Universal Switching Linear Least Squares Prediction

Suleyman S. Kozat, *Member, IEEE*, and Andrew C. Singer, *Senior Member, IEEE*

***Abstract*—In this paper, we consider sequential regression of individual sequences under the square-error loss. We focus on the class of switching linear predictors that can segment a given individual sequence into an arbitrary number of blocks within each of which a fixed linear regressor is applied. Using a competitive algorithm framework, we construct sequential algorithms that are competitive with the best linear regression algorithms for any segmenting of the data as well as the best partitioning of the data into any fixed number of segments, where both the segmenting of the data and the linear predictors within each segment can be tuned to the underlying individual sequence. The algorithms do not require knowledge of the data length or the number of piecewise linear segments used by the members of the competing class, yet can achieve the performance of the best member that can choose both the partitioning of the sequence as well as the best regressor within each segment. We use a transition diagram (F. M. J. Willems, 1996) to compete with an exponential number of algorithms in the class, using complexity that is linear in the data length. The regret with respect to the best member is $O(\ln(n))$ per transition for not knowing the best transition times and $O(\ln(n))$ for not knowing the best regressor within each segment, where n is the data length. We construct lower bounds on the performance of any sequential algorithm, demonstrating a form of min–max optimality under certain settings. We also consider the case where the members are restricted to choose the best algorithm in each segment from a finite collection of candidate algorithms. Performance on synthetic and real data are given along with a Matlab implementation of the universal switching linear predictor.**

***Index Terms*—Piecewise continuous, prediction, transition diagram, universal.**

I. Introduction

In this paper, we apply a competitive algorithm framework to signal processing tasks in attempt to achieve the performance of the best algorithm from a large class of candidate algorithms for each individual sequence, rather than turning the given signal processing task into an intermediate parameter estimation problem and averaging over some assumed statistical knowledge of the signals of interest. This competitive approach has extensive roots in the machine learning [2]–[4], adaptive signal processing [5], and information theory [6], [7] literature, much of which builds on the seminal work of Hannan on prediction of individual sequences [8]. The individual sequence approach has been applied with much success in the universal data compression literature starting with the pioneering work of Fittingoff [9], Davisson [10], Elias [11], Ziv [12] and later Ziv and Lempel [13], Krichevsky and Trofimov [14], Rissanen and Langdon [15], and others [16]–[20]. The connections between universal source coding and sequential decision problems with other loss functions can be traced back to the work of Cover [21], Rissanen [22], Rissanen and Langdon [15], and Ryabko among others [23]–[25]. In the computational learning theory literature, following the work of Vovk on competitive algorithms and aggregating strategies [26], Foster [27], Cesa-Bianchi *et al.* [28], and Haussler *et al.* [29], a number of results for general loss functions have been obtained using mathematical tools that are similar in spirit to those developed in the lossless source coding literature.

Here, the performance measure considered is with respect to the best from this class instead of the usual parametric modeling error between the output of the modeling algorithm and the signal observations. For a number of related problems, it has been shown that by not forcing the algorithms to make hard decisions about a set of parameters at each step, but rather by competing against a rich class of approaches, algorithms can be constructed that compete well with respect to all candidates from a given class such that they sequentially achieve the performance of the best candidate from that class. That is, they perform as well as the best "batch" algorithm that had the ability to observe all of the data in advance, and select the best candidates before even beginning to process the data.

To further define this approach, consider a game in which a player is required to make a set of decisions $b[n]$ and based on these decisions the player incurs some loss $\ell(b[n], x[n])$ that depends on the sequence of outcomes $x[n]$ that are selected by nature. For example, $\ell(b[n], x[n]) = (b[n] - x[n])^2$ for the prediction problem under the square-error loss. The goal of the game is for the player to outperform (or perform at least as well as) every algorithm in some class C, against which the player competes. If each algorithm in the class has a set of decisions $b_c[n]$, $c \in C$, then the total sequentially accumulated loss for each competing algorithm would be given by $\ell_n(b_c, x) = \sum_{t=1}^{n} \ell(b_c[t], x[t])$. The goal of the game is for the player to achieve a sequentially accumulated loss $\ell_n(b, x)$ such that

$$\ell_n(b, x) = \sum_{t=1}^{n} \ell(b[t], x[t]) \leq \inf_{c \in C} \sum_{t=1}^{n} \ell(b_c[t], x[t])$$

uniformly, for every individual sequence $x^n = x[1], \ldots x[n]$. If the player can achieve a sequentially accumulated loss such that

$$\sup_{x^n \in X} \left[\ell_n(b, x) - \inf_{c \in C} \ell_n(b_c, x) \right] = R[n]$$

Manuscript received May 1, 2006; revised March 30, 2007. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Franz Hlawatsch.

S. S. Kozat is with the Electrical and Electronic Engineering Department, Koc University, Sariyer, Istanbul 34450, Turkey (e-mail: skozat@ku.edu.tr).

A. C. Singer is with the Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, Urbana, IL 61801 USA (e-mail: acsinger@uiuc.edu).

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.2007.901161

and $(R[n]/n) \to 0$, $n \gg 1$, then we say that the player has a strategy that is "universal" with respect to the class C of competing strategies for individual sequences in X under the loss $\ell(\cdot,\cdot)$.

For linear regression of real-valued data, several individual sequence results have been obtained under various loss functions in [4], [30], and [31], where the accumulated loss is shown to be asymptotically as small as the best fixed linear regressor for each sequence, taken from the class of all linear regressors of a given order p. In [31], these algorithms are shown to be min–max optimal in a certain sense for regression, and in [30], they are shown to be min–max optimal for prediction of real-valued sequences. In [27], a restricted form of linear regression (via convex combination) is considered for binary data, with a regret that is slightly larger than that in [31] and [30]. These algorithms are constructed by considering a hypothetical performance-weighted (Bayesian) combination of all linear regressors and selecting a prediction based on this "mixture predictor." This mixture approach is different than the classical approach, which typically involves recursively estimating the parameters of a linear regressor based on data observed up to the point at which each decision must be made. This mixture approach has also successfully been applied to problems other than prediction, including portfolio selection [6], lossy source coding [7], and nonlinear signal modeling [5]. We refer to algorithms such as these that are universal with respect to a fixed class of algorithms as *static* universal algorithms, in that the competing class contains a fixed set of algorithms, and performance is compared with the single best, fixed element of the class $\ell_n(b_c, x)$, $c \in C$.

In this paper, we consider the case when the underlying competing class has the added ability to switch among the various static elements. Here, each competing algorithm can divide the observation sequence into an arbitrary collection of segments, say $k+1$ of them, and fit each contiguous segment with the best algorithm from a given class of static algorithms, such as a fixed linear regressor, for that segment. For a path that contains k transitions, there exist $k+1$ segments. The loss incurred by a class member for any such partition P_k with $k+1$ segments is then the sum of the losses of the fixed static algorithms associated with each segment, i.e.,

$$\ell(b_{c,P_k}, x) = \sum_{j=0}^{k} \sum_{t=t_j}^{t_{j+1}-1} \ell(b_{c[j]}[t], x[t]) \tag{1}$$

where the partition P_k is represented by $t_0 = 1 < t_j < t_{k+1} = n+1$ and $c[j] \in C$ for some competition class C. The best partition with a given number of transitions k is then the one which gives the minimum total loss. In this paper, we focus primarily on parametrically continuous linear regressions as the class of experts. Specifically, given a set of observations $y_1[t], \ldots, y_M[t]$ and a sequence of outcomes $x[t]$, a static algorithm in each segment is given as $\vec{w}_j^T \vec{y}[t]$, where $\vec{y}[t] = [y_1[t], \ldots, y_M[t]]^T$ and $\vec{w}_j \in R^M$. Now, the loss incurred by a class member in (1) becomes

$$\ell(b_{c,P_k}, x) = \sum_{j=0}^{k} \sum_{t=t_j}^{t_{j+1}-1} \left(x[t] - \vec{w}_j^T \vec{y}[t]\right)^2 \tag{2}$$

under the square-error loss, where C is the class of all pth order linear regressors. The competing algorithms can determine each $\vec{w}_j$ independently for each segment. Here, we choose to compete against such a switching class without knowing the value of k or n in advance. A natural restriction for the number of possible transitions (switches) is $k < n$, where n is the length of the observation sequence. Unlike the approaches in [3] and [30]–[36], we try to exploit the time-varying nature of the best choice of algorithms for any given realization, since the choice of best algorithm from a class of static algorithms can locally change over time. Once again, rather than trying to find the best partition (possible best switching points) or the best number of transitions, our objective is simply to achieve the performance of the best partition directly and simultaneously for all k. The algorithms we provide are strongly sequential such that they do not need to know the number of transitions, times of these transitions, or length of the data *a priori*. While we focus on parametrically continuous classes of linear regressors as well as arbitrary finite classes of algorithms in (2), the methodology introduced here can be extended to a wide variety of more general competition classes as in [37] or to more general loss functions as in [31]. Our algorithms are generic such that other static universal algorithms as in [3], [31], [33], and [38] can be incorporated to generalize their respective bounds for the static case to the switching framework studied in this paper.

When the competing class can select a different fixed pth-order linear regressor for each segment in a given partition of the data from all pth-order linear regressors, with $k+1$ segments, the sequential algorithms introduced here will be shown to have an excess loss, or regret, of $pA^2(k+1)\ln(n/k) + 2A^2 k \ln(n/k) + O(k+1)$, $k > 0$ [and $O(pA^2 \ln(n)) + O(1)$, $k = 0$], over the performance of the best partition (for data bounded by $\pm A$). For a partition with $k+1$ segments, we also provide a corresponding lower bound on the excess loss of $pA^2(k+1)\ln(n/k) + O(k+1)$, $k > 0$ [and $O(pA^2 \ln(n)) + O(1)$, $k = 0$], for the performance of any sequential algorithm without prior knowledge of k or n. When $k = 0$, i.e., no transitions, these upper and lower bounds match, and the approach is min–max optimal in this sense. The algorithms discussed here have complexity linear in data length n, i.e., $O(n)$ per sample. A version with fixed complexity per sample can be produced using a different weighting method that was introduced in [39] with normalized excess loss of $O(\log(\log(n))/\log(n))$ over the performance of the best partition. However, we observe that even a simplified version of our algorithms with fixed complexity per sample, i.e., $O(M)$ where M is the size of the linear regressors in the class, introduced in Section II-A, nearly achieves the performance of the universal algorithms with the full transition diagram with complexity $O(n)$ per sample.

A closely related problem was investigated in [38] and [40] under the restricted framework of "tracking the best linear predictor." In [38] and [40], three algorithms are investigated and compared with algorithms from [41] and [42]. In [38], the static bounds of these algorithms are extended to the switching case considered here. However, for the bounds in [38] to hold, the underlying competition class is restricted to have $\vec{w}_j$'s in (2)

in certain convex regions. These convex regions are fairly restrictive and only satisfied in certain scenarios, e.g., algorithms that can only choose one algorithm from a finite set for each segment. Even in this restricted case, the performance over that of the best switching algorithm has excess loss of $O(L)$ [not $O(\ln(L))$], where L is the total loss of the best partition and the underlying switching parameters of each algorithm should be chosen *a priori* based on the number of switches k and the length of the data n. The bounds developed in this paper are in terms of the regret with respect to the performance of the best partition and the time-averaged excess loss over the best partition asymptotically vanishes. Furthermore, the static algorithms used in [38] and [40] can also be lifted to the switching case using our algorithms with the same order of growth in the upper bounds, rather than the additional $O(L)$. Through simulations, we also demonstrate that our algorithms with complexity $O(n)$ and the simplified algorithms with complexity $O(M)$ significantly outperform the algorithms introduced in [38] and [40] with similar complexity.

When the static class of algorithms has a finite number of elements as in [32] and the competition class is restricted to choose a single element for each segment from a finite number, this problem has been extensively investigated in the computational learning theory literature and referred to as "tracking the best expert" [4], [40]–[44]. We investigate this framework in Section III. For a process $x^n = \{x[1], \ldots, x[n]\}$ and M experts to choose from (for each segment), one can enumerate all the partitions of the sequence x^n and assign different experts to each segment in order to construct $\binom{n-1}{k} M(M-1)^k$ algorithms for each partition–expert pairing. This finite number of possible algorithms can then be combined by a static universal algorithm, as in [32] and [33] to yield an algorithm with a regret of $O((k+1)\ln(M) + k\ln(n/k))$ over the performance of the best switching algorithm. Nevertheless, this *naive algorithm* [42] is naturally unimplementable due to exponential number of possible algorithms to combine. To overcome this complexity problem, in [42], the authors generalize Vovk's aggregating algorithm (AA) [33] and introduce two main algorithms with complexity $O(M)$ per sample, whose excess loss are of the same order of the *naive algorithm*. These algorithms have excess loss of $O((k+1)\ln(M)+k\ln(n/k))$ and $O((k+1)\ln(M)+k\ln(L/k))$, however, certain parameters should be optimized *a priori* based on k and n. A natural quasi-probabilistic interpretation of these algorithms is given in [41], where improved versions of these algorithms are introduced that do not need *a priori* knowledge of n or k and attain the performance of the *naive algorithm*. Nevertheless, these improved algorithms have complexity growing linearly in data length, i.e., $O(n)$ instead of $O(M)$, per sample.

In [41], Vovk demonstrates that the algorithms in [38] are in fact an application of the AA [33] to combine (or derandomize) a continuum of certain elementary predictors. Taken literally, this combination is infeasible due to the continuum of elementary predictors to combine. However, when the class has a finite number of elements, the algorithms in [42] actually perform this implementation. In this sense, our algorithms extend this notion. In the construction of our algorithms, we begin with the result that a weighted average over a continuum of all such elementary predictors indeed achieves the performance of the best predictor in the class. This is a straightforward result and again directly in analogy with the *naive algorithm*. Our main contribution is that this literally infeasible mixture can be efficiently and *sequentially* implementable for wide variety of competition classes. The key difference between this related work and that developed here is the constructive nature of our results. We illustrate a prediction algorithm with a time complexity that is linear in the data length, which is strongly sequential such that it does not need any knowledge of data length or switching times and whose algebraic operations are explicitly given in the text. For a finite competition class, we present algorithms whose regret are of the same order as those in [42], i.e., $O((k+1)\ln(M)+k\ln(n/k))$ without the need to optimize parameters *a priori*. We also provide corresponding upper and lower bounds when the number of segments is large, e.g., comparable to n. Finally, modifications of the algorithms of [42] are introduced in [44] for the case of large, structured expert classes for which there exist efficient implementations of the exponential weighting, and in [43], to track a small set of experts from a larger set of experts by mixing past posteriors, each with bounds of the same order as [40]. Tracking the best disjunction is investigated by [45] using similar ideas to [42] and tracking the best portfolio is investigated by [46] using similar ideas to [41].

We begin by studying the prediction problem when the static competition class contains fixed-order linear regressors, in Section II. We then continue to provide corresponding lower bounds. In Section III, we investigate the case when the static class contains a finite set of algorithms. This paper is then concluded with simulations of the algorithms on synthetic and real data.

II. Switching Linear Regression

In this section, we investigate linear regression with the square-error loss in a competitive framework for deterministic unknown data. The main results of this section are given in Theorems 1 and 2 as an upper bound and a lower bound on performance, respectively. Here, the real-valued sequence $x^n = \{x[t]\}_{t=1}^n$ and the real-valued vector sequence $\vec{y}^n = \{\vec{y}[t]\}_{t=1}^n$ are assumed to be bounded but otherwise arbitrary, in that $|x[t]| < A_x$ for some $A_x < \infty$ and $|y_r[t]| < A_y$, $r = 1, \ldots, p$, for some $A_y < \infty$. For given sequences x^n and $\vec{y}^n$, a competing algorithm with a transition path $\mathcal{T}_{k,n}$ with k transitions, represented by $(t_1, \ldots, t_k)$, partitions x^n into $k+1$ segments such that x^n and $\vec{y}^n$ are represented by the concatenation

$$\{x[1], \ldots, x[t_1 - 1]\} \times \{x[t_1], \ldots, x[t_2 - 1]\} \ldots \{x[t_k], \ldots, x[n]\}$$

and

$$\{\vec{y}[1], \ldots, \vec{y}[t_1 - 1]\} \times \{\vec{y}[t_1], \ldots, \vec{y}[t_2 - 1]\} \ldots \{\vec{y}[t_k], \ldots, \vec{y}[n]\}$$

respectively. Given the past values of $x[t]$ and $\vec{y}[t]$, a competing algorithm forms an estimate of the desired signal in each segment as

$$\hat{x}[t] = \vec{w}_i^T \vec{y}[t]$$

where $\vec{w}_i = [w_{i,1}, \ldots, w_{i,p}]^T$, $\vec{w}_i \in R^p$, $t_{i-1} \leq t < t_i$, $i = 1, \ldots, k+1$, $\vec{y}[t] = [y_1[t], \ldots, y_p[t]]^T$, $\vec{y}[t] \in [-A_y, A_y]^p$, and $|x[n]| \leq A_x$. For simplicity, we assume $t_0 = 1$ and $t_{k+1} = n+1$. Given n and k, there exist $\binom{n-1}{k}$ such possible transition paths $\mathcal{T}_{k,n}$. We identify the best competing algorithm for any k as the one optimized by selecting the transition path $\mathcal{T}_{k,n}$ and the constants $\vec{w}_i$ based on observing x^n and $\vec{y}^n$ in advance. As such, we try to minimize the regret

$$\sup_{x^n, \vec{y}^n} \left\{ \sum_{t=1}^{n} (x[t] - \hat{x}_q[t])^2 - \inf_{\substack{\vec{w}_1, \ldots, \vec{w}_{k+1} \in R^p \\ t_1, \ldots, t_k \in \{2, \ldots, n\}}} \sum_{i=1}^{k+1} \sum_{t=t_{i-1}}^{t_i - 1} (x[t] - \vec{w}_i^T \vec{y}[t])^2 \right\}$$

where $\hat{x}_q[t]$ is the prediction at time t of any sequential algorithm, $\mathcal{T}_{k,n}$ is any transition path representing $(t_1, \ldots, t_k)$ with k transitions, and $\vec{y}[t] = [y_1[t], \ldots, y_p[t]]^T$. In the linear prediction problem, the observation sequence $\vec{y}[t]$ is formed by the past observations, i.e., $\vec{y}[t] = [x[t-1], \ldots, x[t-p]]^T$. We can also consider a particular choice of $y[t] = 1$ for all t, which corresponds to competition against fixed constant predictions in each segment such that $\hat{x}[t] = c_i$, $c_i \in R$, for each sample of the sequence $x[t]$ for $t = t_{i-1}, \ldots, t_i - 1$. Here, each c_i can be selected independently for each region $i = 1, \ldots, k+1$ (we denote constant predictors using the variable c to avoid confusion with the scalar linear regression problem). Theorem 1 and the final universal piecewise linear predictor can be extended to this case to construct a sequential algorithm for which the regret over the performance of the best piecewise constant predictor is at most $A_x^2(k+1)\ln(n) + 2A_x^2 k \ln(n) + O(k+1)$ for any of $\mathcal{T}_{k,n}$, k or n and with no knowledge of $\mathcal{T}_{k,n}$, k or n *a priori*.

For switching linear regression, we have the following result.

Theorem 1: For all $\delta > 0$, $\epsilon > 0$, a sequential algorithm $\tilde{x}_u[n]$ with complexity linear in data length n can be constructed such that for $x[n]$ and $\vec{y}[n]$ bounded, real-valued arbitrary scalar and vector sequences, such that $|x[n]| \leq A_x$ and $|y_s[n]| \leq A_y$, $s = 1, \ldots, p$, for all n and k

$$R_w[n] \triangleq \left\{ \sum_{t=1}^{n} (x[t] - \tilde{x}_u[t])^2 - \inf_{\substack{\vec{w}_1, \ldots, \vec{w}_{k+1} \in R^p \\ 1 = t_0 < t_1 < \cdots < t_n < t_{k+1} = n+1 \\ t_1, \ldots, t_k \in Z}} \times \sum_{i=1}^{k+1} \left(\sum_{t=t_{i-1}}^{t_i - 1} \left(x[t] - \vec{w}_i^T \vec{y}[t]\right)^2 + \delta \|\vec{w}_i\|^2 \right) \right\}$$

satisfies

$$\frac{R_w[n]}{n} \leq pA_x^2(k+1) \frac{\ln\left(\frac{1 + A_y^2 n}{\delta}\right)}{n} + 2A_x^2(k+\epsilon)\frac{\ln(n)}{n} + \frac{2A_x^2}{n}\left(\log(1+\epsilon) + k \log\frac{1}{\epsilon}\right) \quad (3)$$

and

$$\frac{R_w[n]}{n} \leq pA_x^2(k+1) \frac{\ln\left(\frac{1 + A_y^2 n}{(k\delta)}\right)}{n} + 2A_x^2\left((k+1)\epsilon + k\right) \times \frac{\ln\left(\frac{n}{k}\right)}{n} + \frac{2A_x^2}{n}\left((k+1)\log\frac{1+\epsilon}{\epsilon} + \log\epsilon\right) \quad (4)$$

for any $\mathcal{T}_{k,n}$ representing transition path $(t_1, \ldots, t_k)$ and any k, such that $\tilde{x}_u[t]$ does not depend on $\mathcal{T}_{k,n}$, k or n.

The parameter δ in (3) and (4) is used for regularization purposes and can be selected arbitrarily, such that $\delta > 0$; ϵ in (3) and (4) controls the weight assignments to each transition path and can be set to any value $\epsilon > 0$. The upper bound in (3) is better (tighter) when the number of transitions is small, i.e., $k \ll n$. If the number of transitions is closer to $O(n)$, then the upper bound in (4) is better. Theorem 1 states that the average squared prediction error of the universal linear regressor is within $O(pk\ln(n)/n)$ of the best batch piecewise linear pth-order regression algorithm with k transitions (tuned to the underlying sequence), uniformly, for every individual sequence x^n and vector sequence $\vec{y}^n$.

Proof of Theorem 1: For each possible transition path $\mathcal{T}_{k,n}$ representing $(t_1, \ldots, t_k)$ with k transitions and data length n, we consider a family of predictors $\hat{x}_W[t]$, each with its own regressor vectors $W = [\vec{w}_1 \ldots \vec{w}_{k+1}]^T$ where each $\vec{w}_i$ represents a fixed regression vector for the ith region, such that for ith segment $\hat{x}_W[t] \triangleq \vec{w}_i^T \vec{y}[t]$. For each pairing of $\mathcal{T}_{k,n}$ and W, a measure of the sequential prediction performance or loss of the corresponding competition algorithm $\hat{x}_W[t]$ is defined as

$$l_n(x, \hat{x}_W \mid W, \mathcal{T}_{k,n}) \triangleq \sum_{i=1}^{k+1} \sum_{t=t_{i-1}}^{t_i - 1} \left(x[t] - \vec{w}_i^T \vec{y}[t]\right)^2. \quad (5)$$

We next define a function of the loss, namely, the "probability"

$$P(x^n \mid W, \mathcal{T}_{k,n}) \triangleq \exp\left(-\frac{1}{2a} l_n(x, \hat{x}_W \mid W, \mathcal{T}_{k,n})\right)$$

which can be viewed as a probability assignment of a predictor with transition path $\mathcal{T}_{k,n}$ and with parameters W, to the data $x[t]$, for $1 \leq t \leq n$, induced by the performance of the corresponding algorithm with $\mathcal{T}_{k,n}$ and W on the sequence x^n, where

a is a positive constant. Given any $\mathcal{T}_{k,n}$, the competing algorithm with best fixed predictor in each region assigns to x^n the largest such probability $P(x^n \mid W, \mathcal{T}_{k,n})$, i.e.,

$$\begin{aligned} P^*(x^n \mid \mathcal{T}_{k,n}) &\triangleq \sup_W P(x^n \mid W, \mathcal{T}_{k,n}) \\ &= \exp\left(-\frac{1}{2a}\inf_W \sum_{i=1}^{k+1}\sum_{t=t_{i-1}}^{t_i-1}\left(x[t]-\vec{w}_i^T\vec{y}[t]\right)^2\right) \\ &= \exp\left(-\frac{1}{2a}\sum_{i=1}^{k+1}\sum_{t=t_{i-1}}^{t_i-1}\left(x[t]-\vec{w}_i^{*T}\vec{y}[t]\right)^2\right) \end{aligned}$$

where

$$\vec{w}_i^* = \left[Q_{t_{i-1},\vec{y}\vec{y}}^{t_i-1}\right]^{-1} r_{t_{i-1},x\vec{y}}^{t_i-1}$$

and $Q_{t_{i-1},\vec{y}\vec{y}}^{t_i-1} \triangleq \sum_{k=t_{i-1}}^{t_i-1}\vec{y}[k]\vec{y}^T[k]$ and $r_{t_{i-1},x\vec{y}}^{t_i-1} \triangleq \sum_{k=t_{i-1}}^{t_i-1} x[k]\vec{y}[k]$. Maximizing $P^*(x^n \mid \mathcal{T}_{k,n})$ over all $\mathcal{T}_{k,n}$ (with k transitions) yields

$$\begin{aligned} P^*\left(x^n \mid \mathcal{T}_{k,n}^*\right) &\triangleq \sup_{\mathcal{T}_{k,n}} P^*\left(x^n \mid \mathcal{T}_{k,n}\right) \\ &= \sup_{\mathcal{T}_{k,n},W} P\left(x^n \mid W, \mathcal{T}_{k,n}\right). \end{aligned}$$

Here, $P^*\left(x^n \mid \mathcal{T}_{k,n}^*\right) = P\left(x^n \mid W^*, \mathcal{T}_{k,n}^*\right)$ corresponds to the probability assigned by the best predictor in the class with k transitions. Our goal is to demonstrate a sequential algorithm which achieves $P\left(x^n \mid W^*, \mathcal{T}_{k,n}^*\right)$ for any k and n without knowledge of either k or n. We will accomplish this result using a double mixture approach. We first demonstrate an algorithm achieving the performance of the competing algorithm with the best fixed linear regressor in each region given any $\mathcal{T}_{q,n}$, i.e., $P^*(x^n \mid \mathcal{T}_{q,n})$. Then, we will show that a proper weighted combination of all such algorithms over all $\mathcal{T}_{q,n}$, $q = 1, \ldots, n$, can be used to find a sequential algorithm achieving $P^*\left(x^n \mid \mathcal{T}_{k,n}^*\right)$ for any k.

For any given $\mathcal{T}_{k,n}$, the probability assigned by the algorithm with the best fixed linear regressors in each region $P^*(x^n \mid \mathcal{T}_{k,n})$ can be asymptotically obtained by a sequential predictor that is universal with respect to the class of fixed linear regressors [30], independently for each segment i, i.e.,

$$\tilde{c}_{t_{i-1}}[t-1] \triangleq \vec{w}_{t_{i-1}}[t-1]^T\vec{y}[t] \tag{6}$$

where

$$\vec{w}_{t_{i-1}}[t-1] = \left[Q_{t_{i-1},\vec{y}\vec{y}}^{t} + \delta I\right]^{-1} r_{t_{i-1},x\vec{y}}^{t-1}$$

for some $\delta > 0$. In each segment, this universal algorithm achieves the following regret against the performance of the best fixed predictor for that region [30]

$$\begin{aligned} &\sum_{t=t_{i-1}}^{t_i-1}(x[t]-\tilde{c}_{t_{i-1}}[t-1])^2 \\ &\quad\le \inf_{\vec{w}_i}\left(\sum_{t=t_{i-1}}^{t_i-1}\left(x[t]-\vec{w}_i^T\vec{y}[t]\right)^2 + \delta\|\vec{w}_i\|^2\right) \\ &\qquad + pA_x^2\ln\left(1+A_y^2(t_i-t_{i-1})\delta^{-1}\right) \end{aligned}$$

for any δ. Applying this result for all segments and defining

$$\begin{aligned} &\tilde{P}\left(x^n \mid \mathcal{T}_{k,n}\right) \\ &\quad\triangleq \exp\left(-\frac{1}{2a}\sum_{i=1}^{k+1}\sum_{t=t_{i-1}}^{t_i-1}\left(x[t]-\tilde{c}_{t_{i-1}}[t-1]\right)^2\right) \end{aligned} \tag{7}$$

yields

$$\begin{aligned} &-2a\ln(\tilde{P}(x^n \mid \mathcal{T}_{k,n})) \\ &\quad\le \inf_W\left\{l_n(x,\hat{x}_W \mid W, \mathcal{T}_{k,n}) + \delta\|W\|^2\right\} \\ &\qquad + pA_x^2\sum_{i=1}^{k+1}\ln\left(1+A_y^2(t_i-t_{i-1})\delta^{-1}\right) \end{aligned} \tag{8}$$

where $\|W\|^2 \triangleq \sum_{i=1}^{k+1}\|\vec{w}_i\|^2$. Hence, given $\mathcal{T}_{k,n}$, using $\tilde{c}_{t_{i-1}}[t-1]$ in each segment defines a sequential algorithm that asymptotically achieves the performance of the algorithm with the best fixed linear regressors for each segment. For all $\mathcal{T}_{k,n}$ and k, we construct a similar sequential predictor yielding a total of 2^{n-1} such sequential predictors.

We then define a sequential probability assigned to x^n by a performance-weighted mixture of the probabilities assigned by all such sequential predictors over all possible $\mathcal{T}_{k,n}$ and k

$$\tilde{P}_u(x^n) \triangleq \sum_{k=0}^{n-1}\sum_{\mathcal{T}_{k,n}} P(\mathcal{T}_{k,n})\tilde{P}(x^n \mid \mathcal{T}_{k,n}) \tag{9}$$

with a suitable prior over the partitions $\mathcal{T}_{k,n}$, $P(\mathcal{T}_{k,n})$. It will be convenient in later calculations if the weighting $P(\mathcal{T}_{k,n})$ is nonnegative and satisfies

$$\sum_{k=0}^{n-1}\sum_{\mathcal{T}_{k,n}} P(\mathcal{T}_{k,n}) = 1.$$

Now, we have a probability assignment to the sequence x^n induced by the class of all possible state-constant (fixed) predictors and all possible transition paths. By (9), we can conclude that this probability assignment satisfies

$$\ln\tilde{P}_u(x^n) \ge \ln P(\mathcal{T}_{k,n}) + \ln\tilde{P}(x^n \mid \mathcal{T}_{k,n}) \tag{10}$$

for any transition path $\mathcal{T}_{k,n}$, since $\tilde{P}_u(x^n) \ge P(\mathcal{T}_{k,n})\tilde{P}(x^n \mid \mathcal{T}_{k,n})$. Combining (10) and (8) yields

$$\begin{aligned} -2a\ln(\tilde{P}_u(x^n)) \le &-2a\ln(P(\mathcal{T}_{k,n})) \\ &+\min_W\left\{l_n(x,\hat{x}_W \mid \mathcal{T}_{k,n}) + \delta\|W\|^2\right\} \\ &+pA_x^2\sum_{i=1}^{k+1}\ln\left(1+A_y^2(t_i-t_{i-1})\delta^{-1}\right) \end{aligned} \tag{11}$$

for any transition path $\mathcal{T}_{k,n}$ including $\mathcal{T}_{k,n}^*$.

The assigned probability for $\mathcal{T}_{k,n}$ directly contributes to the regret (excess negative log probability) as $-2a\ln(P(\mathcal{T}_{k,n}))$ over the best batch predictor given any path. Hence, it is desirable that the probability of the "best path" be assigned a large probability. This probability assignment should also be constructed so that the overall probability assignment and the

resulting predictor can be sequentially computable. Although many approaches exist, we will investigate only two such assignments in detail. We focus on these assignments since they yield sequential prediction algorithms with linear complexity in n.

As the first weighting method, we will use a form of estimated probability for $P(\mathcal{T}_{k,n})$ with k transitions, i.e., the Krichevsky–Trofimov (KT) weighting used in [1] and [47]. The KT estimate of the probability of a binary independent and identically distributed source of length $e+b$ with e ones and b zeros is defined as

$$P_{\mathrm{KT}}(e,b) \triangleq \int_0^1 \frac{1}{\pi\sqrt{(1-\theta)\theta}}(1-\theta)^e\theta^b d\theta.$$

This probability assignment admits the sequential updates

$$\begin{aligned} P_{\mathrm{KT}}(e+1,b) &= \frac{\frac{e+1}{2}}{e+b+1}P_{\mathrm{KT}}(e,b) \\ P_{\mathrm{KT}}(e,b+1) &= \frac{\frac{b+1}{2}}{e+b+1}P_{\mathrm{KT}}(e,b) \end{aligned} \tag{12}$$

for all $e \geq 0$ and $b \geq 0$. For a given path $\mathcal{T}_{k,n}$, we construct a binary string in which we represent each transition as a one and each instant without a transition as a zero, forming a binary sequence of length $n-1$; e.g., for $\mathcal{T}_{k,n}$, there exist k ones and a total of $n-k-1$ zeros. We then define $P(\mathcal{T}_{k,n})$ for any $(t_0,t_1,\ldots,t_{k+1})$ as

$$P(\mathcal{T}_{k,n}) \triangleq \left(\prod_{i=0}^{k-1} P_{\mathrm{KT}}(t_{i+1}-t_i-1,1)\right) P_{\mathrm{KT}}(n-t_k,0). \tag{13}$$

This probability is composed of $k+1$ KT estimates. Here, we first estimate the probability of the first transition as $P_{\mathrm{KT}}(t_1-t_0-1,1)$ due to the transition at time t_1 which ends the first segment and then we repeat this process for k segments to get the final probability assignment in (13). In the last segment, there is no transition, hence $P_{\mathrm{KT}}(n-t_k,0)$. It can be shown [1] that $\sum_{k=0}^{n-1}\sum_{\mathcal{T}_{k,n}} P(\mathcal{T}_{k,n}) = 1$ and this probability assignment yields [1], [47]

$$-2a\ln P(\mathcal{T}_{k,n}) \leq 2a\frac{3k+1}{2}\ln\left(\frac{n}{k}\right) + O(k) \tag{14}$$

for any $\mathcal{T}_{k,n}$.

By using the bound in (14), we can provide an upper bound for $-2a\ln(\tilde{P}_u(x^n))$ in (11) as

$$\begin{aligned} -2a\ln(\tilde{P}_u(x^n)) \leq\, & 2a\frac{3k+1}{2}\ln\left(\frac{n}{k}\right) \\ & + \min_W \left\{ l_n(x,\hat{x}_W \mid \mathcal{T}_{k,n}) + \delta\|W\|^2 \right\} \\ & + pA_x^2 \sum_{i=1}^{k+1} \ln\left(1 + A_y^2(t_i - t_{i-1})\delta^{-1}\right). \end{aligned} \tag{15}$$

Two different probability assignments introduced in [39] yield tighter upper bounds for $-2a\ln(P(\mathcal{T}_{k,n}))$

$$\begin{aligned} -2a\ln P(\mathcal{T}_{k,n}) \leq\, & 2a(k+\epsilon)\ln(n) \\ & + 2a\left(\log(1+\epsilon) + k\log\frac{1}{\epsilon}\right) \end{aligned} \tag{16}$$

and

$$\begin{aligned} -2a\ln P(\mathcal{T}_{k,n}) \leq\, & 2a\left(k+(k+1)\epsilon\right)\ln\left(\frac{n}{k}\right) \\ & + 2a\left((k+1)\log\frac{1+\epsilon}{\epsilon} + \log\epsilon\right) \end{aligned} \tag{17}$$

for all $\epsilon > 0$ (which is a parameter used by the probability assignment algorithm), which would give tighter upper bounds on the total regret over the best partition. Here, the upper bound due to the KT probability assignment can be viewed as a special case of (17) when $\epsilon = 1/2$. Other weighting methods used in [1] and [39] (such as the reduced state, quadratic complexity probability assignment) can be extended to yield prediction algorithms using the same methodology used in this paper. For the reduced state algorithm, we can obtain upper bounds on $-2a\ln(P(\mathcal{T}_{k,n}))$ that are larger than both (16) and (17); however, the reduced state algorithm has considerably lower complexity, i.e., complexity that does not grow with n.

We now have a method of assigning a probability to the sequence that achieves, to first order in the exponent, the same sequential probability as that assigned by the best batch predictor, for any partition $\mathcal{T}_{k,n}$, as shown in (15). In this sense, the probability assignment $\tilde{P}_u(x^n)$ is a "universal" probability assignment. It still remains to find a sequential prediction algorithm whose associated probability assignment is as large as $\tilde{P}_u(x^n)$.

For this purpose, we will show that the "conditional probability"

$$\tilde{P}_u(x[n] \mid x^{n-1}) \triangleq \frac{\tilde{P}_u(x^n)}{\tilde{P}_u(x^{n-1})}$$

from which $\tilde{P}_u(x^n) = \prod_{t=1}^n \tilde{P}_u(x[t] \mid x^{t-1})$ can be calculated sequentially in closed form. We will then describe a predictor that achieves this conditional probability for all n, thereby achieving $\tilde{P}_u(x^n)$.

At each time n, we divide the set of all possible paths $\mathcal{T}_{k,n}$, $k=1,\ldots,n$, into n disjoint sets. We label each set by a state variable s_n representing the most recent transition of a corresponding path within the period $1 \leq t \leq n$ such that for any $\mathcal{T}_{k,n}$, $s_n = t_k$. Given n, there can be at most n states $s_n = 1,\ldots,n$. At time n, all transition paths with the same last transition instant $t_k = s$ are represented by the state $s_n = s$. We then define $P_n(s_n,x^n)$ as the assigned probability of all predictions at state s_n at time n. For example, $P_n(s,x^n)$ is the weighted sum of all probabilities assigned to x^n by the sequential predictors whose transition paths ended up at $s_n = s$ state; i.e., for all paths $\mathcal{T}'$ such that the last transition was at $s_n = s$

$$P_n(s,x^n) \triangleq \sum_{\mathcal{T}':s_n=s} P(\mathcal{T}')\tilde{P}(x^n \mid \mathcal{T}').$$

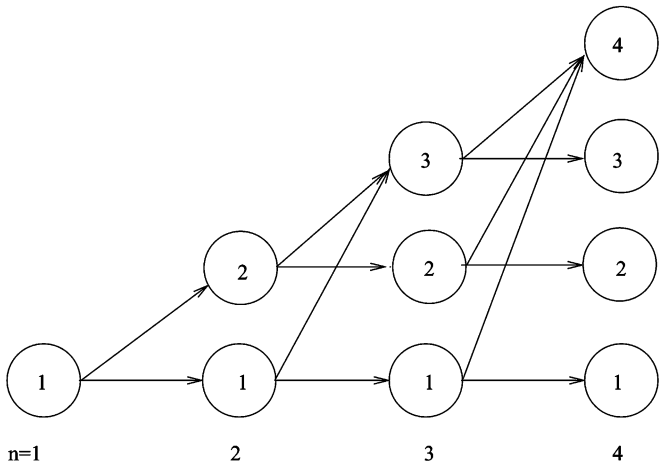


Fig. 1. Transition diagram for $n = 4$. Each circle represents a state; the number inside is the time of the last transition.

Since the states partition the set of paths $\mathcal{T}_{k,n}$

$$\tilde{P}_u(x^n) = \sum_{\mathcal{T}} P(\mathcal{T})\tilde{P}(x^n \mid \mathcal{T}) = \sum_{s_n=1}^{n} P_n(s_n, x^n).$$

To obtain a closed-form expression for $\tilde{P}_u(x[n] \mid x^{n-1})$, we will show that $P_n(s_n, x^n)$, $s_n = 1, \ldots, n$, can be calculated recursively by using a linear transition diagram, similar to that used in [1], and as shown in Fig. 1 (after assigning appropriate weights to each branch). Each box in the Fig. 1 represents a state variable s_n with the corresponding probability $P_n(s_n, x^n)$. In this figure, any directed path represents a transition path where a horizontal move denotes no transition, while an upward move represents a transition. As such, state s_n represents the most recent transition within the period $1 \leq t \leq n$.

We now derive a recursive update for each $P_n(s_n, x^n)$, $s_n = 1, \ldots, n$. From Fig. 1, we see that there exist only two possible transitions from each state. At time $n-1$, all the paths that ended at state $s_{n-1} = s$, $\mathcal{T}_{(\cdot,n-1):s_{n-1}=s}$, will end up in state $s_n = s$ if no transition happens at time n, i.e., a horizontal move in Fig. 1. From (12) and (13), we obtain the transition probability as

$$P_{tr}(s_n = s \mid s_{n-1} = s) \triangleq \frac{P(\mathcal{T}_{(\cdot,n):s_n=s})}{P(\mathcal{T}_{(\cdot,n-1):s_{n-1}=s})} = \frac{n-1-s_{n-1}+\frac{1}{2}}{n-1-s_{n-1}+1} \quad (18)$$

since only the last part of (13) should be changed in all paths. Given that $s_{n-1} = s$, paths ending up at state $s_n = s$ can only come from a horizontal move from $s_{n-1} = s$.

If there exists a transition at time n, i.e., an upward move, the probabilities of all paths from state $s_{n-1} = s$, $P(\mathcal{T}_{(\cdot,n-1):s_{n-1}=s})$, should be adjusted to get $P(\mathcal{T}_{(\cdot,n):s_n=n})$, i.e.,

$$P_{tr}(s_n = n \mid s_{n-1} = s) \triangleq \frac{P(\mathcal{T}_{(\cdot,n):s_n=n})}{P(\mathcal{T}_{(\cdot,n-1):s_{n-1}=s})} = \frac{\frac{1}{2}}{n-1-s_{n-1}+1}.$$

Naturally, $P_{tr}(s_n = s_{n-1} \mid s_{n-1}) + P_{tr}(s_n = n \mid s_{n-1}) = 1$. For the probability assignments of [39], used in (16) and (17), only the transition probabilities $P_{tr}(\cdot|\cdot)$ need to be changed. Hence, we will use $P_{tr}(\cdot|\cdot)$ rather than its evaluation to maintain a generic representation.

In addition to updating path probabilities for the corresponding sequential predictors of (7), the probabilities induced by these predictors in $P_{n-1}(s_{n-1}, x^{n-1})$ should also be updated by the prediction of $x[n]$ leading to $P_n(s_n, x^n)$. For any path $s_{n-1} = s$, $s = 1, \ldots, n-1$, when there is no switch, we need to multiply (7) with

$$\exp\left\{-\frac{1}{2a}(x[n] - \tilde{c}_{s_n}[n-1])^2\right\}$$

where $s_n = s_{n-1} = s$. When there is a switch at time n, the corresponding probabilities from $P_{n-1}(s_{n-1}, x^{n-1})$ should be adjusted by multiplying them with

$$\exp\left\{-\frac{1}{2a}(x[n] - \tilde{c}_n[n-1])^2\right\}.$$

After combining the update for path probabilities and update for prediction probabilities, we conclude for $s_n < n$

$$P_n(s_n, x^n) = P_{n-1}(s_n, x^{n-1})P_{tr}(s_n = s_{n-1} \mid s_{n-1}) \times \exp\left\{-\frac{1}{2a}(x[n] - \tilde{c}_{s_n}[n-1])^2\right\} \quad (19)$$

where

$$\tilde{c}_{s_n}[n-1] = \vec{w}_{s_n}[n-1]^T \vec{y}[n] \quad (20)$$

and $\vec{w}_{s_n}[n-1] = [Q^n_{s_n,\vec{y}\vec{y}} + \delta I]^{-1} r^{n-1}_{s_n,x\vec{y}}$ for some $\delta > 0$, yielding a sequential update for $P_n(s_n, x^n)$, when $s_n < n$.

When $s_n = n$, there exist $n-1$ possible transitions (to generate this new path) from each state s_{n-1}, $s_{n-1} = 1, \ldots, n-1$, at time $n-1$ to form the state $s_n = n$ at time n

$$P_n(n, x^n) = \exp\left\{-\frac{1}{2a}(x[n])^2\right\} \times \sum_{s_{n-1}=1}^{n-1} P_{n-1}(s_{n-1}, x^{n-1})P_{tr}(s_n = n \mid s_{n-1}) \quad (21)$$

since $\tilde{c}_n[n-1] = 0$, hence the sequential update for $P_n(s_n, x^n)$, when $s_n = n$.

A closer look at Fig. 1 and (19) and (21) reveals that $\tilde{P}_u(x^n) = \sum_{s_n=1}^{n} P_n(s_n, x^n)$ can be written as

$$\tilde{P}_u(x^n) = \sum_{s_{n-1}=1}^{n-1} P_{n-1}(s_{n-1}, x^{n-1}) \times \left\{P_{tr}(s_n = s_{n-1} \mid s_{n-1})P_n\left(x[n] \mid x^{n-1}, s_{n-1}\right) + P_{tr}(s_n = n \mid s_{n-1})P_n\left(x[n] \mid x^{n-1}, n\right)\right\}$$

where

$$P_n(x[n] \mid x^{n-1}, s_n) \triangleq \exp\left\{-\frac{1}{2a}(x[n] - \tilde{c}_{s_n}[n-1])^2\right\}.$$

Hence

$$\begin{aligned}\tilde{P}_u(x[n] \mid x^{n-1}) &= \frac{\tilde{P}_u(x^n)}{\tilde{P}_u(x^{n-1})} \\ &= \frac{\sum_{s_n=1}^{n} P_n(s_n, x^n)}{\sum_{s_{n-1}=1}^{n-1} P_{n-1}(s_{n-1}, x^{n-1})} \\ &= \frac{\sum_{s_{n-1}=1}^{n-1} P_{n-1}(s_{n-1}, x^{n-1}) P_{tr}(x[n] \mid x^{n-1}, s_{n-1})}{\sum_{s_{n-1}=1}^{n-1} P_{n-1}(s_{n-1}, x^{n-1})}\end{aligned}$$

where

$$\begin{aligned}P_{tr}\left(x[n] \mid x^{n-1}, s_{n-1}\right) \triangleq\ & P_{tr}(s_n = s_{n-1} \mid s_{n-1}) \\ & \times P_n(x[n] \mid x^{n-1}, s_{n-1}) \\ & + P_{tr}(s_n = n \mid s_{n-1}) \\ & \times P_n(x[n] \mid x^{n-1}, n).\end{aligned}$$

Then

$$\begin{aligned}\tilde{P}_u(x[n] \mid x^{n-1}) = \sum_{s_{n-1}=1}^{n-1} & \mu_{n-1}(s_{n-1}) \\ & \times P_{tr}(x[n] \mid x^{n-1}, s_{n-1}) \quad (22)\end{aligned}$$

where the weights $\mu_{n-1}(s_{n-1})$ are defined as

$$\mu_{n-1}(j) \triangleq \frac{P_{n-1}(j, x^{n-1})}{\sum_{s_{n-1}=1}^{n-1} P_{n-1}(s_{n-1}, x^{n-1})} \quad (23)$$

and are a form of performance-weighting for the states s_{n-1}, emphasizing those states that have the largest $P_{n-1}(s_{n-1}, x^{n-1})$ and, therefore, those states that have the lowest prediction error (i.e., performing well) on the data observed thus far.

If we can find a prediction algorithm with prediction $\tilde{x}_u[n]$ such that

$$\exp\left\{-\frac{1}{2a}(x[n] - \tilde{x}_u[n])^2\right\} \geq P_u(x[n] \mid x^{n-1}) \quad (24)$$

then the result is proven by simply taking the logarithm of both sides. We now introduce two different approaches to finding a suitable $\tilde{x}_u[n]$ satisfying (24). The first is based on a concavity argument and results an algorithm which can be constructed by a simple linear mixture. The second approach is based on the AA of [31] using a search procedure with polynomial-time complexity to construct its final output. This second approach results in the tighter upper bound introduced in Theorem 1. Both approaches use the same transition diagram and states, only differing in the last stage to form the final output.

Observe that $\tilde{P}_u(x[n] \mid x^{n-1})$ can be written as

$$\begin{aligned}\tilde{P}_u(x[n] \mid x^{n-1}) = \sum_{s_{n-1}=1}^{n-1} & \mu_{n-1}(s_{n-1}) \\ & \Big\{P_{tr}(s_n = s_{n-1} \mid s_{n-1}) f_t \\ & \quad \times \left(\tilde{c}_{s_{n-1}}[n-1]\right) \\ & \quad + P_{tr}(s_n = n \mid s_{n-1}) f_t(0)\Big\} \quad (25)\end{aligned}$$

where $f_t(\cdot)$ is defined as

$$f_t(z) \triangleq \exp\left(-\frac{(x[t] - z)^2}{2a}\right). \quad (26)$$

Since

$$\begin{aligned}\sum_{s_{n-1}=1}^{n-1} \mu_{n-1}(s_{n-1}) \Big\{ & P_{tr}(s_n = s_{n-1} \mid s_{n-1}) \\ & + P_{tr}(s_n = n \mid s_{n-1})\Big\} = 1\end{aligned}$$

$P_u(x[n] \mid x^{n-1})$ is sum of a function evaluated at a convex combination of values. In the first method, if the function $f_t(\cdot)$ is concave and $\sum_{i=1}^{n} \theta_i = 1$, then

$$f_t\left(\sum_{i=1}^{n} \theta_i z_i\right) \geq \sum_{i=1}^{n} \theta_i f_t(z_i)$$

by Jensen's inequality. The function defined in (26) will be concave for values of z_i such that $(x[n] - z_i)^2 < a$. This corresponds to

$$-\sqrt{a} \leq (x[n] - \tilde{c}[n]) \leq \sqrt{a}$$

where $\tilde{c}[n]$ is any prediction in (25). Since the signal $|x[n]| \leq A$, then the prediction values in (25) can be chosen such that $|\tilde{c}[n]| \leq A$. Therefore, by Jensen's inequality, whenever $a \geq 4A_x^2$, the function $f_t(\cdot)$ will be concave at all points of the prediction and

$$\begin{aligned}\tilde{P}_u(x[n] \mid x^{n-1}) \leq \exp\Bigg\{ & -\frac{1}{2a}\Bigg(x[n] - \sum_{s_{n-1}=1}^{n-1} \mu_{n-1}(s_{n-1}) \\ & \Big\{P_{tr}(s_n = s_{n-1} \mid s_{n-1}) \tilde{c}_{s_{n-1}}[n-1] \\ & \quad + P_{tr}(s_n = n \mid s_{n-1}) \tilde{c}_n[n-1]\Big\}\Bigg)^2\Bigg\}\end{aligned}$$

which gives the universal predictor as

$$\begin{aligned}\tilde{x}_u[n] = \sum_{s_{n-1}=1}^{n-1} & \mu_{n-1}(s_{n-1}) \\ & \times \left\{P_{tr}(s_n = s_{n-1} \mid s_{n-1}) \tilde{c}_{s_{n-1}}[n-1]\right\}\end{aligned}$$

since $\tilde{c}_n[n-1]=0$. By using (9) and (14) [or (16) or (17)], we conclude that

$$\begin{aligned} R_w[n] &\le pA_x^2 \sum_{i=1}^{k+1} \ln\left(1+A_y^2(t_i-t_{i-1})\delta^{-1}\right) \\ &\quad + 2a\frac{3k+1}{2}\ln\left(\frac{n}{k}\right)+O(k) \\ &\le pA_x^2(k+1)\ln\left(\frac{n}{k}\right)+4A_x^2(3k+1)\ln\left(\frac{n}{k}\right)+O(k) \end{aligned} \tag{27}$$

for any transition path $\mathcal{T}_{k,n}$, for any number of transitions k. The term $pA_x^2(k+1)\ln(n/k)$ in the upper bound can be viewed as the parameter regret resulting from maximizing $\sum_{i=1}^{k+1}\ln\left(1+A_y^2(t_i-t_{i-1})\delta^{-1}\right)$ over t_i. The algorithm is sequential such that it does not require knowledge of $\mathcal{T}_{k,n}$ or k *a priori*, but asymptotically achieves the performance of the best batch algorithm, for all $k<n$, sequentially.

For the second method, $\tilde{P}_u(x[n] \mid x^{n-1})$ in (25) is in the form of the *a posteriori* prediction algorithm (APA) of [31]. For values of $a \ge A_x^2$ from [31], there exists an interval of $\tilde{x}_u[n]$ that satisfies (24) and a value in this interval can be found in polynomial time. Using this value of a yields an upper bound with one fourth of the regret per transition in (27). Hence, using AA in the final stage instead of the convex combination will result in

$$R_w[n] \le pA_x^2(k+1)\ln\left(\frac{n}{k}\right)+A_x^2(3k+1)\ln\left(\frac{n}{k}\right)+O(k).$$

Using the probability assignment from (16) [or (17)] with the AA algorithm yields the bound in Theorem 1

$$R_w[n] \le pA_x^2(k+1)\ln\left(\frac{n}{k}\right)+2A_x^2(k+\epsilon)\ln(n)+O(k).$$

This completes proof of Theorem 1. ∎

We next provide a lower bound for any sequential algorithm competing against the class of piecewise linear predictors.

Theorem 2: Let $\hat{x}_q[t]$ be the prediction from any sequential prediction algorithm applied to $x[n]$ and $\vec{y}[n]=[y_1[n],\ldots,y_p[n]]^T$, a bounded, real-valued arbitrary sequence and vector sequence, such that $|x[n]|<A_x$ and $|y_s[t]|<A_y$, $s=1,\ldots,p$, for all n. Then, for any $\epsilon>0$, there exists a constant G such that

$$\begin{aligned} \inf_{q\in\mathcal{Q}}\sup_{x^n,\vec{y}^n}&\left\{\sum_{t=1}^{n}(x[t]-\hat{x}_q[t])^2-\inf_{W,\mathcal{T}_{k,n}} l_n(x,\hat{x}_W \mid W,\mathcal{T}_{k,n})\right\} \\ &\ge pA_x^2(1-\epsilon)(k+1)\ln\left(\frac{n}{(k+1)}\right)+G \end{aligned}$$

where $\mathcal{Q}$ is the class of all sequential algorithms and $l_n(x,\hat{x}_W \mid W,\mathcal{T}_{k,n})=\sum_{i=1}^{k+1}\sum_{t=t_{i-1}}^{t_i-1}(x[t]-\vec{w}_i^T\vec{y}[t])^2$.

Outline of proof of Theorem 2: We begin by noting that, for any distribution on x^n

$$\begin{aligned} &\inf_{q\in\mathcal{Q}}\sup_{x^n}\left\{\sum_{t=1}^{n}(x[t]-\hat{x}_q[t])^2-\inf_{W,\mathcal{T}_{k,n}} l_n(x,\hat{x}_W \mid W,\mathcal{T}_{k,n})\right\} \\ &\ge \inf_{q\in\mathcal{Q}} E_{x^n}\left\{\sum_{t=1}^{n}(x[t]-\hat{x}_q[t])^2-\inf_{W,\mathcal{T}_{k,n}} l_n(x,\hat{x}_W \mid W,\mathcal{T}_{k,n})\right\} \end{aligned} \tag{28}$$

where $E_{x^n}(\cdot)$ is the expectation taken with respect to any distribution on x^n. Hence, to obtain a lower bound on the total regret, we need only to lower bound the right-hand side of (28). Given any transition path $\mathcal{T}_{k,n}$, consider the following way of generating the sequence x^n. For each segment i, let θ_i be a random variable drawn from a beta distribution with parameters (C_i,C_i). For each segment, we generate $x[n]$ from a Markov distribution with parameter θ_i and we let $x[n]$ to have only two possible values, i.e., A_x or $-A_x$. In the ith segment, we generate $x[n]$ such that $x[n]=-x[n-1]$ with probability θ_i and $x[n]=x[n-1]$ with probability $(1-\theta_i)$. At the boundary point of each segment, we select $x[n]=A_x$ or $x[n]=-A_x$ with equal probability. The result now follows from Theorem 2 of [30]. ∎

A. Algorithmic Description

In this section, we give a Matlab implementation of the switching linear regression algorithm using pth order recursive least squares (RLS) regressors in (20). For one-step ahead prediction, the regression vectors should be replaced by $\vec{y}[n]=[x[n-1],\ldots,x[n-p]]^T$ and for piecewise constant prediction $\vec{y}[n]=1$. The complete algorithm is given in Fig. 2. The corresponding implementation uses Willems' [1] weighting for transition probabilities. For other weighting methods, only the calculation of the transition vector needs to be changed, i.e., vT in Fig. 2. For this particular implementation, at time n, we have a vector of state probabilities $vS=[P_{n-1}(1,x^{n-1}),\ldots,P_{n-1}(n-1,x^{n-1})]^T$, a vector of state predictions $vP=[\tilde{c}_1[n-1],\ldots,\tilde{c}_{n-1}[n-1]]^T$, and a vector of state transitions $vT=[P_{tr}(s_n=s_{n-1}|s_{n-1}=1),\ldots,P_{tr}(s_n=s_{n-1}|s_{n-1}=n-1)]^T$. For each time, $vM=[\mu_{n-1}(1),\ldots,\mu_{n-1}(n-1)]^T$ contains the normalized state probabilities as in (24). These vectors are updated at each iteration and the size of each vector is expanded by one for each new sample.

The complexity of the universal algorithm in Fig. 1 is $O(n)$, i.e., the complexity grows with data length. A simplified version of this algorithm with a constant complexity per sample (e.g., complexity linear in the prediction order) can be constructed by pruning the low probably states from the full transition diagram. In this version of the algorithm, at each time instant n, we sort the state probabilities vS and keep only the states corresponding to top K states. Only these K states remain for the next sample and all other states are dropped. The update equations for vS remain the same, except now vM is calculated based on only these K state probabilities and final prediction is given as the weighted combination of the prediction from these K states.

III. Switching Adaptive Filters

In this section, we investigate competing against a finite set of static algorithms in each region. Given a real-valued, bounded sequence $x[t]$, $t=1,\ldots,n$, we now consider a class of M different algorithms producing estimates $\hat{x}_m[t]$, $m=1,\ldots,M$, $t=1,\ldots,n$. For an observation sequence $x^n=\{x[1],\ldots,x[n]\}$ and outputs of these algorithms $\hat{x}_m^n=\{\hat{x}_m[1],\ldots,\hat{x}_m[n]\}$, $m=1,\ldots,M$ in the class, a transition path $\mathcal{T}_{k,n}$ with k transitions, represented by $(t_1,\ldots,t_k)$,

Variables:

m: data length; p: model order for linear regressor (predictor); d: a small positive constant;

mX: a matrix of size $p \times m$. The column $mX(:, n)$ is the corresponding regressor vector at time n.

vY: a vector of size $1 \times m$, i.e., $vY(n)$ is the desired signal at time n.

vS: the vector of state probabilities. vP: the vector of state predictions. vT: the vector of state transition weights.

```
% Inputs mX and vY;
xu = zeros(1, m);
vS = 1;
vP = 0;
mW = [zeros(p, 1)];
mQ = [inv(d) * eye(p)];
for n = 1 : m,
        vT = (n - [1 : n]' + 1/2)./(n - [1 : n]' + 1);
        vM = vS/sum(vS);
        xu(n) = vM' * (vT. * vP);
        tmps = 0;
        for j = 1 : n,
                vK = mQ(:, j : j + p - 1)/(1 + mX(:, n)' * mQ(:, j : j + p - 1) * mX(:, n));
                mQ(:, j : j + p - 1) = mQ(:, j : j + p - 1) - vK * mX(:, n)' * mQ(:, j : j + p - 1);
                e = vY(t) - mW(:, j)' * mX(:, n);
                tmps = tmps + vS(j) * exp(-(1/(2 * A * A)) * vY(n)^2) * (1 - vT(j));
                vS(j) = vS(j) * exp(-1/(2 * A * A) * e^2) * vT(j);
        end
        mQ = [mQ  inv(d) * eye(p)];
        mW = [mW  zeros(p, 1)];
        vS = [vS  tmps];
end
```

Fig. 2. Complete implementation of the universal piecewise linear algorithm for linear regression.

divides x^n and $\hat{x}_m^n$ into $k+1$ segments such that x^n and each $\hat{x}_m^n$ can be represented as a concatenation of

$$\{x[1], \ldots, x[t_1 - 1]\} \times \{x[t_1], \ldots, x[t_2 - 1]\} \ldots \{x[t_k], \ldots, x[n]\}$$

and

$$\{\hat{x}_m[1], \ldots, \hat{x}_m[t_1 - 1]\} \times \{\hat{x}_m[t_1], \ldots, \hat{x}_m[t_2 - 1]\} \ldots \{\hat{x}_m[t_k], \ldots, \hat{x}_m[n]\}$$

respectively. A more general class of competing algorithms could then independently select for each segment a different algorithm from the class of M algorithms.

Given any $\mathcal{T}_{k,n}$, the best competing algorithm with minimum total square error would choose the best algorithm for each segment from the class of M algorithms. Here, we demonstrate a sequential algorithm with no knowledge of k, $\mathcal{T}_{k,n}$, or n *a priori* that asymptotically achieves the performance of

$$\sum_{i=1}^{k+1} \sum_{t=t_{i-1}}^{t_i - 1} (x[t] - \hat{x}^i[t])^2 = \sum_{i=1}^{k+1} \min_{j=1,\ldots,M} \sum_{t=t_{i-1}}^{t_i - 1} (x[t] - \hat{x}_j[t])^2 \tag{29}$$

for any transition path $\mathcal{T}_{k,n}$ such that the difference between the total loss of this sequential algorithm and the loss defined in (29), normalized with n, will vanish as n goes to infinity. Here, $\hat{x}^i[t]$ is the best algorithm for the ith segment such that $\hat{x}^i[t] = \hat{x}_m[t]$ if

$$\sum_{t=t_{i-1}}^{t_i - 1} (x[t] - \hat{x}_m[t])^2 \leq \sum_{t=t_{i-1}}^{t_i - 1} (x[t] - \hat{x}_j[t])^2 \tag{30}$$

for $j \neq m$, $j = 1, \ldots, M$.

Theorem 3: For all $\epsilon > 0$, a sequential algorithm $\tilde{x}_u[n]$ with complexity linear in the data length n per prediction can be constructed for any bounded, real-valued arbitrary sequence $x[n]$, $|x[n]| < A$, using predictions of M arbitrary algorithms $\hat{x}_m[n]$, $m = 1, \ldots, M$, at time n such that

$$\frac{1}{n} \left\{ \sum_{t=1}^{n} (x[t] - \tilde{x}_u[t])^2 - \sum_{i=1}^{k+1} \sum_{t=t_{i-1}}^{t_i - 1} (x[t] - \hat{x}^i[t])^2 \right\} \leq 2A^2(k+1) \frac{\ln(M)}{n} + 2A^2(k+\epsilon) \frac{\ln(n)}{n} + \frac{2A_x^2}{n} \left(\log(1+\epsilon) + k \log \frac{1}{\epsilon} \right)$$

and

$$\frac{1}{n}\left\{\sum_{t=1}^{n}(x[t]-\tilde{x}_u[t])^2-\sum_{i=1}^{k+1}\sum_{t=t_{i-1}}^{t_i-1}(x[t]-\hat{x}^i[t])^2\right\} \le 2A^2(k+1)\frac{\ln(M)}{n}+2A^2\left((k+1)\epsilon+k\right) \times\frac{\ln\left(\frac{n}{k}\right)}{n}+\frac{2A_x^2}{n}\left((k+1)\log\frac{1+\epsilon}{\epsilon}+\log\epsilon\right)$$

for any $\mathcal{T}_{k,n}$ represented by $(t_1,\ldots,t_k)$ and k, without any knowledge of $\mathcal{T}_{k,n}$, k or n *a priori*, where $x^i[t]$ is the prediction of the best algorithm in the sense of (30) and $\tilde{x}_u[t]$ does not depend on $\mathcal{T}_{k,n}$, k or n.

The proof follows from Theorem 1 in [33]. We note that a linear complexity lattice filter-based algorithm [32] can achieve

$$\sum_{t=1}^{n}(x[t]-\tilde{x}_u[t])^2-\sum_{i=1}^{k+1}\sum_{t=t_{i-1}}^{t_i-1}(x[t]-\hat{x}^i[t])^2\le 8A^2(k+1)\ln(M)+8A^2(k+\epsilon)\ln(n)+O(k+1).$$

Proof of Theorem 3: The proof of Theorem 3 parallels that of Theorem 1, so we only highlight the main differences. Suppose the predictions of the M algorithms are given as $\hat{x}_m[t]$, $m=1,\ldots,M$, $t=1,\ldots,n$. Since we consider the prediction of a bounded signal $x[n]$, i.e., $x[n]\in[-A,A]$, any prediction $\hat{x}_k[n]$ made by any of the constituent algorithms in the mixture which is outside the interval $[-A,A]$ can be replaced by A if $\hat{x}_k[n]\ge A$ or $-A$ if $\hat{x}_k[n]\le -A$. This replacement will only improve the performance of the underlying predictor $\hat{x}_k[n]$. Hence, without loss of generality we can assume that each constituent prediction algorithm outputs a bounded prediction in $[-A,A]$. Here, an algorithm from the competing class with transition path $\mathcal{T}_{k,n}$ will select a single algorithm for each segment independently. Then, for each such transition path $\mathcal{T}_{k,n}$, with k transitions, a measure of the sequential prediction performance of the best competing algorithm is constructed

$$l_n(x,\hat{x}_{\text{best}}\mid\mathcal{T}_{k,n})\triangleq\sum_{i=1}^{k+1}\sum_{t=t_{i-1}}^{t_i-1}(x[t]-\hat{x}^i[t])^2$$

where $x^i[t]$ is the best algorithm for each segment as in (30).

For such a path $\mathcal{T}_{k,n}$, we define the probability

$$P^*(x^n\mid\mathcal{T}_{k,n})\triangleq\exp\left(-\frac{1}{2a}\sum_{i=1}^{k+1}\sum_{t=t_{i-1}}^{t_i-1}(x[t]-\hat{x}^i[t])^2\right)=\exp\left(-\frac{1}{2a}l_n(x,\hat{x}_{\text{best}}\mid\mathcal{T}_{k,n})\right)$$

which can be viewed as a probability assignment of $\mathcal{T}_{k,n}$ and the best competing algorithm to the data x^n. Maximizing $P^*(x^n\mid\mathcal{T}_{k,n})$ over all $\mathcal{T}_{k,n}$ (with k transitions) yields $P^*(x^n\mid\mathcal{T}^*_{k,n})$. Here, $P^*(x^n\mid\mathcal{T}^*_{k,n})$ corresponds to the best predictor in the competition class with k transitions. Our goal is to demonstrate a sequential algorithm that achieves $P^*(x^n\mid\mathcal{T}^*_{k,n})$ for all k and n without *a priori* knowledge of k or n.

Given any $\mathcal{T}_{k,n}$, if we use the AA of [33] to combine the M static algorithms in each segment, the AA for that segment achieves the performance of the best predictor for that segment, i.e.,

$$\sum_{t=t_{i-1}}^{t_i-1}(x[t]-\tilde{x}^{\text{aa}}_{t_{i-1}}[t])^2\le\sum_{t=t_{i-1}}^{t_i-1}(x[t]-\hat{x}^i[t])^2+2A^2\ln(M)+O(1)$$

where $x^{\text{aa}}_{t_{i-1}}[t]$ is the prediction of the AA running on the M constituent algorithms starting from time t_{i-1} to time $t-1$. Applying this result for all segments

$$\tilde{P}(x^n\mid\mathcal{T}_{k,n})\triangleq\exp\left(-\frac{1}{2a}\sum_{i=1}^{k+1}\sum_{t=t_{i-1}}^{t_i-1}(x[t]-\tilde{x}^{\text{aa}}_{t_{i-1}}[t])^2\right)\tag{31}$$

yields

$$-2a\ln(\tilde{P}(x^n\mid\mathcal{T}_{k,n}))\le\sum_{i=1}^{k+1}\sum_{t=t_{i-1}}^{t_i-1}(x[t]-\hat{x}^i[t])^2+2A^2(k+1)\ln(M)+O(k+1).$$

For a given $\mathcal{T}_{k,n}$, running an independent AA for each segment yields a sequential predictor, similar to the sequential predictor represented in (7). After this point, the derivation follows the proof of Theorem 1, where we combine sequential algorithms as in (31) with proper probability assignment $P(\mathcal{T}_{k,n})$. For construction of the universal algorithm, we only need to replace the prediction algorithm in (20) with the AA

$$\tilde{c}_{s_n}[n-1]=\hat{x}^{\text{aa}}_{s_n}[n-1]$$

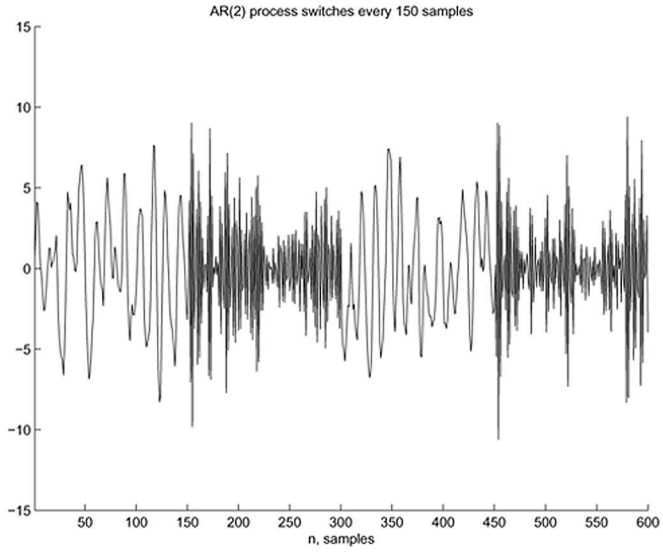
where $\hat{x}^{\text{aa}}_{s_n}[n-1]$ is the AA operating on the constituent algorithms from time s_n to $n-1$. ■

IV. Simulations

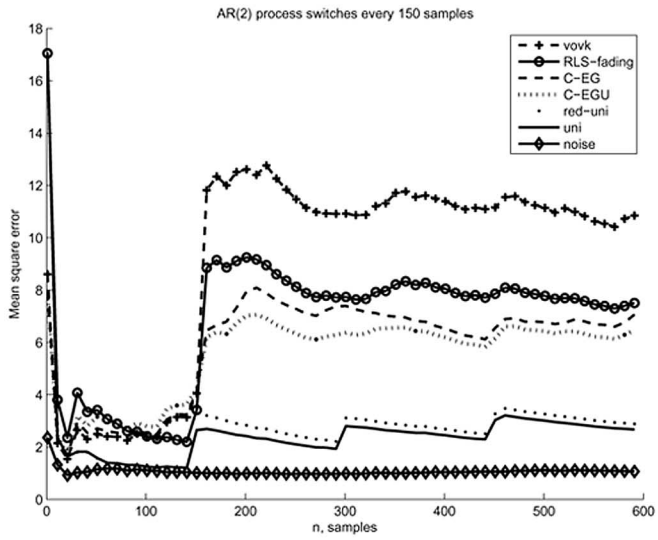
In this section, we demonstrate the performance of the universal algorithms with several different examples. We first investigate the one-step-ahead prediction of a second-order process that switches its parameters every 150 samples, i.e., the process switches between

$$\begin{aligned}x[n]&=1.6x[n-1]-0.8+w[n]\\x[n]&=-1.6x[n-1]-0.8+w[n]\end{aligned}\tag{32}$$

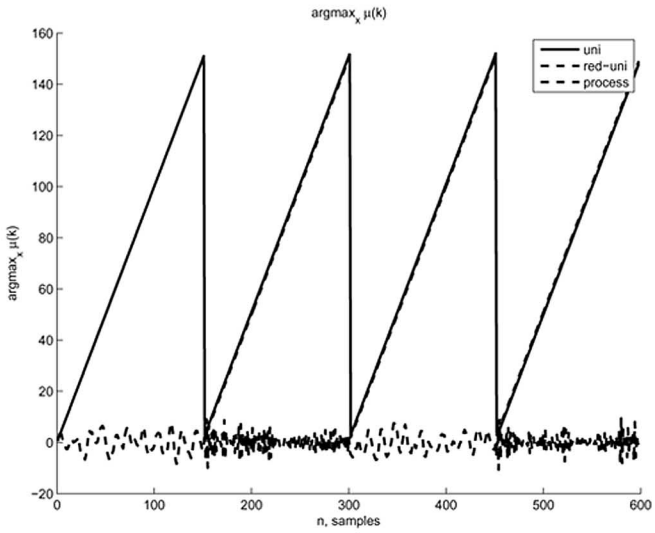
every 150 samples. Here, $w[n]$ is a sample function of a Gaussian process with zero mean and unit variance. For this process, the number of transitions is small compared to



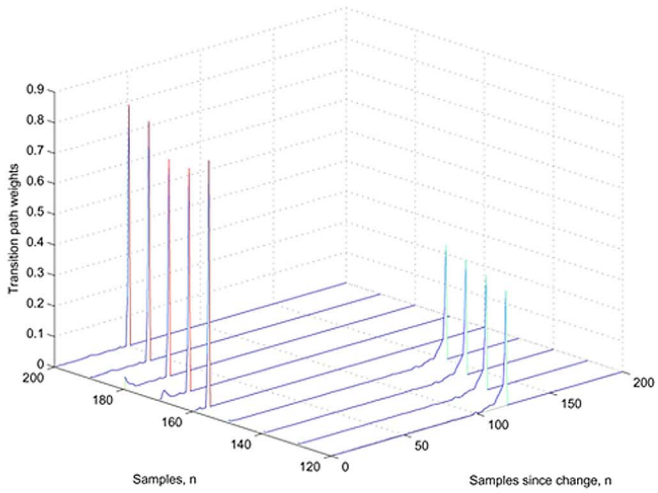
(a)



(b)



(c)



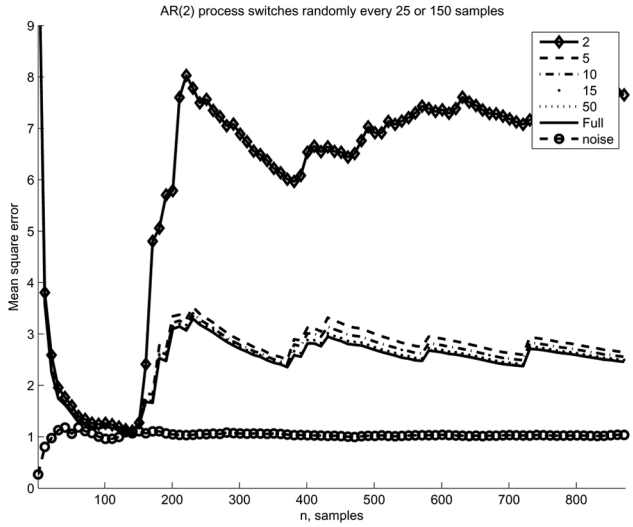
(d)

Fig. 3. (a) Prediction result of a sample function of a second-order autoregressive process as in (33) switching every 150 samples. (b) Normalized accumulated sequential prediction error $l_n(x,\hat{x})/n$ for universal piecewise linear predictor "uni," universal predictor with reduced transitions diagram "red-uni," C-EGU "C-EGU," C-EG "C-EG," derandomized algorithm by Vovk "vovk," fading RLS "RLS-fading," and normalized accumulated power of $w[n]$ from (33) "noise." (c) $\arg\max_k \mu_n(k)$ in (24). (d) Distribution of the weights $\mu_n(k)$ in (24) at $n = 120, 140, 160, 180, 200$.

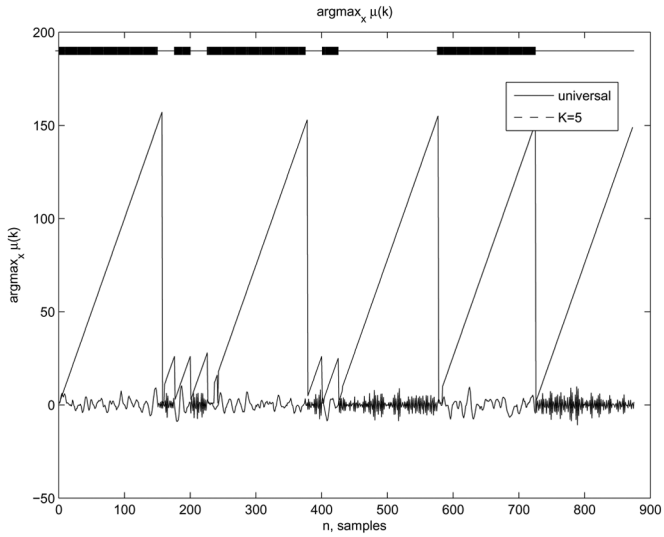
the data length, i.e., $k \ll n$. A sample function is given in Fig. 3(a). Since the main results of this paper are on prediction of individual sequences, in Fig. 3(b), we plot the normalized accumulated square prediction error of several algorithms for a sample function of the process in (32). In Fig. 3(b), we plot normalized mean square prediction error of the universal algorithm with full transition diagram, the universal algorithm with a reduced transition diagram as introduced in Section II-A, two algorithms introduced in [38] including constraint unnormalized exponentiated gradient (C-EGU) and constraint normalized exponentiated gradient (C-EG), an improved version of the fixed share algorithm by Vovk [41], and an RLS predictor with fading memory; and the sample noise floor, i.e., normalized accumulated power of $w[n]$. For each of these algorithms, we chose $\vec{y}[n] = [x[n-1]\ x[n-2]]^T$. We also simulated the constraint gradient descent (C-GD) algorithm from [38], however, the performance of this algorithm was significantly worse than the rest. Although we observe this in our simulations, it is shown in [38] that the GD algorithm can do better than EG on certain types of data. We note that for these types of data one expects that C-GD should give better results than C-EG. Since the algorithms introduced in [38] have complexity $O(M)$, $M = 2$, the universal algorithm with reduced transition diagram keeps only two paths (or states) in order to make a fair comparison. The fading parameter of the RLS algorithm is selected as $(1 - 1/150)$ based on the length of the stationary segments. The switching rates are given by 1/100 and 1/100 for C-EG and C-EGU, respectively, and 4 for

C-GD. The learning rate chosen for C-EG is larger (and gives better results) than the optimal switching rate prescribed in [38] since the optimal switching rate suggested in [38] is usually overly pessimistic. However, we avoided further optimizing the switching rates to be fair to the other algorithms. For the universal algorithm, the parameter a is estimated based on the observation sequence using standard deviation of the process observed so far (i.e., sequentially); however, we observed that the performance of the algorithm was fairly insensitive to changes in a. We observe that the universal algorithm suffers considerably less mean square error (MSE) at each transition of the underlying process. Although not as good as the full algorithm, the universal algorithm with the reduced diagram keeping only the top two states also suffers considerably less MSE at each transition. This behavior can be seen in Fig. 3(c), where we plot for each time n, the index of the state on the transition diagram with the largest weight, i.e., $\arg\max_k \mu_n(k)$ from (23) for the universal algorithm and the universal algorithm with reduced transition diagram. In this figure, the y-axis represents the sample since the last transition. The universal algorithm assigns, and continues to assign, the largest weight $\mu_n(k)$ in (23) to the transition paths that switch exactly at the transition times of the underlying process, i.e., every 150 samples. We also plot the distribution of the weights $\mu_n(k)$ in (32) among the states in Fig. 3(d). The universal algorithm appears to prefer the paths which have switching times in accordance with the transition times of the underlying process. We point out that although the reduced state algorithm correctly selects the paths with the switching pattern in accordance with the switching pattern of the underlying process, its performance is not as good as the full universal algorithm. Although, it is not a fair comparison, the extended version of the fixed share algorithm by [41] was included for completeness.

As the next example, we investigate the performance of the reduced state universal algorithm as a function of the number of states kept at each step. Here, the underlying second-order autoregressive (AR) process in (32) switches its parameters every 25 or 150 samples randomly, i.e., the duration of each stationary segment can be either 25 samples or 150 samples. In Fig. 4(a), we plot the normalized total squared prediction error of the universal piecewise linear prediction algorithm with the full transition diagram and universal algorithms with reduced state diagrams that keep only $K = 2, 5, 10, 15, 50$ states. As the number of the alive states that we keep on the diagram increases, the performance of the reduced state universal algorithms approaches to the performance of the universal algorithm with full transition diagram. The performance of the reduced state algorithm with $K = 10$ is nearly identical to the performance of the universal algorithm with full diagram. Furthermore, in Fig. 4(b), we plot for each time n, the index of the state on the transition diagram with the largest weight, i.e., $\arg\max_k \mu_n(k)$ from (23) for the universal algorithm and universal algorithm with a reduced transition diagram when $K = 5$. In the same plot, we also show the underlying process, at the bottom, and the duration of the randomly selected stationary segments, on the top. For this process, the switching times are given as $t = 150, 175, 200, 225, 375, 400, 425, 575, 725$. The universal algorithm assigns, and continues to assign, the largest weight in (23)



(a)



(b)

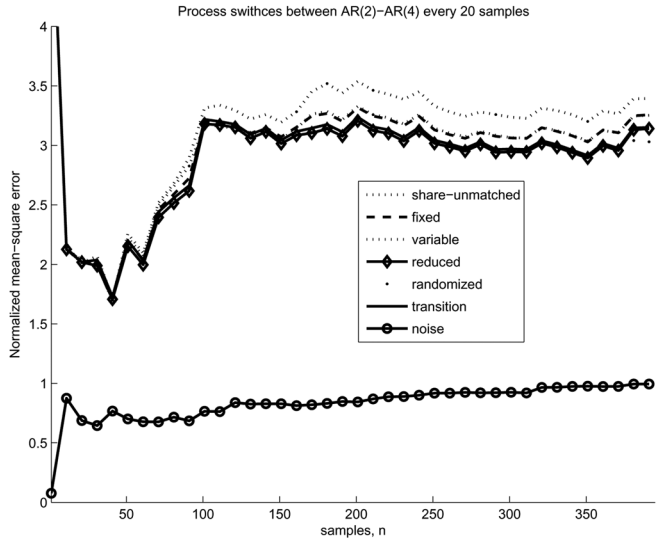
Fig. 4. Prediction result for a sample function of a second-order autoregressive process as in (33) switching randomly every 25 or 150 samples. (a) Normalized accumulated sequential prediction error $l_n(x, \hat{x})/n$ for universal piecewise linear predictor with full transition diagram "full," universal piecewise linear predictors with reduced transition diagrams, with $K = 2, 5, 10, 15, 50$ states, and normalized accumulated power of $w[n]$ from (33) "noise." (b)$\arg\max_k \mu_n(k)$ in (24) for the universal predictor with full diagram and for the reduced state universal predictor with $K = 5$. The stationary segment of the process are also shown in the same figure on the top and the underlying sample process is shown on the bottom.

to the transition paths that switch exactly at the transition times of the underlying process. This is also true for the reduced state universal algorithm for $K = 5$.

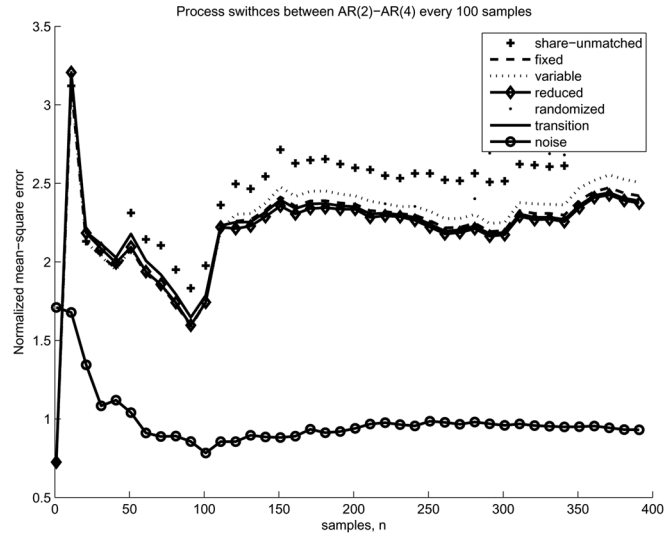
As the next example, we consider prediction of a process that switches between a second- and fourth-order process every 100 samples, i.e.,

$$\begin{aligned} x[n] &= 0.4x[n-1] - 0.85x[n-2] + w[n] \\ x[n] &= -1.6x[n-1] - 1.1x[n-2] - 1.142x[n-3] \\ &\quad - 0.8245 + w[n] \end{aligned} \tag{33}$$

where $w[n]$ is a sample function of a Gaussian process with zero mean and unit variance. Here, we simulate the performance of



(a)



(b)

Fig. 5. (a) Prediction results for a sample function of the process switching between second- and fourth-order process for every 100 samples as in (9). The normalized accumulated sequential prediction error $l_n(x,\hat{x})/n$ for a universal algorithm with full diagram for a finite class of algorithms as in Section III "transition," a universal algorithm with five states alive on the transition diagram "reduced," fixed-share algorithm with optimal switching parameter "fixed," variable-share algorithm with optimal switching parameter "variable," fixed-share algorithm extended by Vovk [41] "randomized," fixed-share algorithm with nonoptimal switching parameter "share-unmatched," and normalized accumulated power of $w[n]$ from (33) "noise." (b) Same algorithms when the process changes its parameters every 20 samples.

universal algorithms introduced in Section III, i.e., competing against a finite class of algorithms. As the competition class, we select first- to fifth-order RLS predictors, running in parallel on the underlying sequence. In Fig. 5(a), we plot the normalized MSE for the universal algorithm using the full transition diagram, the universal algorithm with the reduced transition diagram, two algorithms, fixed share and variable share from [42] with switching parameter tuned to the underlying switching rate of the process, i.e., 1/100, the fixed-share algorithm with the switching parameter different (much larger) from the underlying switching rate of the process, i.e., 10/100, and the noise floor, i.e., the normalized power of $w[n]$. For the universal algorithm with the reduced diagram, we keep only five paths to make a fair comparison with other algorithms. When the switching rate of the fixed- and variable-share algorithms are selected based on the switching rate of the underlying process, their performance is identical to the performance of the universal algorithms developed in this paper. Nevertheless, when the switching rate does not match, their performance degrades. We observe that the performance of the universal algorithm with the reduced transition diagram is nearly identical to the universal algorithm with the full transition diagram and the algorithm by Vovk [41], although, they have complexity $O(M)$, $O(n)$, and $O(n)$, respectively. Hence, we suggest using the universal algorithm with the reduced diagram for applications when the computational resources are limited. We observed that maintaining a fixed number (ten in our simulations) or more states in the full diagram gives nearly identical performance results to the algorithm using the full diagram. We next simulate the performance of the same algorithm when the underlying process switches every 20 samples. The normalized MSE of the algorithms are given in Fig. 5(b). In this figure, we observe similar behavior as before.

We next try to predict the daily closing value of the NASDAQ Composite Index. In Fig. 6(a), we plot the daily closing value of the NASDAQ Composite Index from January 2005 to January 2006 [48]. We present the MSE of the following four different algorithms in Fig. 6(b): the universal algorithm using 20th-order RLS predictors (without fading), the RLS predictor with optimal fading where the optimal fading parameter is selected *a priori* using all the available data from January 2005 to January 2006, the RLS predictor with optimal transition paths where the optimal transition paths are selected *a priori* using all the available data for each time n, and an ordinary RLS algorithm. The optimal path for the RLS algorithm is determined individually for each time t, $t = 1,\ldots,n$, using dynamic programming. The performance of the universal algorithm outperforms the performance of the ordinary RLS predictor, the RLS predictor with optimal fading and is as good as the RLS predictor with the best transition paths. We also plot the difference between the MSE of the universal algorithm and the RLS algorithm with the optimal transition paths in Fig. 6(d). We see that although the optimal transition paths are selected in hindsight for each time to minimize the MSE, the universal predictor is as good as this algorithm in several regions and outperforms it at the start of the process. Hence, combining several paths and using a performance-weighted mixture, the universal algorithm can outperform the algorithm with the best transition paths. We also plot the distribution of the weights $\mu_n(k)$ in (23) in Fig. 6(c) for days 20, 70, 120, 170, and 220.

V. Conclusion

In this paper, we investigated sequential regression of individual sequences under square-error loss. We developed strongly sequential algorithms, without *a priori* knowledge of the data length or the number of piecewise constant segments that achieve the performance of the best switching linear (or constant) regression algorithm tuned to the underlying sequence. To achieve this, a performance-weighted mixture of an exponential number of sequential predictors, one for

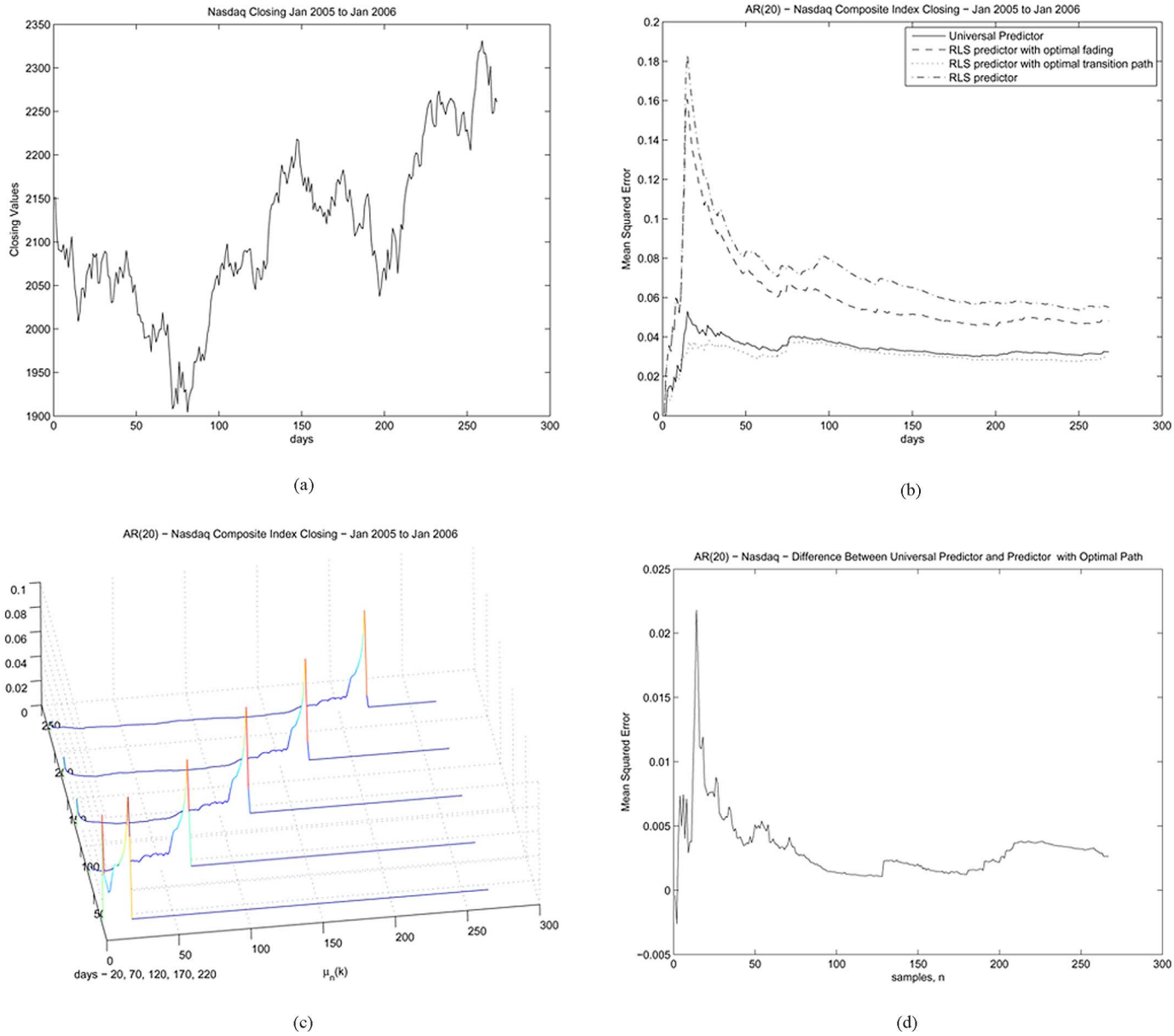


Fig. 6. Prediction of Nasdaq Composite Index using 20th-order predictors. (a) Daily closing value of Nasdaq Composite Index from January 2005 to January 2006. (b) Running average prediction results: universal piecewise linear predictor (solid line), an RLS algorithm with optimal fading (dashed line), an RLS predictor with the optimal transition paths (dotted line), and an ordinary RLS predictor (dotted–dashed line). (c) Weights $\mu_n(k)$ in (23) assigned by the universal algorithm to the transition paths at days 20, 70, 120, 170, and 220. (d) Difference between the MSE of the universal algorithm and the RLS algorithm with the best transition paths at each time.

each transition path, was shown to asymptotically achieve the performance of the best algorithm given any number of piecewise constant segments. We then showed that this exponential number of algorithms can be implicitly implemented with linear complexity in the data length using a transition diagram similar to [1]. We then considered the case when the members of the static class include only a finite collection of algorithms; we demonstrated an algorithm that can asymptotically achieve the best subpartitioning of the data into segments within which the best among the finite collection of algorithms are chosen.

REFERENCES

[1] F. M. J. Willems, "Coding for a binary independent piecewise-identically-distributed source," *IEEE Trans. Inf. Theory*, vol. 42, no. 6, pt. 2, pp. 2210–2217, Nov. 1996.

[2] V. Vovk, "Competitive on-line linear regression," in *Advances in Neural Information Processing Systems*. Cambridge, MA: MIT Press, 1998, pp. 364–370.

[3] N. Cesa-Bianchi, Y. Freund, D. Haussler, D. P. Helmbold, R. E. Schapire, and M. K. Warmuth, "How to use expert advice," *J. Assoc. Comput. Mach.*, vol. 44, no. 3, pp. 427–485, 1997.

[4] N. Littlestone and M. K. Warmuth, "The weighted majority algorithm," *Inf. Comput.*, vol. 108, no. 2, pp. 212–261, 1994.

[5] S. S. Kozat, A. C. Singer, and G. Zeitler, "Universal piecewise linear prediction via context trees," *IEEE Trans. Signal Process.*, vol. 55, no. 7, pt. 2, pp. 3730–3745, Jul. 2007.

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

[7] T. Linder and G. Lugosi, "A zero-delay sequential scheme for lossy coding of individual sequences," *IEEE Trans. Inf. Theory*, vol. 46, no. 1, pp. 190–207, Jan. 2001.

[8] J. Hannan, *Approximation to Bayes Risk in Repeated Play*. Princeton, NJ: Princeton Univ. Press, 1957, vol. III, Contributions to the Theory of Games, pp. 97–139.

[9] B. Fittingoff, "Universal methods of coding for the case of unknown statistics," in *Proc. 5th Symp. Inf. Theory*, Moscow/Gorky, U.S.S.R., 1972, pp. 129–135.
[10] L. D. Davisson, "Universal noiseless coding," *IEEE Trans. Inf. Theory*, vol. IT-19, no. 6, pp. 783–795, Nov. 1973.
[11] P. Elias, "Universal codeword sets and representations of the integers," *IEEE Trans. Inf. Theory*, vol. IT-21, no. 2, pp. 194–203, Mar. 1975.
[12] J. Ziv, "Coding of sources with unknown statistics, Part I: Probability of encoding error," *IEEE Trans. Inf. Theory*, vol. IT-18, no. 3, pp. 373–343, May 1972.
[13] J. Ziv and A. Lempel, "Compression of individual sequences via variable-rate coding," *IEEE Trans. Inf. Theory*, vol. IT-24, no. 5, pp. 530–536, Sep. 1978.
[14] R. E. Krichevsky and V. K. Trofimov, "The performance of universal encoding," *IEEE Trans. Inf. Theory*, vol. IT-27, no. 2, pp. 199–207, Mar. 1981.
[15] J. Rissanen and G. G. Langdon, "Universal modeling and coding," *IEEE Trans. Inf. Theory*, vol. IT-27, no. 1, pp. 12–23, Jan. 1984.
[16] J. Rissanen, "A universal data compression system," *IEEE Trans. Inf. Theory*, vol. IT-29, no. 5, pp. 656–664, Sep. 1983.
[17] B. Y. Ryabko, "Twice-universal coding," *Prob. Inf. Trans.*, vol. 20, no. 3, pp. 173–177, Jul.–Sep. 1984.
[18] P. A. Chou, M. Effros, and R. M. Gray, "Universal quantization of parametric sources has redundancy k/2 logn/n," in *Proc. IEEE Int. Symp. Inf. Theory*, 1995, pp. 371–371.
[19] M. Effros, P. A. Chou, and R. M. Gray, "Rates of convergence in adaptive universal vector quantization," in *Proc. IEEE Int. Symp. Inf. Theory*, 1994, pp. 456–456.
[20] R. Zamir and M. Feder, "On universal quantization by randomized uniform/lattice quantizers," *IEEE Trans. Inf. Theory*, vol. 38, no. 2, pp. 428–36, Mar. 1992.
[21] T. M. Cover, "Universal gambling schemes and the complexity measures of Kolmogorov and Chaitin," Dept. Statist., Stanford Univ., Stanford, CA, Tech. Rep. 12, 1974.
[22] J. Rissanen, "Universal coding, information, prediction, and estimation," *IEEE Trans. Inf. Theory*, vol. IT-30, no. 4, pp. 629–636, Jul. 1984.
[23] B. Ya Ryabko, "Prediction of random sequences and universal coding," *Prob. Inf. Trans.*, vol. 24, no. 2, pp. 87–96, Apr.–Jun. 1988.
[24] N. Merhav and M. Feder, "Universal prediction," *IEEE Trans. Inf. Theory*, vol. 44, no. 10, pp. 2124–2147, Oct. 1998.
[25] P. Algoet, "Universal schemes for prediction, gambling, and portfolio selection," *Ann. Probab.*, pp. 901–941, 1992.
[26] V. G. Vovk, M. Fulk and J. Case, Eds., "Aggregating strategies (learning)," in *Proc. 3rd Annu. Workshop Comput. Learn. Theory*, San Mateo, CA, 1990, pp. 371–383.
[27] D. Foster, "Prediction in the worst case," *Annu. Statist.*, vol. 19, no. 2, pp. 1084–1090, 1991.
[28] N. Cesa-Bianchi, Y. Freund, D. Helmbold, D. Haussler, R. Schapire, and M. Warmuth, "How to use expert advice," in *Proc. Annu. Assoc. Comput. Mach. Symp. Theory Comput.*, 1993, pp. 382–391.
[29] D. Haussler, J. Kivinen, and M. K. Warmuth, P. Vitanyi, Ed., "Tight worst-case loss bounds for predicting with expert advice," in *Proc. 2nd Eur. Conf. Comput. Learn. Theory (EuroCOLT)*, Mar. 1995, pp. 69–83.
[30] 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.
[31] V. Vovk, "A game of prediction with expert advice," *J. Comput. System Sci.*, vol. 56, pp. 153–173, 1998.
[32] 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.
[33] V. Vovk, "Aggregating strategies," in *Proc. Comput. Learn. Theory*, 1990, pp. 371–383.
[34] D. Haussler, J. Kivinen, and M. K. Warmuth, "Sequential prediction of individual sequences under general loss functions," *IEEE Trans. Inf. Theory*, vol. 44, no. 5, pp. 1906–1925, Sep. 1998.
[35] J. Kivinen and D. Haussler, "Additive versus exponentiated gradient updates for linear prediction," *J. Inf. Comput.*, vol. 132, no. 1, pp. 1–64, 1997.
[36] 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.
[37] D. Luengo, S. S. Kozat, and A. C. Singer, "Universal piecewise linear least squares prediction," in *Proc. Int. Symp. Inf. Theory*, 2004, p. 198.
[38] M. Herbster and M. K. Warmuth, "Tracking the best regressor," in *Proc. 11th Annu. Conf. Comput. Learn. Theory*, 1998, pp. 24–31.
[39] G. I. Shamir and N. Merhav, "Low-complexity sequential lossless coding for piecewise-stationary memoryless sources," *IEEE Trans. Inf. Theory*, vol. 45, no. 5, pp. 1498–1519, Jul. 1999.
[40] M. Herbster and M. K. Warmuth, "Tracking the best linear predictor," *J. Mach. Learn. Res.*, pp. 281–309, 2001.
[41] V. Vovk, "Derandomizing stochastic prediction strategies," *Mach. Learn.*, vol. 35, pp. 247–282, 1999.
[42] M. Herbster and M. K. Warmuth, "Tracking the best expert," in *Proc. Int. Conf. Mach. Learn.*, 1995, pp. 286–294.
[43] O. Bousquet and M. K. Warmuth, "Tracking a small set of experts by mixing past posteriors," *J. Mach. Learn. Res.*, vol. 3, pp. 363–396, 2000.
[44] A. Gyorgy, T. Linder, and G. Lugosi, "Tracking the best of many expert," in *Proc. Comput. Learn. Theory*, 2005, pp. 204–216.
[45] P. Auer and M. K. Warmuth, "Tracking the best disjuction," *J. Mach. Learn.*, vol. 32, no. 2, pp. 127–150, 1998.
[46] Y. Singer, "Switching portfolios," in *Proc. 14th Conf. Uncertainty Artif. Intell.*, 1998, pp. 1498–1519.
[47] R. E. Krichevsky and V. K. Trofimov, "The performance of universal encoding," *IEEE Trans. Inf. Theory*, vol. IT-27, no. 2, pp. 190–207, Mar. 1981.
[48] P. Wessa, Free Statistics Software. ver. 1.1.22–r1, Office for Research Development and Education, 2007 [Online]. Available: http://www.wessa.net

Suleyman S. Kozat (M'04) was born in Ankara, Turkey. He received the B.S. degree in electrical engineering from Bilkent University, Ankara, Turkey, in 1998, and the M.S. and Ph.D. degrees from the Coordinated Science Laboratory, Electrical and Computer Engineering Department, University of Illinois, Urbana–Champaign, in 2001 and 2004, respectively, where he was in the Signal Processing group, under the supervision of Dr. A. C. Singer. He received full-time scholarship from Bilkent University during his undergraduate studies.

Until 2007, he was a full-time Research Staff Member at the IBM Research, Pervasive Speech Technologies group, T. J. Watson Research Center, Yorktown, NY. Currently, he is an Assistant Professor at the Electrical and Electronic Engineering Department, Kuc University, Istanbul, Turkey. His research interests include machine learning, signal processing, speech processing, and statistical signal processing.

Dr. Kozat has served as a Reviewer for the IEEE TRANSACTIONS ON SIGNAL PROCESSING, the IEEE TRANSACTIONS ON IMAGE PROCESSING, the IEEE TRANSACTIONS ON INFORMATION THEORY as well as several conferences such as the IEEE International Conference on Signal Processing (ICASSP), IEEE International Conference on Image Processing (ICIP), International Conference on Spoken Language Processing (ICSLP), and the IEEE International Workshop on Machine Learning for Signal Processing. He is a member of the IEEE Signal Processing Society and the IEEE Information Theory Society.

Andrew C. Singer (S'92–M'96–SM'05) received the S.B., S.M., and Ph.D. degrees, all in electrical engineering and computer science, from the Massachusetts Institute of Technology (MIT), Cambridge, in 1990, 1992, and 1996, respectively.

During the academic year 1996, he was a Postdoctoral Research Affiliate at the Research Laboratory of Electronics, MIT. From 1996 to 1998, he was a Research Scientist at Sanders, A Lockheed Martin Company, Manchester, NH. Since 1998, he has been on the faculty of the Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, Urbana, where he is currently an Associate Professor at the Electrical and Computer Engineering Department, a Research Associate Professor at the Coordinated Science Laboratory, and a Willett Faculty Scholar. Since 2005, he has also served as the Director of the Technology Entrepreneur Center in the College of Engineering. His research interests include statistical signal processing, communication systems, and machine learning.

Dr. Singer serves as an Associate Editor for the IEEE TRANSACTIONS ON SIGNAL PROCESSING. He was a Hughes Aircraft Masters Fellow and the recipient of the Harold L. Hazen Memorial Award for excellence in teaching. In 2000, he received the National Science Foundation CAREER Award. In 2001, he received the Xerox Faculty Research Award. He is a member of the MIT Educational Council and of Eta Kappa Nu and Tan Beta Pi.