Switching Strategies for Sequential Decision Problems With Multiplicative Loss With Application to Portfolios

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

***Abstract*—A wide variety of problems in signal processing can be formulated such that decisions are made by sequentially taking convex combinations of vector-valued observations and these convex combinations are then multiplicatively compounded over time. A "universal" approach to such problems might attempt to sequentially achieve the performance of the best fixed convex combination, as might be achievable noncausally, by observing all of the outcomes in advance. By permitting different piecewise-fixed strategies within contiguous regions of time, the best algorithm in this broader class would be able to switch between different fixed strategies to optimize performance to the changing behavior of each individual sequence of outcomes. Without knowledge of the data length or the number of switches necessary, the algorithms developed in this paper can achieve the performance of the best piecewise-fixed strategy that can choose both the partitioning of the sequence of outcomes in time as well as the best strategy within each time segment. We compete with an exponential number of such partitions, using only complexity linear in the data length and demonstrate that the regret with respect to the best such algorithm is at most $O(\ln(n))$ in the exponent, where n is the data length. Finally, we extend these results to include finite collections of candidate algorithms, rather than convex combinations and further investigate the use of an arbitrary side-information sequence.**

***Index Terms*—Convex combinations, portfolio, sequential decisions, side information, switching, universal.**

I. Introduction

In this paper, we consider sequential decision problems when the metric of performance arises from a convex combination of observations and is multiplicatively compounded from decision to decision, such as in gambling, investing or probabilistic model fitting. We pay specific attention to problems in which the decisions made at each time amount to picking a strategy that defines a convex combination of the next set of vector-valued outcomes. Once these outcomes are observed, the gains associated with all past decisions are then multiplied by the gain achieved by the most recent convex combination of outcomes to construct the accumulated gains over the data sequence observed so far. For the case of sequential investing, we consider a market with a finite number, say m, of stocks to trade. The same framework applies to sequential gambling problems, in which there are a finite number of outcomes on which to place bets. Similarly, for probabilistic model fitting, when the models are constrained to take convex combinations of a fixed set of given models and the resulting model likelihood conditioned on the past is taken as this convex combination.

Statistical language models (SLMs) are widely used in a variety of signal processing applications including automatic speech recognition (ASR), speech-to-speech translation, and natural language processing (NLP) [1]. In its most basic form, a statistical language model assigns conditional probabilities to successive words in a sequence of words $(w_1, w_2, \ldots, w_n)$ based on the time history of the sequence. For a given statistical model, the conditional probability of a word w_i is represented by $p(w_i|h_i)$, where h_i may represent the context, or the history, of the word w_i in $(w_1, w_2, \ldots, w_n)$. The performance of a particular SLM on a sequence of words is given by $\prod_{i=1}^{n} p(w_i|h_i)$, where the square root of $2^{-(1)/(n)\ln\prod_{i=1}^{n} p(w_i|h_i)}$ is defined as the perplexity (or the average word branching factor) of the language model [1]. In most ASR or NLP applications, multiple statistical language models are combined to construct a new statistical model in order to exceed the performance of each of the constituent language models [1], [2]. As an example, if the domain of an application is not known, then instead of extensively training a new SLM for this new domain, several existing SLMs (trained on other domains, for example) can be combined using a fairly small amount of training data (or held-out data) by tailoring the combination to this new domain [3]. The combined language model is usually constructed as a weighted mixture, or convex combination, of the constituent language models. Given m different SLMs, each assigning conditional probabilities $p_j(w_i|h_i)$, the probability assigned to each successive word by the weighted combination is given by $p(w_i|h_i) = \sum_{j=1}^{m} b_j p_j(w_i|h_i)$. These weights b_j's are usually optimized using a held-out set and chosen to maximize $\prod_{i=1}^{n} p(w_i|h_i) = \prod_{i=1}^{n}(\sum_{j=1}^{m} b_j p_j(w_i|h_i))$. However, when the held-out set is not present or the underlying statistics are changing, then the weights might be optimized online and adjusted accordingly to follow the statistical changes over time [4], [2].

Gaussian mixture models (GMMs) have been used extensively in the signal processing and pattern recognition to model or approximate a wide variety of density functions as weighted sums of Gaussian distributions [1]. Given a feature vector $\boldsymbol{y}$, a

Manuscript received October 15, 2007; accepted September 27, 2008. Current version published May 15, 2009. 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 Engineering Department, Koc University, Istanbul 06660, 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@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.2009.2013906

GMM with m such Gaussian distributions in its mixture, assigns $\boldsymbol{y}$ the probability $p(\boldsymbol{y}|\text{GMM}) = \sum_{j=1}^{m} b_j[\exp(-(1)/(2)(\boldsymbol{y}-\boldsymbol{\mu_j})^T\Sigma_j^{-1}(\boldsymbol{y}-\boldsymbol{\mu_j}))/((2\pi)^{N/2}|\Sigma_j|^{1/2})]$, where $\boldsymbol{\mu}_j$ and Σ_j are the mean vector and the covariance matrix, respectively, for the jth Gaussian distribution in the mixture. The probability assigned to a sequence of vectors $(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n)$ is then given as $\prod_{i=1}^{n} p(\boldsymbol{y}_i|\text{GMM})$. Such GMMs are used in most speaker recognition applications to model state occupation probabilities [1], in speaker recognition applications to represent probabilistic models of individual speakers [5] and in classifier design to represent class probability distributions [6]. In most of these applications, the weights of a mixture are optimized using a maximum likelihood criterion on an offline set of training data to maximize $\prod_{i=1}^{m} p(\boldsymbol{y}_i|\text{GMM})$, where $(\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_n)$ are the corresponding training data. However, to follow the variations or changes in the application statistics, it is often desirable to use online algorithms to adjust such weights or sequentially update them.

Similar weighted mixtures are used to combine probabilities arising from different modalities or attributes in multimedia signal processing applications. In audio–visual applications, probabilities derived from different modalities, each having its own probabilistic model, are combined as a weighted mixture to yield the final probability assigned to a given audio–visual observation [7]. In other speech applications, probabilities derived from models trained on cepstrum coefficients and LPC coefficients (or from different streams, or trained with different sets of speakers) are combined through a weighted mixture of probabilities to yield a final assigned probability for each feature vector [8]. In all these applications, the weights for the combination could be trained online or optimized adaptively to further improve sequential performance.

II. Portfolios as Weighted Combinations

In this paper, we look at weighted combinations of not only such model-based probabilities, but also general vectors $\boldsymbol{x}_i$ with only the constraint that they are nonnegative, i.e., each element of $\boldsymbol{x}_i$ is greater than or equal to zero. For simplicity of exposition, and by analogy to the sequential investing scenario, with such arbitrary $\boldsymbol{x}_i$, we refer to such weighted combinations as "portfolios," noting that the methods developed can be applied to any problem with convex decisions and a multiplicative performance metric.

By their nature, time series taken from prices of financial instruments, such as stock in publicly traded companies, are often both difficult to predict and exhibit nonstationary behavior. Within the signal processing community, a number of signal prediction, modeling and smoothing algorithms have been applied to such financial data, as many methods used for prediction and handling nonstationarity behavior arise naturally in other applications of interest to the community. Examples abound, but a brief listing of such general methods applied to financial data include correlative learning applied for prediction of option prices [9], nonlinear signal modeling [10], methods based on support vector machines [11] and particle filtering methods for smoothing stock price data [12].

Portfolio selection in particular is examined in [13] using meta-controlled Boltzmann machines and in [14] using ideas from universal lossless source coding. Portfolio modeling is investigated in [15] using nonlinear vector multiresolution methods and in [16] by perceptual indexing.

The behavior of a market with m stocks is governed by the sequence of prices of each stock over time, i.e., $p_j[t], j = 1, \ldots, m, t = 0, \ldots, n$. Since the gain or loss in an investment position held in any stock is a function of the relative change, rather than absolute price, in that stock, we are more interested in the sequence of price relatives, or gains, $x_j[t]$ in each stock. These gain vectors may either denote daily changes in the stock price, i.e., $x_j[t] = p_j[t]/p_j[t-1]$, or ratios of closing-to-opening prices in the stock, i.e., $x_j[t] = p_j^c[t]/p_j^o[t]$. As such, the market is modeled by a sequence of vectors $\{\boldsymbol{x}[t]\}_{t\geq 1}, \boldsymbol{x}[t] \in R_+^m$, where the jth entry, $x_j[t]$, of this vector $\boldsymbol{x}[t]$ represents the gain achieved from the jth stock on the tth trading day. Decisions made at day t then amount to investment decisions, or a "portfolio assignment," represented by the vector $\boldsymbol{b}[t]$ in the positive orthant, i.e., $\boldsymbol{b}[t] \in R_+^m$ and $\sum_{j=1}^{m} b_j[t] = 1$ for all t. Each entry $b_j[t]$ corresponds to the portion of the wealth invested in the stock j at day t. The wealth achieved after n trading periods on $\boldsymbol{x}^n$ is then given by $\prod_{t=1}^{n} \boldsymbol{b}^T[t]\boldsymbol{x}[t]$. When the vector $\boldsymbol{b}[t] = \boldsymbol{b}$, i.e., the apportionment of assets at each point in time is a fixed constant convex combination $\boldsymbol{b}$ over all time, the strategy is called a "constant rebalanced portfolio." This name arises from the need to rebalance the distribution of assets among stocks before each trading period, due to the changes in asset value that occur from the previous trading period. For example, when $m = 2$, and $\boldsymbol{b} = [0.5 \ 0.5]^T$, if $\boldsymbol{x}[t] = [2 \quad 1]^T$, then after trading period t, the assets in Stock 1 will be double those in Stock 2. Hence, to maintain the constant portfolio assignment $\boldsymbol{b} = [0.5 \quad 0.5]^T$, the assets need to be rebalanced, i.e., a portion of the assets in Stock 1 need to be sold and invested into Stock 2, such that the asset allocation between the two stocks is again equal. This framework is extensively discussed in [17]. In the context of probabilistic model combinations, such a "constant rebalanced portfolio" would amount to a constant weighting among constituent probabilistic models, i.e., an *a priori* weighting.

For our competitive framework, the performance measure is defined with respect to that of the best algorithm from a class of competing algorithms. As an example of this framework, Cover [17] presented a portfolio selection algorithm which achieves the rate of wealth growth of the best constant rebalanced portfolio (CRP) from the class of all constant rebalanced portfolios (CRPs) for any sequence of price relative vectors in hindsight. We refer to such algorithms that can asymptotically achieve the performance of the best algorithm from a given class of algorithms (for any sequence of outcomes) "static universal algorithms," since the competition class contains a fixed set of algorithms, and performance is compared with that of the best fixed element from the class applied to the entire outcome sequence.

In this paper, we extend results for static algorithms to a framework where the underlying competition class includes the ability to switch among the various static elements. We first investigate this problem, when each competing algorithm can divide the sequence of outcomes into arbitrary segments, say k

of them, and fit each contiguous segment with the best "portfolio" from a given class of static algorithms for that segment, such as a fixed CRP. For k such transitions, there exist $k+1$ segments. The total gain achieved by a class member with such a partition is the combined gains of all fixed static algorithms associated with each segment. The best partition is the one that yields the maximum total gain. We can also let the competing algorithm choose the number of possible switches k. A natural restriction for the number of possible segments (switches) is $k < n$, where n is the length of the sequence of outcomes. Unlike [17]–[19], here we try to exploit the time varying nature of the best choice of algorithm for any given sequence of outcomes, since the choice of best algorithm from a class of static algorithms can change over time. Nevertheless, instead of trying to find the best partition (best possible switching points) or best number of transitions, our objective is simply to achieve the performance of the best partition directly. The algorithms we provide are strongly sequential such that they do not need the number of transitions, times of these transitions, or length of the data n *a priori*; however, they can asymptotically achieve the gain of the best algorithm in the competition class which can select the number of transitions, locations of these transitions, and the best static algorithm for each segment, based on observing the whole sequence of outcomes in advance.

Let $\{\boldsymbol{x}[t]\}_{t\geq 1}$ represent a sequence of price relative vectors that is deterministic and arbitrary. For all n, a competing algorithm with a transition path $\mathcal{T}_{k,n}$ with k transitions, represented by $(t_1,\ldots,t_k)$, divides $\boldsymbol{x}^n$ into $k+1$ segments such that $\boldsymbol{x}^n$ is represented by the concatenation of

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

Given n and k, there exist $\binom{n-1}{k}$ such possible transition paths $\mathcal{T}_{k,n}$. Given the past values of the price relatives $\boldsymbol{x}[t], t = 1,\ldots,n-1$, a competing algorithm assigns a vector $\boldsymbol{b}_i$ in each segment as $\hat{\boldsymbol{b}}[t] = \boldsymbol{b}_i$, where $t_{i-1} \leq t < t_i, i = 1,\ldots,k+1$. For notational simplicity we assume $t_0 = 1$ and $t_{k+1} = n+1$. Here, the competing class contains all CRPs in each segment that have the same $\boldsymbol{b}_i$ for each sample of the sequence $\boldsymbol{x}[t]$ for $t = t_{i-1},\ldots,t_i-1$, where each $\boldsymbol{b}_i$ can be selected independently for each region $i = 1,\ldots,k+1$. In determining the best algorithm in the competing class, we attempt to outperform all such portfolios, including the one that has been selected by choosing the transition path $\mathcal{T}_{k,n}$, the number of transitions k and the CRP vectors $\boldsymbol{b}_i$ in each segment based on observing the entire sequence $\boldsymbol{x}^n$ in advance. As such, we try to find a sequential strategy when applied to $\{\boldsymbol{x}[t]\}_{t\geq 1}$ will minimize for any n the following gain ratio:

$$\sup_{\boldsymbol{x}^n} \frac{\sup_{\substack{\boldsymbol{b}_1,\ldots,\boldsymbol{b}_{k+1}\in\boldsymbol{R}_+^m \\ t_1<\ldots<t_k\in\{2,\ldots,n\}}} \prod_{i=1}^{k+1}\prod_{t=t_{i-1}}^{t_i-1} \boldsymbol{b}_i^T\boldsymbol{x}[t]}{\prod_{t=1}^{n}\hat{\boldsymbol{b}}^T[t]\boldsymbol{x}[t]} \tag{1}$$

where $\hat{\boldsymbol{b}}[t]$ is a sequential assignment at time t, i.e., $\hat{\boldsymbol{b}}[t]$ may be a function of $\boldsymbol{x}[1],\ldots,\boldsymbol{x}[t-1]$ but does not depend on the future, $\mathcal{T}_{k,n}$ is any transition path representing $(t_1,\ldots,t_k)$ with an arbitrary number of transitions k. In other words, we seek an algorithm such that even in the worst case, it will achieve the performance of the best algorithm in the competition class, uniformly for all sequences $\{\boldsymbol{x}[t]\}_{t\geq 1}$ and all n. We will construct a sequential portfolio selection algorithm for which the logarithm of this gain ratio, or regret, is at most $(k+1)(m-1)\ln n/2 + k\ln n + O(k+1)$ for any $\mathcal{T}_{k,n}, k$ or n without any knowledge of $\mathcal{T}_{k,n}, k$ or n *a priori*. Hence, for any $\{\boldsymbol{x}[t]\}_{t\geq 1}$, the performance of our algorithm on $\boldsymbol{x}^n$ for all n is at most $(k+1)(m-1)\ln n/2 + k\ln n + O(k+1)$ different than the optimal algorithm in the competition class that is tuned for $\boldsymbol{x}^n$ for all n. We recognize the term $(k+1)(m-1)\ln n/2$ as a parameter regret or additional loss due to uncertainty of the best CRP vector in each of the $k+1$ separate regions and the term $k\ln n$ as the transition path regret due to uncertainty in the best transitions times. In a separate paper, we demonstrated an algorithm similar to the algorithm introduced in here that achieves the performance of the best algorithm in the competition class with a rate $O((k+1)(m-1)\ln n)$, when there are fixed-rate transaction costs involved [20].

We then continue our discussion when there are a finite number of such strategies, e.g., $\hat{\boldsymbol{b}}_1[t],\ldots,\hat{\boldsymbol{b}}_M[t]$ for some M, for the competing algorithm to choose within each segment [21]. As an example when we compete against "pure strategies," each $\hat{\boldsymbol{b}}_r[t]$ corresponds to selecting only one element from the mixture (i.e., a single stock). In this case, a competing algorithm chooses both the transition path and one of these strategies (single mixture element) for each segment of the transition path. In determining the best algorithm in the class, we attempt to outperform all such strategies, including the one that has been selected by choosing the transition path $\mathcal{T}_{k,n}$, the number of transitions k and the best strategy $\hat{\boldsymbol{b}}_r[t]$ in each segment based on observing the entire sequence $\boldsymbol{x}^n$ in advance. Hence, we try to minimize the gain ratio shown in (2) at the bottom of the page, where $\hat{\boldsymbol{b}}[t]$ is a sequential portfolio assignment at time t, i.e., $\hat{\boldsymbol{b}}[t]$ is a function of $\boldsymbol{x}[1],\ldots,\boldsymbol{x}[t-1]$ but does not depend on the future, $\mathcal{T}_{k,n}$ is any transition path representing $(t_1,\ldots,t_k)$ with an arbitrary number of transitions k. We will show that we can construct a sequential selection algorithm for which the logarithm of this regret is at most

$$\sup_{\boldsymbol{x}^n} \frac{\sup_{t_1<\ldots<t_k\in\{2,\ldots,n\}} \prod_{i=1}^{k+1}\sup_{r\in\{1,\ldots,M\}}\prod_{t=t_{i-1}}^{t_i-1}\hat{\boldsymbol{b}}_r^T[t]\boldsymbol{x}[t]}{\prod_{t=1}^{n}\hat{\boldsymbol{b}}^T[t]\boldsymbol{x}[t]} \tag{2}$$

$(k+1)\ln M + k\ln n + O(k+1)$ for any of $\mathcal{T}_{k,n}, k$ or n with no knowledge of $\mathcal{T}_{k,n}, k$ or n *a priori*.

We then consider the case when the switching among the various static elements, e.g., $\hat{\boldsymbol{b}}_1[t], \ldots, \hat{\boldsymbol{b}}_M[t]$ is represented by a side-information sequence y^n, where $y[t] \in \{1, \ldots, S\}$. We use the side-information sequence to label the sequence of outcomes, e.g., without loss of generality, if $y[t] = r$, then we choose $\hat{\boldsymbol{b}}_r[t]$ for time t. Here, there exist S labels to choose from for each time. A competing algorithm can divide the sequence of vectors using this side-information sequence and assign a different algorithm to each label. The natural restriction is $M \geq S$. Unlike [17], [19], we do not assume that the side-information sequence or the algorithm to generate the side information are known *a priori*. In determining the best algorithm in the competing class, we attempt to outperform all such portfolios, including the one that has been selected by choosing the side-information sequence y^n and the best algorithm $\hat{\boldsymbol{b}}_r[t]$ for each label based on observing the entire sequence $\boldsymbol{x}^n$ in advance. In this case, we try to minimize

$$\sup_{\boldsymbol{x}^n} \frac{\sup_{y^n} \prod_{t=1}^{n} \hat{\boldsymbol{b}}_{y[t]}^T[t]\boldsymbol{x}[t]}{\prod_{t=1}^{n} \hat{\boldsymbol{b}}^T[t]\boldsymbol{x}[t]}$$

where $\hat{\boldsymbol{b}}[t]$ is a sequential portfolio assignment at time t, i.e., $\hat{\boldsymbol{b}}[t]$ may be a function of $\boldsymbol{x}[1], \ldots, \boldsymbol{x}[t-1]$ but does not depend on the future, y^n is any side-information sequence with S labels. We will show that the sequential algorithm constructed for the corresponding $\mathcal{T}_{k,n}$ in (2) representing y^n can be used to construct a sequential selection algorithm for which the logarithm of this regret is at most $(k+1)\ln M + k\ln n + O(k+1)$ for any y^n (and corresponding k), S, or n with no knowledge of y^n, S, or n *a priori*.

Universal algorithms that can compete against CRPs or against a set of finite portfolio selection algorithms are studied in [18], [19], [22], and [23]. In [19], the authors introduce a portfolio selection algorithm based on the multiplicative update rule of [24]. Although their bounds are inferior to [17], complexities of their algorithms are linear in n, i.e., the number of trading times. In [18], Vovk interpreted Cover's universal algorithm as a special case of his aggregating algorithm (AA) [25] and extended it to include different learning rates. In [26], a heuristic algorithm motivated by Cover's rebalancing strategy is introduced that is shown to have excellent performance on historical data. However, this algorithm is not a universal algorithm, i.e., it does not guarantee a low or vanishing regret which is the framework studied in this paper. Although we use Cover's universal algorithm in our derivations for the corresponding bounds, the methods we use are generic. The algorithms we introduce can easily build on other algorithms such as [19] instead of [17], or any other algorithm that is universal with respect to a static class, such as [18], [19], [22], and [23]. The additional complexity of our algorithms over the complexity of the static algorithms used in the construction is linear in the data size n. Transaction costs (applicable both in the context of investing as well as to the probabilistic model mixture scenario as a regularization penalizing paths that switch too rapidly) can also be included within the framework of universal portfolios. In [27], after providing a simple analysis of [17], the authors show that an extended version of Cover's algorithm is also competitive in the presence of such costs. Extension of the basic universal portfolio selection problem with arbitrary observation vectors and positive portfolio vectors is also studied for bounded stocks and portfolios allowing margin and short sales in [28] and [18] and stocks with restrictions in [29]. Moreover, in [22], the authors extend the notion of internal regret to online portfolio selection. They refer to the form of regret studied in this paper, the regret between an online algorithm and the best from a class of algorithms, an "external regret." The examples of observation sequences provided in [22] that show good internal regret, but poor external regret, exhibit the type of piecewise behavior exploited by the algorithms developed in this paper, hence motivate this work.

Competition against portfolio selection algorithms that can switch among a finite number of M static algorithms (not against CRPs) is studied in [30]. In [30], the authors apply a sequential portfolio selection algorithm for each transition path $\mathcal{T}_{k,n}$ independently to construct universal algorithms that can compete against the class of all switching portfolios. Initially, they use a geometric distribution to weight each transition path $\mathcal{T}_{k,n}$. The first algorithm introduced requires a constant parameter to be selected *a priori* for the geometric distribution. This algorithm is along the lines of [24], where a similar algorithm is used for tracking the best expert in a universal prediction context. Nevertheless, due to the unbounded loss in this framework, i.e., $\ln \boldsymbol{w}^T\boldsymbol{x}$, the derivations in [24] do not extend to this case. This algorithm, with the *a priori* selected constant, has an additional regret of $O((k+1)\ln M) + O(n-k)$ over the algorithm with the best transition path. This initial algorithm [30] is then extended by allowing the constant parameter vary in time to appropriately weight all transition paths based on performance. This algorithm then achieves the performance of the best algorithm with the best transition path in the competition class with a regret $O((k+1)\ln M + (3/2)k\ln(n/k))$. This approach is similar to weighting methods introduced by [31] for tracking the best expert in the prediction context. Nevertheless, in [30], the authors are unable to provide a final portfolio assignment that achieves this performance. In this paper, we not only show that by using different weighting methods, we can improve the final regret to $O((k+1)\ln M + k\ln(n/k))$, but also show that we can *actually construct the portfolios* that can achieve this overall gain. The methods of [30] do not extend to include the best CRP in each region, due to the transition path structure used. We provide this by using a full transition path diagram to keep all the switching or segmentation information, together with a universal CRP for each such segment. The approach taken in this paper leverages work from universal source coding [32] and universal prediction [33]. The linear transition diagram of [32] was used in [33] for the problem of universal piecewise linear least squares prediction. Aside from using a similar data structure (the linear transition diagram), the methods of [33] apply to accumulated square error, rather than the loss considered here. Hence, the methods of [33] are not applicable to problem considered in this paper.

A related class of sequential decision problems with of a similar metric to that used in this paper are also considered in [34], where the authors investigate the portfolio selection problem

using context trees. Rather than considering a possible temporal switching among various constant mixtures, in [34], the space of past observation vectors is partitioned into disjoint regions represented using a context tree. A different mixture vector is then fitted to each region independently. The underlying mixture vector therefore becomes data dependent and can approximate any nonlinear function of the past observations through finer partitions within the context tree. However, unlike here, the context tree algorithm used to partition the sequence of past observations is fixed and can be thought as a static algorithm in the nomenclature of this paper. The context tree algorithm can be chosen as a constituent algorithm among many to be selected for each "region" as considered here. As an example, one can choose context trees with different partitions to compete against each other for each segment.

The organization of this paper is as follows. In Section II, we provide the main results of the paper as guaranteed performance results given through respective upper bounds on regret. The construction of the actual portfolios are provided in Section III along with the proofs of each theorem, where we give a complete Matlab implementation of the switching portfolio selection algorithm. We then conclude the paper with some example applications of these algorithms to simulated and real stock markets.

III. Upper Bounds

The main results of this section are given as guaranteed performance bounds contained in following theorems. We first investigate CRPs in each segment in Section III-A. We then extend our discussion to the case when there are only a finite class of algorithms in each segment to choose from in Section III-B. Competition against a finite class of algorithms in each segment is then extended to portfolios with side-information in Section III-C. The corresponding universal and strongly sequential portfolio algorithms that achieve these performance guarantees are constructed at the end of each proof in Section IV.

A. Constant Rebalanced Portfolios in Each Segment

Let $\{\boldsymbol{x}[t]\}_{t\geq 1}$ represent a sequence of price relative vectors. For all n, a competing portfolio selection algorithm with a transition path $\mathcal{T}_{k,n}$ with k transitions, represented by $(t_1,\ldots,t_k)$, divides $\boldsymbol{x}^n$ into $k+1$ segments such that $\boldsymbol{x}^n$ is represented by the concatenation of

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

Given the past values of the desired price relatives $\boldsymbol{x}[t], t=1,\ldots,n-1$, a competing algorithm assigns a portfolio vector $\boldsymbol{b}_i$ in each segment as $\hat{\boldsymbol{b}}[t]=\boldsymbol{b}_i$ where $t_{i-1}\leq t<t_i, i=1,\ldots,k+1$. For notational simplicity, we assume $t_0=1$ and $t_{k+1}=n+1$. For this setting we have the following theorem.

Theorem 1: We have a sequential strategy $\tilde{\boldsymbol{b}}_u[t]$ such that for any sequence of price relative vectors $\{\boldsymbol{x}[t]\}_{t\geq 1}, \boldsymbol{x}[t]\in\boldsymbol{R}_+^m$ (and some components of $\boldsymbol{x}[t]$ can be zero), the performance of $\tilde{\boldsymbol{b}}_u[t]$ on $\boldsymbol{x}^n$ for all n, defined as the following wealth ratio:

$$R_{\boldsymbol{b}}[n]=\frac{\sup_{\substack{\boldsymbol{b}_1,\ldots,\boldsymbol{b}_{k+1}\in\boldsymbol{R}_+^m\\ t_1<\ldots<t_k\in\{2,\ldots,n\}}}\prod_{i=1}^{k+1}\prod_{t=t_{i-1}}^{t_i-1}\boldsymbol{b}_i^T\boldsymbol{x}[t]}{\prod_{t=1}^{n}\tilde{\boldsymbol{b}}_u^T[t]\boldsymbol{x}[t]} \tag{3}$$

for all n and k, satisfies

$$\frac{\ln R_{\boldsymbol{b}}[n]}{n}\leq\frac{(k+1)(m-1)}{2}\frac{\ln(n+1)}{n}+(k+1)\frac{\ln 2}{n}+(k+\epsilon)\frac{\ln n}{n}+\frac{1}{n}\left(\log(1+\epsilon)+k\log\frac{1}{\epsilon}\right) \tag{4}$$

and

$$\frac{\ln R_{\boldsymbol{b}}[n]}{n}\leq\frac{(k+1)(m-1)}{2}\frac{\ln(n+1)}{n}+(k+1)\frac{\ln 2}{n}+\frac{3k+1}{2}\frac{\ln n}{n}+O\left(\frac{k}{n}\right) \tag{5}$$

for any $\mathcal{T}_{k,n}$ representing transition path $(t_1,\ldots,t_k)$ and any k, such that $\tilde{\boldsymbol{b}}_u[t]$ does not depend on $\mathcal{T}_{k,n}, k$ or n. The introduced $\tilde{\boldsymbol{b}}_u[t]$ has computational complexity linear in t^{m+1} for each investment period.

Construction of $\tilde{\boldsymbol{b}}_u[t]$ uses the portfolio introduced in Theorem 2 of Cover and Ordentlich [17]. The computational complexity, i.e., number of additions and multiplications, $O(t^{m+1})$ of $\tilde{\boldsymbol{b}}_u[t]$ per investment period is due to the calculation of Cover's universal portfolios [17] which has computational complexity $O(t^m)$ per investment. The actual complexity of the combination algorithm is $O(t)$. As an example, if one replaces Cover's algorithm with [19], the complexity overall would be $O(t)$, albeit with different performance guarantees. To implement $\tilde{\boldsymbol{b}}_u[t]$, we need to store and update t auxiliary variables at each investment period. Theorem 1 states that the regret of the universal sequential portfolio $\tilde{\boldsymbol{b}}_u[t]$ is within $O((k+1)m\ln n)$ of the best batch piecewise CRPs with k transitions (tuned to the underlying sequence in hindsight), uniformly, for every sequence of price relatives $\boldsymbol{x}^n$ and n.

B. Finite Number of Portfolio Selection Algorithms for Each Segment

Here, we compete against a finite set of algorithms in each region for a given k-partition. Let $\{\boldsymbol{x}[t]\}_{t\geq 1}$ represent a sequence of price relative vectors. We now consider a class of M different portfolio strategies producing $\hat{\boldsymbol{b}}_r[t], r=1,\ldots,M, t\geq 1$ working in parallel. For all n and a sequence of price relative vectors $\boldsymbol{x}^n=\{\boldsymbol{x}[1],\ldots,\boldsymbol{x}[n]\}$, a transition path $\mathcal{T}_{k,n}$ with k transitions, represented by $(t_1,\ldots,t_k)$, divides $\boldsymbol{x}^n$ and $\hat{\boldsymbol{b}}_r^n$ into $k+1$ segments such that $\boldsymbol{x}^n$ and each $\hat{\boldsymbol{b}}_r^n$ can be represented as a concatenation of

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

and

$$\{\hat{\boldsymbol{b}}_r[1],\ldots,\hat{\boldsymbol{b}}_r[t_1-1]\}\{\hat{\boldsymbol{b}}_r[t_1],\ldots,\hat{\boldsymbol{b}}_r[t_2-1]\}\ldots\{\hat{\boldsymbol{b}}_r[t_k],\ldots,\hat{\boldsymbol{b}}_r[n]\}$$

respectively. For all n, the competing algorithm achieving the maximum total wealth would choose the best k, the best transition path $\mathcal{T}_{k,n}$ and the best algorithms for each segment from the class of these M algorithms. Here, we have a sequential strategy $\tilde{\boldsymbol{b}}_M[t]$ when applied to $\{\boldsymbol{x}[t]\}_{t\geq 1}$, that achieves, for all n and $\boldsymbol{x}^n$, the wealth

$$\prod_{i=1}^{k+1}\prod_{t=t_{i-1}}^{t_i-1}\hat{\boldsymbol{b}}_{v[i]}^T[t]\boldsymbol{x}[t]=\prod_{i=1}^{k+1}\sup_{r=1,\ldots,M}\prod_{t=t_{i-1}}^{t_i-1}\hat{\boldsymbol{b}}_r^T[t]\boldsymbol{x}[t]$$

with no knowledge of $k,\mathcal{T}_{k,n}$, or n, for any transition path $\mathcal{T}_{k,n}$, where $\hat{\boldsymbol{b}}_{v[i]}[t]$ is the best algorithm for the ith segment such that $\hat{\boldsymbol{b}}_{v[i]}[t]=\hat{\boldsymbol{b}}_r[t]$ if

$$\prod_{t=t_{i-1}}^{t_i-1}\hat{\boldsymbol{b}}_r^T[t]\boldsymbol{x}[t]\geq\prod_{t=t_{i-1}}^{t_i-1}\hat{\boldsymbol{b}}_j^T[t]\boldsymbol{x}[t] \tag{6}$$

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

Theorem 2: We have a sequential strategy $\tilde{\boldsymbol{b}}_M[t]$ such that for any sequence of price relative vectors $\{\boldsymbol{x}[t]\}_{t\geq 1},\boldsymbol{x}[t]\in\boldsymbol{R}_+^m$ (and some components of $\boldsymbol{x}[t]$ can be zero), the performance of $\tilde{\boldsymbol{b}}_M[t]$ on $\boldsymbol{x}^n$ for all n, defined as the wealth ratio shown in the equation at the bottom of the page, for all n and k, satisfies

$$\frac{\ln R_M[n]}{n}\leq(k+1)\frac{\ln M}{n}+(k+\epsilon)\frac{\ln n}{n}+\frac{1}{n}\left(\log(1+\epsilon)+k\log\frac{1}{\epsilon}\right) \tag{7}$$

and

$$\frac{\ln R_M[n]}{n}\leq(k+1)\frac{\ln M}{n}+\frac{3k+1}{2}\frac{\ln n}{n}+O\left(\frac{k}{n}\right) \tag{8}$$

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*. The introduced $\tilde{\boldsymbol{b}}_M[t]$ does not depend on $\mathcal{T}_{k,n},k$ or n and have computational complexity $O(t)$ per investment period.

Theorem 2 states that, for all n, the regret of the universal sequential portfolio $\tilde{\boldsymbol{b}}_M[t]$ is within $O((k+1)\ln n)$ of an algorithm that can select the best algorithm for each segment from the class of M algorithms, with k transitions and the location of these transitions, i.e., $\mathcal{T}_{k,n}$, (tuned to the underlying sequence), uniformly, for every sequence of price relatives $\boldsymbol{x}^n$.

C. Competition Against the Best Side-Information

Let $\{\boldsymbol{x}[t]\}_{t\geq 1}$ represent a sequence of price relatives and $\{\hat{\boldsymbol{b}}_r[t]\}_{t\geq 1},r=1,\ldots,M$, represent portfolios working in parallel on $\{\boldsymbol{x}[t]\}_{t\geq 1}$. We use a side-information sequence $\{y[t]\}_{t\geq 1}$, where $y[t]\in\{1,\ldots,S\}$ for all t, to label the sequence of price relative vectors. Naturally, $S\leq M$. A competing algorithm with side-information sequence $\{y[t]\}_{t\geq 1}$ assigns, $\hat{\boldsymbol{b}}[t]=\hat{\boldsymbol{b}}_{y[t]}[t]$, for each trading day t. Previously, this framework is investigated in [17] and [19] for a restricted case when the side information or the mechanism generating the side information for labeling the data were known by the universal algorithm. In [17], the side-information sequence is used both by the competition class and the universal algorithm to partition the sequence of price relative vectors. However, for our framework, the side information is unknown to the universal algorithm. We compete against an algorithm that can observe the whole sequence of price relative vectors and in hindsight choose the best side-information sequence to label the data. For this setting, we have the following result.

Corollary 3: We have a sequential strategy $\tilde{\boldsymbol{b}}_y[t]$ such that for any sequence of price relative vectors $\{\boldsymbol{x}[t]\}_{t\geq 1},\boldsymbol{x}[t]\in\boldsymbol{R}_+^m$ (and some components of $\boldsymbol{x}[t]$ can be zero), the performance of $\tilde{\boldsymbol{b}}_y[t]$ on $\boldsymbol{x}^n$ for all n, defined as the following wealth ratio:

$$R_y[n]=\frac{\sup_{y^n}\prod_{t=1}^n\hat{\boldsymbol{b}}_{y[t]}^T[t]\boldsymbol{x}[t]}{\prod_{t=1}^n\tilde{\boldsymbol{b}}_y^T[t]\boldsymbol{x}[t]} \tag{9}$$

satisfies

$$\frac{\ln R_y[n]}{n}\leq(k+1)\frac{\ln M}{n}+(k+\epsilon)\frac{\ln n}{n}+\frac{1}{n}\left(\log(1+\epsilon)+k\log\frac{1}{\epsilon}\right) \tag{10}$$

and

$$\frac{\ln R_y[n]}{n}\leq(k+1)\frac{\ln M}{n}+\frac{3k+1}{2}\frac{\ln n}{n}+O\left(\frac{k}{n}\right) \tag{11}$$

where $y[t]\in\{1,\ldots,S\}$ for any y^n with k transitions among labels $\{1,\ldots,S\}$, such that $\tilde{\boldsymbol{b}}_y[t]$ does not depend on y^n,k,S, or n.

Corollary 3 states that the regret of universal sequential portfolio $\tilde{\boldsymbol{b}}_y[t]$ is within $O((k+1)\ln n)$ of the best batch portfolio with the best side-information sequence that has k transitions among labels $\{1,\ldots,S\}$ on $t=1,\ldots,n$ (that is tuned to the underlying sequence), uniformly, for every sequence of price relative vectors $\{\boldsymbol{x}[t]\}_{t\geq 1}$.

$$R_M[n]=\frac{\sup_{t_1<\cdots<t_k\in\{2,\ldots,n\}}\prod_{i=1}^{k+1}\sup_{r\in\{1,\ldots,M\}}\prod_{t=t_{i-1}}^{t_i-1}\hat{\boldsymbol{b}}_r^T[t]\boldsymbol{x}[t]}{\prod_{t=1}^n\tilde{\boldsymbol{b}}_M^T[t]\boldsymbol{x}[t]}$$

IV. Construction of the Universal Portfolios

Proof of Theorem 1: The proof of Theorem 1 follows along similar lines to the proof of Theorem 2 from [33], where the authors use ideas from universal lossless source coding to combine an exponential number of switching predictors. Here, we present the general method and mainly focus on the differences.

Let $\{\boldsymbol{x}[t]\}_{t\geq 1}$ represent a sequence of price relative vectors. Our goal is to find a strongly sequential portfolio $\tilde{\boldsymbol{b}}_u[t]$ such that the logarithm of the wealth achieved by $\{\boldsymbol{b}_u[t]\}_{t\geq 1}$ over $\boldsymbol{x}^n$ for all n, i.e., $\ln\prod_{t=1}^{n}\tilde{\boldsymbol{b}}_u^T[t]\boldsymbol{x}[t]$, is asymptotically as large as

$$\sup_{\substack{\boldsymbol{b}_1,\ldots,\boldsymbol{b}_{k+1}\in R_+^m\\ t_1,\ldots,t_k\in\{2,\ldots,n\}}} \ln\prod_{i=1}^{k+1}\prod_{l=t_{i-1}}^{t_i-1}\boldsymbol{b}_i^T\boldsymbol{x}[l]+O((k+1)\ln n)$$

for any k and for any possible deterministic $\{\boldsymbol{x}[t]\}_{t\geq 1}$. In doing this, $\tilde{\boldsymbol{b}}_u[t]$ can only use $\boldsymbol{x}[1],\ldots,\boldsymbol{x}[t-1]$ to invest at time t, nothing else from the future.

For all n, we can construct 2^{n-1} different transition paths, $\mathcal{T}_{k,n}, k=0,\ldots,n-1$, each representing a partition of $\boldsymbol{x}^n$. For a transition path $\mathcal{T}_{k,n}$ representing $(t_1,\ldots,t_k)$ with k transitions, we can assign constant vector of portfolios $\boldsymbol{B}_k=[\boldsymbol{b}_1,\ldots,\boldsymbol{b}_{k+1}]$, where each $\boldsymbol{b}_i$ represents a CRP vector for the ith region. The wealth that will be achieved by this pairing on $\boldsymbol{x}^n$ is given by $W(\boldsymbol{x}^n\,|\,\boldsymbol{B}_k,\mathcal{T}_{k,n})\triangleq\prod_{i=1}^{k+1}\prod_{t=t_{i-1}}^{t_i-1}\boldsymbol{b}_i^T\boldsymbol{x}[t]$. If $\boldsymbol{x}^n$ were known, one could first optimize CRPs in each region $W(\boldsymbol{x}^n\,|\,\boldsymbol{B}_k^*,\mathcal{T}_{k,n})\triangleq\sup_{\boldsymbol{B}_k}W(\boldsymbol{x}^n\,|\,\boldsymbol{B}_k,\mathcal{T}_{k,n})$ holding $\mathcal{T}_{k,n}$ fixed and then select $\mathcal{T}_{k,n}$ among all transition paths with k transitions to yield $W(\boldsymbol{x}^n\,|\,\boldsymbol{B}_k^*,\mathcal{T}_{k,n}^*)\triangleq\sup_{\mathcal{T}_{k,n}}W(\boldsymbol{x}^n\,|\,\boldsymbol{B}_k^*,\mathcal{T}_{k,n})$. Our goal is to find $\tilde{\boldsymbol{b}}_u[t]$, which achieves $W(\boldsymbol{x}^n\,|\,\boldsymbol{B}_k^*,\mathcal{T}_{k,n}^*)$ for all n, for any k and for any $\{\boldsymbol{x}[t]\}_{t\geq 1}$.

Clearly, $\boldsymbol{x}^n$ is not known to a sequential algorithm, hence, this optimization is an impossible task. However, we will demonstrate that $W(\boldsymbol{x}^n\,|\,\boldsymbol{B}_k^*,\mathcal{T}_{k,n}^*)$ can be achieved asymptotically by a sequential portfolio $\{\tilde{\boldsymbol{b}}_u[t]\}_{t\geq 1}$ for all n, uniformly without any knowledge of k or n. This algorithm will be a double mixture algorithm [32] such that we will first find a sequential algorithm that achieves asymptotically $W(\boldsymbol{x}^n\,|\,\boldsymbol{B}_k^*,\mathcal{T}_{k,n})$ given a $\mathcal{T}_{k,n}$ and then combine all such sequential algorithms corresponding to all possible paths $\mathcal{T}_{k,n}$. The final sequential portfolio $\tilde{\boldsymbol{b}}_u[t]$ we seek will be given as the portfolio that achieves this combined wealth, hence $W(\boldsymbol{x}^n\,|\,\boldsymbol{B}_k^*,\mathcal{T}_{k,n}^*)$.

For all n, there exist 2^{n-1} possible partitions of $\boldsymbol{x}^n$. For every possible such $\mathcal{T}_{k,n}=(t_1,\ldots,t_k), k<n$, one can apply the sequential portfolio from [17] on $\boldsymbol{x}^n$ in each segment i independently

$$\tilde{\boldsymbol{b}}_{t_{i-1}}[t]\triangleq\left(\int_{\boldsymbol{B}}\boldsymbol{b}\prod_{k=t_{i-1}}^{t-1}\boldsymbol{b}^T\boldsymbol{x}[k]\boldsymbol{\mu}(\boldsymbol{b})\right)\Bigg/\left(\int_{\boldsymbol{B}}\prod_{k=t_{i-1}}^{t-1}\boldsymbol{b}^T\boldsymbol{x}[k]\boldsymbol{\mu}(\boldsymbol{b})\right)\quad(12)$$

where $\boldsymbol{\mu}(\boldsymbol{b})$ is an mth order Dirichlet distribution, $\boldsymbol{\mu}(\boldsymbol{b})=D(1/2,\ldots,1/2)$, $\boldsymbol{B}$ is the simplex, i.e., $\boldsymbol{B}=\{\boldsymbol{b}\in\boldsymbol{R}_+^m:\sum_z\boldsymbol{b}_z=1\}$ and $t_{i-1}\leq t<t_i$ in (12). Due to Theorem 2 of Cover and Ordentlich [17], in each segment this algorithm asymptotically achieves the performance of the best CRP for that region in the following sense [17]:

$$\ln\left(\prod_{t=t_{i-1}}^{t_i-1}\tilde{\boldsymbol{b}}_{t_{i-1}}^T[t-1]\boldsymbol{x}[t]\right)\geq\sup_{\boldsymbol{b}_i}\ln\left(\prod_{t=t_{i-1}}^{t_i-1}\boldsymbol{b}_i^T\boldsymbol{x}[t]\right)-\frac{(m-1)}{2}\ln(1+(t_i-t_{i-1}))-\ln 2\quad(13)$$

where here $t_{i-1}\leq t<t_i$. Applying this result for all segments

$$\tilde{W}(\boldsymbol{x}^n\,|\,\mathcal{T}_{k,n})\triangleq\prod_{i=1}^{k+1}\prod_{t=t_{i-1}}^{t_i-1}\tilde{\boldsymbol{b}}_{t_{i-1}}^T[t]\boldsymbol{x}[t]\quad(14)$$

yields

$$\ln(\tilde{W}(\boldsymbol{x}^n\,|\,\mathcal{T}_{k,n}))\geq\sup_{\boldsymbol{B}_k}\ln\{W(\boldsymbol{x}^n\,|\,\boldsymbol{B}_k,\mathcal{T}_{k,n})\}-\sum_{i=1}^{k+1}\frac{(m-1)}{2}\ln(1+(t_i-t_{i-1}))-(k+1)\ln 2.\quad(15)$$

Hence given $\mathcal{T}_{k,n}$, using $\tilde{\boldsymbol{b}}_{t_{i-1}}[t]$ in each segment defines a sequential portfolio that asymptotically achieves the performance of the algorithm with the best CRP for each segment.

For all $\mathcal{T}_{k,n}$ and k, we can construct a similar sequential algorithm yielding a total of 2^{n-1} such sequential portfolios. Hypothetically, we could have invested a portion of our initial wealth, $P(\mathcal{T}_{k,n})$, to each of the 2^{n-1} sequential algorithms and then collect the combined wealth at the end. The final combined wealth of 2^{n-1} such sequential portfolios that correspond to all possible $\mathcal{T}_{k,n}$ and k, is given by

$$\tilde{W}_u(\boldsymbol{x}^n)\triangleq\sum_{k=0}^{n-1}\sum_{\mathcal{T}_{k,n}}P(\mathcal{T}_{k,n})\tilde{W}(\boldsymbol{x}^n\,|\,\mathcal{T}_{k,n}).\quad(16)$$

We will next show that by a proper selection of $P(\mathcal{T}_{k,n})$, $\tilde{W}_u(\boldsymbol{x}^n)$ can be made asymptotically as large as $W(\boldsymbol{x}^n\,|\,\boldsymbol{B}_k^*,\mathcal{T}_{k,n}^*)$ for all n and any k. The constructed $\{\boldsymbol{b}_u[t]\}_{t\geq 1}$ will not explicitly implement all of these 2^{n-1} algorithms for each n. However, it will obtain a wealth of $\tilde{W}_u(\boldsymbol{x}^t)=\prod_{l=1}^{t}\tilde{\boldsymbol{b}}_u^T[l]\boldsymbol{x}[l]$ for all $t\geq 1$ including $t=n$. Hence, $\{\tilde{\boldsymbol{b}}_u[t]\}_{t\geq 1}$ achieves $W(\boldsymbol{x}^n|B_k^*,\mathcal{T}_{k,n}^*)$ for all n.

Let us show how to select $P(\mathcal{T}_{k,n})$ to make $\tilde{W}_u(\boldsymbol{x}^n)$ as large as $W(\boldsymbol{x}^n\,|\,\boldsymbol{B}_k^*,\mathcal{T}_{k,n}^*)$ for all n. Clearly, the portion of initial wealth invested in each $\mathcal{T}_{k,n}$, i.e., the weight of each path, should satisfy $\sum_{k=0}^{n-1}\sum_{\mathcal{T}_{k,n}}P(\mathcal{T}_{k,n})=1$ and the combined wealth $\tilde{W}_u(\boldsymbol{x}^n)$ is as large as the wealth of any portfolio in the mixture, i.e.,

$$\ln\tilde{W}_u(\boldsymbol{x}^n)\geq\ln P(\mathcal{T}_{k,n})+\ln\tilde{W}(\boldsymbol{x}^n\,|\,\mathcal{T}_{k,n}).\quad(17)$$

Combining (17) and (15) yields

$$\ln(\tilde{W}_u(\boldsymbol{x}^n))\geq\ln(P(\mathcal{T}_{k,n}))+\sup_{\boldsymbol{B}_k}\{W(\boldsymbol{x}^n\,|\,\boldsymbol{B}_k,\mathcal{T}_{k,n})\}-\frac{(m-1)}{2}\sum_{i=1}^{k+1}\ln(1+(t_i-t_{i-1}))-(k+1)\ln 2$$

for any transition path $\mathcal{T}_{k,n}$ including $\mathcal{T}^*_{k,n}$. Obviously, the proportion of the wealth invested on $\mathcal{T}_{k,n}, P(\mathcal{T}_{k,n})$, directly contributes to the wealth $\tilde{W}_u(\boldsymbol{x}^n)$. Hence, it is desirable that the portion of wealth invested on the "best rebalancing path" (i.e., the path with the largest wealth gain, which is not known *a priori*) be as large as possible. In addition, the portion wealth invested on each $\mathcal{T}_{k,n}, P(\mathcal{T}_{k,n})$, should also be sequentially constructed for each new n so that the resulting combined portfolio can be sequentially computable. We will use two such sequential weighting algorithms with linear complexity in n. These weightings are introduced in [32] and [35] for universal lossless source coding to assign probabilities to binary sequences and then later used in prediction context in [33] to construct universal switching linear predictors under square error loss. The first form of initial investment, $P(\mathcal{T}_{k,n})$, is generated using a Krichevsky–Trofimov (KT) weighting. Given a binary sequence of length $a+b$ with a ones and b zeros, the KT weight assigned to this binary sequence is given by $P_{\mathrm{KT}}(a,b) = \int_0^1 (1-\theta)^a \theta^b / [\pi\sqrt{(1-\theta)\theta}] d\theta$. This weighting admits sequential updates, i.e.,

$$P_{\mathrm{KT}}(a+1,b) = \frac{a+1/2}{a+b+1} P_{\mathrm{KT}}(a,b)$$

and

$$P_{\mathrm{KT}}(a,b+1) = \frac{b+1/2}{a+b+1} P_{\mathrm{KT}}(a,b) \tag{18}$$

for all $a \geq 0, b \geq 0$. For a given $\mathcal{T}_{k,n}$, we first construct a binary sequence in which each transition represents a one and lack of a transition represents a zero, forming a binary sequence of length $n-1$. Hence, for $\mathcal{T}_{k,n}$, there are k ones and $n-k-1$ zeros. We next define $P(\mathcal{T}_{k,n}), \mathcal{T}_{k,n} = (t_0, \ldots, t_{k+1})$, using $k+1$ KT weights as follows:

$$P(\mathcal{T}_{k,n}) \triangleq \left(\prod_{i=0}^{k-1} P_{\mathrm{KT}}(t_{i+1} - t_i - 1, 1) \right) P_{\mathrm{KT}}(n - t_k, 0). \tag{19}$$

In (19), we first assign weight to the first transition at time $t = t_1$ as $P_{\mathrm{KT}}(t_1 - t_0 - 1, 1)$, which ends the first segment. We repeat this for all k segments. Since, in the last segment there is no transition, we have $P_{\mathrm{KT}}(n - t_k, 0)$ for the last segment. It can be shown [32] that $\sum_{k=0}^{n-1} \sum_{\mathcal{T}_{k,n}} P(\mathcal{T}_{k,n}) = 1$ and this weight assignment satisfies [32], [36]

$$-\ln P(\mathcal{T}_{k,n}) \leq \frac{3k+1}{2} \ln n + O(k) \tag{20}$$

for any $\mathcal{T}_{k,n}$. By using this bound we obtain

$$\ln \tilde{W}_u(\boldsymbol{x}^n) \geq \ln \tilde{W}(\boldsymbol{x}^n \,|\, \mathcal{T}_{k,n}) - \frac{3k+1}{2} \ln n + O(k). \tag{21}$$

For a tighter upper bound on $P(\mathcal{T}_{k,n})$, we can use the weighting introduced in [35]. In this case, instead of using KT estimates for each segment in (19), we replace each term $P_{\mathrm{KT}}(t_{i+1} - t_i - 1, 1)$ with $[\prod_{t=t_{i-1}}^{t_i - 1} (Z(\infty) - Z(t))/(Z(\infty) - Z(t-1))](Y(t_i - 1))/(Z(\infty) - Z(t_i - 1))$, where $Y(j) \triangleq 1/j^{1+\epsilon}, Z(t) \triangleq \sum_{j=1}^{t} Y(j)$, $Z(\infty) \triangleq \sum_{j=1}^{\infty} Y(j)$ and ϵ is any positive constant. This weighting will yield [35]

$$-\ln P(\mathcal{T}_{k,n}) \leq (k+\epsilon) \ln n + \left(\log(1+\epsilon) + k \log \frac{1}{\epsilon} \right)$$

which is a tighter upper bound than (20). We will implement our algorithms in a generic manner such that either of these two weight assignments can be used in the implementation.

Hence, we now have a sequential strategy which invests the portion $P(\mathcal{T}_{k,n})$ of wealth on each $\mathcal{T}_{k,n}$ and has a combined wealth asymptotically achieving, to the first order in the exponent, the same wealth as that achieved by any rebalancing path $\mathcal{T}_{k,n}$ as shown in (21). In this sense, $\tilde{W}_u(\boldsymbol{x}^n)$ is a "universal" portfolio selection method for the class of all switching CRPs. It still remains to find a sequential algorithm of reasonable complexity whose wealth is as large as $\tilde{W}_u(\boldsymbol{x}^n)$ (the wealth achieved by all sequential algorithms represented in (14) weighted by the corresponding $P(\mathcal{T}_{k,n}), k = 1, \ldots, n$).

We are now ready to find the actual universal portfolio strategy. We observe that, by definition

$$\tilde{W}_u(\boldsymbol{x}^n) = \prod_{t=1}^{n} \frac{\tilde{W}_u(\boldsymbol{x}^t)}{\tilde{W}_u(\boldsymbol{x}^{t-1})}$$

for all n. Thus, if we can find a strongly sequential portfolio $\tilde{\boldsymbol{b}}_u[t]$ which satisfies for all t

$$\frac{\tilde{W}_u(\boldsymbol{x}^t)}{\tilde{W}_u(\boldsymbol{x}^{t-1})} = \tilde{\boldsymbol{b}}_u^T[t] \boldsymbol{x}[t] \tag{22}$$

then this portfolio will achieve $\tilde{W}_u(\boldsymbol{x}^t)$, for all $t \geq 1$ (including $t = n$), i.e.,

$$\tilde{W}_u(\boldsymbol{x}^t) = \prod_{r=1}^{t} \frac{\tilde{W}_u(\boldsymbol{x}^r)}{\tilde{W}_u(\boldsymbol{x}^{r-1})} = \prod_{r=1}^{t} \tilde{\boldsymbol{b}}_u^T[r] \boldsymbol{x}[r]$$

for all $t \geq 1$. We recall that at each time $t, \tilde{W}_u(\boldsymbol{x}^t)$ asymptotically achieves the optimal wealth, $W(\boldsymbol{x}^t | B_q^*, T_q^*)$, for any q, on $\boldsymbol{x}^t$ by definition. To find $\tilde{\boldsymbol{b}}_u[t]$, we look at $(\tilde{W}_u(\boldsymbol{x}^t))/(\tilde{\boldsymbol{W}}_u(\boldsymbol{x}^{t-1}))$ closely. By (16), we have

$$\frac{\tilde{W}_u(\boldsymbol{x}^t)}{\tilde{W}_u(\boldsymbol{x}^{t-1})} \frac{\sum_{k=0}^{t-1} \sum_{\mathcal{T}_{k,t}} P(\mathcal{T}_{k,t}) \tilde{W}(\boldsymbol{x}^t \,|\, \mathcal{T}_{k,t})}{\sum_{j=0}^{t-2} \sum_{\mathcal{T}_{j,t-1}} P(\mathcal{T}_{j,t-1}) \tilde{W}(\boldsymbol{x}^{t-1} \,|\, \mathcal{T}_{j,t-1})}$$

where $\tilde{W}(\boldsymbol{x}^t \,|\, \mathcal{T}_{k,t})$ is the wealth achieved by the sequential portfolio corresponding to $\mathcal{T}_{k,t}$ on $\boldsymbol{x}^t$. Suppose, with an abuse of notation, we represent $\mathcal{T}_{k,t} = (t_0 = 1, t_1, \ldots, t_k, t_{k+1} = t+1)$, then $\tilde{W}(\boldsymbol{x}^t \,|\, \mathcal{T}_{k,t}) = \prod_{i=1}^{k+1} \prod_{l=t_{i-1}}^{t_i - 1} \tilde{\boldsymbol{b}}_{t_{i-1}}^T[l] \boldsymbol{x}[l]$, where $\tilde{\boldsymbol{b}}_{t_{i-1}}[l]$ is Cover's portfolio that started running at time t_{i-1} and investing at time l, trained on $(\boldsymbol{x}[t_{i-1}], \ldots, \boldsymbol{x}[l-1])$. Thus, we have

$$\frac{\tilde{W}_u(\boldsymbol{x}^t)}{\tilde{W}_u(\boldsymbol{x}^{t-1})} = \frac{\sum_{k=0}^{t-1} \sum_{\mathcal{T}_{k,t}} P(\mathcal{T}_{k,t}) \prod_{i=1}^{k+1} \prod_{l=t_{i-1}}^{t_i - 1} \tilde{\boldsymbol{b}}_{t_{i-1}}^T[l] \boldsymbol{x}[l]}{\tilde{W}_u(\boldsymbol{x}^{t-1})}. \tag{23}$$

Conceptually, to calculate the numerator of (23), we need to run 2^{t-1} algorithms in parallel. Each of these 2^{t-1} sequential algorithms will produce a portfolio vector to invest at time t. Since,

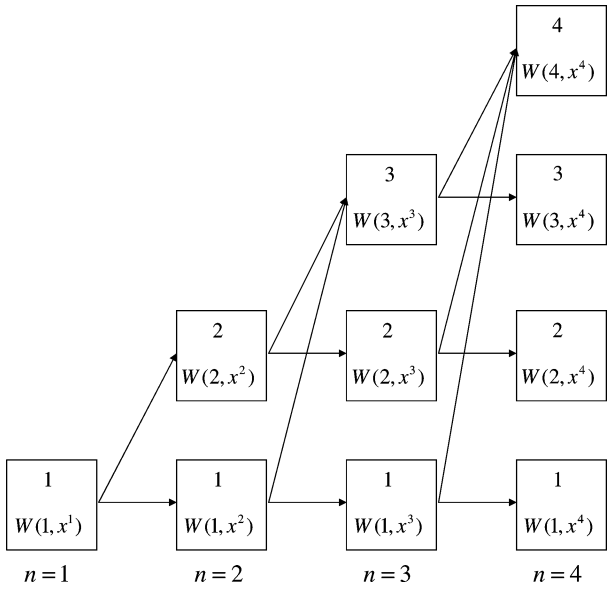

Fig. 1. Transition diagram for $n = 4$. Each box represents a state, where each number in the top of the box is the time of the last transition.

for all these algorithms, on the last segment, the last investment is on $\boldsymbol{x}[t]$, for the numerator of (23), we have

$$\prod_{i=1}^{k+1}\prod_{l=t_{i-1}}^{t_i-1}\tilde{\boldsymbol{b}}_{t_{i-1}}^{T}[l]\boldsymbol{x}[l] = \left(\prod_{i=1}^{k}\prod_{l=t_{i-1}}^{t_i-1}\tilde{\boldsymbol{b}}_{t_{i-1}}^{T}[l]\boldsymbol{x}[l]\right)\left(\prod_{z=t_k}^{t_{k+1}-2}\tilde{\boldsymbol{b}}_{t_k}^{T}[z]\boldsymbol{x}[z]\right)\tilde{\boldsymbol{b}}_{t_k}^{T}[t]\boldsymbol{x}[t] \tag{24}$$

where $t_{k+1} = t + 1$ (and $t_{k+1} - 2 = t - 1$). Hence, (23) becomes (25), shown at the bottom of the page. Thus, we can choose our sequential portfolio as (26), shown at the bottom of the page. Equation (26) suggests the following algorithmic simplification. To invest at time t (before observing $\boldsymbol{x}[t]$), we collect portfolio vectors from all 2^{t-1} sequential algorithms. We then combine all these portfolios with the appropriate weights in the numerator of (26), where the weights are the combination of the performance of the sequential algorithm corresponding to $\mathcal{T}_{k,t}$ on $\boldsymbol{x}^{t-1}$ and $P(\mathcal{T}_{k,t})$. This gives the numerator in (26). We then normalize this combination by $\tilde{W}_u(\boldsymbol{x}^{t-1})$, since we desire the final portfolio $\tilde{\boldsymbol{b}}_u[t]$ to sum up to 1, (this normalization to 1 will be proved later in the derivations). Clearly, $\tilde{\boldsymbol{b}}_u[t]$ has no access to n, k or switching times; however, by construction, when applied to $\{\boldsymbol{x}[t]\}_{t\geq 1}$, it achieves $\tilde{W}_u(\boldsymbol{x}^t)$ for all $t \geq 1$. We have that $\tilde{\boldsymbol{b}}_u[t]$ is calculated by just observing $\boldsymbol{x}[1], \ldots, \boldsymbol{x}[t-1]$. Although we know all of the terms to calculate $\tilde{\boldsymbol{b}}_u[t]$, in this form, $\tilde{\boldsymbol{b}}_u[t]$ needs to combine all 2^{t-1} portfolios with the corresponding weights for all t. Our goal is to calculate $\tilde{\boldsymbol{b}}_u[t]$ efficiently with a computational complexity linear in t.

We first show that $\tilde{W}_u(\boldsymbol{x}^{t-1})$ (and $\tilde{W}_u(\boldsymbol{x}^t)$) can be calculated efficiently, by grouping certain sequential algorithms and updating their wealths together. At time $t-1$, we divide the set of all 2^{t-2} sequential strategies into $t-1$ classes (or sets) based on their last transition times, and label them with state variables $s_{t-1} = 1, \ldots, t-1$, as in Fig. 1. At time $t-1$, there exist only $t-1$ such classes, each represented as a box on the Fig. 1. We note that all sequential portfolios having the same last transition time will use the same Cover portfolio in their last segment. We then define $W_{t-1}(s_{t-1}, \boldsymbol{x}^{t-1})$ as the combined wealth achieved on $\boldsymbol{x}^{t-1}$ by all sequential strategies that have the last transition time at s_{t-1}, i.e., if $s_{t-1} = s$ then

$$W_{t-1}(s, \boldsymbol{x}^{t-1}) \stackrel{\triangle}{=} \sum_{\mathcal{T}':s_{t-1}=s} P(\mathcal{T}')\tilde{W}(\boldsymbol{x}^{t-1} \mid \mathcal{T}')$$

$$\frac{\tilde{W}_u(\boldsymbol{x}^t)}{\tilde{W}_u(\boldsymbol{x}^{t-1})} = \frac{\sum_{k=0}^{t-1}\sum_{\mathcal{T}_{k,t}} P(\mathcal{T}_{k,t})\left[\left(\prod_{i=1}^{k}\prod_{l=t_{i-1}}^{t_i-1}\tilde{\boldsymbol{b}}_{t_{i-1}}^{T}[l]\boldsymbol{x}[l]\right)\left(\prod_{z=t_k}^{t-1}\tilde{\boldsymbol{b}}_{t_k}^{T}[z]\boldsymbol{x}[z]\right)\tilde{\boldsymbol{b}}_{t_k}^{T}[t]\boldsymbol{x}[t]\right]}{\tilde{W}_u(\boldsymbol{x}^{t-1})}. \tag{25}$$

$$\tilde{\boldsymbol{b}}_u[t] = \frac{\sum_{k=0}^{t-1}\sum_{\mathcal{T}_{k,t}} P(\mathcal{T}_{k,t})\left[\left(\prod_{i=1}^{k}\prod_{l=t_{i-1}}^{t_i-1}\tilde{\boldsymbol{b}}_{t_{i-1}}^{T}[l]\boldsymbol{x}[l]\right)\left(\prod_{z=t_k}^{t-1}\tilde{\boldsymbol{b}}_{t_k}^{T}[z]\boldsymbol{x}[z]\right)\tilde{\boldsymbol{b}}_{t_k}[t]\right]}{\tilde{W}_u(\boldsymbol{x}^{t-1})}. \tag{26}$$

where $(\mathcal{T}' : s_{t-1} = s)$ represents all paths of length $t-1$ with the last transition time at s. At time $t-1$, each box in Fig. 1 is assigned a variable $W_{t-1}(s_{t-1}, \boldsymbol{x}^{t-1})$. Since these sets are disjoint and cover all possible paths, this yields

$$\tilde{W}_u(\boldsymbol{x}^{t-1}) = \sum_{s_{t-1}=1}^{t-1} W_{t-1}(s_{t-1}, \boldsymbol{x}^{t-1}).$$

Similarly, we have for $\tilde{W}_u(\boldsymbol{x}^{t-1})$,

$$\tilde{W}_u(\boldsymbol{x}^{t}) = \sum_{s_t=1}^{t} W_t(s_t, \boldsymbol{x}^{t}).$$

The next step is to show that each term $W_t(s_t, \boldsymbol{x}^t), s_t = 1, \ldots, t$, can be recursively calculated from $W_{t-1}(s_{t-1}, \boldsymbol{x}^{t-1})$, $s_{t-1} = 1, \ldots, t-1$, provided that $P(\mathcal{T}_{k,t})$ permits such a sequential update. While doing this update, we will also a get a recursive calculation for $\tilde{\boldsymbol{b}}_u[t]$. We now derive a recursive update for each $W_t(s_t, \boldsymbol{x}^t)$. In Fig. 1, any directed path represents a transition path, where a horizontal move represents no transition, while an upward move represents a transition to a new segment. We derive the recursive update first for $W_t(s_t, \boldsymbol{x}^t), s_t = 1, \ldots, t-1$ and then for $W_t(s_t, \boldsymbol{x}^t), s_t = t$.

At time $t-1$, all the paths (or the sequential algorithms corresponding to these paths) that were included in state $s_{t-1} = s$ will end up in state $s_t = s$ if no transition happens at time t, i.e., a horizontal move in Fig. 1. Hence, the combined wealth $W_{t-1}(s, \boldsymbol{x}^{t-1}), s = 1, \ldots, t-1$, should be updated as follows to get $W_t(s, \boldsymbol{x}^t)$. We note that all the sequential algorithms in $W_t(s, \boldsymbol{x}^t), s = 1, \ldots, t-1$ should come from the sequential algorithms in $W_{t-1}(s, \boldsymbol{x}^{t-1})$ (with no update at time t). Hence, first all the sequential algorithms represented in $W_{t-1}(s, \boldsymbol{x}^{t-1})$ observe a new sample $\boldsymbol{x}[t]$ at time t. Since $s_{t-1} = s_t = s$, the Cover portfolio used in the last segments starts from the same time instant s for all algorithms in $W_{t-1}(s, \boldsymbol{x}^{t-1})$ and in $W_t(s, \boldsymbol{x}^t)$. This will result an additional gain of $\tilde{b}_s[t]^T \boldsymbol{x}[t]$ in $W_t(s, \boldsymbol{x}^t)$ over $W_{t-1}(s, \boldsymbol{x}^{t-1})$, where $\tilde{b}_s[t]$ is the Cover portfolio started at time s, trained on $(\boldsymbol{x}[s], \ldots, \boldsymbol{x}[t-1])$. However, the sequential algorithms in $W_t(s, \boldsymbol{x}^t)$ have different path weights $P(\mathcal{T}_{k,t})$ than those in $W_{t-1}(s, \boldsymbol{x}^t)$, i.e., $P(\mathcal{T}_{k,t-1})$, since the path lengths are increased by one due to the new sample $\boldsymbol{x}[t]$. Hence, all path weights $P(\mathcal{T}_{k,t-1})$ should be updated to get $P(\mathcal{T}_{k,t-1})$. Since all paths that end up in $W_t(s, \boldsymbol{x}^t)$ should come from the paths in $W_{t-1}(s, \boldsymbol{x}^{t-1})$ with no transition, we need to scale path weights using (18) and (19) to obtain the scaling

$$\begin{aligned} P_{\mathrm{tr}}(s_t = s \,|\, s_{t-1} = s) &\triangleq \frac{P(\mathcal{T}_{(\cdot,t):s_t=s})}{P(\mathcal{T}_{(\cdot,t-1):s_{t-1}=s})} \\ &= \frac{t-1-s+1/2}{t-1-s+1} \end{aligned} \tag{27}$$

since only the last term of (19) should be changed in all paths. Here, $P(\mathcal{T}_{(\cdot,t-1):s_{t-1}=s})$ is the weight of a path of length $t-1$ that ended at $s_{t-1} = s$ and $P(\mathcal{T}_{(\cdot,t):s_t=s})$ is the continuation of that path to length t. Hence, combining these two terms yields, for $s = 1, \ldots, t-1$.

$$W_t(s, \boldsymbol{x}^t) = W_{t-1}(s, \boldsymbol{x}^{t-1}) P_{\mathrm{tr}}(s_t = s \,|\, s_{t-1} = s) \tilde{\boldsymbol{b}}_s^T[t] \boldsymbol{x}[t]. \tag{28}$$

Hence, the recursive update for $W_t(s, \boldsymbol{x}^t)$ for $s = 1, \ldots, t-1$. We also need $W_t(t, \boldsymbol{x}^t)$. For all the sequential algorithms (or paths) that end up in $W_t(t, \boldsymbol{x}^t)$, there should be a new transition at time t. All of these paths (or sequential algorithms) can be generated from the sequential algorithms in $W_{t-1}(s, \boldsymbol{x}^{t-1}), s = 1, \ldots, t-1$ by a new transition at time t. Since, to obtain the wealths of the sequential algorithms in $W_t(t, \boldsymbol{x}^t)$ from the wealths of the sequential algorithms in $W_{t-1}(s, \boldsymbol{x}^{t-1}), s = 1, \ldots, t-1$, we need to first change the path lengths. However, there is now a transition at time t, i.e.,

$$\begin{aligned} P_{\mathrm{tr}}(s_t = t \,|\, s_{t-1} = s) &\triangleq \frac{P(\mathcal{T}_{(\cdot,t):s_t=t})}{P(\mathcal{T}_{(\cdot,t-1):s_{t-1}=s})} \\ &= \frac{1/2}{t-1-s+1} \end{aligned} \tag{29}$$

$s = 1, \ldots, t-1$, i.e., the weights for all the paths $P(\mathcal{T}_{(\cdot,t-1):s_{t-1}=s})$ that end up in $P(\mathcal{T}_{(\cdot,t):s_t=t})$ with a switch are changed. Next, for the sequential algorithms in $W_t(t, \boldsymbol{x}^t)$ coming from $W_{t-1}(s_{t-1}, \boldsymbol{x}^{t-1})$, we observe a new sample $\boldsymbol{x}[t]$. Since, there is a transition at time $s_t = t$, i.e., no data to train on after the switch, all these algorithms use a uniform portfolio in m stocks, where we set $\mathbf{1}/m$ as their portfolio and the $\mathbf{1}$ is vector of m ones. Hence, combining these yields

$$W_t(t, \boldsymbol{x}^t) = \sum_{s=1}^{t-1} W_{t-1}(s, \boldsymbol{x}^{t-1}) P_{\mathrm{tr}}(s_t = t \,|\, s_{t-1} = s) \frac{\mathbf{1}}{m}^T \boldsymbol{x}[t]. \tag{30}$$

Hence, the recursive update for $W_t(s_t, \boldsymbol{x}^t)$, when $s_t = t$.

Combining all updates (28) and (30) on $W_t(s_t, \boldsymbol{x}^t)$ yields

$$\begin{aligned} &\tilde{W}_u(\boldsymbol{x}^t) \\ &\quad = \sum_{s_t=1}^{t} W_t(s_t, \boldsymbol{x}^t) \\ &\quad = \left(\sum_{s=1}^{t-1} W_t(s, \boldsymbol{x}^t) \right) + W_t(t, \boldsymbol{x}^t) \\ &\quad = \left(\sum_{s=1}^{t-1} W_{t-1}(s, \boldsymbol{x}^{t-1}) P_{\mathrm{tr}}(s_t = s \,|\, s_{t-1} = s) \tilde{b}_s^T[t] \boldsymbol{x}[t] \right) \\ &\qquad + \sum_{s=1}^{t-1} W_{t-1}(s, \boldsymbol{x}^{t-1}) P_{\mathrm{tr}}(s_t = t \,|\, s_{t-1} = s) \frac{\mathbf{1}}{m}^T \boldsymbol{x}[t] \\ &\quad = \sum_{s=1}^{t-1} W_{t-1}(s, \boldsymbol{x}^{t-1}) \Big\{ P_{\mathrm{tr}}(s_t = s \,|\, s_{t-1} = s) \tilde{b}_s^T[t] \boldsymbol{x}[t] \\ &\qquad + P_{\mathrm{tr}}(s_t = t \,|\, s_{t-1} = s) \frac{\mathbf{1}}{m}^T \boldsymbol{x}[t] \Big\}. \end{aligned}$$

This yields [see the equation shown at the bottom of the page]. Hence from (22), the portfolio vector we seek is given by

$$\tilde{\boldsymbol{b}}_u[t] = \sum_{s=1}^{t-1} \mu_{t-1}(s) \Big\{ P_{\mathrm{tr}}(s_t = s \,|\, s_{t-1} = s) \tilde{\boldsymbol{b}}_s[t] + P_{\mathrm{tr}}(s_t = t \,|\, s_{t-1} = s) \frac{\mathbf{1}}{m} \Big\} \quad (31)$$

where the weights $\mu_{t-1}(s)$ are defined as

$$\mu_{t-1}(s) \triangleq \frac{W_{t-1}(s, \boldsymbol{x}^{t-1})}{\sum_{q=1}^{t-1} W_{t-1}(q, \boldsymbol{x}^{t-1})}. \quad (32)$$

Clearly, in (31), all the terms, i.e., all of the weights $P_{\mathrm{tr}}(\cdot), \mu_{t-1}(j)$ and all portfolios $\tilde{\boldsymbol{b}}_{s_{t-1}}[t], (\mathbf{1})/(m)$, sums to 1. Hence, $\tilde{\boldsymbol{b}}_u[t]$ adds up to 1 and is a valid portfolio.

To summarize the algorithm: at each time t, after we make the investment, $\tilde{\boldsymbol{b}}_u[t]$, we observe $\boldsymbol{x}[t]$. We then update weights $W(s_{t-1}, \boldsymbol{x}^{t-1})$ to get $W(s_t, \boldsymbol{x}^t)$ using (28) and (30). Then, we will calculate $\tilde{\boldsymbol{b}}_u[t+1]$ using $W(s_t, \boldsymbol{x}^t)$ to invest at time $t+1$. We continue like this for all $t \geq 1$. Hence, we combine effectively an exponential number of algorithms, i.e., 2^{t-1}, at each time t, with complexity linear in t. This algorithm has computational complexity $O(t^{m+1})$ per investment period due to calculation of t Cover's portfolios for each segment for each investment period. Combining 2^{t-1} sequential algorithms to obtain (31) only requires $O(t)$ computation per sample. This completes the proof of Theorem 1. ∎

Proof of Theorem 2: The proof of Theorem 2 parallels that of Theorem 1. Suppose portfolio vectors of M algorithms are represented as $\hat{\boldsymbol{b}}_r[t], r = 1, \ldots, M, t \geq 1$. For all n, 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, wealth of the best competing algorithm is constructed, $W^*(\boldsymbol{x}^n \,|\, \mathcal{T}_{k,n}) \triangleq \prod_{i=1}^{k+1} \prod_{t=t_{i-1}}^{t_i - 1} \hat{\boldsymbol{b}}_{v[i]}^T[t] \boldsymbol{x}[t]$, where $\hat{\boldsymbol{b}}_{v[i]}[t]$ is the best algorithm for each segment as in (6). Maximizing $W^*(\boldsymbol{x}^n \,|\, \mathcal{T}_{k,n})$ over all $\mathcal{T}_{k,n}$ (with k transitions) yields $W^*(\boldsymbol{x}^n \,|\, \mathcal{T}^*_{k,n})$. Here, $W^*(\boldsymbol{x}^n \,|\, \mathcal{T}^*_{k,n})$ corresponds to the best algorithm in the competition class with k transitions. Our goal is to demonstrate a sequential algorithm $\tilde{\boldsymbol{b}}_M[t]$, when applied to $\{\boldsymbol{x}[t]\}_{t \geq 1}$, achieves $W^*(\boldsymbol{x}^n \,|\, \mathcal{T}^*_{k,n})$ for all k and n without *a priori* knowledge of k or n.

Given any $\mathcal{T}_{k,n}$, we will now demonstrate an algorithm which combines the M static algorithms in each segment (independently), and for that segment achieves the performance of the best strategy for that segment, i.e., $\prod_{t=t_{i-1}}^{t_i - 1} \hat{\boldsymbol{b}}_{v[i]}^T[t] \boldsymbol{x}[t]$. This algorithm is in this certain sense a discrete version of Cover's algorithm in (12). The derivation of this algorithm closely follows the derivation of (12) in [17]. We will then apply this algorithm to each segment independently to construct the sequential algorithm for $\mathcal{T}_{k,n}$. Then, as in Theorem 1, we will conceptually combine all these sequential algorithms corresponding to all $\mathcal{T}_{k,n}$ to get the final $\tilde{\boldsymbol{b}}_M[t]$.

Given $\mathcal{T}_{k,n}$, in each segment i, we first define a weighted wealth achieved by the M portfolio strategies as, for $t_{i-1} \leq l < t_i$,

$$\tilde{W}_{i,u}(\boldsymbol{x}^l_{t_{i-1}}) \triangleq \sum_{j=1}^{M} \frac{1}{M} \prod_{z=t_{i-1}}^{l} \hat{\boldsymbol{b}}_j^T[z] \boldsymbol{x}[z] \quad (33)$$

where each algorithm has equal weight $(1/M)$, similar to selecting a uniform distribution on $\boldsymbol{\mu}(\boldsymbol{b})$ in (12). Since, $\ln \tilde{W}_{i,u}(\boldsymbol{x}^l_{t_{i-1}}) \geq \ln(\prod_{z=t_{i-1}}^{l} \hat{\boldsymbol{b}}_j^T[z] \boldsymbol{x}[z]) - \ln(M)$ for any $j = 1, \ldots, M$, we conclude $\ln \tilde{W}_{i,u}(\boldsymbol{x}^l_{t_{i-1}}) \geq \sup_j \ln(\prod_{z=t_{i-1}}^{l} \hat{\boldsymbol{b}}_j^T[z] \boldsymbol{x}[z]) - \ln(M)$. Hence, it is enough to demonstrate a sequential algorithm that achieves the wealth $\tilde{W}_{i,u}(\boldsymbol{x}^l_{t_{i-1}})$. By definition

$$\tilde{W}_{i,u}\left(\boldsymbol{x}^l_{t_{i-1}}\right) = \prod_{z=t_{i-1}}^{l} \frac{\tilde{W}_{i,u}\left(\boldsymbol{x}^z_{t_{i-1}}\right)}{\tilde{W}_{i,u}\left(\boldsymbol{x}^{z-1}_{t_{i-1}}\right)}. \quad (34)$$

If we look at each term closely in (34)

$$\frac{\tilde{W}_{i,u}\left(\boldsymbol{x}^z_{t_{i-1}}\right)}{\tilde{W}_{i,u}\left(\boldsymbol{x}^{z-1}_{t_{i-1}}\right)} = \frac{\sum_{j=1}^{M} \frac{1}{M} \prod_{o=t_{i-1}}^{z} \hat{\boldsymbol{b}}_j^T[o] \boldsymbol{x}[o]}{\tilde{W}_{i,u}\left(\boldsymbol{x}^{z-1}_{t_{i-1}}\right)}$$

we conclude that

$$\begin{aligned} \frac{\tilde{W}_{i,u}\left(\boldsymbol{x}^z_{t_{i-1}}\right)}{\tilde{W}_{i,u}\left(\boldsymbol{x}^{z-1}_{t_{i-1}}\right)} &= \frac{\left(\sum_{j=1}^{M} \frac{1}{M} \left(\prod_{o=t_{i-1}}^{z-1} \hat{\boldsymbol{b}}_j^T[o] \boldsymbol{x}[o]\right) \hat{\boldsymbol{b}}_j^T[z]\right) \boldsymbol{x}[z]}{\tilde{W}_{i,u}\left(\boldsymbol{x}^{z-1}_{t_{i-1}}\right)} \\ &= \left(\sum_{j=1}^{M} \mu_{z-1}\left(j, \boldsymbol{x}^{z-1}_{t_{i-1}}\right) \hat{\boldsymbol{b}}_j^T[z] \right) \boldsymbol{x}[z] \end{aligned} \quad (35)$$

$$\begin{aligned} \frac{\tilde{W}_u(\boldsymbol{x}^t)}{\tilde{W}_u(\boldsymbol{x}^{t-1})} &= \frac{\sum_{s_t=1}^{t} W_t(s_t, \boldsymbol{x}^t)}{\sum_{s_{t-1}=1}^{t-1} W_{t-1}(s_{t-1}, \boldsymbol{x}^{t-1})} \\ &= \frac{\left(\sum_{s=1}^{t-1} W_{t-1}(s, \boldsymbol{x}^{t-1}) \left\{ P_{\mathrm{tr}}(s_t = s \,|\, s_{t-1} = s) \tilde{\boldsymbol{b}}_s^T[t] + P_{\mathrm{tr}}(s_t = t \,|\, s_{t-1} = s) \frac{\mathbf{1}}{m}^T \right\}\right) \boldsymbol{x}[t]}{\sum_{q=1}^{t-1} W_{t-1}(q, \boldsymbol{x}^{t-1})}. \end{aligned}$$

MATLAB-IMPLEMENTATION:

N: length of trading period;
m: number of stocks.
X: matrix of price relatives, i.e., X=[$\boldsymbol{x}[1], \ldots, \boldsymbol{x}[N]$].
mB: matrix of size $mN \times N$ of offline calculated portfolios.
The column mX(j*m+1:(j+1)*m,n) is the corresponding portfolio vector at time n.
vW: the vector of state wealth.

```
vW=1;
bu=[]; % Switching universal portfolio given in (31)
wu=zeros(1,N); % The wealth achieved by switching universal portfolio.
for n=1:N,
    vT=(n-[1:n]'+1/2)./(n-[1:n]'+1);
    tempvW=zeros(1,n);
    tempsum=0;
    tempbu=zeros(m,1);
    for j=1:n,
        tempvW(j) = mB(j*m+1:j*m,n)'*X(:,n+1)*(1-vT(j))*vW(j);
        tempsum = tempsum+((ones(m,1)/m)'*X(:,n+1))*(vT(j)*vW(j));
        tempbu = tempbu+ mB(j*m+1:j*m,n)*(1-vT(j))*vW(j)+(ones(m,1)/m)*vT(j)*vW(j);
    end
    vW = [tempvW tempsum]
    wu(n+1) = sum(vW);
    bu = [bu tempbu/(sum(tempbu))]; % normalize the weight so that bu(:,n) sums up to 1.
end
```

(a)

PSEUDO-CODE

initialize at $t = 1$, $W_1(1, \boldsymbol{x}^1) = 1$ and $\tilde{\boldsymbol{b}}_u[t] = \frac{\mathbf{1}}{m}$
for $t = 1, \ldots, n$ do
  invest using $\tilde{\boldsymbol{b}}_u[t]$
  receive $\boldsymbol{x}[t]$
  for $s = 1, \ldots, t-1$ do
    $W_t(s, \boldsymbol{x}^t) = W_{t-1}(s, \boldsymbol{x}^{t-1}) P_{\mathrm{tr}}(s_t = s \mid s_{t-1} = s) \tilde{b}_s^T[t] \boldsymbol{x}[t]$
    % here, $\tilde{b}_s[t]$ is Cover's portfolio started running at time s.
  endfor
  $W_t(t, \boldsymbol{x}^t) = \sum_{s=1}^{t-1} W_{t-1}(s, \boldsymbol{x}^{t-1}) P_{\mathrm{tr}}(s_t = t \mid s_{t-1} = s) \frac{\mathbf{1}}{m}^T \boldsymbol{x}[t]$.
  for $s = 1, \ldots, t$ do
    calculate $\tilde{b}_s[t+1]$
  endfor
  $\tilde{\boldsymbol{b}}_u[t+1] = \sum_{s=1}^{t} \mu_t(s)$
  $\left\{ P_{\mathrm{tr}}(s_{t+1} = s \mid s_t = s) \tilde{\boldsymbol{b}}_s[t+1] + P_{\mathrm{tr}}(s_{t+1} = t \mid s_t = s) \frac{\mathbf{1}}{m} \right\}$

(b)

Fig. 2. (a) Complete implementation of the universal switching algorithm in Matlab. (b) Pseudocode for the algorithm, where $\tilde{b}_s[t]$ is Cover's portfolio started running at time s. Here, $\tilde{b}_u[t]$ can be replaced by other sequential portfolios such as [19] started at time s.

where $\mu_{z-1}(j, \boldsymbol{x}_{t_{i-1}}^{z-1}) = \prod_{o=t_{i-1}}^{z-1} (1/M) \hat{\boldsymbol{b}}_j^T[o] \boldsymbol{x}[o] / \tilde{W}_{i,u}(\boldsymbol{x}_{t_{i-1}}^{z-1})$ are the weights of each algorithm j. Hence, our algorithm which achieves $\tilde{W}_{i,u}(\boldsymbol{x}_{t_{i-1}}^{l})$ in each segment i is given by

$$\tilde{\boldsymbol{b}}_{\mathrm{mix},t_{i-1}}[l] = \sum_{j=1}^{M} \mu_{l-1}\left(j, \boldsymbol{x}_{t_{i-1}}^{l-1}\right) \hat{\boldsymbol{b}}_j[l] \tag{36}$$

which is the in certain sense a discrete version of (12). Applying this result for all segments

$$\tilde{W}(\boldsymbol{x}^n \mid \mathcal{T}_{k,n}) \triangleq \prod_{i=1}^{k+1} \prod_{t=t_{i-1}}^{t_i-1} \tilde{\boldsymbol{b}}_{\mathrm{mix},t_{i-1}}^T[t] \boldsymbol{x}[t] \tag{37}$$

yields

$$\ln(\tilde{W}(\boldsymbol{x}^n \mid \mathcal{T}_{k,n})) \geq \ln\left(\prod_{i=1}^{k+1} \prod_{t=t_{i-1}}^{t_i-1} \hat{\boldsymbol{b}}_{v[i]}^T[t] \boldsymbol{x}[t] \right) - (k+1) \ln M.$$

For a given $\mathcal{T}_{k,n}$, running an independent (36) for each segment yields a sequential portfolio assignment algorithm, similar to the sequential portfolio assignment algorithm represented in (14). After this point the derivation follows the proof of Theorem 1,

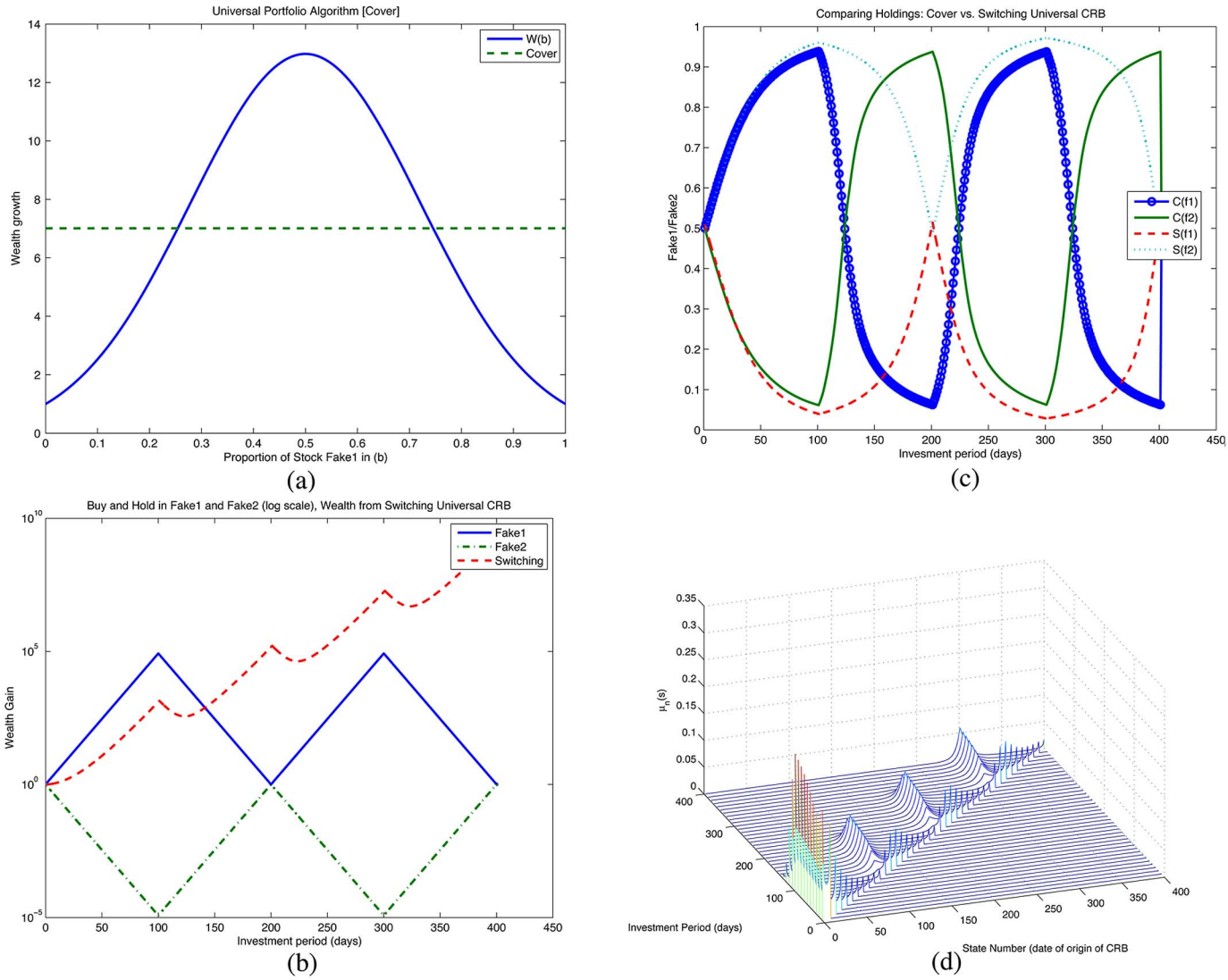

Fig. 3. Performance of switching CRPs for two simulated stocks. (a) Performance of CRPs for different proportions of Fake1 stock. The dashed line is the performance of Cover's universal algorithm which is the weighted average of all CRPs. (b) Wealth gain achieved by buy-and-hold and switching universal algorithm. (c) Portfolio vectors suggested by Cover's algorithm and switching algorithm. C(f1) and S(f1) are the proportions of stock Fake1 in Cover's algorithm and switching universal algorithm respectively. Naturally, at all times, $S(f1) + S(f2) = 1$ and $C(f1) + C(f2) = 1$. (d) Distribution of the weights $\mu_n(k)$ in (32) at days $n = 1, \ldots, 400$.

where we combine sequential algorithms as in (37) with a proper $P(\mathcal{T}_{k,n})$ to define a weighted mixture of the total wealth,

$$\tilde{W}_u(\boldsymbol{x}^n) \triangleq \sum_{k=0}^{n-1} \sum_{\mathcal{T}_{k,n}} P(\mathcal{T}_{k,n}) \tilde{W}(\boldsymbol{x}^n \,|\, \mathcal{T}_{k,n}). \tag{38}$$

Hence, for the construction of the universal algorithm, we need only replace the sequential portfolio vector in (31) with $\tilde{\boldsymbol{b}}_{s_n}[n] = \tilde{\boldsymbol{b}}_{\text{mix},s_n}[n]$, where $\tilde{\boldsymbol{b}}_{\text{mix},s_n}[n]$ is the algorithm in (36) trained on the constituent algorithms from time s_n to $n-1$. i.e.,

$$\tilde{\boldsymbol{b}}_M[n] = \sum_{s=1}^{n-1} \mu_{n-1}(s) \left\{ P_{\text{tr}}(s_n = s \,|\, s_{n-1} = s) \tilde{\boldsymbol{b}}_{\text{mix},s}[n] + P_{\text{tr}}(s_n = n \,|\, s_{n-1} = s) \frac{\mathbf{1}}{m} \right\} \tag{39}$$

where the weights $\mu_{n-1}(s)$ are defined as $\mu_{n-1}(s) \triangleq (W_{n-1}(s, \boldsymbol{x}^{n-1}))/(\sum_{j=1}^{n-1} W_{n-1}(j, \boldsymbol{x}^{[n-1]}))$. This completes the proof of Theorem 2. ■

Proof of Corollary 3: Here, we will show that the universal portfolio selection algorithm introduced in (39) also satisfies the bound in Corollary 3. For any y^n, there corresponds a transition path $\mathcal{T}_{k,n}$. Hence, for any y^n applying the universal algorithm to $\boldsymbol{x}^n$ yields the upper bounds in Theorem 2. Since the bounds in Theorem 2 are with respect to the best algorithm in each segment, i.e., the universal algorithm achieved the performance of the best static algorithm for that segment, they are also correct for the particular static algorithm that is selected by the side-information sequence. Hence,

$$\tilde{\boldsymbol{b}}_y[t] = \tilde{\boldsymbol{b}}_M[t].$$

This completes the proof of Corollary 3. ■

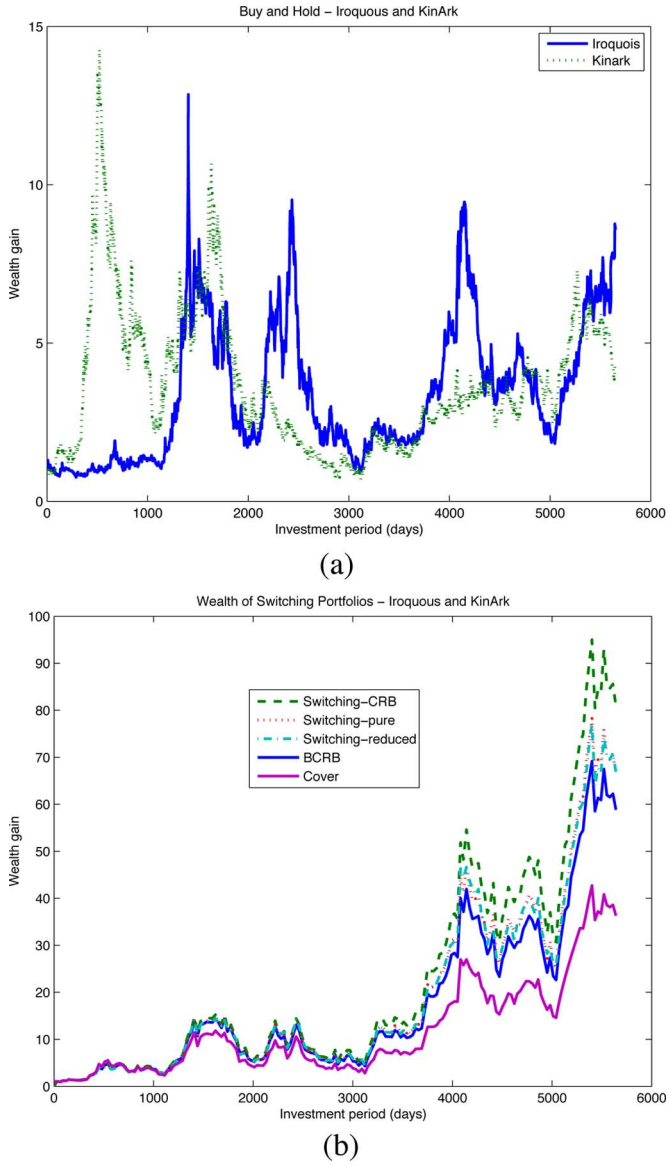

Fig. 4. Performance of various portfolio selection algorithms on Kinark–Iroquois stock pairs. (a) Wealth gain for buy-and-hold strategies on Kinark and Iroquois. (b) Wealth gain for: switching portfolios with finite class of algorithms; switching portfolios with simplified Cover's algorithm; switching portfolios with CRPs; best CRP; Cover's universal portfolio.

A. Algorithmic Description

In this section, we provide a Matlab implementation of the switching CRP. The corresponding implementation uses Willems' weighting [32] for transition probabilities. For other weighting methods only the calculation of the transition vector needs to be changed, i.e., $\boldsymbol{vT}$ in Fig. 2. For N trading days and m stocks, we also construct an offline matrix of portfolio vectors $\boldsymbol{mB}$ to make the implementation generic, where $\boldsymbol{mB}$ is an $mN \times N$ matrix. For example, if one wishes to use Cover's algorithm in (12) at trading day n, the entries $\boldsymbol{mB}((j-1)*m+1:j*m,n)$ (for $j=1,\ldots,n-1$) would be equal to Cover's portfolio vector trained on price relative vectors for trading days j up to $n-1$, i.e., $\boldsymbol{x}[j],\ldots,\boldsymbol{x}[n-1]$. For [19], the same entries would be the portfolio vector of [19] trained for the same trading sequence. This matrix $\boldsymbol{mB}$ can be constructed using any such algorithm from [17]–[19] and [26]. For this particular implementation, we also have at time n: a vector of state wealth, $\boldsymbol{vW} = [W_{n-1}(1,\boldsymbol{x}^{n-1}),\ldots,W_{n-1}(n-1,\boldsymbol{x}^{n-1})]^T$; a vector of state transitions, $\boldsymbol{vT} = [P_{\mathrm{tr}}(s_n = s_{n-1}|s_{n-1} = 1),\ldots,P_{\mathrm{tr}}(s_n = s_{n-1}|s_{n-1} = n-1)]^T$. These vectors are updated at each iteration, and the size of each vector is expanded by one for each new trading day. Apart from the complexity of constructing $\boldsymbol{mB}$, the complexity of the universal algorithm in Fig. 2 is $O(n)$, i.e., the complexity grows with data length. Although we calculate $\boldsymbol{mB}$ offline for illustration, we point out that in the algorithm we use $\boldsymbol{mB}$ sequentially. In Fig. 2, we also provided a pseudocode implementation.

V. Simulations

In this section, we demonstrate the performance of the universal algorithms with several different examples. We first investigate switching CRPs for simulated data of two stocks. In this example, the first stock increases its value by 1.12 during initial 100 days and then decreases its value by $1/1.12$ during the following 100 days. It then switches back and forth for a total of 400 days, i.e., sequence of price relatives for stock Fake1 is

$$[1.12,\ldots,1.12,1/1.12,\ldots,1/1.12,1.12,\ldots,1.12,\\ 1/1.12,\ldots,1/1.12].$$

Price relatives for stock Fake2 is just the opposite, i.e.,

$$[1/1.12,\ldots,1/1.12,1.12,\ldots,1.12,1/1.12,\ldots,\\ 1/1.12,1.12,\ldots,1.12].$$

Clearly, there is no gain for buy-and-hold strategies. In Fig. 3, the straight line displays the performance of CRPs for different proportions of Fake1 to Fake2. Obviously, the maximum wealth is achieved for CRP of $\boldsymbol{b} = [1/2 \quad 1/2]^T$ since the price relatives are symmetric. The dashed line is the performance of Cover's universal algorithm, which is the weighted average of all such CRPs. We observe that due to hedging against all possible sequences of price relatives, the performance of Cover's portfolio is worse than the performance of the optimal $\boldsymbol{b}$. We plot the wealth gain achieved in buy-and-hold strategies as well as the gain achieved by the switching universal algorithm in Fig. 3(b) on a logarithmic scale. Fig. 3(b) shows the exponential gain achieved by the switching universal algorithm. It is clear that at each trend change point, i.e., days 100, 200, 300, and 400, the algorithm quickly adjusts to the new trend. We next provide the portfolio suggested by the switching algorithm in (31) and Cover's portfolio in Fig. 3(c) to illustrate the learning behavior of both algorithms. Due to the averaging performed in Cover's algorithm, this algorithm slowly adjusts to the trend change while the switching algorithm quickly picks up the change and transfers weights from the poorly performing stock to the other accordingly at the switching days 100, 200, 300, and 400. This behavior can be seen in Fig. 3(d), where we plot for each time n, the index of the state on the transition diagram and its associated weight, i.e., $\mu_n(k)$ from (32) for the switching universal algorithm. In this figure, the y axis represents the state number, i.e., date of origin of the CRP used by the state with the largest

TABLE I

LANGUAGE MODEL COMBINATION BETWEEN UNIGRAM, BIAGRAM, AND TRIAGRAM LANGUAGE MODELS USING THE BEST FIXED OFFLINE WEIGHTING, COVER'S WEIGHTING, AND UNIVERSAL ALGORITHM OF THEOREM 1. SCORES SHOWN ARE PERPLEXITIES, SUCH THAT A LOWER SCORE IMPLIES A BETTER MODEL

	unigram & biagram	biagram & triagram	unigram & triagram
Best weighting	27.3231	23.5054	13.9269
Cover's alg.	27.8073	23.9238	14.2004
Switching alg.	27.1082	23.2358	13.8567

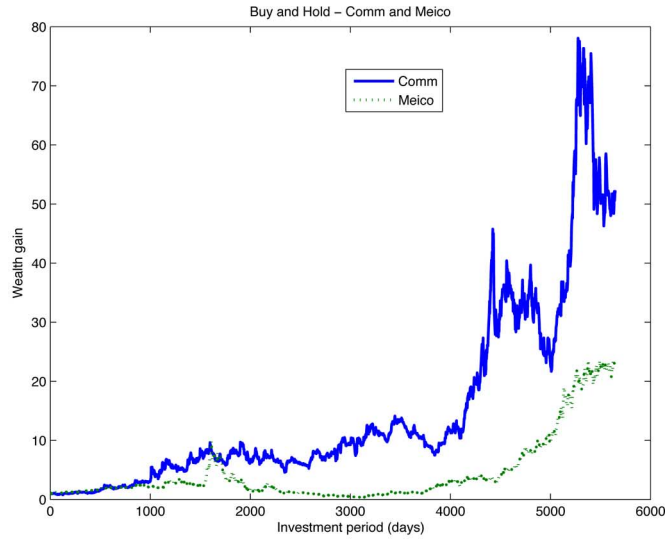

Fig. 5. Performance of various portfolio selection algorithms on Kinark–Iroquois stock pairs. (a) Wealth gain for buy-and-hold strategies on Meico and Commercial Metals. (b) Wealth gain for: switching portfolios with finite class of algorithms; switching portfolios with simplified Cover's algorithm; switching portfolios with CRPs; best CRP; Cover's universal portfolio.

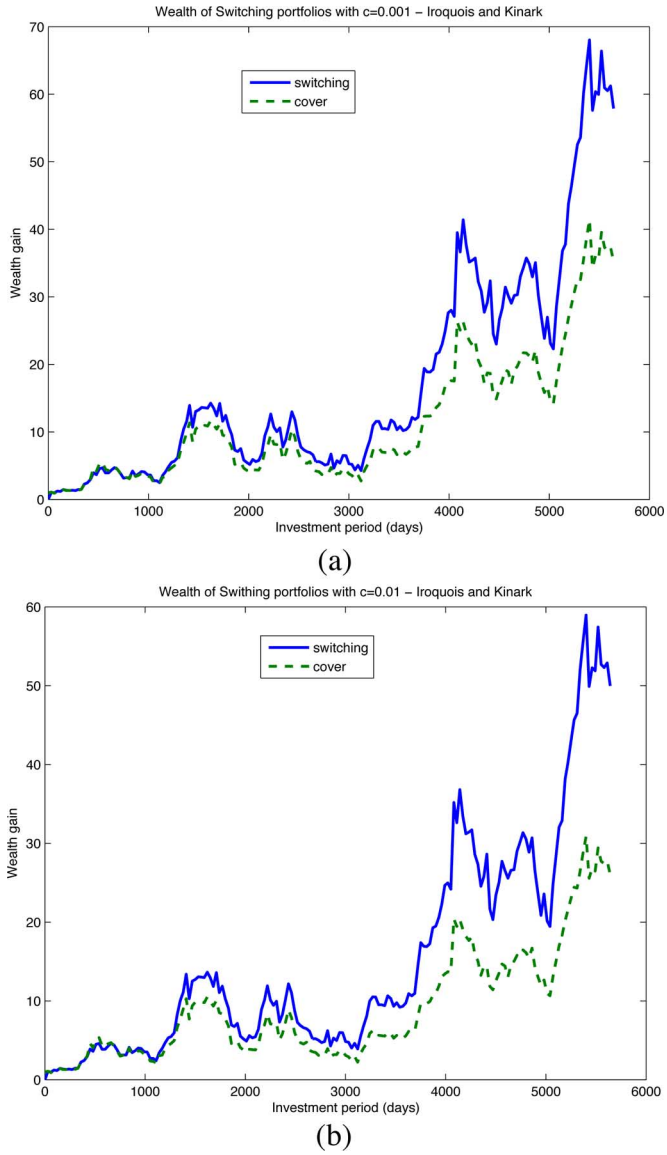

Fig. 6. Performance of switching CRPs and Cover's universal algorithm under varying transaction costs: (a) wealth gain when $c = 0.001$ and (b) wealth gain when $c = 0.01$.

weight. The universal algorithm assigns, and continues to assign, the largest weight $\mu_n(k)$ in (32) to the transition paths that switches at the transition times of the underlying price relative sequence i.e., every 100 days.

As the next example, we apply our algorithms to historical data from [37] from the New York Stock Exchange over a 22-year period. We first apply our algorithms on the Kinark–Iroquois pair as shown in Fig. 4, which are chosen because of their volatility. In Fig. 4(a), we first show the performance of buy-and-hold strategies. We next plot wealth gain of the following: the switching CRP algorithm from Section III-A; the switching finite portfolio algorithm from Section III-B; best CRP that is tuned for the underlying data set; and Cover's universal algorithm, in Fig. 4(b). The switching algorithm with a finite number of algorithms uses only buy-and-hold strategies and as seen from Fig. 4(b), it outperforms other algorithms for this data set. We project that hedging against all CRPs degrades the performance of both Cover's algorithm as well

as the switching CRP algorithm. Hence, in the same figure, we also introduce and plot a simplified (or reduced) switching CRP algorithm. In this algorithm, instead of averaging over all CRPs in (12) for all $\boldsymbol{b} \in R_+^2$, we only average over, three CRPs, $[1/2 \quad 1/2]^T, [1 \quad 0]^T, [0 \quad 1]^T$. The performance of this algorithm improves over the switching CRP selection algorithm. We next show the same set of experiments for another pair of stocks, i.e., Meico and Commercial Metals in Fig. 5. We observe similar behavior to that of the Kinark–Iroquois pair.

We next present results that show the performance of our algorithms with varying amounts of transaction costs. In Fig. 6, we present results on Kinark and Iroquois for a mild transaction cost $c = 0.001$ and a hefty transaction cost $c = 0.01$, where c is the fraction paid in commission for each transaction, i.e., $c = 0.01$ is a 1% commission. Rebalancing is done everyday. We observe that rebalancing weekly or daily gives similar results. In Fig. 6(a), we plot both switching CRPs and Cover's algorithm for $c = 0.001$. The performance is better for small transaction costs, however, in both cases, the switching algorithm outperforms Cover's. Although we observe that transaction costs effect the performance of the switching algorithm due to extensive rebalancing, we can provide similar bounds on the performance of our algorithms with respect to the best switching strategy under a fixed-percent transaction cost model [20].

Finally, we simulate our algorithms for language model combination. Here, we train unigram, biagram, and triagram language models using modified Knesser Ney Smoothing with data collect on the Web. We then score a related test set of 200 words, where each word is scored with respect to the language model to get the corresponding probability of the word. For each pair of language models, to sequentially find the optimal weight combination parameters, we apply first the algorithm from (31) and then Cover's algorithm. The comparison of the final perplexities, calculated from the combined model probabilities, including the optimal (static) combination weight calculated offline "best weighting," Cover's algorithm "Cover's alg." and switching algorithm (31) "switching alg." are presented in Table I. For these simulations, the switching algorithm is able to outperform the other algorithms for all pairs.

VI. Conclusion

In this paper, we considered sequential (online) decisions taken as convex combinations of observations and multiplicatively compounded over time. We focussed on sequential portfolio selection for individual price relative sequences. 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 CRP selection algorithm tuned to the underlying sequence of price relatives. To achieve this, a performance-weighted mixture of an exponential number of sequential portfolios, one for 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 [32]. We then considered the case when the members of the static class include only a finite collection of portfolio selection algorithms.

Acknowledgment

The authors would like to thank E. Ordentlich for providing them with the historical data used in their experiments.

References

[1] L. Rabiner and B.-H. Juang, *Fundamentals of Speech Recognition*. Englewood Cliffs, NJ: Prentice-Hall, 1993.
[2] S. E. Schwarm, I. Bulyko, and M. Ostendorf, "Adaptive language modeling with varied sources to cover new vocabulary items," *IEEE Trans. Speech Audio Process.*, vol. 12, no. 3, pp. 334–342, May 2004.
[3] E. Charniak, *Statistical Language Learning*. Cambridge, MA: MIT Press, 1993.
[4] A. Kalai, S. Chen, A. Blum, and R. Rosenfeld, "On-line algorithms for combining language models," in *Proc. Int. Conf. Acoustics, Speech, Signal Processing (ICASSP)*, 1999, pp. 1–13.
[5] L. Rabiner and R. W. Schafer, *Digital Processing of Speech Signals*. Englewood Cliffs, NJ: Prentice-Hall, 1978.
[6] S. S. Kozat, K. Visweswariah, and R. Gopinath, "Efficient, low latency adaptation for speech recognition," in *Proc. Int. Conf. Acoustics, Speech, Signal Processing (ICASSP)*, 2007, pp. 777–780.
[7] H. Glotin, D. Vergyr, C. Neti, G. Potamianos, and J. Luettin, "Weighting schemes for audio–visual fusion in speech recognition," in *Proc. Int. Conf. Acoustics, Speech, Signal Processing (ICASSP)*, 2001, pp. 173–176.
[8] A. Morris, A. Hagen, H. Glotin, and H. Bourlard, "Multistream adaptive evidence combination to noise robust ASR," in *Proc. Int. Conf. Acoustics, Speech, Signal Processing (ICASSP)*, 2001, pp. 173–176.
[9] S. Haykin, Z. Chen, and S. Becker, "Stochastic correlative learning algorithms," *IEEE Trans. Signal Process.*, vol. 52, no. 8, pp. 2200–2209, Aug. 2004.
[10] E. Rafajlowicz, "Testing (non-)existence of input-output relationships by estimating fractal dimensions," *IEEE Trans. Signal Process.*, vol. 52, no. 11, pp. 3151–3159, Nov. 2004.
[11] E. Osuna, R. Freund, and F. Girosi, "An improved training algorithm for support vector machines," in *Proc. IEEE Workshop Neural Networks Signal Processing*, 1997, pp. 276–285.
[12] K. Nakamura and T. Tsuchiya, "A recursive recomputation approach for smoothing in nonlinear state-space modeling: An attempt for reducing space complexity," *IEEE Trans. Signal Process.*, vol. 55, no. 11, pp. 5167–5178, Nov. 2007.
[13] D. Jeng, J. Watada, and T. Watanabe, "Solving quadratic programming problems via meta-controlled Boltzmann machine," in *Proc. IEEE Int. Workshop Intelligent Signal Processing*, 2005, pp. 310–315.
[14] S. S. Kozat and A. C. Singer, "Universal constant rebalanced portfolios with switching," in *Proc. Int. Conf. Acoustics, Speech, Signal Processing (ICASSP)*, 2007, pp. 1129–1132.
[15] M. Gupta and A. Gilbert, "Nonlinear vector multiresolution analysis," in *Proc. 34th Asilomar Conf. Signals, Systems, Computers*, 2000, pp. 1077–1081.
[16] A. Mojsilovic, "Perceptual indexing of multivariate time series," in *Proc. Int. Conf. Acoustics, Speech, Signal Processing (ICASSP)*, 2007, pp. 533–536.
[17] T. Cover and E. Ordentlich, "Universal portfolios with side-information," *IEEE Trans. Inf. Theory*, vol. 42, no. 2, pp. 348–363, 1996.
[18] V. Vovk and C. Watkins, "Universal portfolio selection," in *Proc. Conf. Learning Theory (COLT)*, 1998, pp. 12–23.
[19] D. P. Helmbold, R. E. Schapire, Y. Singer, and M. K. Warmuth, "On-line portfolio selection using multiplicative updates," *Math. Finance*, vol. 8, no. 4, pp. 325–347, 1998.
[20] S. S. Kozat and A. C. Singer, "Universal switching portfolios under transaction costs," in *Proc. Int. Conf. Acoustics, Speech, Signal Processing (ICASSP)*, 2008, pp. 5404–5407.
[21] N. Cesa-Bianchi and G. Lugosi, *Prediction, Learning and Games*. Cambridge, U.K.: Cambridge Univ. Press, 2006.
[22] G. Stoltz and G. Lugosi, "Internal regret in online portfolio selection," *Mach. Learn.*, vol. 59, pp. 125–159, 2005.
[23] A. Kalai and S. Vempala, "Efficient algorithms for universal portfolios," in *Proc. IEEE Symp. Foundations Computer Science*, 2000, pp. 486–491.

[24] M. Herbster and M. K. Warmuth, "Tracking the best expert," in *Proc. Int. Conf. Machine Learning*, 1995, pp. 286–294.
[25] V. Vovk, "A game of prediction with expert advice," *J. Comput. Syst. Sci.*, vol. 56, pp. 153–173, 1998.
[26] A. Borodin, R. El-Yaniv, and V. Govan, "Can we learn to beat the best stock," *J. Artific. Intell. Res.*, vol. 21, pp. 579–594, 2004.
[27] A. Blum and A. Kalai, "Universal portfolios with and without transaction costs," *Mach. Learn.*, vol. 30, no. 1, pp. 23–30, 1998.
[28] T. Cover and E. Ordentlich, "Universal portfolios with short sales and margin," in *Proc. ISIT*, 1998, p. 174.
[29] T. Cover, "Minimax regret portfolios for restricted stock sequences," in *Proc. ISIT*, 2004, p. 141.
[30] Y. Singer, "Swithcing portfolios," in *Proc. Conf. Uncertainty in AI*, 1998, pp. 1498–1519.
[31] V. Vovk, "Derandomizing stochastic prediction strategies," *Mach. Learn.*, vol. 35, pp. 247–282, 1999.
[32] F. M. J. Willems, "Coding for a binary independent piecewise-identically-distributed source," *IEEE Trans. Inf. Theory*, vol. 42, pp. 2210–2217, 1996.
[33] 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.
[34] S. S. Kozat and A. C. Singer, "Switching strategies for sequential decision problems with multiplicative loss with application to portfolios," *IEEE Trans. Signal Process.*, submitted for publication.
[35] 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, 1999.
[36] R. E. Krichevsky and V. K. Trofimov, "The performance of universal encoding," *IEEE Trans. Inf. Theory*, vol. 27, no. 2, pp. 190–207, May 1981.
[37] T. Cover, "Universal portfolios," *Math. Finance*, vol. 1, no. 1, pp. 1–29, 1991.

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 University of Illinois, Urbana–Champaign, in 2001 and 2004, respectively.

In 2007, he was a full-time Research Staff Member at IBM Research, Pervasive Speech Technologies Group, in the T. J. Watson Research Center, Yorktown, NY. He is currently an Assistant Professor in the Electrical Engineering Department of Koc University, Istanbul, Turkey. His research interests include machine learning, signal processing, speech processing, telecommunications, and statistical signal processing.

Dr. Kozat received a full-time scholarship from Bilkent University during his undergraduate studies. He has served as a reviewer for IEEE TRANSACTIONS ON SIGNAL PROCESSING, the IEEE TRANSACTIONS ON IMAGE PROCESSING, the IEEE TRANSACTIONS ON INFORMATION THEORY as well as several conferences, such as IEEE International Conference on Signal Processing, the IEEE International Conference on Image Processing, the International Conference on Spoken Language Processing, 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.

Since 1998, he has been on the faculty of the Department of Electrical and Computer Engineering at the University of Illinois at Urbana-Champaign, where he is currently a Professor in the Electrical and Computer Engineering (ECE) Department, a Research Professor in the Coordinated Science Laboratory, and a Willett Faculty Scholar. During the academic year 1996, he was a Postdoctoral Research Affiliate in the Research Laboratory of Electronics at MIT. From 1996 to 1998, he was a Research Scientist at Sanders, a Lockheed Martin Company, Manchester, NH, where he designed algorithms, architectures, and systems for a variety of DOD applications. In 2005, he was appointed as the Director of the Technology Entrepreneur Center (TEC) in the College of Engineering. He also co-founded Intersymbol Communications, Inc., a venture-funded fabless semiconductor IC company, based in Champaign, IL. A developer of signal processing enhanced chips for ultra-high-speed optical communications systems, Intersymbol was acquired by Finisar Corporation in 2007. His research interests include statistical signal processing, communication systems, and machine learning.

Dr. Singer was a Hughes Aircraft Masters Fellow, and was the recipient of the Harold L. Hazen Memorial Award for excellence in teaching in 1991. In 2000, he received the National Science Foundation CAREER Award, in 2001 he received the Xerox Faculty Research Award, and in 2002 was named a Willett Faculty Scholar. He serves as an Associate Editor for the IEEE TRANSACTIONS ON SIGNAL PROCESSING and is a member of the MIT Educational Council, and of Eta Kappa Nu and Tau Beta Pi.