B that this algorithm can be efficiently implemented with a computational complexity linear in the hierarchy depth, hence resulting in a tremendous reduction in the computational cost. In Sections III-C and III-D, we extend our discussions for different linear combination weights and cost functions, respectively. We then demonstrate the performance of our algorithm through simulations in Section IV and finalize our paper with concluding remarks in Section V.

II. Finite State Prediction via Hierarchical Structures

We consider the generic prediction framework under an arbitrary convex loss function with Lipschitz gradient [42], where we sequentially observe a real valued sequence $x_1, x_2, \ldots$ and produce an output $\hat{d}_t$ based on $x_1, \ldots, x_t$ at each time t. Then, the true output d_t is revealed yielding the loss $\ell(d_t, \hat{d}_t)$, e.g., $\ell(d_t, \hat{d}_t) = (d_t - \hat{d}_t)^2$. Over any data length n, the performance of the predictor is evaluated by its time accumulated loss, i.e., $\sum_{t=1}^{n} \ell(d_t, \hat{d}_t)$, or by its accumulated expected loss, i.e., $\sum_{t=1}^{n} E[\ell(d_t, \hat{d}_t)]$. Nevertheless, we emphasize that our methods can incorporate different loss functions such as the accumulated weighted loss, i.e., $\sum_{t=1}^{n} \lambda^{n-t}\, \ell(d_t, \hat{d}_t)$, where $0 < \lambda \leq 1$ represents the forgetting factor.

In order to produce the output $\hat{d}_t$, we use an FS predictor. In its most general form, an FS predictor has a sequential prediction function

$$\hat{d}_t = f_t(s_t), \tag{1}$$

where $s_t \in \mathcal{S}$ is the current state taking values from a finite set $\mathcal{S}$, e.g., the set of relative ordering patterns. Upon the observation of the new data x_{t+1}, the states are traversed according to the next state function

$$s_{t+1} = g(s_t, x_{t+1}). \tag{2}$$

One can use different variations for (1) and (2), e.g., include different samples of the observed sequence in the function definitions or use time varying functions, e.g., $\hat{d}_t = f(s_t, s_{t-1})$ or $s_{t+1} = g_t(s_t, x_t, x_{t-1}, x_{t-2})$. However, either these variations can be covered by our basic setup by defining a super set of states or our results can be straightforwardly extended to these configurations [34].

In this framework, we consider a finite set of states $\mathcal{S}$ that is selected according to the underlying prediction task. As an example, for prediction problem, one can choose the past values of a sequence as the set of states, i.e., at each time t, we can use the last l samples of the sequence history $x_{t-l+1}, \ldots, x_t$ to define equivalence classes or states. Similarly, for portfolio selection problem, the set of states can be chosen according to the ratio of the closing prices of stocks to the opening prices [43]. In our example, we consider the relative ordering patterns of the sequence history as our states.

Example 1: Consider the setting where at each time t, we use the last 3 samples of the sequence history x_{t-2}, x_{t-1}, x_t to define equivalence classes or states. In this setting, we can have $3! = 6$ different possible patterns, i.e., $\mathcal{S} = \{(1,2,3), (1,3,2), (2,1,3), (2,3,1), (3,1,2), (3,2,1)\}$, where "3" represents the location of the largest value and "1" represents the location of the smallest value, e.g., the sequence $(x_{t-2}, x_{t-1}, x_t) = (5, 10, -1)$ corresponds to the pattern or ordering $(2,3,1)$. At time instant t, naturally, the current state s_t can correspond to only one of these states in $\mathcal{S}$, i.e., the state set is complete. $\mathcal{S}$ forms the highest level of hierarchy in our notation and framework and each element of $\mathcal{S}$ denotes a different node in this level. Note that, the nodes that are higher in the hierarchy are positioned in the lower levels of the graph in our notation, e.g., the node at the top of the hierarchy is in level-0. After assigning the states at the highest level, we tie certain states into certain equivalence classes and create lower levels in the hierarchical graph. As an example, for the order preserving patterns, we have the aforementioned states at the highest level (see Fig. 2). Assume that our aim is to create a hierarchical structure of depth $h = l - 1$ according to the place of the greatest element in each equivalence class. Hence, we tie the 6 states at level-2 according to the place of their greatest element, e.g., we tie the states $(1,2,3)$ and $(2,1,3)$ into $(\cdot,\cdot,3)$. Thus, we obtain the equivalence classes $(\cdot,\cdot,3)$, $(\cdot,3,\cdot)$, and $(3,\cdot,\cdot)$ at level-1. Continuing this procedure, we obtain $(\cdot,\cdot,\cdot)$ at level-0.

In the generic scenario, let $c_{i,j}$ represent the jth equivalence class at the ith hierarchy level and let $\mathcal{H}_i$ represent the set of equivalence classes at hierarchy level i, where $0 \leq i \leq h$. We start constructing our equivalence classes by setting $\mathcal{H}_h = \mathcal{S}$ and tie certain equivalence classes according to a tying function

$$\phi^{(i+1 \to i)}(c_{i+1,j}) = c_{i,k}, \tag{3}$$

for some $c_{i+1,j} \in \mathcal{H}_{i+1}$ and $c_{i,k} \in \mathcal{H}_i$, where $0 \leq i \leq h-1$. With an abuse of notation, we use $\phi(c_{i,j+1})$ as our tying function in the rest of the paper.

Having obtained a hierarchical structure using a tying function $\phi(\cdot)$, we assign an FS predictor to each of the equivalence classes (we emphasize that each of the original states at the highest level also corresponds to an equivalence class). In Example 1, for ease of exposition, let us assume that our aim is to determine the relative gain/loss of the upcoming value of the sequence compared to its current value, i.e.,

$$d_t = \begin{cases} 1, & \text{if } x_{t+1} \geq x_t \\ -1, & \text{otherwise} \end{cases}. \tag{4}$$

Then, we can use a sequential binary prediction algorithm in each equivalence class on the hierarchical model. For this, we can assign a universal binary predictor (such as [44]) to each

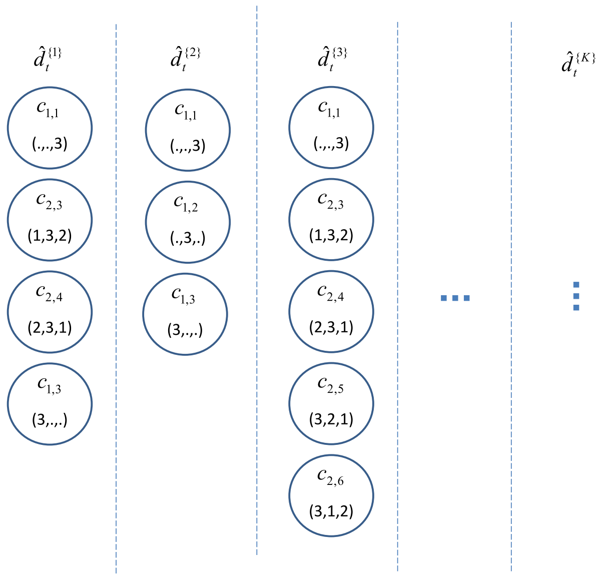


Fig. 3. 3 possible FS predictors for the equivalence classes in Fig. 1 with $h = 3$.

equivalence class $c_{i,j}$ as follows

$$\hat{d}_t^{\{c_{i,j}\}} = \frac{\sum_{\tau=1}^{t-1} I_\tau^{\{c_{i,j}\}} d_\tau}{\max\left(1, \sum_{\tau=1}^{t-1} I_\tau^{\{c_{i,j}\}}\right)} \tag{5}$$

to predict d_t, where $\hat{d}_t^{\{c_{i,j}\}}$ is the prediction of the equivalence class $c_{i,j}$ at time t and $I_\tau^{\{c_{i,j}\}}$ is the indicator function representing whether the length-h sequence corresponds to the equivalence class $c_{i,j}$, i.e.,

$$I_\tau^{\{c_{i,j}\}} \triangleq \begin{cases} 1, & \text{if } s_\tau \in c_{i,j} \\ 0, & \text{otherwise} \end{cases}. \tag{6}$$

Having fixed the state definitions and the sequential predictors at each equivalence class, we then consider all different FS predictors defined on the hierarchical model. An example FS predictor considers all nodes at the hth hierarchy level (highest level) such that its output is set as $\hat{d}_t = \hat{d}_t^{\{c_{h,j}\}}$, when $s_t \in c_{h,j}$, where $c_{h,j} \in \mathcal{H}_h$. This FS predictor is constructed only from the finest states and requires most samples to train. Similarly, one can construct an FS predictor by considering the single equivalence class at the lowest level, i.e., $c_{0,1}$, and directly set $\hat{d}_t = \hat{d}_t^{\{c_{0,1}\}}$ for all s_t. In Fig. 3, we illustrate 3 possible FS predictors for Example 1 of order preserving patterns where $l = 3$ and $h = 2$.

We emphasize that FS predictors that are formed using equivalence classes at higher hierarchy levels include more states. For instance, in Fig. 3, the second FS predictor is formed using the equivalence classes at level-1 and includes 3 states, whereas the third FS predictor includes some equivalence classes from level-2, thus includes 5 states. Clearly, FS predictors containing more states require longer data sequences in order to sufficiently train each sequential equivalence class (or state) predictor $\hat{d}_t^{\{c_{i,j}\}}$ they use. In this sense, a hierarchical structure introduces different coarse and fine FS predictors. Hence, at the beginning of the learning process, one can use coarser FS predictors that can be quickly trained and then gradually switch to the finer FS predictors, e.g., the complete model defined on the highest level of hierarchical graph. However, a careless switching between coarser and finer FS predictors can significantly deteriorate the performance of the system and the optimal selection of such a switching is highly data dependent [4]. Furthermore, the effectiveness of FS predictors may change over time, especially when the underlying data is highly nonstationary. Then, finest model defined on the highest level of hierarchical graph may never have enough data to adequately train its equivalence class predictors even if it observes a data sequence of infinite length. To this end, in the following section, we introduce a sequential algorithm that elegantly and effectively performs such decisions by intrinsically implementing and combining a huge number of FS predictors.

III. Prediction Over Hierarchical Structures

For a hierarchical structure with depth h, we can have at least 2^{2^h} different FS predictors provided that each equivalence class at any level is formed by tying at least two equivalence classes in the above level [45]. As an example, for the order preserving patterns, we have $K_h \approx 2^{(h/e)^h}$ different FS predictors since $K_h = (K_{h-1})^{h+1} + 1$ as can be seen in Fig. 1. In the generic scenario, over $K > 2^{2^h}$ different FS predictors, i.e., $\hat{d}_t^{\{k\}}, k = 1, \ldots, K$ (where we drop the subscript h for notational simplicity), one of them is optimal for the current observations. However, as we observe new samples of the data, the optimal FS predictor can change over time. For example, when there is not enough data, the coarsest FS predictor only having the equivalence class at the lowest hierarchy level, i.e., $c_{0,1}$, is expected to learn much faster than the finest FS predictor having the equivalence classes at the highest hierarchy level, i.e., $\forall c_{h,j} \in \mathcal{H}_h$. However, one expects the finest model to perform better as the observed data length increases considering its higher modeling power for stationary data. On the other hand, when the data is nonstationary, making an efficient switching from coarser to finer FS predictors may not be possible. Therefore, in this paper, instead of committing to one of these FS predictors or switching between them, we use a mixture-of-experts approach to adaptively combine the outputs of all these FS predictors, $\hat{d}_t^{\{k\}}, k = 1, \ldots, K$.

In a general weight update framework for mixture weights, the update rule corresponds to balancing the desire to utilize new information and the desire to retain the past information. An update rule minimizes an objective function to find the new weight such that

$$\boldsymbol{w}_t = \arg\min_{\boldsymbol{w}} \left[D(\boldsymbol{w}_{t-1}, \boldsymbol{w}) + \mu \ell(d_{t-1}, \boldsymbol{w}^T \hat{\boldsymbol{d}}_{t-1}) \right], \tag{7}$$

where $\ell(\cdot,\cdot)$ is a convex loss function with Lipschitz gradient and $D(\cdot,\cdot)$ is a general distance measure between the weight vectors [8]. As an example, if the distance measure is selected to be Kullbeck-Leibler divergence [46], than the weight updates correspond to the Exponentiated Gradient (EG) algorithm [8], such that

$$\boldsymbol{w}_t[k] = \frac{\boldsymbol{w}_{t-1}[k] \exp\left(-\mu \epsilon_{t-1} \hat{\boldsymbol{d}}_{t-1}[k]\right)}{\sum_{r=1}^K \boldsymbol{w}_{t-1}[r] \exp\left(-\mu \epsilon_{t-1} \hat{\boldsymbol{d}}_{t-1}[r]\right)}, \tag{8}$$

where $\boldsymbol{w}_t[k]$ and $\hat{\boldsymbol{d}}_{t-1}[k]$ corresponds to the k^{th} element of the weight vector $\boldsymbol{w}_t$ and the predictor output vector $\hat{\boldsymbol{d}}_t$ respectively, and ϵ_t is the gradient of the loss function $\ell(\cdot,\cdot)$ with respect to the final prediction $\hat{d}_t$, which is equal to $\boldsymbol{w}_t^T\hat{\boldsymbol{d}}_t$. Instead of a linear update using the gradient as in Gradient Descent, the update is done in the exponent [8]. The denominator normalizes the updated weights to make their sum 1, and thus creates a probability simplex.

We use EG algorithm [8] in combining the predictions of our FS predictors, since EG produces sublinear regret for generic loss functions and the weight updates are multiplicative. Since the additive updates are done in the exponent, an hierarchical structure is straightforward to implement. Although there exists various alternatives of the EG algorithm in the signal processing and machine learning literatures [8], [9], the main advantage of the EG algorithm is its superior tracking performance compared to its well-known alternatives, e.g., the least mean squares (LMS) [47]. Therefore, our algorithm can track abrupt changes or nonstationary data better than its alternatives. Furthermore, our algorithm achieves the optimal linear combination of doubly exponential number of FS predictors, whereas the conventional methods can only track the best expert and achieves its performance in an asymptotic manner.

A. Sequential Combination of FS Predictors

Suppose we construct all possible FS predictors $\hat{d}_t^{\{k\}}$, $k=1,\ldots,K$ and run them in parallel to predict d_t. When used with the EG algorithm to combine the outputs of all FS predictors to produce the final output

$$\hat{d}_t \triangleq \sum_{k=1}^{K} w_t^{\{k\}}\hat{d}_t^{\{k\}}, \tag{9}$$

the mixture algorithm has the performance

$$\sum_{t=1}^{n}\ell(d_t,\hat{d}_t) \le \min_{\|\boldsymbol{w}\|=1}\sum_{t=1}^{n}\ell(d_t,\boldsymbol{w}^T\hat{\boldsymbol{d}}_t) + O\left(\sqrt{n\log K}\right), \tag{10}$$

where the combination weights are recursively calculated as

$$w_t^{\{k\}} = \frac{w_{t-1}^{\{k\}}\exp\left(-\mu\epsilon_{t-1}\hat{d}_{t-1}^{\{k\}}\right)}{\sum_{r=1}^{K} w_{t-1}^{\{r\}}\exp\left(-\mu\epsilon_{t-1}\hat{d}_{t-1}^{\{r\}}\right)}, \tag{11}$$

for $k=1,\ldots,K$, with $\epsilon_t \triangleq \ell'(d_t,\hat{d}_t)$ representing the first derivative of $\ell(d_t,\hat{d}_t)$ with respect to $\hat{d}_t$ and $\mu>0$ representing a positive constant controlling the learning rate. The mixture algorithm achieves the performance in (10) for any n without any knowledge on the optimal $\hat{d}_t^{\{k\}}$, the future values of the sequences, and the data length n, where $\hat{\boldsymbol{d}}_t \triangleq [\hat{d}_t^{\{1\}},\ldots,\hat{d}_t^{\{K\}}]^T$ and $\ell(\cdot,\cdot)$ is a convex loss function with Lipschitz gradient. We emphasize that there exist various different extensions of the update in (11). Yet, our derivations straightforwardly generalize to these updates [8] as shown in Section III-C.

Although one can use various cost functions $\ell(\cdot,\cdot)$ in (11), a widely used one is $\ell(d_t,\hat{d}_t)=(d_t-\hat{d}_t)^2$ in many signal processing applications when no statistical information about the data is available [47]. On the other hand, in various stochastic settings, steady-state MSE can be more meaningful in terms of analyzing the convergence performance of the algorithm.

Theorem 1: If the random variables $\left\|\bar{\boldsymbol{d}}_t\right\|^2_{\bar{\boldsymbol{J}}_t^T\boldsymbol{\Sigma}\bar{\boldsymbol{J}}_t}$ and ϵ_t^2 are asymptotically uncorrelated, the weighted mixture algorithm in (9) achieves the following the steady-state MSE performance for sufficiently small learning rate

$$\mathrm{MSE} = \frac{2\sigma_n^2}{2-\mu\mathrm{Tr}\left\{\boldsymbol{\Lambda}\bar{\boldsymbol{J}}^T\right\}}, \tag{12}$$

where σ_n^2 is the variance of the weighted excess error at steady-state, $\boldsymbol{\Lambda}$ is a diagonal matrix containing the eigenvalues of the autocorrelation matrix of the FS predictor outputs, and $\bar{\boldsymbol{J}}$ is related to the eigen decomposition of this autocorrelation matrix and the Jacobian matrix of the logarithm of the unnormalized combination weights, $\boldsymbol{\Sigma}$ is the covariance of the noise, ϵ_t represents the first derivative of $\ell(d_t,\hat{d}_t)$ with respect to $\hat{d}_t$, and $\bar{\boldsymbol{d}}_t$ is a parameter linear in the estimation vector $\hat{\boldsymbol{d}}_t$.

Proof: The proof is given in Appendix A. ■

The MSE performance in (12) illustrates that we can decrease the steady-state MSE by decreasing the learning rate of the algorithm and achieve the performance of the MMSE optimal batch predictor, i.e., the best convex combination of the FS predictors chosen with the full knowledge of the observation sequence $\{x_t\}_{t=1}^n$. Thus, the Excess Mean Square Error (EMSE), i.e., the difference between our algorithm's MSE and the MMSE of the optimal batch predictor can be arbitrarily set according to the performance requirements of the application by tuning the learning rate of the algorithm.

Although (9) is guaranteed to achieve the performance of the optimal combination over doubly exponential number of experts (i.e., FS predictors), it is not possible to practically implement such an algorithm even for a moderate hierarchy depth. As an example, for the order preserving patterns example, even for a small history length such as $l=4$ and a small hierarchy depth such as $h=3$, we have $K=6562$ different FS predictors. However, in practical applications such a huge number of FS predictors cannot be implemented and run in parallel to combine their outputs. Hence, in this form, the algorithm (9) cannot be directly implemented since we need to run K different FS predictors in parallel and monitor their performances to construct (9). To solve this problem, we next introduce a method that implements (9) with complexity only linear in the hierarchy depth h.

B. Efficient Sequential Combination of FS Predictors

We first observe that although there are K different FS predictors, the states used by each of the FS predictors are unions of a relatively small number of equivalence classes, i.e., nodes. Consider the order pattern classes in Example 1 and Fig. 1 where the ordering patterns for a length 3 observation, i.e., the last 3 samples, are illustrated. In this example, the observation $(x_{t-2},x_{t-1},x_t)=(1,5,-1)$ corresponds to the equivalence classes of the node $c_{2,4}$ and its super nodes, i.e., $c_{1,2}$ and $c_{0,1}$, which corresponds to the orderings $(2,3,1)$, $(.,3,.)$, $(.,.,.)$ respectively, where the ordering is made in an ascending manner. The predictors illustrated on Fig. 3 uses only certain nodes, i.e., classes, corresponding to a disjoint partition of the observation space. We use $\hat{d}_t^{\{k\}}$ to refer to the prediction of the k^{th} FS predictor at time t and $\hat{d}_t^{\{c_{i,j}\}}$ to refer to the prediction at time t of the equivalence class corresponding to the node $c_{i,j}$ in the hierarchical graph. As an example, $\hat{d}_t^{\{1\}}$, given in Fig. 3, uses only the equivalence class predictors $\hat{d}_t^{\{c_{1,1}\}}$, $\hat{d}_t^{\{c_{2,3}\}}$, $\hat{d}_t^{\{c_{2,4}\}}$ and $\hat{d}_t^{\{c_{1,3}\}}$, which corresponds to the orderings $(.,.,3), (1,3,2), (2,3,1)$ and $(3,.,.)$ respectively. The FS predictor $\hat{d}_t^{\{1\}}$ only considers the location of the biggest sample value except the case when that location is the middle. Therefore, the FS predictor uses a mixture of level-1 and level-2 equivalence classes. If this FS predictor observes the sequence $(x_{t-2}, x_{t-1}, x_t) = (1, 5, -1)$ at time t, which indicates the ordering $(2,3,1)$, then $\hat{d}_t^{\{1\}}$ uses the state predictor $\hat{d}_t^{\{c_{2,4}\}}$ to give its final output as $\hat{d}_t^{\{1\}} = \hat{d}_t^{\{c_{2,4}\}}$.

Each FS predictor is constructed from the nodes of hierarchical graph such that the regions corresponding to these nodes are disjoint regions whose union gives the whole observation space, i.e., the region of the node $c_{0,1}$. During the construction of the hierarchical model, each node at the highest level in the hierarchical graph is connected to a single equivalence class at each of the lower levels. Each FS predictor is constructed through combination of disjoint nodes in the hierarchical graph. Consider Ex. 1 and Fig. 1, and suppose that at time t, we observed the pattern $(x_{t-2}, x_{t-1}, x_t) = (1, 5, -1)$, which is included in the equivalence classes $c_{2,4} = (2,3,1)$ at the highest level, i.e., level-2. This equivalence class is connected to the equivalence classes $c_{1,2} = (\cdot, 3, \cdot)$ at level-1 and $c_{0,1} = (\cdot,\cdot,\cdot)$ at level-0. Thus, the nodes $c_{2,4}$, $c_{1,2}$, $c_{0,1}$ are all correspond to the region of the observation space including the sample pattern $(2,3,1)$. Hence, each FS predictor only includes one of these three nodes, i.e., states, in its prediction. In other words, an FS predictor cannot have more than one of these equivalence classes since these equivalence classes overlap with each other. Since there are $h+1$ levels in the hierarchical graph such that there is a hierarchical relationship between $h+1$ nodes (states) from the top level to the bottom level, each observation falls within the region of $h+1$ nodes that exist in different levels of the hierarchical graph. Hence, an FS predictor can only use one of these nodes in its prediction since these nodes are super sets of one another.

Remark 1: Each FS predictors uses the output of only one of the $h+1$ different equivalence class predictors based on the current state.

Thus, although the summation (9) is over K terms, there are actually $h+1$ unique equivalence class predictors (each comes from a different hierarchy level) to be combined. However, we emphasize that although we use only $h+1$ different equivalence class predictors, the sequential algorithm in (9) requires all K weights in (11). Since all FS predictors have different states, their weights $w_t^{\{k\}}$ are different. In the following, we will show that both the summation in (9) and weight calculations in (11) can be efficiently implemented yielding the algorithm in Algorithm 1, which illustrates a method to calculate (9) and (11) in $O(h)$ computation.

To illustrate this, we first introduce a technique to recursively calculate the total sum in the denominator of (11). Based on this recursion, we next introduce methods to calculate the numerator of (11) and to perform a sequential update of the combined loss. This sequential formulation is then proven to be able to produce the exact same output as in (9), however with a significantly reduced computational cost, i.e., only linear in the hierarchy depth h.

1) A Recursive Calculation of the Denominator of (11): We first note that after some algebra, we can write (11) as follows

$$w_t^{\{k\}} = \frac{w_0^{\{k\}} \exp\left(-\mu \sum_{\tau=1}^{t-1} \epsilon_\tau \hat{d}_\tau^{\{k\}}\right)}{\sum_{r=1}^{K} w_0^{\{r\}} \exp\left(-\mu \sum_{\tau=1}^{t-1} \epsilon_\tau \hat{d}_\tau^{\{r\}}\right)}. \tag{13}$$

Then, by assigning equal prior weight to each FS predictor, i.e., $w_0^{\{k\}} = 1/K$ for all $k = 1, \ldots, K$, we obtain

$$w_t^{\{k\}} = \frac{\exp\left(-\mu \sum_{\tau=1}^{t-1} \epsilon_\tau \hat{d}_\tau^{\{k\}}\right)}{\sum_{r=1}^{K} \exp\left(-\mu \sum_{\tau=1}^{t-1} \epsilon_\tau \hat{d}_\tau^{\{r\}}\right)}. \tag{14}$$

Here, we represent the sum in the denominator of (14) as follows

$$L_t \triangleq \sum_{k=1}^{K} L_t^{\{k\}}, \tag{15}$$

where

$$L_t^{\{k\}} \triangleq \exp\left(-\mu \sum_{\tau=1}^{t-1} \epsilon_\tau \hat{d}_\tau^{\{k\}}\right) \tag{16}$$

represents the total loss of the kth FS predictor at time t. We then define a function of loss for each equivalence class (cf. Fig. 1) as follows

$$L_t^{\{c_{i,j}\}} \triangleq \exp\left(-\mu \sum_{\tau=1}^{t-1} I_\tau^{\{c_{i,j}\}} \epsilon_\tau \hat{d}_\tau^{\{c_{i,j}\}}\right), \tag{17}$$

where the indicator function $I_\tau^{\{c_{i,j}\}}$ is defined as in (6). $L_t^{\{c_{i,j}\}}$ defined in (17) is the cumulative loss acquired from the region of the observation space corresponding to the node $c_{i,j}$. Note that, every node at the same level of the hierarchical graph covers disjoint regions whose union gives the whole observation space. The initial value of the exponentiated loss at time t, $L_t^{\{c_{i,j}\}}$, of the node $c_{i,j}$, which is the j^{th} node of the i^{th} level in the hierarchical graph, is 1, i.e., $L_1^{\{c_{i,j}\}} = 1$ for all $c_{i,j}$ in the graph by definition.

According to these definitions, we observe that each $L_t^{\{k\}}$ can be written as a product of the errors of its equivalence class predictors. Then, letting $\mathcal{C}^{\{k\}}$ denote the states (or equivalence classes) of the kth FS predictor, we have

$$L_t^{\{k\}} = \prod_{c_{i,j} \in \mathcal{C}^{\{k\}}} L_t^{\{c_{i,j}\}}. \tag{18}$$

As an example, for the first FS predictor in Fig. 3, we have $\mathcal{C}^{\{1\}} = \{c_{1,1}, c_{2,3}, c_{2,4}, c_{1,3}\}$. Based on this observation in (18), we next use a recursive formulation in order to efficiently calculate the sum L_t in (15). To accomplish this, we start from the highest hierarchy level and go to the lower hierarchy levels by recursively defining intermediate parameters that are a function of the total accumulated loss as follows

$$T_t^{\{c_{i,j}\}} \triangleq L_t^{\{c_{i,j}\}} + \prod_{c_{i+1,k} \in \mathcal{C}^{\{c_{i,j}\}}} T_t^{\{c_{i+1,k}\}}, \tag{19}$$

for each equivalence class $c_{i,j}$, where $\mathcal{C}^{\{c_{i,j}\}}$ represents the set of equivalence classes at hierarchy level $i+1$ that are connected to $c_{i,j}$. As an example, for the equivalence class $c_{0,1}$, we have $\mathcal{C}^{\{c_{0,1}\}} = \{c_{1,1}, c_{1,2}, c_{1,3}\}$. $T_t^{\{c_{i,j}\}}$ is the intermediate variable that gives the effective cumulative loss of each node in the hierarchical graph. The recursion in (19) is true for all the nodes except the highest level, i.e., level h, since at the highest level we have the finest nodes whose intermediate variables are given by the simple equation

$$T_t^{\{c_{h,j}\}} \triangleq L_t^{\{c_{h,j}\}}. \tag{20}$$

Using the recursive equations in (19) and (20), one can find the initial values $T_1^{\{c_{i,j}\}}$. As an example, consider the hierarchical graph given in Fig. 1, the intermediate variables at the top level, i.e., level-2, are given by $T_1^{\{c_{2,j}\}} = 1,\ j \in \{1,2,3,4,5,6\}$. The initial intermediate variables of the nodes at level-1, are given by $T_1^{\{c_{1,j}\}} = 2,\ j \in \{1,2,3\}$ and the initial value at the bottom node is given by $T_1^{\{c_{0,1}\}} = 9$. Note that, the initial value of $T_1^{\{c_{0,1}\}}$ gives the total number of FS predictors used in the Online Hierarchical Predictor. The initial values of $\widetilde{T}_t^{\{c_{i,j}\}}$ are dependent on the initial prediction conditions which are whatever is set in the constructed algorithm.

The following lemma illustrates that these intermediate parameters can be used to recursively calculate the denominator of (14).

Lemma 1: The recursions in (18) and (19) yield $T_t^{\{c_{0,1}\}} = L_t$.

Proof: The proof is given in Appendix B. ■

We next use Lemma 1 and the recursive relationship in (19) to efficiently calculate the final output of the algorithm in (9).

2) Construction of the Final Predictor $\hat{d}_t$ in (9): Using Lemma 1 in (14) and putting (14) in (9), we obtain

$$\begin{aligned}\hat{d}_t &= \sum_{k=1}^{K} w_t^{\{k\}} \hat{d}_t^{\{k\}} \\ &= \sum_{k=1}^{K} \frac{\exp\left(-\mu \sum_{\tau=1}^{t-1} \epsilon_\tau \hat{d}_\tau^{\{k\}}\right)}{\sum_{r=1}^{K} \exp\left(-\mu \sum_{\tau=1}^{t-1} \epsilon_\tau \hat{d}_\tau^{\{r\}}\right)} \hat{d}_t^{\{k\}} \\ &= \frac{1}{L_t} \sum_{k=1}^{K} L_t^{\{k\}} \hat{d}_t^{\{k\}}. \end{aligned} \tag{21}$$

In order to compactly represent the term inside the sum in (21), we introduce another intermediate parameter as follows

$$\widetilde{T}_t^{\{c_{i,j}\}} \triangleq L_t^{\{c_{i,j}\}} \hat{d}_t^{\{c_{i,j}\}} + \widetilde{T}_t^{\{c_{i+1,j'}\}} \prod_{\substack{c_{i+1,k} \in \mathcal{C}^{\{c_{i,j}\}} \\ c_{i+1,k} \neq c_{i+1,j'}}} T_t^{\{c_{i+1,k}\}}, \tag{22}$$

for all $c_{i,j}$ that contain the current pattern (i.e., $\forall c_{i,j} : s_t \in c_{i,j}$), where $s_t \in c_{i+1,j'}$. In the following lemma, we show that the intermediate parameter in (22) can be used to efficiently calculate the final output of the algorithm.

Lemma 2: The recursions in (18), (19), and (22) yield

$$\widetilde{T}_t^{\{c_{0,1}\}} = \sum_{k=1}^{K} L_t^{\{k\}} \hat{d}_t^{\{k\}}, \tag{23}$$

with overall computational complexity of $O(h)$.

Proof: The proof is given in Appendix C. ■

Using Lemma 2, we can construct the final output of our algorithm $\hat{d}_t$ in (9) as follows

$$\hat{d}_t = \frac{\widetilde{T}_t^{\{c_{0,1}\}}}{T_t^{\{c_{0,1}\}}}. \tag{24}$$

According to Lemma 2, $\widetilde{T}_t^{\{c_{0,1}\}}$ can be calculated with computational complexity $O(h)$. Therefore, if we can find a recursive method to calculate the denominator of (24), i.e., $T_t^{\{c_{0,1}\}}$, using the past values $T_{t-1}^{\{c_{i,j}\}}$, then we can sequentially obtain $\hat{d}_t$ at each time t with a computational complexity $O(h)$. In the following section, we address this recursion.

3) Sequential Calculation of $T_{t+1}^{\{c_{0,1}\}}$ Using (19): Suppose we performed our prediction $\hat{d}_t$, then d_t is revealed and we calculated $\epsilon_t = \ell'(d_t, \hat{d}_t)$. Our task is now to calculate $T_{t+1}^{\{c_{0,1}\}}$ from $T_t^{\{c_{0,1}\}}$ with a computational complexity $O(h)$. Naturally, a recursive formulation for $T_{t+1}^{\{c_{0,1}\}}$ also holds as in (19), where we have the terms $T_{t+1}^{\{c_{i,j}\}}$ and $L_{t+1}^{\{c_{i,j}\}}$ instead of the terms $T_t^{\{c_{i,j}\}}$ and $L_t^{\{c_{i,j}\}}$, respectively. However, from time t to $t+1$, due to the indicator function in (17), only the equivalence classes that contain the state s_t are effected by this update. Therefore, we have

$$L_{t+1}^{\{c_{i,j}\}} = \begin{cases} L_t^{\{c_{i,j}\}} \exp\left(-\mu \epsilon_t \hat{d}_t^{\{c_{i,j}\}}\right), & \text{if } s_t \in c_{i,j} \\ L_t^{\{c_{i,j}\}}, & \text{otherwise} \end{cases}, \tag{25}$$

and similarly we also have $T_{t+1}^{\{c_{i,j}\}} = T_t^{\{c_{i,j}\}}, \forall c_{i,j} : s_t \notin c_{i,j}$. Hence, according to (19), we have

$$\begin{aligned} T_{t+1}^{\{c_{i,j}\}} &= L_t^{\{c_{i,j}\}} \exp\left(-\mu \epsilon_t \hat{d}_t^{\{c_{i,j}\}}\right) \\ &\quad + T_{t+1}^{\{c_{i+1,j'}\}} \prod_{\substack{c_{i+1,k} \in \mathcal{C}^{\{c_{i,j}\}} \\ c_{i+1,k} \neq c_{i+1,j'}}} T_{t+1}^{\{c_{i+1,k}\}}, \end{aligned} \tag{26}$$

$\forall c_{i,j} : s_t \in c_{i,j}$, where $c_{i+1,j'} \in \mathcal{C}^{\{c_{i,j}\}} : s_t \in c_{i+1,j'}$.

Lemma 3: The recursions in (25) and (26) can be calculated in $O(h)$ operations.

Proof: The proof is given in Appendix D. ■

This lemma concludes that (24) can be efficiently calculated without any approximations with a computational complexity of $O(h)$. Hence, the introduced algorithm achieves the same deterministic and stochastic performance guarantees as the original algorithm, whose computational complexity is doubly exponential in h.

4) Summary of the Algorithm: The summary of these steps for the generic case can be outlined as follows (and the complete description of the algorithm is given in Algorithm 1). At time t, we have the variables $T_t^{\{c_{i,j}\}}$, $L_t^{\{c_{i,j}\}}$ and the predictions $\hat{d}_t^{\{c_{i,j}\}}$ for each equivalence class in the entire state diagram.

- After we find the current state s_t, we recursively calculate $\widetilde{T}_t^{\{c_{i,j}\}}$, $\forall c_{i,j} : s_t \in c_{i,j}$ starting from the hierarchy level $i = h-1$ to get the numerator of (21). After this recursion, the output of our algorithm is found as

Algorithm 1: Online Hierarchical Predictor.

```
1: % Initialization: $L_0^{\{c_{i,j}\}} = 1$, calculate $T_0^{\{c_{i,j}\}}$.
2: for $t = 1$ to $T$ do
3:    % Find the current state $s_t$ by $s_t = g(s_{t-1}, x_t)$.
4:    % Find the set of equivalence classes $\mathcal{E}_t$ that contain
      the current state $s_t$, i.e., $\mathcal{E}_t = \{c_{i,j} : s_t \in c_{i,j}\}$.
5:    % Prediction
6:    for all $c_{i,j} \in \mathcal{E}_t$ (from $i = h-1$ to 0) do
7:       if $i = h - 1$ then
8:          $\widetilde{T}_t^{\{c_{i,j}\}} = L_t^{\{c_{i,j}\}} \hat{d}_t^{\{c_{i,j}\}}$
9:       else
10:         $P_t^{\{c_{i,j}\}} = \left(T_t^{\{c_{i,j}\}} - L_t^{\{c_{i,j}\}}\right) / T_t^{\{c_{i+1,j'}\}}$
11:         $\widetilde{T}_t^{\{c_{i,j}\}} = L_t^{\{c_{i,j}\}} \hat{d}_t^{\{c_{i,j}\}} + \widetilde{T}_t^{(c_{i+1,j'})} P_t^{\{c_{i,j}\}}$
12:      end if
13:   end for
14:   $\hat{d}_t = \widetilde{T}_t^{\{c_{0,1}\}} / T_t^{\{c_{0,1}\}}$
15:   $\epsilon_t = \ell'(d_t, \hat{d}_t)$
16:   % Update
17:   for all $c_{i,j} \in \mathcal{E}_t$ (from $i = h-1$ to 0) do
18:      % Update $\hat{d}_t^{\{c_{i,j}\}}$ as desired such as in (5).
19:      $L_{t+1}^{\{c_{i,j}\}} = L_t^{\{c_{i,j}\}} \exp\left(-\mu \epsilon_t \hat{d}_t^{\{c_{i,j}\}}\right)$
20:      if $i = h - 1$ then
21:         $T_{t+1}^{\{c_{i,j}\}} = L_{t+1}^{\{c_{i,j}\}}$
22:      else
23:         $T_{t+1}^{\{c_{i,j}\}} = L_{t+1}^{\{c_{i,j}\}} + T_{t+1}^{\{c_{i+1,j'}\}} P_t^{\{c_{i,j}\}}$
24:      end if
25:   end for
26: end for
```

$\hat{d}_t = \widetilde{T}_t^{\{c_{0,1}\}} / T_t^{\{c_{0,1}\}}$, which, in total, can be found in $O(h)$ calculations (as $T_t^{\{c_{0,1}\}}$ is calculated at the previous step).

- After the true output d_t is revealed, the variables $L_{t+1}^{\{c_{i,j}\}}$ and $T_{t+1}^{\{c_{i,j}\}}$ should be updated. For the equivalence classes that contain the current pattern (in total, $h+1$ of them), the updates of $L_{t+1}^{\{c_{i,j}\}}$ and $T_{t+1}^{\{c_{i,j}\}}$ are done using the recursions in (25) and (26), where we emphasize that for the equivalence classes that do not include the current state, no update is necessary.
- Lastly, for the equivalence classes that contain the current state, the equivalence class predictions $\hat{d}_{t+1}^{\{c_{i,j}\}}$ can be updated using any desired method, such as the one in (5).

In the following two subsections, we extend our discussions for different linear combination weights and different cost functions, respectively.

C. Positive and Negative Weights

In many mixture-of-experts frameworks, the weighting parameters are usually restricted to be positive and sum up to 1 as in the case for the EG algorithm (cf. (14)). In order to overcome this limitation, we can use the EG algorithm with positive and negative weights [8]. To this end, we consider the output of each FS predictor, say $\hat{d}_t^{\{k\}}$, and instead of directly scaling this value with a weighting parameter, we consider $\beta \hat{d}_t^{\{k\}}$ and $-\beta \hat{d}_t^{\{k\}}$ as the outputs of two different experts. We then scale these values by the weighting parameters $w_{t,+}^{\{k\}}$ and $w_{t,-}^{\{k\}}$, respectively.

Hence, we obtain the output of the final predictor as follows

$$\hat{d}_t = \sum_{k=1}^{K} \left(w_{t,+}^{\{k\}} - w_{t,-}^{\{k\}}\right) \hat{d}_t^{\{k\}}, \tag{27}$$

where the combination weights are recursively calculated as follows

$$w_{t+1,\pm}^{\{k\}} = \beta \frac{w_{t,\pm}^{\{k\}} \exp\left(\mp \mu \beta \epsilon_t \hat{d}_t^{\{k\}}\right)}{\sum_{r=1}^{K} w_{t,+}^{\{r\}} \exp\left(-\mu \beta \epsilon_t \hat{d}_t^{\{r\}}\right) + w_{t,-}^{\{r\}} \exp\left(\mu \beta \epsilon_t \hat{d}_t^{\{r\}}\right)}. \tag{28}$$

In this manner, while we have $w_{t,+}^{\{k\}}, w_{t,-}^{\{k\}} > 0$, $\sum_{k=1}^{K} w_{t,+}^{\{k\}} = \beta$, and $\sum_{k=1}^{K} w_{t,-}^{\{k\}} = \beta$, the resulting combination weights in (27), i.e., $w_{t,+}^{\{k\}} - w_{t,-}^{\{k\}}$ can take any value satisfying $\sum_{k=1}^{K} \left|w_{t,+}^{\{k\}} - w_{t,-}^{\{k\}}\right| \leq \beta$.

Following similar lines to Section III-B, we define

$$L_t \triangleq L_{t,+} + L_{t,-}, \tag{29}$$

where

$$L_{t,\pm} \triangleq \exp\left(\mp \mu \beta \sum_{z=1}^{t} \epsilon_t \hat{d}_t^{\{k\}}\right). \tag{30}$$

After defining the equivalence class losses and recursion parameters as in Section III-B, we obtain independent recursions over parameters $L_{t,\pm}^{\{c_{i,j}\}}$ and $T_{t,\pm}^{\{c_{i,j}\}}$, where $L_{t,\pm}^{\{c_{i,j}\}}$ represents the loss of the equivalence class $c_{i,j}$ for the positive and negative weights, respectively, which is defined similar to (17) and the parameters $T_{t,\pm}^{\{c_{i,j}\}}$ can be calculated with independent recursions as follows

$$T_{t,\pm}^{\{c_{i,j}\}} = L_{t,\pm}^{\{c_{i,j}\}} + \prod_{c_{i+1,k} \in \mathcal{C}^{\{c_{i,j}\}}} T_{t,\pm}^{\{c_{i+1,k}\}}. \tag{31}$$

Using these definitions, one can obtain the final algorithm after following similar lines to Section III-B.

D. Implementation of the Algorithm With Forgetting Factor

In this section, we consider the weighted loss as our cost function, i.e.,

$$\sum_{t=1}^{n} \lambda^{n-t} \, \ell(d_t, \hat{d}_t), \tag{32}$$

where $0 < \lambda < 1$ represents the forgetting factor. We emphasize that the objective function in (32) is used in various different applications, especially when the aim is to track a drifting parameter [47]. However, directly using (32) as our objective function may significantly deteriorate the performance of the algorithm since $\hat{d}_t$ is generated according to the current state s_t and even for a moderate h, the time difference between two consecutive appearances of the same state may take significantly long time, e.g., for the order preserving patterns scenario this recurrence time is $\sim (h/e)^h$. Therefore, instead of using (32) in its current form, we use different forgetting factors for each equivalence class as follows

$$\sum_{t_\tau \in \mathcal{T}_n^{\{c_{i,j}\}}} \lambda^{|\mathcal{T}_n^{\{c_{i,j}\}}|-\tau} \ell(d_t, \hat{d}_t), \tag{33}$$

where $\mathcal{T}_t^{\{c_{i,j}\}}$ represents the time instances (up to t) at which the current state is included in the equivalence class $c_{i,j}$, i.e., $\mathcal{T}_t^{\{c_{i,j}\}} \triangleq \{1 \le t_\tau \le t : s_t \in c_{i,j}\}$, and $t_\tau \in \mathcal{T}_t^{\{c_{i,j}\}}$ represents the τ^{th} smallest value in the set $\mathcal{T}_t^{\{c_{i,j}\}}$. By this formulation we can overcome the aforementioned recurrence time issue and also assign different forgetting factors for each equivalence class $c_{i,j}$, i.e., using $\lambda_{c_{i,j}}$ instead of λ.

Owing to the comprehensive structure of the introduced algorithm such recursive cost functions can be directly incorporated in our framework. To this end, we can define our new total loss function as follows

$$L_t = \sum_{k=1}^{K} L_t^{\{k\}}, \tag{34}$$

where

$$L_t^{\{k\}} = \exp\left(-\mu \sum_{\tau=1}^{t-1} \lambda^{t-\tau-1} \epsilon_\tau \hat{d}_\tau^{\{k\}}\right) \tag{35}$$

represents the total weighted loss of the kth FS predictor at time t. Following similar lines to Section III-B, we define $T_t^{\{c_{i,j}\}}$ similar to (19). Then, the recursion of the parameter $L_t^{\{c_{i,j}\}}$ is updated as follows

$$L_{t+1}^{\{c_{i,j}\}} = \exp\left(\lambda \ln L_t^{\{c_{i,j}\}}\right) \exp\left(-\mu \epsilon_t \hat{d}_t^{\{c_{i,j}\}}\right), \tag{36}$$

$\forall c_{i,j} : s_t \in c_{i,j}$, whereas the parameter $T_t^{\{c_{i,j}\}}$ is updated as in (26). Note that for the equivalence classes that do not contain the current state, no update is necessary. Following similar steps to Section III-B, one can construct the desired algorithm after some algebra.

IV. Simulations

In this section, we illustrate the performance of our algorithm in various scenarios. Throughout this section, we set $\mu = 1$ for a fair performance comparison between our algorithm and its competitors. The code used in the experiments and all of the data are accessible from http://www.ee.bilkent.edu.tr/∼vanli.

A. Real Life Energy Profile Forecasting

In this experiment, we consider prediction of the energy consumption in Turkish energy markets using real data.[1] Particularly, we forecast the energy consumption of consumers using their past consumption patterns, where the aim is to predict the consumption trend such that $d_t = 1$ if $x_{t+1} \ge x_t$ and $d_t = -1$, otherwise, i.e., we try to forecast an increasing or decreasing trend in the energy consumption patterns using real data collected in Turkish markets. Note that this scenario perfectly matches with the example framework we have illustrated throughout the paper. In this experiment, we illustrate the performance guarantee in Theorem 1 and the mitigation of undertraining as well as convergence issues by our algorithm. We set $h = 4$ for this real life experiment. In Fig. 4, the time averaged square error performances of the proposed algorithms are compared, where "OHP" represents the online hierarchical predictor introduced in this paper and "$h = i$" represents the predictor using all equivalence classes at the ith hierarchy level as its states (e.g., see example hierarchy levels in Fig. 1 for $h = 2$).

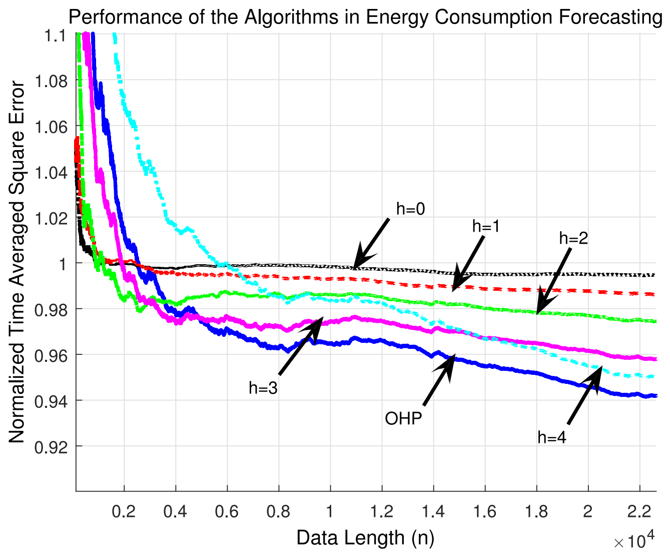


Fig. 4. Normalized time averaged square errors of the proposed algorithms for the real life electricity consumption data.

Fig. 4 illustrates that the performance of the OHP algorithm is comparable with the performances of the coarser predictors (e.g., $h = 0$ and $h = 1$) when there is not sufficient amount of data to train finer energy consumption patterns (equivalence classes). However, as the data length increases, the performances of the coarser predictors deteriorate with respect to the predictors having finer equivalence classes (e.g., $h = 3$ and $h = 4$). Nevertheless, the performance of the OHP algorithm is still better than the finest predictor even after a significantly large amount of observations.

We emphasize that as the pattern order h increases or when the underlying data is highly nonstationary, the convergence performance of the OHP algorithm will significantly outperform the performance of the finer predictors since the finer predictors may not be able to observe enough training sequences to achieve a satisfactory performance. This result is also apparent in Fig. 4, where over short data sequences the performance of the finer predictors is worse compared to the coarser predictors and the OHP algorithm. Hence, the OHP algorithm outperforms the constituent FS predictors by exploiting the time-dependent nature of the best choice among constituent FS predictors that are defined on the hierarchical structure.

B. Synthetic MSE Analysis

In this section, we illustrate the steady-state MSE performance of our algorithm. To this end, we consider the following

[1]This data set can be achieved from http://www.ee.bilkent.edu.tr/∼vanli

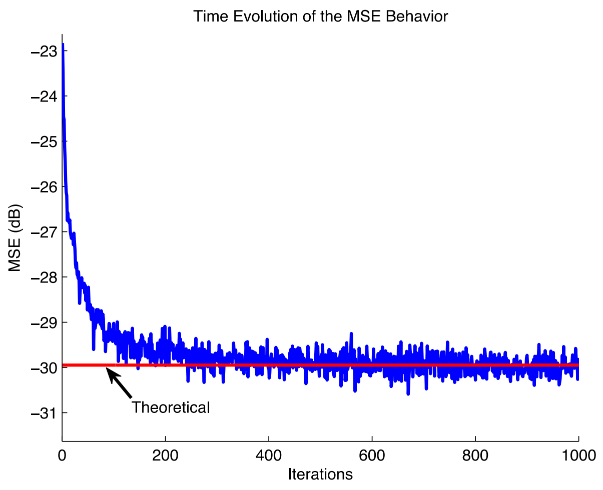


Fig. 5. The experimental MSE of the proposed algorithm converges to the theoretical steady-state MSE performance. The results are averaged over 500 independent trials.

prediction model

$$x_t = 0.5x_{t-1} + 0.5x_{t-2} + n_t, \qquad (37)$$

where n_t represents the realization of the i.i.d. additive Gaussian noise with zero mean and variance $\sigma_n^2 = 10^{-3}$ at time t.

For our state selection, we have considered the observation space $x_{t-1} \times x_{t-2}$, i.e., the past data of the given prediction model. We have used a depth $h = 2$ partitioning and this observation space has been partitioned such that partitions at the highest level, i.e., finest partitions are given by the regions of the space $(x_{t-1} \geq 0, x_{t-2} \geq 0)$, $(x_{t-1} \geq 0, x_{t-2} < 0)$, $(x_{t-1} < 0, x_{t-2} \geq 0)$, $(x_{t-1} < 0, x_{t-2} < 0)$ respectively. The unbounded space of $x_{t-1} \times x_{t-2}$ has been split into these disjoint four regions according to the boundaries given and the finest nodes at the top level are created. Then, at the lower level, i.e., $h = 1$, we combine them such that we create the regions $x_{t-1} \geq 0$ and $x_{t-1} < 0$ and match them with the corresponding nodes. At the bottom level, as usual, the node for the whole observation space is created.

The four regions corresponding to the finest nodes in our hierarchical graph are assigned a linear predictor. The combination of these four nodes at the lower levels are also assigned a linear predictor. The nodes in the hierarchical graph are trained only using the past observations corresponding to their partition space, i.e., the linear predictors of the top level nodes are trained only with the samples in their respective quadrants, while the linear predictors of the nodes at level-1 are trained with the samples corresponding to the left and right half planes. The linear predictor of the mother node at the bottom is trained by using all of the past observation samples. Except the bottom node, each different combination of the nodes at the levels 1 and 2 jointly represent a piecewise linear predictor, while the bottom node represents a completely linear predictor. These nodes are then trained and updated using our algorithm, and the combination gradually converges to the true linear model given in (37). We emphasize that such piecewise linear prediction scenarios are extensively studied in the literature, cf. [16], [30], [48] (and references therein).

In Fig. 5, we illustrate the theoretical and experimental MSE results averaged over 500 independent trials. This figure shows that the actual MSE behavior of our algorithm can be quite accurately represented by the theoretical steady-state MSE result in (12).

C. SETAR Time Series Prediction

In this set of experiments, we consider the prediction of the signals generated from self-exciting threshold autoregressive (SETAR) models. As the first experiment, we consider the following SETAR model

$$x_t = \begin{cases} 1.71x_{t-1} - 0.81x_{t-2} + 0.356 + \varepsilon_t, & \text{if } x_{t-1} > 0 \\ -0.562x_{t-2} - 3.91 + \varepsilon_t, & \text{otherwise} \end{cases}, \qquad (38)$$

where ε_t represents the realization of the i.i.d. additive Gaussian noise with zero mean and unit variance at time t. Note that this SETAR model is used in various other papers, e.g., [48]. Here, we normalize the both dimensions of this space between $[-1, 1]$ to provide a fair comparison between algorithms. Note that, this normalization effectively reduces the power of ε_t.

In the hierarchical model, we have partitioned the observation space into $2^4 = 16$ equal partitions such that the normalized observation space of $x_{t-1} \times x_{t-2}$, which is $[-1, 1] \times [-1, 1]$, i.e., a square of edge length 2, has been partitioned into 16 identical squares of side 0.5. Then these regions have been assigned to the finest nodes at the top level and are combined in pairs (corresponding to the adjacent regions) in each level of the hierarchical graph to create a depth 4 hierarchy similar to a binary tree.

Throughout this section, "OHP" represents the online hierarchical predictor proposed in this paper, "CTW" represents the context tree weighting algorithm of [16], "VF" represents the Volterra filter [49], and "FNF" represents the Fourier nonlinear filter of [50]. We set the depths of the OHP and CTW algorithms to 4 and the order of the VF and FNF algorithms to 3 for a fair performance comparison (since the computational complexities of the OHP and CTW algorithms scale linearly with their depth, whereas the computational complexities of the VF and FNF algorithms scale exponentially with their order). To train the linear predictors at each node of the OHP and CTW algorithms and to update the VF and FNF algorithms, we use the recursive least squares (RLS) method [47].

In Fig. 6, we present the cumulative square errors of the OHP, CTW, VF, and FNF algorithms. This figure indicates that our algorithm has a significantly faster convergence rate especially with respect to other tree based learning methods such as the CTW algorithm. That is because, our algorithm achieves the optimal combination of the different FS predictors, whereas the CTW algorithm uses ad-hoc weights to achieve universal performance guarantees. Furthermore, the OHP algorithm achieves a significantly lower square error performance with respect to its competitors.

To illustrate the convergence rate of our algorithm, we generate a time series of length 5000 using the SETAR model in (38) and then switch to the following SETAR model

$$x_t = \begin{cases} -1.71x_{t-1} + 0.81x_{t-2} - 0.356 + \varepsilon_t, & \text{if } x_{t-1} \leq 0 \\ 0.562x_{t-2} + 3.91 + \varepsilon_t, & \text{otherwise} \end{cases}, \qquad (39)$$

and generate an additional time series of length 5000. The dataset has again been normalized to $[-1, 1]$. However, since

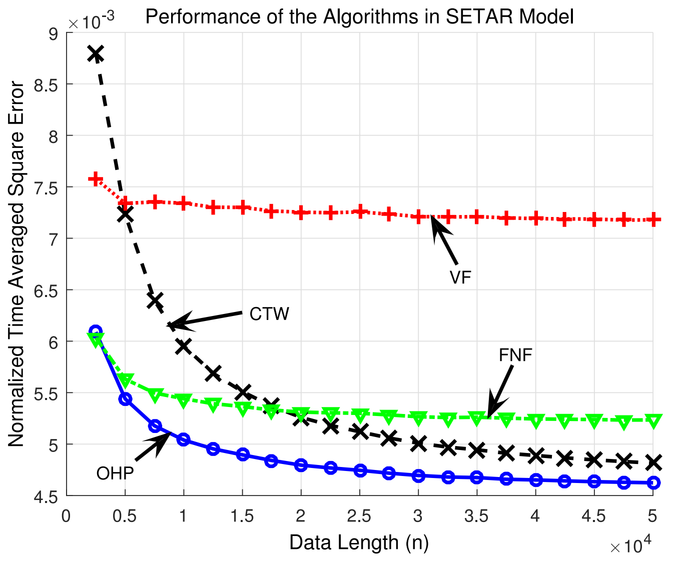


Fig. 6. Normalized time averaged square errors of the proposed algorithms for the SETAR model in (38).

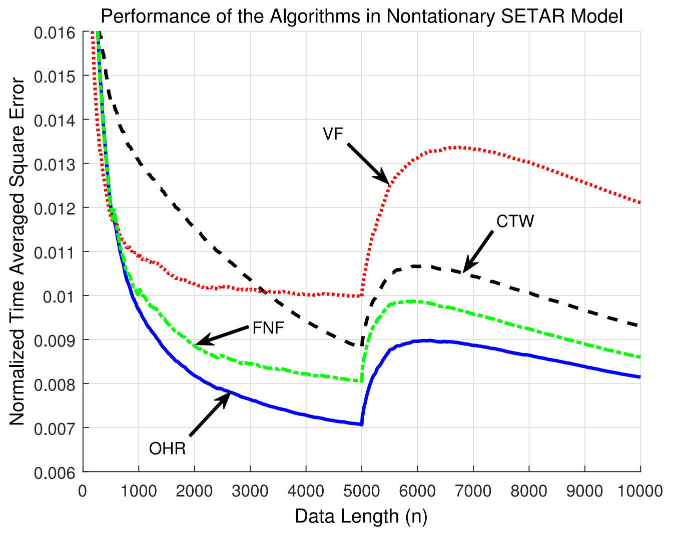


Fig. 7. Normalized time averaged square errors of the proposed algorithms for the SETAR models in (38) and (39). Here, the first 5000 samples of the data are generated using (38), whereas the last 5000 are generated using (39).

we created the dataset from two different SETAR models, the normalization factor is different from previous case, hence the power of ε_t is different. For the hierarchical model, we have partitioned the space similarly to the stationary SETAR model and created the hierarchical graph accordingly.

In Fig. 7, we present the cumulative square errors of the proposed algorithms. This figure illustrates that the convergence rate of our algorithm is exceptionally better than ones of the competitor algorithms. Particularly, our algorithm presents a faster convergence compared to the similar tree based methods such as the CTW algorithm. That is because, our algorithm is based on the EG method, which dynamically updates the combination weights (owing to the exponentiated steps), whereas the CTW algorithm cannot perform such quick adaptations due to its universality over the entire data history. In this sense, our algorithm is highly efficient for applications involving nonstationary data.

V. Conclusion

In this paper, we introduce a sequential FS prediction algorithm for real valued sequences, where we construct hierarchical structures to define states. Instead of directly using the equivalence classes at the highest level of hierarchy, which can result in a prohibitively large number of states even for moderate hierarchy depths, we define hierarchical equivalence classes by recursively tying certain states to avoid undertraining problems. With this hierarchical equivalence class definitions, we construct a doubly exponential number (in the hierarchy depth) of FS predictors. By using the EG algorithm, we show that we can sequentially achieve the performance of the optimal combination of all FS predictors that can be defined on this hierarchical structure with a computational complexity only linear in the length of the hierarchy depth. Our results are generic such that they can be directly used with numerous weighting methods for a wide range of hierarchical equivalence class definitions, and hold for a wide range of loss functions.

Appendix A
Proof of Theorem 1

The weighted mixture algorithm in (9) has the weight update rule given by

$$w_t^{\{k\}} = \frac{w_0^{\{k\}} \exp\left(-\mu \sum_{\tau=1}^{t-1} \epsilon_\tau \hat{d}_\tau^{\{k\}}\right)}{\sum_{r=1}^{K} w_0^{\{r\}} \exp\left(-\mu \sum_{\tau=1}^{t-1} \epsilon_\tau \hat{d}_\tau^{\{r\}}\right)}. \quad (40)$$

Considering the weights at $t = 0$ are all unit weights, we define an intermediate variable $z_t^{\{k\}}$ such that the weights are given by

$$w_t^{\{k\}} = \frac{e^{-z_t^{\{k\}}}}{\sum_{r=1}^{K} e^{-z_t^{\{r\}}}},$$

where $z_t^{\{k\}}$ is defined as $z_t^{\{k\}} = -\mu \sum_{\tau=1}^{t-1} \epsilon_\tau \hat{d}_\tau^{\{k\}}$ for the k^{th} FS predictor. We also define correspondingly a function $\boldsymbol{w}_t = f(\boldsymbol{z}_t)$, where $\boldsymbol{w}_t \triangleq [w_t^{\{1\}}, \cdots, w_t^{\{K\}}]^T$ and $\boldsymbol{z}_t \triangleq [z_t^{\{1\}}, \cdots, z_t^{\{K\}}]^T$. The function $f(\boldsymbol{z_t})$ is the nonlinear transformation from the intermediate variables of the FS predictors to their EG weights. Then, the definition of $\boldsymbol{z}_t$ yields

$$\boldsymbol{z}_t = \boldsymbol{z}_{t-1} - \mu \hat{\boldsymbol{d}}_{t-1} \epsilon_{t-1}. \quad (41)$$

We then apply the Euler discretization technique [51] to (41) and obtain

$$\boldsymbol{w}_t = \boldsymbol{w}_{t-1} + \mu J\left(f(\boldsymbol{z}_{t-1})\right) \hat{\boldsymbol{d}}_{t-1} \epsilon_{t-1}, \quad (42)$$

where $J\left(f(\boldsymbol{z}_{t-1})\right)$ is the Jacobian matrix of $f(\boldsymbol{z}_{t-1})$ with respect to $\boldsymbol{z}_{t-1}$. In particular, $J\left(f(\boldsymbol{z}_t)\right)$ is given by

$$J\left(f(\boldsymbol{z}_t)\right) = \frac{1}{\left(\sum_{r=1}^{K} e^{-z_t^{\{r\}}}\right)^2} \left\{ \left(\sum_{r=1}^{K} e^{-z_t^{\{r\}}} \right) \boldsymbol{D}_t - \boldsymbol{Z}_t \right\}, \quad (43)$$

where

$$\boldsymbol{D}_t \triangleq \begin{bmatrix} e^{-z_t^{\{1\}}} & & \\ & \ddots & \\ & & e^{-z_t^{\{K\}}} \end{bmatrix} \tag{44}$$

and

$$\boldsymbol{Z}_t \triangleq \begin{bmatrix} \left(e^{-z_t^{\{1\}}}\right)^2 & e^{-z_t^{\{1\}}}e^{-z_t^{\{2\}}} & \cdots & e^{-z_t^{\{1\}}}e^{-z_t^{\{K\}}} \\ e^{-z_t^{\{2\}}}e^{-z_t^{\{1\}}} & \left(e^{-z_t^{\{2\}}}\right)^2 & \cdots & e^{-z_t^{\{2\}}}e^{-z_t^{\{K\}}} \\ \vdots & \vdots & \ddots & \vdots \\ e^{-z_t^{\{K\}}}e^{-z_t^{\{1\}}} & e^{-z_t^{\{K\}}}e^{-z_t^{\{2\}}} & \cdots & \left(e^{-z_t^{\{K\}}}\right)^2 \end{bmatrix}.$$

Letting

$$\boldsymbol{w}_o = \arg\min_{\boldsymbol{w}\in\Delta^K} \sum_{t=1}^{\infty} \ell\left(d_t, \hat{\boldsymbol{d}}_t^T \boldsymbol{w}\right), \tag{45}$$

where

$$\Delta^K \triangleq \left\{\boldsymbol{w}\in\mathbb{R}^K \mid w^{(i)} \geq 0\,\forall i \in \{1,\cdots,K\}, \|\boldsymbol{w}\|_1 = 1\right\}$$

denotes K-dimensional unit simplex, we define a deviation parameter $\tilde{\boldsymbol{w}}_t = \boldsymbol{w}_o - \boldsymbol{w}_t$. Then, (42) yields

$$\tilde{\boldsymbol{w}}_t = \tilde{\boldsymbol{w}}_{t-1} - \mu J\left(f(\boldsymbol{z}_{t-1})\right)\hat{\boldsymbol{d}}_{t-1}\epsilon_{t-1}. \tag{46}$$

Here, note that the outputs of the predictors are correlated. Hence, let the autocorrelation matrix of the outputs of the FS predictors at the steady-state be defined as

$$\boldsymbol{R} \triangleq \lim_{t\to\infty} E\left[\hat{\boldsymbol{d}}_t \hat{\boldsymbol{d}}_t^T\right], \tag{47}$$

where $\hat{\boldsymbol{d}}_t \triangleq [\hat{d}_t^{\{1\}}, \cdots, \hat{d}_t^{\{K\}}]^T$. We can decompose $\boldsymbol{R}$ through the eigen decomposition as follows $\boldsymbol{R} = \boldsymbol{T}\boldsymbol{\Lambda}\boldsymbol{T}^T$. Multiplying both sides of (46) by $\boldsymbol{T}^T$, we obtain

$$\bar{\boldsymbol{w}}_t = \bar{\boldsymbol{w}}_{t-1} - \mu \boldsymbol{T}^T J\left(f(\boldsymbol{z}_{t-1})\right)\boldsymbol{T}\bar{\boldsymbol{d}}_{t-1}\epsilon_{t-1}, \tag{48}$$

where $\bar{\boldsymbol{w}}_t \triangleq \boldsymbol{T}^T \boldsymbol{w}_t$ and $\bar{\boldsymbol{d}}_t \triangleq \boldsymbol{T}^T \hat{\boldsymbol{d}}_t$. For notational simplicity, we denote $\bar{\boldsymbol{J}}_{t-1} \triangleq \boldsymbol{T}^T J\left(f(\boldsymbol{z}_{t-1})\right)\boldsymbol{T}$. Then, the weighted energy recursion of (42) is given by

$$E\|\bar{\boldsymbol{w}}_t\|^2_{\boldsymbol{\Sigma}} = E\|\bar{\boldsymbol{w}}_{t-1}\|^2_{\boldsymbol{\Sigma}} - 2\mu E\left[\bar{\boldsymbol{w}}_{t-1}^T \boldsymbol{\Sigma}\bar{\boldsymbol{J}}_{t-1}\bar{\boldsymbol{d}}_{t-1}\bar{\boldsymbol{d}}_{t-1}^T\bar{\boldsymbol{w}}_{t-1}\right] + \mu^2 E\left[\epsilon_{t-1}^2 \bar{\boldsymbol{d}}_{t-1}^T \bar{\boldsymbol{J}}_{t-1}^T \boldsymbol{\Sigma}\bar{\boldsymbol{J}}_{t-1}\bar{\boldsymbol{d}}_{t-1}\right], \tag{49}$$

where $\boldsymbol{\Sigma}$ is a positive semi-definite weight matrix.

We assume that the random variables $\left\|\bar{\boldsymbol{d}}_t\right\|^2_{\bar{\boldsymbol{J}}_t^T \boldsymbol{\Sigma}\bar{\boldsymbol{J}}_t}$ and ϵ_t^2 are asymptotically uncorrelated. Hence, at steady-state, we have

$$\lim_{t\to\infty} E\|\bar{\boldsymbol{w}}_t\|^2_{\boldsymbol{\Sigma}\bar{\boldsymbol{J}}\boldsymbol{\Lambda}} = \lim_{t\to\infty} \frac{\mu}{2} E\left[\epsilon_t^2\right] \mathrm{Tr}\left\{\boldsymbol{\Lambda}\bar{\boldsymbol{J}}^T\boldsymbol{\Sigma}\bar{\boldsymbol{J}}\right\}, \tag{50}$$

where we approximate the Jacobian matrix as follows

$$\bar{\boldsymbol{J}} \triangleq \boldsymbol{T}^T J(f(\boldsymbol{z}_o))\boldsymbol{T}, \tag{51}$$

and define

$$\boldsymbol{z}_o \triangleq f(\boldsymbol{w}_0)^{-1}. \tag{52}$$

We note that $\epsilon_t = \bar{\boldsymbol{w}}_t^T \bar{\boldsymbol{d}}_t + n_t$, where

$$n_t = \sum_{k=1}^{K} w_o^{\{k\}} e_t^{\{k\}}, \tag{53}$$

and $e_t^{\{k\}} = d_t - \hat{d}_t^{\{k\}}$. This yields that

$$\lim_{t\to\infty} E\left[\epsilon_t^2\right] = \lim_{t\to\infty} E\|\bar{\boldsymbol{w}}_t\|^2_{\boldsymbol{\Lambda}} + E\left[n_t^2\right]. \tag{54}$$

Letting $\boldsymbol{\Sigma} = \bar{\boldsymbol{J}}^{-1}$ and

$$\sigma_n^2 = \lim_{t\to\infty} E[n_t^2], \tag{55}$$

we obtain the steady-state excess MSE $\zeta \triangleq \lim_{t\to\infty} E\|\bar{\boldsymbol{w}}_t\|^2_{\boldsymbol{\Lambda}}$ as

$$\zeta = \frac{\mu\sigma_n^2 \mathrm{Tr}\left\{\boldsymbol{\Lambda}\bar{\boldsymbol{J}}^T\right\}}{2 - \mu\mathrm{Tr}\left\{\boldsymbol{\Lambda}\bar{\boldsymbol{J}}^T\right\}}, \tag{56}$$

and the steady-state MSE is given by

$$\mathrm{MSE} = \frac{2\sigma_n^2}{2 - \mu\mathrm{Tr}\left\{\boldsymbol{\Lambda}\bar{\boldsymbol{J}}^T\right\}}. \tag{57}$$

Appendix B
Proof of Lemma 1

In order to prove $T_t^{\{c_{0,1}\}} = L_t$, we use mathematical induction. First, for $h = 1$, we have a single state and the equation $T_t^{\{c_{0,1}\}} = L_t$ is trivially satisfied for this case. Assuming that $T_t^{\{c_{0,1}\}} = L_t$ holds for some $h \geq 1$, let us consider the term $T_t^{\{c_{0,1}\}}$ for $h+1$.

For ease of exposition let us drop the time index t from the subscript, and use $T_h^{\{c_{i,j}\}}$ to refer to the induction hypothesis and $T_{h+1}^{\{c_{i,j}\}}$ to refer to the objective function. Similarly, we let $\mathcal{C}_{h+1}^{\{c_{i,j}\}}$ represent the set of equivalence classes at hierarchy level $i+1$ that are connected to $c_{i,j}$ for the induction case. According to the definition in (19), we have

$$T_{h+1}^{\{c_{0,1}\}} = L_{h+1}^{\{c_{0,1}\}} + \prod_{c_{1,j}\in\mathcal{C}_{h+1}^{\{c_{0,1}\}}} T_{h+1}^{\{c_{1,j}\}} \tag{58}$$

$$= L_{h+1}^{\{c_{0,1}\}} + \prod_{j=1}^{J_{h+1}} L_{h,j}, \tag{59}$$

where the last line follows from the induction hypothesis with $J_{h+1} \triangleq \left|\mathcal{C}_{h+1}^{\{c_{0,1}\}}\right|$ and $L_{h,j}$ representing the loss of the jth equivalence class in $\mathcal{C}_{h+1}^{\{c_{0,1}\}}$. Note that for hierarchical structure of depth $h+1$, we have $K_{h+1} = (K_h)^{J_{h+1}} + 1$, which can be immediately observed from (59). Inserting the loss definition in

(15) to (59), we obtain

$$T_{h+1}^{\{c_{0,1}\}} = L_{h+1}^{\{c_{0,1}\}} + \prod_{j=1}^{J_{h+1}} \left(\sum_{k=1}^{K_h} L_{h,j}^{\{k\}} \right)$$
$$= L_{h+1}^{\{c_{0,1}\}} + \sum_{k=1}^{(K_h)^{J_{h+1}}} L_{h+1}^{\{k\}},$$

which shows that $T_{h+1}^{\{c_{0,1}\}} = L_{h+1}$ holds, where the last line follows from (18) and from the following observation: Combining the states of an arbitrary FS predictor with hierarchy level h from each J_{h+1} paths, we obtain an FS predictor with hierarchy level $h+1$. Hence, the proof is concluded. ■

Appendix C
Proof of Lemma 2

We first consider the right hand side of the equality in (23) and observe that it is similar to

$$L_{t+1}^{\{c_{0,1}\}} = \sum_{k=1}^{K} L_t^{\{k\}} \exp\left(-\mu \epsilon_t \hat{d}_t^{\{k\}}\right), \quad (60)$$

where only the last exponential term in (60) is replaced by $\hat{d}_t^{\{k\}}$. Hence, following similar lines to the proof of Lemma 1 (i.e., using an induction method), it can be shown that the equality (23) can be achieved using the recursion in (22).

We then consider the product term in (22) and using (19), we obtain this product term (i.e., the last term in (22)) as follows

$$P_t^{\{c_{i,j}\}} \triangleq \prod_{\substack{c_{i+1,k} \in \mathcal{C}^{\{c_{i,j}\}} \\ c_{i+1,k} \neq c_{i+1,j'}}} T_t^{\{c_{k,l}\}}$$
$$= \frac{T_t^{\{c_{i,j}\}} - L_t^{\{c_{i,j}\}}}{T_t^{\{c_{i+1,j'}\}}}. \quad (61)$$

Putting (61) back in (22), we obtain

$$\widetilde{T}_t^{\{c_{i,j}\}} = L_t^{\{c_{i,j}\}} \hat{d}_t^{\{c_{i,j}\}} + \widetilde{T}_t^{\{c_{i',j'}\}} P_t^{\{c_{i,j}\}}. \quad (62)$$

Hence, $\widetilde{T}_t^{\{c_{0,1}\}}$ can be calculated with computational complexity $O(h)$. ■

Appendix D
Proof of Lemma 3

Since s_t is an element of one and only one equivalence class from each hierarchy level $0 \le i \le h$, computing $L_{t+1}^{\{c_{i,j}\}}$ from $L_t^{\{c_{i,j}\}}$ using (25) requires only $h+1$ updates. Hence, (25) can be calculated in $O(h)$ computations. We next analyze the number of computations required to perform (26).

Since the product term in (26) contains the equivalence classes in $\mathcal{C}^{\{c_{i,j}\}}$ that do not contain s_t, they are not updated from time t to $t+1$, i.e., $T_{t+1}^{\{c_{i+1,k}\}} = T_t^{\{c_{i+1,k}\}}$, $\forall c_{i+1,k} \in \mathcal{C}^{\{c_{i,j}\}} : s_t \notin c_{i+1,k}$. Then, we have $P_{t+1}^{\{c_{i,j}\}} = P_t^{\{c_{i,j}\}}$ according to (61). Putting (61) back in (26), we obtain

$$T_{t+1}^{\{c_{i,j}\}} = L_t^{\{c_{i,j}\}} \exp\left(-\mu \hat{d}_t^{\{c_{i,j}\}} \epsilon_t\right) + T_{t+1}^{\{c_{i+1,j'}\}} P_{t+1}^{\{c_{i,j}\}}, \quad (63)$$

which, in its current form, can be calculated with a computational complexity $O(h)$. ■

References

[1] T. Moon and T. Weissman, "Universal FIR MMSE filtering," *IEEE Trans. Signal Process.*, vol. 57, no. 3, pp. 1068–1083, Mar. 2009.

[2] T. Moon, "Universal switching FIR filtering," *IEEE Trans. Signal Process.*, vol. 60, no. 3, pp. 1460–1464, Mar. 2012.

[3] A. Gyorgy, T. Linder, and G. Lugosi, "Efficient algorithms and minimax bounds for zero-delay lossy source coding," *IEEE Trans. Signal Process.*, vol. 52, no. 8, pp. 2337–2347, Aug. 2004.

[4] S. S. Kozat and A. C. Singer, "Universal switching linear least squares prediction," *IEEE Trans. Signal Process.*, vol. 56, no. 1, pp. 189–204, Jan. 2008.

[5] N. D. Vanli and S. S. Kozat, "A unified approach to universal prediction: Generalized upper and lower bounds," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 3, pp. 646–651, Mar. 2015.

[6] A. C. Singer, S. S. Kozat, and M. Feder, "Universal linear least squares prediction: Upper and lower bounds," *IEEE Trans. Inf. Theory*, vol. 48, no. 8, pp. 2354–2362, Aug. 2002.

[7] T. Moon and T. Weissman, "Universal filtering via hidden Markov modeling," *IEEE Trans. Inf. Theory*, vol. 54, no. 2, pp. 692–708, Feb. 2008.

[8] J. Kivinen and M. K. Warmuth, "Exponentiated gradient versus gradient descent for linear predictors," *J. Inf. Comput.*, vol. 132, no. 1, pp. 1–62, 1997.

[9] N. Cesa-Bianchi and G. Lugosi, *Prediction, Learning, and Games*. Cambridge, U.K.: Cambridge Univ. Press, 2006.

[10] A. Gyorgy, T. Linder, and G. Lugosi, "Tracking the best of many expert," in *Proc. 18th Annu. Conf. Learn. Theory*, 2005, pp. 204–216.

[11] T. Linder and G. Lugosi, "A zero-delay sequential scheme for lossy coding of individual sequences," *IEEE Trans. Inf. Theory*, vol. 46, no. 6, pp. 190–207, Sep. 2001.

[12] A. H. Sayed, *Fundamentals of Adaptive Filtering*. Hoboken, NJ, USA: Wiley, 2003.

[13] K. Kim, N. Kalantarova, S. S. Kozat, and A. C. Singer, "Linear MMSE-optimal turbo equalization using context trees," *IEEE Trans. Signal Process.*, vol. 61, no. 12, pp. 3041–3055, Jun. 2013.

[14] F. M. J. Willems, Y. M. Shtarkov, and T. J. Tjalkens, "The context-tree weighting method: Basic properties," *IEEE Trans. Inf. Theory*, vol. 41, no. 3, pp. 653–664, May 1995.

[15] F. M. J. Willems, "Coding for a binary independent piecewise-identically-distributed source," *IEEE Trans. Inf. Theory*, vol. 42, no. 6, pp. 2210–2217, Nov. 1996.

[16] S. S. Kozat, A. C. Singer, and G. C. Zeitler, "Universal piecewise linear prediction via context trees," *IEEE Trans. Signal Process.*, vol. 55, no. 7, pp. 3730–3745, Jul. 2007.

[17] Y. Yilmaz and S. S. Kozat, "Competitive randomized nonlinear prediction under additive noise," *IEEE Signal Process. Lett.*, vol. 17, no. 4, pp. 335–339, Apr. 2010.

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

[19] H. Ozkan, N. Vanli, and S. Kozat, "Online classification via self-organizing space partitioning," *IEEE Trans. Signal Process.*, vol. 64, no. 15, pp. 3895–3908, Aug. 2016.

[20] T. Cover and E. Ordentlich, "Universal portfolios with side-information," *IEEE Trans. Inf. Theory*, vol. 42, no. 2, pp. 348–363, Mar. 1996.

[21] M. Feder, N. Merhav, and M. Gutman, "Universal prediction of individual sequences," *IEEE Trans. Inf. Theory*, vol. 38, no. 4, pp. 1258–1270, Jul. 1992.

[22] S. S. Kozat and A. C. Singer, "Universal semiconstant rebalanced portfolios," *Math. Finance*, vol. 21, no. 2, pp. 293–311, Oct. 2010.

[23] J. Kim *et al.*, "Order-preserving matching," *Theor. Comput. Sci.*, vol. 525, pp. 68–79, 2014.

[24] M. Kubica, T. Kulczyski, J. Radoszewski, W. Rytter, and T. Wale, "A linear time algorithm for consecutive permutation pattern matching," *Inf. Process. Lett.*, vol. 113, no. 12, pp. 430–433, 2013.

[25] P. Gawrychowski and P. Uznanski, "Order-preserving pattern matching with k mismatches," in *Proc. 25th Annu. Symp. Combinatorial Pattern Matching*, 2013, pp. 130–139.

[26] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, New York, NY, USA: Springer, 2001.

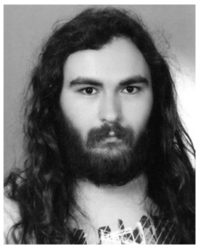
[27] S. S. Kozat, A. T. Erdogan, A. C. Singer, and A. H. Sayed, "Steady state MSE performance analysis of mixture approaches to adaptive filtering," *IEEE Trans. Signal Process.*, vol. 58, no. 8, pp. 4050–4063, Aug. 2010.

[28] M. Feder and N. Merhav, "Hierarchical universal coding," *IEEE Trans. Inf. Theory*, vol. 42, no. 5, pp. 1354–1364, Sep. 1996.

[29] A. György, T. Linder, and G. Lugosi, "Efficient tracking of large classes of experts," in *Proc. IEEE Int. Symp. Inf. Theory Proc.*, pp. 885–889, 2012.
[30] D. P. Helmbold and R. E. Schapire, "Predicting nearly as well as the best pruning of a decision tree," *Mach. Learn.*, vol. 27, no. 1, pp. 51–68, 1997.
[31] E. Takimoto, A. Maruoka, and V. Vovk, "Predicting nearly as well as the best pruning of a decision tree through dynamic programming scheme," *Theor. Comput. Sci.*, vol. 261, no. 1, pp. 179–209, 2001. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0304397500001389
[32] F. M. J. Willems, Y. M. Shtarkov, and T. J. Tjalkens, "Context weighting for general finite-context sources," *IEEE Trans. Inf. Theory*, vol. 42, no. 5, pp. 1514–1520, Sep. 1996.
[33] R. Diestel, *Graph Theory (Graduate Texts in Mathematics)*. New York, NY, USA: Springer-Verlag, Aug. 2005. [Online]. Available: http://www.amazon.ca/exec/obidos/redirect?tag=citeulike04-20&path=ASIN/3540261826
[34] L. Rabiner and R. Schafer, *Digital Processing of Speech Signals*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1978.
[35] E. Takimoto and M. K. Warmuth, "Predicting nearly as well as the best pruning of a planar decision graph," *Theor. Comput. Sci.*, vol. 288, no. 2, pp. 217–235, 2002.
[36] V. Vovk, "Derandomizing stochastic prediction strategies," *Mach. Learn.*, vol. 35, pp. 247–282, 1999.
[37] V. Vovk, "Aggregating strategies," in *Proc. 3rd Annu. Workshop Comput. Learn. Theory*, 1990, pp. 371–383.
[38] A. C. Singer and M. Feder, "Universal linear prediction by model order weighting," *IEEE Trans. Signal Process.*, vol. 47, no. 10, pp. 2685–2699, Oct. 1999.
[39] M. Herbster and M. K. Warmuth, "Tracking the best regressor," in *Proc. 11th Annu. Conf. Comput. Learn. Theory*, 1998, pp. 24–31.
[40] J. Arenas-Garcia, A. R. Figueiras-Vidal, and A. H. Sayed, "Mean-square performance of a convex combination of two adaptive filters," *IEEE Trans. Signal Process.*, vol. 54, no. 3, pp. 1078–1090, Mar. 2006.
[41] N. Cesa-Bianchi, Y. Freund, D. Helmbold, D. Haussler, R. Schapire, and M. Warmuth, "How to use expert advice," in *Proc. Annu. ACM Symp. Theory Comput.*, 1993, pp. 382–391.
[42] D. Haussler, J. Kivinen, and M. K. Warmuth, "Sequential prediction of individual sequences under general loss functions," *IEEE Trans. Inf. Theory*, vol. 44, no. 2, pp. 1906–1925, Sep. 1998.
[43] S. Tunc, M. Donmez, and S. Kozat, "Optimal investment under transaction costs: A threshold rebalanced portfolio approach," *IEEE Trans. Signal Process.*, vol. 61, no. 12, pp. 3129–3142, Jun. 2013.
[44] T. Weissman and N. Merhav, "Universal prediction of individual binary sequences in the presence of noise," *IEEE Trans. Inf. Theory*, vol. 47, no. 6, pp. 2151–2173, Sep. 2001.
[45] A. V. Aho and N. J. A. Sloane, "Some doubly exponential sequences," *Fibonacci Quart.*, vol. 11, pp. 429–437, 1970.
[46] S. Kullback and R. A. Leibler, "On information and sufficiency," *Ann. Math. Statist.*, vol. 22, no. 1, pp. 79–86, 1951. [Online]. Available: http://dx.doi.org/10.1214/aoms/1177729694
[47] A. H. Sayed, *Fundamentals of Adaptive Filtering*. Hoboken, NJ, USA: Wiley, 2003.
[48] O. J. J. Michel, A. O. Hero, and A.-E. Badel, "Tree-structured nonlinear signal modeling and prediction," *IEEE Trans. Signal Process.*, vol. 47, no. 11, pp. 3027–3041, Nov. 1999.
[49] V. Mathews, "Adaptive polynomial filters," *IEEE Signal Process. Mag.*, vol. 8, no. 3, pp. 10–26, Jul. 1991.
[50] A. Carini and G. L. Sicuranza, "Fourier nonlinear filters," *Signal Process.*, vol. 94, pp. 183–194, 2014.
[51] S. I. Hill and R. C. Williamson, "Convergence of exponentiated gradient algorithms," *IEEE Trans. Signal Process.*, vol. 49, no. 6, pp. 1208–1215, Jun. 2001.

N. Denizcan Vanli received the B.S. and M.S. degrees in electrical and electronics engineering from Bilkent University, Ankara, Turkey, in 2013 and 2015, respectively.

He is currently working toward the Ph.D. degree in electrical engineering and computer science at Massachusetts Institute of Technology, Cambridge, MA, USA. His research interests include convex optimization, online learning, and distributed optimization.



Kaan Gokcesu received the B.S. degree with high honors in electrical and electronics engineering from Bilkent University, Ankara, Turkey, in 2015.

He is currently working toward the M.S. degree in the Department of Electrical and Electronics Engineering at Bilkent University. His research interests include sequential learning, adaptive filtering, machine learning, bandit problems, and convex optimization.

Muhammed O. Sayin received the B.S. and M.S. degrees in electrical and electronics engineering from Bilkent University, Ankara, Turkey, in 2013 and 2015, respectively.

He is currenty working toward the Ph.D. degree in electrical and computer engineering from the University of Illinois at Urbana-Champaign, Champaign, IL, USA. His current research interests include signaling games, optimal information disclosure, and dynamic games for cyber security applications.

Hikmet Yildiz received the B.S. degree with high honors in electrical and electronics engineering from Bilkent University, Ankara, Turkey, in 2015, and the M.S. degree in electrical engineering from the California Institute of Technology (Caltech), Pasadena, CA, USA, in 2016. He is currently working toward the Ph.D. degree in electrical engineering at Caltech.

Suleyman Serdar Kozat (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.

After graduation, he joined IBM Research, T. J. Watson Research Lab, Yorktown, New York, as a Research Staff Member (and later became a Project Leader) in the Pervasive Speech Technologies Group, where he focused on problems related to statistical signal processing and machine learning. While doing his Ph.D., he was also working as a Research Associate at Microsoft Research, Redmond, Washington, in the Cryptography and Anti-Piracy Group. He holds several patent inventions due to his research accomplishments at IBM Research and Microsoft Research. He is currently an Associate Professor at the Electrical and Electronics Engineering Department at Bilkent University. He is the elected President of the IEEE Signal Processing Society, Turkey Chapter. He coauthored more than 100 papers in refereed high impact journals and conference proceedings and has several patent inventions (currently used in several different Microsoft and IBM products such as the MSN and the ViaVoice). He holds many international and national awards. Overall, his research interests include cyber security, anomaly detection, big data, data intelligence, adaptive filtering and machine learning algorithms for signal processing.