Correspondence

Universal Randomized Switching

Suleyman S. Kozat and Andrew C. Singer

***Abstract*—In this paper, we consider a competitive approach to sequential decision problems, suitable for a variety of signal processing applications where at each of a succession of times, a selection must be made from among a fixed set of strategies (or outcomes). For each such decision and outcome pair, loss is incurred, and it is the time-accumulation of these losses that is sought to be minimized. Rather than using a statistical performance measure, our goal in this pursuit is to sequentially accumulate loss that is no larger than that of the best loss that could be obtained through a partitioning of the sequence of observations into an arbitrary fixed number of segments and independently selecting a different strategy for each segment. For this purpose, we introduce a randomized sequential algorithm built upon that of Kozat and Singer that asymptotically achieves the performance of a noncausal algorithm that would be able to choose the number of segments and the best algorithm for each segment, based on observing the whole observation process *a priori*. In addition to improving upon the bounds of Kozat and Singer as well as Gyorgy *et al.*, the results we provide hold for more general loss functions than the square-error loss studied therein.**

***Index Terms*—Prediction, quantization, randomized, sequential decisions, switching, universal.**

I. Introduction

In many signal processing applications that employ sequential decisions, such as prediction and quantization, performance is often measured using an assumed statistical ensemble over the set of possible outcomes, such as the mean-squared error, variance, or entropy. However, for this paper, we follow the individual sequence approach taken in universal source-coding, where we choose not to place any statistical distribution over the set of possible outcomes [1]. This gives rise to the challenge of how to assess performance of our results. Following the universal source coding literature [3], we simply create a "competition class" of algorithms. If this competition class is sufficiently rich, then by measuring performance of *any* algorithm with respect to the *best* member of this class, for each individual sequence of outcomes, if we can approach this best-in-class performance, then we are satisfied that our approach is, in this sense, a "good" approach. Note that the competition class is introduced solely for the purpose of creating a performance benchmark and we do not require our approach to actually be drawn from this class. A common misconception of this approach is that one could achieve the best-in-class performance by simply selecting the best member of the competition class for each individual sequence *a priori*. This would require observation of the entire sequence of outcomes before the first decision is made. However, due to the strict sequentiality of the algorithms we desire, we are not able to this, and what is being "learned" in this approach is the *cost* of learning the "best" member from this class *sequentially*. For certain estimation and modeling problems, one does have the ability to see all of the data in advance, and in such applications, could simply select the best from the class. This form of *batch* estimation or learning is not possible for the problems we consider here.

In this paper, the desired real-valued individual sequence is represented by $\{y[t]\}_{t\geq 1}$ and $y[t]$ is revealed sequentially. At each time t, we observe outcomes of m different algorithms producing estimates $\hat{y}_j[t]$, $j = 1,\ldots,m$, of $y[t]$. Although these algorithms are not constrained to be sequential, we observe the outputs of these algorithms sequentially, i.e., at time t, we only have access to $\{\hat{y}_j[1],\ldots,\hat{y}_j[t]\}$, $j = 1,\ldots,m$. Based on these estimates, at each time t, each algorithm suffers a loss $l(y[t],\hat{y}_j[t])$, where $l(\cdot,\cdot) : \mathbb{R}\times\mathbb{R}\to[0,A]$, $A\in\mathbb{R}^+$, $A<\infty$, is any bounded loss function. For any n, the accumulated loss of the jth algorithm up to time n is given by $\sum_{t=1}^{n} l(y[t],\hat{y}_j[t])$, $j = 1,\ldots,m$. We define, for any n, a partition of $\{1,\ldots,n\}$ into $r+1$ segments at times $1<t_1<t_2<\ldots t_r<n+1$ as $\boldsymbol{t}_{r,n}=(t_1,\ldots,t_r)$ such that $\{1,\ldots,n\}$ can be represented as the concatenation of

$$\{1,\ldots,t_1-1\}\{t_1,\ldots,t_2-1\}\ldots\{t_r,\ldots,n\}.$$

For notational simplicity, we take $t_0=1$ and $t_{r+1}=n+1$. We permit, for any n, at most $r\leq n-1$ switches among algorithms within the competition class. For any r transitions (or switches), i.e., $r+1$ segments, there exist $\binom{n-1}{r}$ possible ways of choosing a switching pattern $\boldsymbol{t}_{r,n}$ to partition $\{1,\ldots,n\}$. Since the number of switches can be $0\leq r\leq n-1$, there exist a total of $2^{n-1}=\sum_{r=0}^{n-1}\binom{n-1}{r}$ possible switching patterns to partition $\{1,\ldots,n\}$.

For the purpose of a comparison benchmark, for any n, given any partition $\boldsymbol{t}_{r,n}$ of $\{1,\ldots,n\}$ into $r+1$ segments, a constituent algorithm can be selected from the class of m algorithms for each segment independently. We represent this selection by the assignment $\boldsymbol{a}_{r+1}=(a_1,\ldots,a_{r+1})$, $a_i\in\{1,\ldots,m\}$, e.g., if for region i, $a_i=j$, then for the ith segment $\hat{y}_j[t]$ is the selected output. For $r+1$ segments, this results $m(m-1)^r$ different mappings $\boldsymbol{a}_{r+1}$. Corresponding to any pairing of a switching pattern $\boldsymbol{t}_{r,n}=(t_1,\ldots,t_r)$ and a mapping $\boldsymbol{a}_{r+1}$, we can construct a sequential algorithm with outputs given by $\hat{y}_{(\boldsymbol{t}_{r,n},\boldsymbol{a}_{r+1})}[t]$. For each time t, this sequential algorithm will produce the estimate

$$\hat{y}_{(\boldsymbol{t}_{r,n},\boldsymbol{a}_{r+1})}[t]=\hat{y}_{a_i}[t]$$

if $t_{i-1}\leq t\leq t_i-1$, $t\leq n$. For each n, there exists a total of $\sum_{r=0}^{n-1}\binom{n-1}{r}m(m-1)^r$ such sequential switching algorithms using all pairings of all switching patterns and algorithm mappings. We define the accumulated sequential loss of the switching algorithm $\hat{y}_{(\boldsymbol{t}_{r,n},\boldsymbol{a}_{r+1})}[t]$ as

$$\begin{aligned}\delta(n,\boldsymbol{t}_{r,n},\boldsymbol{a}_{r+1}) &\triangleq \sum_{t=1}^{n} l\left(y[t],\hat{y}_{(\boldsymbol{t}_{r,n},\boldsymbol{a}_{r+1})}[t]\right)\\ &= \sum_{i=1}^{r+1}\sum_{t=t_{i-1}}^{t_i-1} l\left(y[t],\hat{y}_{a_i}[t]\right).\end{aligned}$$

Manuscript received January 29, 2008; accepted September 14, 2009. First published November 20, 2009; current version published February 10, 2010. This work is supported in part by TUBITAK Career Award, under Contract 108E195. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. William A. Sethares.

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

A. C. Singer is with the Department of Electrical and Computer Engineering, University of Illinois, Urbana, IL 61801 USA (e-mail: acsinger@illinois.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.2009.2037062

We consider the class of all such switching algorithms $\hat{y}_{(\boldsymbol{t}_{r,n},\boldsymbol{a}_{r+1})}[t],\ \forall \boldsymbol{t}_{r,n},\ \forall \boldsymbol{a}_{r+1},\ \forall r$, as a performance benchmark and seek a sequential algorithm that asymptotically achieves the performance of even the best algorithm in this class, uniformly for any $\{y[t]\}_{t\geq 1}$ and all n. Specifically, our goal is to find a sequential algorithm whose output, say $\hat{Y}[t]$, is constructed using the past samples of the desired signal $\{y[1],\ldots,y[t-1]\}$ and the outputs of the m constituent algorithms, $\{\hat{y}_j[1],\ldots,\hat{y}_j[t]\}$, $j=1,\ldots,m$, that, when used for estimation of $y[t]$ achieves an accumulated sequential loss, $\sum_{t=1}^{n} E[l(y[t],\hat{Y}[t])]$, nearly as small as the accumulated loss of any algorithm in the switching class for all n. Note that in defining the accumulated loss of $\hat{Y}[t]$, we take an expectation over the loss, since we allow randomized algorithms and the expectation is with respect to this randomization. However, the results introduced here hold uniformly for any deterministic $\{y[t]\}_{t\geq 1}$, under any bounded loss function. In this sense, we seek a sequential algorithm, with output $\hat{Y}[t]$, which does not depend on n, or any specific r, $\boldsymbol{t}_{r,n}$ or $\boldsymbol{a}_{r+1}$, however, when applied to any $\{y[t]\}_{t\geq 1}$, will satisfy

$$\frac{1}{n}\left\{\left(\sum_{t=1}^{n} E\left[l\left(y[t],\hat{Y}[t]\right)\right]\right)-\delta(n,\boldsymbol{t}_{r,n},\boldsymbol{a}_{r+1})\right\}\leq \frac{o(rn)}{n} \quad (1)$$

for all $n>0$, any fixed r, $r=0,\ldots,n-1$, all switching patterns $\boldsymbol{t}_{r,n}$ and all assignments $\boldsymbol{a}_{r+1}$, where $\lim_{n\to\infty}(o(n)/n)=0$. The desired algorithm is sequential in the sense that $\hat{Y}[t]$ can only depend on $y[1],\ldots,y[t-1]$ and the outputs of the m adaptive algorithms, $\{\hat{y}_j[1],\ldots,\hat{y}_j[t]\}$, $j=1,\ldots,m$, but nothing from the future or present, $y[\tau]$, $\tau\geq t$, explicitly. However, we place no such constraints on the output of the constituent algorithms.

For this purpose, we introduce a randomized algorithm that is built upon the sequential algorithm of [1, Th. 3]. At each time, the algorithm randomly selects one of the m constituent algorithms based on certain weights assigned to each constituent algorithm and repeats the outcome of this selected algorithm as its own. The sequential algorithm of [1] generates an outcome, however, as a convex combination of the outcomes of the m constituent algorithms using similar weights to those derived here. The results presented in [1] hold for bounded real-valued observations under square-error loss, however, we permit more general loss functions here. We introduce a randomized version of [1] for certain signal processing problems where one is expected to *choose* a particular strategy from a class of strategies at each time, instead of producing a new outcome based on the outcomes of the constituent algorithms. A well-studied problem that fits this framework is tracking a finite class of finite-delay scalar quantizers [2], [4]. A fixed-rate finite-delay sequential scalar quantizer is defined as a quantizer-decoder pair, where at each time, the observation process $y[t]$ is quantized, or mapped, into a finite set of symbols. These symbols can then be stored or noiselessly transmitted to a decoder for the reconstruction of the original signal. Hence, one can take each quantizer-decoder pair as a constituent estimation algorithm and use an algorithm such as the one introduced here, with the coding scheme of [4], to track the best scalar quantizer from a class with a finite number of quantizer-decoder pairs [2], to achieve the performance of the best switching scalar quantizer-decoder pair. Note that selecting the best quantizer with the corresponding quantization level at each time t and transmitting this information to the decoder requires transmission of indexes of both the best quantizer and the quantization level. Accordingly, one can determine the corresponding weights assigned to each algorithm and send these weights with the corresponding quantization level of each quantizer to produce a weighted combination of these outputs using the method introduced in [1] or a committee-based method [5]. However, such an approach will require transmission of the quantization indexes for all quantizers. It has been shown in [6] that using a randomized algorithm such as the one introduced here, which randomly selects a quantizer based on assigned weights and only transmits the corresponding quantization level index, along with the weights, can achieve the performance of the best switching quantizer. This method only needs a prediction algorithm achieving a redundancy rate such as the one in (1). We introduce such an algorithm with complexity only linear in data length. Furthermore, it has been also demonstrated in [6] that with a certain time allocation scheme, such an algorithm will only need to send $\log M$ bits (where M is the number of quantization levels) per transmission to achieve the intended performance.

The problem formulation studied in this paper has been described in the computational learning theory literature under the guise of "tracking the best expert" [5], [7]–[9] and later in the signal processing literature as universal switching algorithms [1]. As detailed in [1] and [7], without any consideration to algorithmic complexity, this problem can be readily solved using a *naive algorithm* [9] which combines, at each time n, all $\sum_{r=0}^{n-1}\binom{n-1}{r}m(m-1)^r$ sequential switching algorithms using a convex combination [10] (or using the Aggregating Algorithm (AA) [11]) or by a randomized algorithm [7]. These universal algorithms will yield a regret of $O((r+1)\ln(m)+r\ln(n/r))$ [10] or $O(\sqrt{(r+1)\ln(m)+r\ln(n/r)})$ [7] respectively over the best algorithm with the best transition times. Nevertheless, this *naive algorithm* [7] needs to implement and track weights for an exponential number of possible algorithms, which is not a plausible task.

In [7], the authors introduced two algorithms that store only m weights, one for each constituent algorithm, and that have complexity $O(m)$ per sample. The excess loss of these algorithms over the best switching algorithm is of the same order as that of the *naive algorithm*, however, to achieve these performance bounds, certain parameters, such as the switching rate of the underlying process, must be optimized *a priori*. The algorithms introduced in [7] were later used in [2] for tracking the best scalar quantizer. These algorithms were then demonstrated to be an application of the AA to combine (or derandomize) a continuum of certain elementary predictors in [9], using a quasi-probabilistic interpretation. Using this point of view, the author in [9] introduced extended versions of these algorithms attaining the performance of the *naive algorithm*, that do not need any *a priori* information. However, these new algorithms have complexity that is $O(n)$ per sample and they are in the same spirit of [1], where instead of using a convex combination, the outcomes of the constituent algorithms are merged using the AA to produce a new outcome. The approach taken in this paper builds on that of the *naive algorithm* in [9] using the idea that an appropriately weighted average over all elementary predictors (as in [9]) asymptotically achieves the performance of the best predictor in the class of all switching algorithms. The main contribution of this paper is that this impractically complex, and therefore infeasible, mixture can actually be efficiently implemented and done so in a sequential manner. In the process of developing bounds, we construct a prediction algorithm, with a time complexity that is linear in the data length, that is sequential in that it does not require knowledge of the present or future data, or the length of the data or the times for optimal switching.

We begin by introducing the randomized universal algorithm and the corresponding performance result in Section II. We then provide the proof of the performance result. We conclude the paper with simulations of the algorithm for the problem of tracking the best scalar quantizer.

II. Randomized Algorithm

In this framework, the desired real-valued individual sequence is represented by $\{y[t]\}_{t\geq 1}$. At each time t, we observe outcomes

of m algorithms producing estimates $\hat{y}_j[t]$, $j = 1,\ldots,m$ of $y[t]$. We note that these constituent algorithms need not to be sequential. However, we observe the outputs of these algorithms sequentially, i.e., at time t, we have only access to $\{\hat{y}_j[1],\ldots,\hat{y}_j[t]\}$, $j = 1,\ldots,m$. Based on these estimates, each algorithm suffers a loss $l(y[t],\hat{y}_j[t])$, $j = 1,\ldots,m$. For any n, the accumulated loss of each algorithm is given by $\sum_{t=1}^{n} l(y[t],\hat{y}_j[t])$. For any n, we also have $\sum_{r=0}^{n-1}\binom{n-1}{r}m(m-1)^r$ switching algorithms with outputs $\hat{y}_{(\boldsymbol{t}_{r,n},\boldsymbol{a}_{r+1})}[t]$, for all $r = 0,\ldots,n-1$, $\boldsymbol{t}_{r,n}$, and $\boldsymbol{a}_{r+1}$. Each such switching algorithm will suffer a loss $l(y[t],\hat{y}_{(\boldsymbol{t}_{r,n},\boldsymbol{a}_{r+1})}[t])$, and, for any n, will have an accumulated sequential loss $\delta(n,\boldsymbol{t}_{r,n},\boldsymbol{a}_{r+1})$. We introduce a randomized sequential algorithm which has access to $\{y[1],\ldots,y[t-1]\}$, $\{\hat{y}_j[1],\ldots,\hat{y}_j[t]\}$, $j = 1,\ldots,m$, and selects, at each time t, a constituent algorithm from the set of m constituent algorithms and repeats the output of this selected algorithm as its own to estimate $y[t]$. As an example, if the universal algorithm selects the ith constituent algorithm at time t, then the output of the universal algorithm is $\hat{Y}[t] = \hat{y}_i[t]$ at time t. The randomized algorithm, which does not depend on a particular switching pattern, is given as follows:

Algorithm

Step 1: Fix constants η and ϵ, such that $\eta,\epsilon \in \mathbb{R}^+$. At time $t = 1$, initialize weights $w_j[1] = 1/m$ and auxiliary variables $S[0,0,j] = 1/m$, $j = 1,\ldots,m$.

Step 2: At each time t

Step 2.1 Select one of the m constituent algorithms randomly based on the following probabilities: $P(\hat{y}_i[t] \text{ selected}) = w_i[t]$, where $\sum_{i=1}^{m} w_i[t] = 1$.

Step 2.2 Output $\hat{Y}[t] = \hat{y}_i[t]$.

Step 2.3 Update auxiliary variables.

For $s = 1,\ldots,t-1$ and $j = 1,\ldots,m$: $S[t,s,j] = S[t-1,s,j]W_{t-1}(s|s)\exp(-\eta l(y[t],\hat{y}_j[t]))$.

For $s = t$ and $j = 1,\ldots,m$:

$$S[t,t,j] = \sum_{s=1}^{t-1}\left[\left(\sum_{z=1,z\neq j}^{m} S[t-1,s,z]\right)W_{t-1}(t|s)\frac{1}{m-1} \times \exp\left(-\eta l\left(y[t],\hat{y}_j[t]\right)\right)\right].$$

Step 2.4 Update weights for each constituent algorithm using auxiliary functions:

$$w_j[t+1] = \sum_{s=1}^{t}\frac{S[t,s,j]}{\sum_{p=1}^{t}\sum_{z=1}^{m} S[t,p,z]}.$$

Here, $W_{t-1}(t|s) \triangleq 1/t^{1+\epsilon}/(Z_\infty - Z_{t-1})$, where $Z_t \triangleq \sum_{i=1}^{t}(1/i^{1+\epsilon})$ and $W_{t-1}(s|s) = 1 - W_{t-1}(t|s)$. As given in the proof of the following theorem, selecting different functions for $W_{t-1}(t|s)$ will yield different algorithms with corresponding different performance bounds as detailed in [1] and [12]. Hence, we use a generic $W_{t-1}(t|s)$ enabling a general implementation. We note that the results we introduce for this algorithm hold for any real number $\epsilon > 0$ as shown in the proof of the theorem and ϵ does not need to be optimized to yield the corresponding performance bounds. We further point out that optimal selection of the parameter η can be surpassed using a doubling approach [6], [7], as explained after we introduce the corresponding theorem. Hence, we observe that while combining (or tracking) 2^{n-1} different transition paths and $m(m-1)^r$ different assignments for each path, the randomized algorithm requires only $O(n)$ computations per output and stores $O(n)$ variables per sample. We achieve this by compactly storing the transition path information using the auxiliary variable $S[t,s,j]$ as illustrated in the proof of the following theorem. For the universal algorithm with output $\hat{Y}[t]$, we have the following result.

Theorem: Let $\{y[t]\}_{t\geq 1}$ represent the desired arbitrary real-valued individual sequence. At each time t, outputs of m different constituent algorithms, $\hat{y}_j[t]$, $j = 1,\ldots,m$, producing estimates of $y[t]$ are observed. Given parameters η and ϵ, such that $\eta,\epsilon \in \mathbb{R}^+$, the universal algorithm with output $\hat{Y}[t]$, which does not depend on the number of switches or switching patterns, when applied to $\{y[t]\}_{t\geq 1}$, satisfies, for all n, r, $r = 0,\ldots,n-1$, $\boldsymbol{t}_{r,n}$, $\boldsymbol{a}_{r+1}$,

$$\frac{1}{n}\sum_{t=1}^{n} E\left[l\left(y[t],\hat{Y}[t]\right)\right] \leq \frac{\delta(n,\boldsymbol{t}_{r,n},\boldsymbol{a}_{r+1})}{n} + \frac{(r\ln(n)+(r+1)\ln(m))}{\eta n} + \frac{\eta A^2}{8} + O\left(\frac{r+1}{n}\right) + O\left(\frac{1}{\epsilon n}\right)$$

uniformly for any arbitrary deterministic $\{y[t]\}_{t\geq 1}$, under any bounded loss function $l(\cdot,\cdot) : R \times R \to [0,A]$, $A \in \mathbb{R}^+$, $A < \infty$. Selecting η to minimize the right-hand side yields

$$\frac{1}{n}\sum_{t=1}^{n} E\left[l\left(y[t],\hat{Y}[t]\right)\right] \leq \frac{\delta(n,\boldsymbol{t}_{r,n},\boldsymbol{a}_{r+1})}{n} + \sqrt{\frac{A^2\left(r\ln(n)+(r+1)\ln(m)\right)}{2n}} + O\left(\frac{r+1}{n}\right) + O\left(\frac{1}{\epsilon n}\right).$$

The theorem states that the performance of the randomized algorithm is asymptotically as good as the performance of the best switching algorithm (that can only be chosen in hindsight), and the additional regret over the best switching algorithm is only $O(\sqrt{\ln(n)/n})$. We point out that for the optimized value of η, one needs to know n ahead of time, i.e., for the minimum regret, $\eta^* = \sqrt{(8(r\ln(n)+(r+1)\ln(m)))/A^2 n}$. However, this need for *a priori* information can be easily surpassed as in [6] and [7] by applying the randomized algorithm to known fixed-length intervals, where the length of each interval is exponentially increased and the algorithm resets itself at the start of each interval. For each interval, the optimized value of the η for that interval can be used. However, overall with such a scheme, we do not need to known the total length n.

Proof of the Theorem: The proof of the theorem follows similar lines as those of the proof of Theorem 3 of [1]. Hence, we only present significant differences. Given a transition pattern $\boldsymbol{t}_{r,n}$ with r transitions and a mapping of constituent algorithms to each segment $\boldsymbol{a}_{r+1}$, we define a function of the accumulated loss of the algorithm corresponding to this pair as $P(\boldsymbol{t}_{r,n},\boldsymbol{a}_{r+1}) \triangleq \exp(-\eta\delta(n,\boldsymbol{t}_{r,n},\boldsymbol{a}_{r+1}))$, where $\eta > 0$. Clearly, $P(\boldsymbol{t}_{r,n},\boldsymbol{a}_{r+1})$ reflects the performance of the pair $(\boldsymbol{t}_{r,n},\boldsymbol{a}_{r+1})$ on $\{y[t]\}_{t\geq 1}$ over a period of n samples. We then define a weighted sum of all such functions corresponding to all pairs $(\boldsymbol{t}_{r,n},\boldsymbol{a}_{r+1})$, $r = 0,\ldots,n-1$, as

$$S[n] \triangleq \sum_{\boldsymbol{t}_{r,n},\boldsymbol{a}_{r+1}} W(\boldsymbol{t}_{r,n},\boldsymbol{a}_{r+1})\exp\left(-\eta\delta(n,\boldsymbol{t}_{r,n},\boldsymbol{a}_{r+1})\right) \quad (2)$$

where the $W(\boldsymbol{t}_{r,n},\boldsymbol{a}_{r+1})$'s are weights to be selected as follows. The weighting for each pair, $W(\boldsymbol{t}_{r,n},\boldsymbol{a}_{r+1})$, is constructed as a mixture of two terms, i.e., $W(\boldsymbol{t}_{r,n},\boldsymbol{a}_{r+1}) = W_1(\boldsymbol{t}_{r,n})W_2(\boldsymbol{a}_{r+1})$. The first term,

$W_1(\boldsymbol{t}_{r,n})$, assigns weight to the corresponding switching pattern $\boldsymbol{t}_{r,n}$ and the second term, $W_2(\boldsymbol{a}_{r+1})$, assigns weight to $\boldsymbol{a}_{r+1}$. We select the first term $W_1(\boldsymbol{t}_{r,n})$ as the weight assigned to the binary sequence generated from $\boldsymbol{t}_{r,n}$ using ideas from universal source coding literature. From $\boldsymbol{t}_{r,n}$, we first generate a binary sequence of length n, by assigning the value 1 when there is a "switch" and the value 0 when there is "no switch." Hence, the binary sequence generated from $\boldsymbol{t}_{r,n}$ has r ones and $n-r$ zeros. Given this binary sequence, we then use any of the three different weight assignments for $W_1(\boldsymbol{t}_{r,n})$ introduced in [3] and [12]. It can be shown that all these three weight assignments $W_1(\boldsymbol{t}_{r,n})$ are sequentially computable and individually satisfy the following bounds:

$$-\ln W_{1,1}(\boldsymbol{t}_{r,n}) \le \frac{3r+1}{2}\ln(n/r) + O(r) \tag{3}$$

$$-\ln W_{1,2}(\boldsymbol{t}_{r,n}) \le (r+\epsilon)\ln(n) + \left(\ln(1+\epsilon) + r\ln\frac{1}{\epsilon}\right) \tag{4}$$

$$-\ln W_{1,3}(\boldsymbol{t}_{r,n}) \le (r+(r+1)\epsilon)\ln(n/r) + \left((r+1)\ln\frac{1+\epsilon}{\epsilon} + \ln\epsilon\right) \tag{5}$$

for all $\epsilon > 0$ and any $\boldsymbol{t}_{r,n}$, where $W_{1,1}(\boldsymbol{t}_{r,n})$ is from [3], $W_{1,2}(\boldsymbol{t}_{r,n})$ and $W_{1,3}(\boldsymbol{t}_{r,n})$ are from [12]. We will implement our algorithm generically such that any of these three weight assignments can be used in the implementation in place of $W_1(\boldsymbol{t}_{r,n}) = W_{1,i}(\boldsymbol{t}_{r,n})$, $i = 1,2,3$. For the presentation of the theorem, the bound in (4) is used. The second term is selected as $W_2(\boldsymbol{a}_{r+1}) = 1/m(m-1)^r$, since for $r+1$ segments, one can choose $\boldsymbol{a}_{r+1}$ among $m(m-1)^r$ different assignments. We assign each such assignment an equal weight, i.e., $1/m(m-1)^r$. It can be shown that $\sum_{\boldsymbol{t}_{r,n},\boldsymbol{a}_{r+1}} W(\boldsymbol{t}_{r,n},\boldsymbol{a}_{r+1}) = 1$ [3]. We observe that from (2)

$$-\ln(S[n]) \le \eta\delta(n,\boldsymbol{t}_{r,n},\boldsymbol{a}_{r+1}) + \ln\left(W(\boldsymbol{t}_{r,n},\boldsymbol{a}_{r+1})\right).$$

As seen, the logarithm of the total sum, $\ln(S[n])$, is as small as the total (η-weighted) error corresponding to any $\boldsymbol{t}_{r,n}$ and $\boldsymbol{a}_{r+1}$ pairing, plus the regret term $\ln(W(\boldsymbol{t}_{r,n},\boldsymbol{a}_{r+1}))$. However, using any of the bounds from (3), (4), or (5), as an example the one from (4), we have

$$-\ln(S[n]) \le \eta\delta(n,\boldsymbol{t}_{r,n},\boldsymbol{a}_{r+1}) + (r+\epsilon)\ln(n) + \left(\ln(1+\epsilon) + r\ln\frac{1}{\epsilon}\right) + (r+1)\ln(m). \tag{6}$$

Hence, it remains to show sequential universal algorithm that has an exponentiated error as small as $S[n]$.

To accomplish this, we will first demonstrate that $S[n]$ can be calculated recursively using the transition diagram introduced in [3], which was later used in a prediction context in [1]. To calculate $S[n]$, we need to sum $P(\boldsymbol{t}_{r,n},\boldsymbol{a}_{r+1})$ through every pairing $(\boldsymbol{t}_{r,n},\boldsymbol{a}_{r+1})$, for $r = 0,\ldots,n-1$. However, we observe that most of these switching paths share common features that can be used to combine the reconstruction error of many paths together. One way of characterizing switching patterns is to look at the most recent transition time, e.g., for a path $\boldsymbol{t}_{r,n}$ with r transitions, $r = 0,\ldots,n-1$, the last transition time is t_r. At time $t = n$, for all the paths that have the most recent transition time $t_r = s$ and using $\hat{y}_j[n]$ from $t = s$ to $t = n$, we combine their exponentiated error with the corresponding weights and define auxiliary variables

$$S[n,s,j] \triangleq \sum_{\boldsymbol{t}'_{r,n},\boldsymbol{a}'_{r+1}} W(\boldsymbol{t}_{r,n},\boldsymbol{a}_{r+1})\exp\left(-\eta\delta(n,\boldsymbol{t}_{r,n},\boldsymbol{a}_{r+1})\right) \tag{7}$$

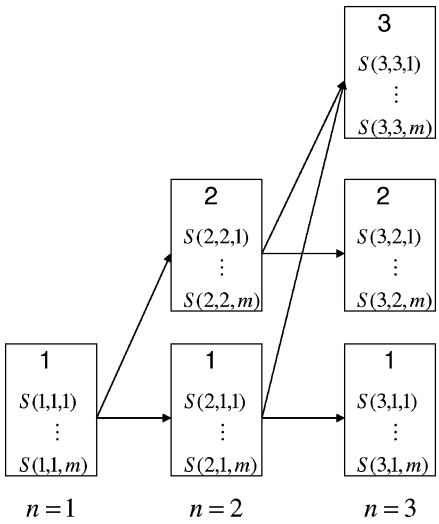

Fig. 1. Transition diagram for $n = 3$. On each box, the number on the top of the box is the time of the last transition. Each box stores m auxiliary variables, $S[n,s,j]$, $j = 1,\ldots,m$, each corresponding to a particular constituent algorithm.

where $\boldsymbol{t}'_{r,n}$ and $\boldsymbol{a}'_{r+1}$ are the paths and assignments such that the last transition time for $\boldsymbol{t}'_{r,n}$ is s and in the last segment $\hat{y}_j[t]$ is used, i.e., $a_{r+1} = j$. At time $t = n$, s can take n possible values $s = 1,\ldots,n$ and $j = 1,\ldots,m$. Hence, at each n, we have nm auxiliary variables as shown on Fig. 1.

We observe that when $s = 1,\ldots,n-1$, $S[n,s,j]$ can only be generated from the paths that also have the last transition time s, i.e., no transition at time n coming from $n-1$. Hence, it is enough to update $S[n-1,s,j]$ to get $S[n,s,j]$ when $s = 1,\ldots,n-1$. From $S[n-1,s,j]$ to $S[n,s,j]$, we need to first adjust the path weights $W_1(\boldsymbol{t}'_{r,n-1})$ in (7) since the length of the paths are increased by 1. Then, we need to account for the loss we acquired at $t = n$ using $\hat{y}_j[n]$. Combining these two effects yields

$$S[n,s,j] = S[n-1,s,j]W_{n-1}(s|s)\exp\left(-\eta l\left(y[n],\hat{y}_j[n]\right)\right)$$

$s = 1,\ldots,n-1$, $j = 1,\ldots,m$, where we define $W_{n-1}(s|s)$ as the weight adjustment to scale path weights $W_2(\boldsymbol{t}_{r,n})$. The exact forms of $W_{n-1}(s|s)$ to give different weight assignments of (3), (4), or (5) are given in [1] and [3].

However, if there is a new transition at time $n-1$, i.e., $s = n$, to a new algorithm other than j, then this switch can come from any $s = 1,\ldots,n-1$ and $z = 1,\ldots,m$, such that $z \ne j$. Hence

$$S[n,n,j] = \sum_{s=1}^{n-1}\left[\left(\sum_{z=1,z\ne j}^{m} S[n-1,s,z]\right)W_{n-1}(n|s)\frac{1}{m-1} \times \exp\left(-\eta l\left(y[n],\hat{y}_j[n]\right)\right)\right]$$

where $W_{n-1}(s|s) + W_{n-1}(n|s) = 1$, since each algorithm $z \ne j$ has equal weight of $1/(m-1)$. Hence, the update of $S[n,s,j]$ for all $s = 1,\ldots,n$ is given.

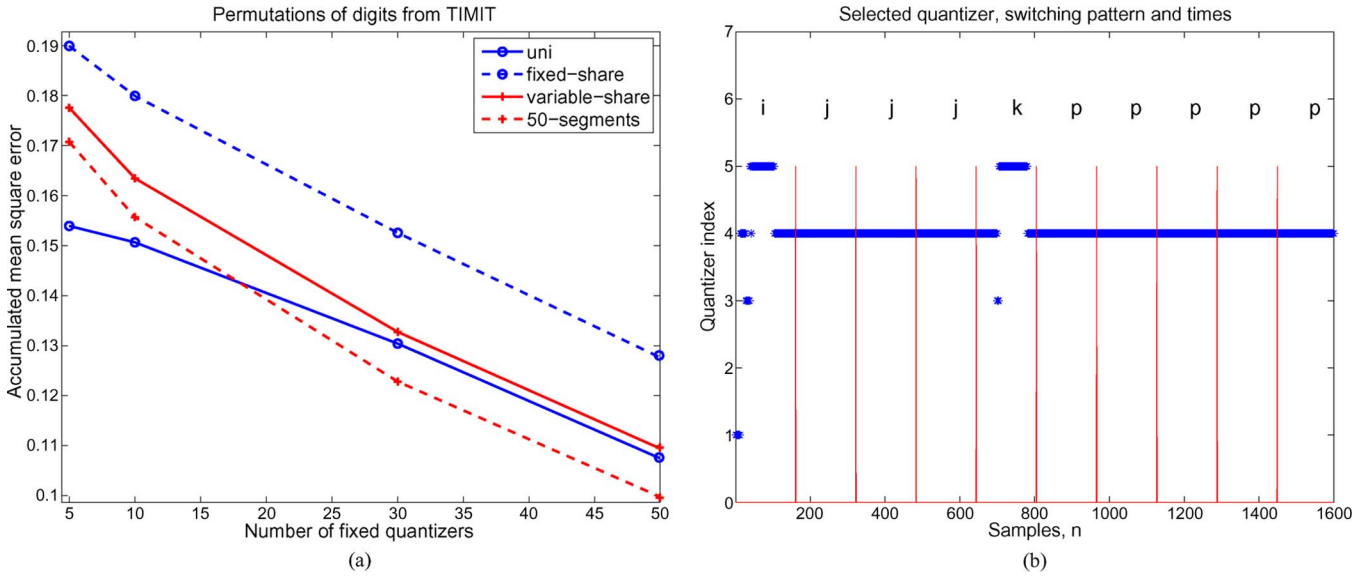

Fig. 2. (a) Normalized mean-square prediction error: universal algorithm "uni"; fixed-share [7] "fixed-share"; variable-share [7] "variable-share"; the best offline switching scalar quantizer with 50 segments. (b) Index of the scalar quantizer selected by the universal algorithm, blue stars, with the switching pattern on the top.

We observe from Fig. 1 that at each time $n-1$, from each $S[n-1,s,j]$, we have m possible transitions, i.e., either stay in the current algorithm or switch to another algorithm. Hence, we can write

$$\begin{aligned} S[n] &= \sum_{s=1}^{n-1}\sum_{j=1}^{m} S[n,s,j] \\ &= \sum_{s=1}^{n-1}\sum_{j=1}^{m} S[n-1,s,j] \\ &\quad \times\left\{ W_{n-1}(s|s)\exp\left(-\eta l\left(y[n],\hat{y}_j[n]\right)\right) + W_{n-1}(t|s) \right. \\ &\quad \left. \times\left[\sum_{z=1,z\neq j}^{m}\frac{1}{m-1}\exp\left(-\eta l\left(y[n],\hat{y}_z[n]\right)\right)\right]\right\} \end{aligned} \tag{8}$$

which is the recursive update for $S[n]$ from $n-1$ to n.

We next show that the randomized algorithm has an exponentiated total error as large as $S[n]$. By definition

$$S[n] = \prod_{t=1}^{n}\frac{S[t]}{S[t-1]} \tag{9}$$

which yields

$$\ln\left(S[n]\right) = \sum_{t=1}^{n}\ln\left(\frac{S[t]}{S[t-1]}\right). \tag{10}$$

However, for each term in (9), we use (8) to get

$$\frac{S[t]}{S[t-1]} = \frac{\sum_{s=1}^{t-1}\sum_{j=1}^{m} S\left[(t-1),s,j\right]B(t,s,j)}{\sum_{p=1}^{t-1}\sum_{i=1}^{m} S\left([t-1],p,i\right)} \tag{11}$$

where

$$\begin{aligned} B(t,s,j) \triangleq &\left\{ W_{n-1}(s|s)\exp\left(-\eta l\left(y[t],\hat{y}_j[t]\right)\right)\right. \\ &\left. + W_{n-1}(t|s)\left[\sum_{z=1,z\neq j}^{m}\frac{1}{m-1}\exp\left(-\eta l\left(y[t],\hat{y}_z[t]\right)\right)\right]\right\}. \end{aligned}$$

However, this yields

$$\frac{S[t]}{S[t-1]} = \sum_{s=1}^{t-1}\sum_{j=1}^{m}\frac{S\left[(t-1),s,j\right]}{\sum_{p=1}^{t-1}\sum_{i=1}^{m} S\left[(t-1),p,i\right]}B(t,s,j). \tag{12}$$

We observe that in the right-hand side of (12), without the exponential terms, we have

$$\begin{aligned} &\left\{\sum_{s=1}^{t-1}\sum_{j=1}^{m}\frac{S\left[(t-1),s,j\right]}{\sum_{p=1}^{t-1}\sum_{i=1}^{m} S\left[(t-1),p,i\right]}\left[W_{t-1}(s|s)(1)+\right.\right. \\ &\qquad \left.\left. W_{t-1}(t|s)\sum_{z=1,z\neq j}^{m}\frac{1}{m-1}(1)\right]\right\} = 1 \end{aligned} \tag{13}$$

since each term sums up to 1. We point out that by Hoeffding's inequality, $E[\exp(aX)] \leq \exp(aE[X] + (a^2A^2/8))$, for bounded random variables X such that $X \in [0,A]$, $A \in \mathbb{R}^+$ and $a \in \mathbb{R}$. Using this identity in the right-hand side of (12), due to (13), yields

$$\begin{aligned} \frac{S[t]}{S[t-1]} \leq &\exp\left\{-\eta\left[\sum_{s=1}^{t-1}\sum_{j=1}^{m}\mu_{t-1}(s,j)\right.\right. \\ &\times\left(W_{t-1}(s|s)l\left(y[t],\hat{y}_j[t]\right)+W_{t-1}(t|s)\right. \\ &\left.\left.\left.\times\frac{1}{m-1}\sum_{z=1,z\neq j}^{m} l\left(y[t],\hat{y}_z[t]\right)\right)\right]+\frac{\eta^2A^2}{8}\right\} \end{aligned} \tag{14}$$

where

$$\mu_{t-1}(s,j) \triangleq \frac{S\left((t-1),s,j\right)}{\sum_{p=1}^{t-1}\sum_{i=1}^{i} S\left((t-1),p,i\right)}$$

and A is an upper bound on the instantaneous loss. By definition of the weights of the universal randomized algorithm, we have

$$\begin{aligned} E\left[l\left(y[t],\hat{Y}[t]\right)\right] = &\sum_{s=1}^{k-1}\sum_{j=1}^{m}\mu_{t-1}(s,j)\left[W_{t-1}(s|s)l\left(y[t],\hat{y}_j[t]\right)\right. \\ &\left. + W_{t-1}(t|s)\frac{1}{m-1}\sum_{z=1,z\neq j} l\left(y[t],\hat{y}_z[t]\right)\right]. \end{aligned}$$

This yields, by (14)

$$S[t]/S[t-1] \leq \exp(-\eta E[l(y[t],\hat{Y}[t])] + (\eta^2 A^2/8)).$$

Finally, by (9)

$$S[n] \leq \exp(-\eta E[\sum_{t=1}^{n} l(y[t],\hat{Y}[t])] + (n\eta^2 A^2/8))$$

and using (6), we get

$$E\left[\sum_{t=1}^{n} l\left(y[t],\hat{Y}[t]\right)\right] \leq \delta(n,\boldsymbol{t}_{r,n},\boldsymbol{a}_{r+1}) + (r+\epsilon)\frac{\ln(n)}{\eta} + \frac{1}{\eta}\left(\ln(1+\epsilon) + r\ln\frac{1}{\epsilon}\right) + (r+1)\frac{\ln(m)}{\eta} + \frac{n\eta A^2}{8}$$

for any $\boldsymbol{t}_{r,n}$ and $\boldsymbol{a}_{r+1}$, $r = 1,\ldots,n-1$. This completes the proof of the theorem. □

III. Simulations and Conclusion

We illustrate the performance of the introduced algorithm when it is applied to the problem of tracking the best scalar quantizer. Here, speech signals, each containing recording of a digit uttered by different speakers, are randomly selected from TIMIT[1] database. Each speech file is decimated appropriately after antialiasing filtering is applied. Then, different permutations of speech files are concatenated to get 10-digit numbers with a specific pattern, i.e., $ijjjkppppp$, where i,j,k,p are different speech files. We then randomly generated 8-level quantizers as our constituent algorithms and plot in Fig. 1 the accumulated and normalized mean-squared error as a function of the number of quantizers, i.e., m, included in the constituent set, when the results are averaged over 200 different permutations. At each time t, the square prediction of a scalar quantizer, say Q_j, $j = 1,\ldots,m$, is the closest point to $y[t]$ in Q_j in the Euclidean norm, i.e., $q^*[t] = \arg\min_{q\in Q_j}(y[t]-q)^2$ and the prediction error is given as $(y[t]-q^*[t])^2$. We simulate four different algorithms including: the universal randomized algorithm "uni"; the fixed-share algorithm "fixed-share" [7]; the variable-share algorithm "variable-share"; and the best offline switching scalar quantizer with 50 equally spaced segments, in Fig. 2(a). For all algorithms, the learning rate is set to $\eta = 1$. For fixed-share and variable-share algorithms, we use a switching rate parameter of 1/160, which is the switching rate to achieve the optimal guaranteed performance bounds. We also plot in Fig. 2(b), the selected quantizer by the randomized algorithm and the switching pattern $ijjjkppppp$ on the top. We observe that, for these simulations, the randomized algorithm correctly captures the switching instants and has superior performance among the online algorithms in a mean-squared error sense.

In this paper, we presented a randomized algorithm that asymptotically performs as well as the best algorithm that can partition an observation sequence into an arbitrary number of segments and fit in each segment the best algorithm from a finite set of algorithms by observing the whole process *a priori*. The algorithm introduced chooses one of the constituent algorithms in this finite set based on assigned probabilities and replicates the outcome of the selected algorithm. The universal algorithm is a modified version of one of the algorithms introduced in [1] for this problem setup and is shown to have an excess loss of only $O(\sqrt{\ln(n)/n})$ over the best switching algorithm, with complexity only linear in data length per sample.

[1]TIMIT Acoustic-Phonetic Continuous Speech Corpus, Linguistic Data Consortium, Philadelphia, 1993.

References

[1] 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.

[2] A. Gyorgy, T. Linder, and G. Lugosi, "Tracking the best quantizer," *IEEE Trans. Inf. Theory*, vol. 54, no. 4, pp. 1604–1625, Apr. 2008.

[3] 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.

[4] N. Merhav, "On the minimum description length principle for sources with piecewise constant parameters," *IEEE Trans. Inf. Theory*, vol. 39, no. 6, pp. 1962–1967, Nov. 1993.

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

[6] 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.

[7] M. Herbster and M. K. Warmuth, "Tracking the best expert," in *Proc. Int. Conf. Machine Learning*, 1995, pp. 286–294.

[8] M. Herbster and M. K. Warmuth, "Tracking the best linear predictor," *J. Mach. Learn. Res.*, pp. 281–309, 2001.

[9] V. Vovk, "Derandomizing stochastic prediction strategies," *Mach. Learn.*, vol. 35, pp. 247–282, 1999.

[10] 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.

[11] V. Vovk, "Aggregating strategies," in *Proc. COLT*, 1990, pp. 371–383.

[12] 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.