Improved Regret Bounds for Tracking Experts with Memory

Anonymous Author(s)
Affiliation
Address
email

Abstract

We address the problem of sequential *prediction with expert advice* in a non-stationary environment with long-term memory guarantees in the sense of Bousquet and Warmuth [4]. We give a linear-time algorithm that improves on the best known regret bounds [26]. This algorithm incorporates a relative entropy projection step. This projection is advantageous over previous weight-sharing approaches in that weight updates may come with implicit costs as in for example portfolio optimization. We give an algorithm to compute this projection step in linear time, which may be of independent interest.

1 Introduction

We consider the classic problem of online prediction with expert advice [32] in a non-stationary environment. In this model `nature` sequentially generates outcomes which `learner` attempts to predict. Before making each prediction, `learner` listens to a set of n experts who each make their own predictions. `Learner` bases its prediction on the advice of the experts. After the prediction is made and the true outcome is revealed by `nature`, the accuracies of `learner`'s prediction and the expert predictions are measured by a loss function. `Learner` receives information on all expert losses on each trial. We make no distributional assumptions about the outcomes generated, indeed `nature` may be assumed to be adversarial. The goal of `learner` is to predict well relative to a predetermined comparison class of predictors, in this case the set of experts themselves. Unlike the standard regret model, where `learner`'s performance is compared to the single best predictor in hindsight, our aim is for `learner` to predict well relative to a sequence of comparison predictors. That is, "switches" occur in the data sequence and different experts are assumed to predict well at different times.

In this work our focus is on the case when this sequence consists of a few unique predictors relative to the number of switches. Thus most switches return to a previously "good" expert, and a `learner` that can exploit this fact by "remembering" the past can adapt more quickly than a `learner` who has no memory and must re-learn the experts after every switch. The problem of switching with memory in online learning is part of a much broader and fundamental problem in machine learning: how a system can adapt to new information yet retain knowledge of the past. This is an area of research in many fields, including for example, catastrophic forgetting in artificial neural networks [10, 33].

Contributions. In this paper we present an $\mathcal{O}(n)$-time per trial projection-based algorithm for which we prove the best known regret bound for tracking experts with memory. Our projection-based algorithm is intimately related to a more traditional "weight-sharing" algorithm, which we show is a new method for *Mixing Past Posteriors* (MPP) [4]. We show that surprisingly this method

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

corresponds to the algorithm with the previous best known regret bound for this problem [26]. We also give an efficient $\mathcal{O}(n)$-time algorithm for computing exact relative entropy projection onto a simplex with non-uniform (lower) box constraints. Finally, we provide a guarantee which favors projection-based updates over weight-sharing updates when updating weights may incur costs.

The paper is organized as follows. We first introduce the model and discuss related work, giving a detailed overview of the previous results on which we improve. In Section 3 we give our main results, a regret bound which holds for two algorithms, and an algorithm to compute relative entropy projection with non-uniform lower box constraints in linear time. In Section 4 we derive a new "geometric-decay" method for MPP, and show the correspondence to the current best known algorithm [26]. We give a few concluding remarks in Section 5. All proofs are contained in the appendices.

1.1 Preliminaries

We first introduce notation. Let $\Delta_n := \{\boldsymbol{u} \in [0,1]^n : \|\boldsymbol{u}\|_1 = 1\}$ be the $(n-1)$-dimensional probability simplex. Let $\Delta_n^\alpha := \{\boldsymbol{u} \in [0,\alpha]^n : \|\boldsymbol{u}\|_1 = \alpha\}$ be a scaled simplex. Let $\mathbf{1}$ denote the vector $(1,\ldots,1)$ and $\mathbf{0}$ denote the vector $(0,\ldots,0)$. Let $\boldsymbol{e}_i$ denote the ith standard basis vector. We define $D(\boldsymbol{u},\boldsymbol{w}) := \sum_{i=1}^n u_i \log \frac{u_i}{w_i}$ to be the relative entropy between $\boldsymbol{u}$ and $\boldsymbol{w}$. We denote component-wise multiplication as $\boldsymbol{u} \odot \boldsymbol{w} := (u_1 w_1, \ldots, u_n w_n)$. For $p \in [0,1]$ we define $\mathcal{H}(p) := -p \ln p - (1-p)\ln(1-p)$ to be the binary entropy of p, using the convention that $0 \ln 0 = 0$. We define ri S to be the relative interior of the set S. For any positive integer n we define $[n] := \{1,\ldots,n\}$. We overload notation such that $[\texttt{pred}]$ is equal to 1 if the predicate `pred` is true and 0 otherwise. For two vectors $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ we say $\boldsymbol{\alpha} \preceq \boldsymbol{\beta}$ iff $\alpha_i \leq \beta_i$ for all $i = 1,\ldots,n$.

2 Background

In sequential prediction with expert advice `nature` generates elements from an outcome space, $\mathcal{Y}$ while the predictions of `learner` and the experts are elements from a prediction space, $\mathcal{D}$ (e.g., we may have $\mathcal{Y} = \{0,1\}$ and $\mathcal{D} = [0,1]$). Given a non-negative loss function $\ell : \mathcal{D} \times \mathcal{Y} \to [0,\infty)$, learning proceeds in trials. On each trial $t = 1,\ldots,T$: 1) `learner` receives the expert predictions $\boldsymbol{x}^t \in \mathcal{D}^n$, 2) `learner` makes a prediction $\hat{y}^t \in \mathcal{D}$, 3) `nature` reveals the true label $y^t \in \mathcal{Y}$, and 4) `learner` suffers loss $\ell^t := \ell(\hat{y}^t, y^t)$ and expert i suffers loss $\ell_i^t := \ell(x_i^t, y^t)$ for $i = 1,\ldots,n$. Common to the algorithms we consider in this paper is a weight vector, $\boldsymbol{w}^t \in \Delta_n$, where w_i^t can be interpreted as the algorithm's confidence in expert i on trial t. `Learner` uses a prediction function $\texttt{pred} : \Delta_n \times \mathcal{D}^n \to \mathcal{D}$ to generate its prediction $\hat{y}^t = \texttt{pred}(\boldsymbol{w}^t, \boldsymbol{x}^t)$ on trial t. A classic example is to predict with the weighted average of the expert predictions, that is, $\texttt{pred}(\boldsymbol{w}^t, \boldsymbol{x}^t) = \boldsymbol{w}^t \cdot \boldsymbol{x}^t$, although for some loss functions improved bounds are obtained with different prediction functions (see e.g., [40]). In this paper we assume (c,η)-realizability of ℓ and `pred` [4, 17, 38]. That is, there exists constants $c, \eta > 0$ such that for all $\boldsymbol{w} \in \Delta_n$, $\boldsymbol{x} \in \mathcal{D}^n$, and $y \in \mathcal{Y}$, $\ell(\texttt{pred}(\boldsymbol{w},\boldsymbol{x}), y) \leq -c \ln \sum_{i=1}^n v_i e^{-\eta \ell(x_i, y)}$. This includes η-exp-concave losses when $\texttt{pred}(\boldsymbol{w}^t, \boldsymbol{x}^t) = \boldsymbol{w}^t \cdot \boldsymbol{x}^t$ and $c = \frac{1}{\eta}$. For simplicity we present regret bound guarantees that assume $(c, \frac{1}{c})$-realizability, that is $c\eta = 1$. This includes the log loss with $c = 1$, and the square loss with $c = \frac{1}{2}$. The absolute loss is *not* (c,η)-realizable. Generalizing our bounds for general bounded, convex losses in the sense of online convex optimization [42] and the Hedge setting [12] is straightforward. For any comparison sequence of experts $i_1,\ldots,i_T \in [n]$ the regret of `learner` with respect to this sequence is defined as

$$\mathcal{R}(i_{1:T}) = \sum_{t=1}^T \ell^t - \sum_{t=1}^T \ell_{i_t}^t \,.$$

We consider and derive algorithms which belong to the family of "exponential weights" (EW) algorithms (see e.g., [40, 24, 32]). After receiving the expert losses the EW algorithm applies the following incremental loss update to the expert weights,

$$\dot{w}_i^t = \frac{w_i^t e^{-\eta \ell_i^t}}{\sum_{j=1}^n w_j^t e^{-\eta \ell_j^t}} \,. \tag{1}$$

Static setting. In the static setting `learner` competes against a single expert (i.e., $i_1 = \ldots = i_T$). For the static setting the EW algorithm sets $\boldsymbol{w}^{t+1} = \dot{\boldsymbol{w}}^t$ for the next trial, and for $(c, \frac{1}{c})$-realizable losses and prediction functions achieves a static regret bound of $\mathcal{R}(i_{1:T}) \leq c \ln n$.

Switching. In the switching (without memory) setting `learner` competes against a sequence of experts $i_1, \ldots, i_T$ with $k := \sum_{t=1}^{T-1} [i_t \neq i_{t+1}]$ switches. The well-known Fixed-Share algorithm [22] solves the switching problem with the update

$$\boldsymbol{w}^{t+1} = (1-\alpha)\dot{\boldsymbol{w}}^t + \alpha \frac{\mathbf{1}}{n}\,, \tag{2}$$

by forcing each expert to "share" a fraction of its weight *uniformly* with all experts.[1] The update is parameterized by a "switching" parameter, $\alpha \in [0, 1]$, and the regret with respect to the best sequence of experts with k switches is

$$\mathcal{R}(i_{1:T}) \leq c\left((k+1)\ln n + (T-1)\mathcal{H}\left(\frac{k}{T-1}\right)\right) \leq c\left((k+1)\ln n + k \ln \frac{T-1}{k} + k\right). \tag{3}$$

Switching with memory. Freund [11] gave an open problem to improve on the regret bound (3) when the comparison sequence of experts is comprised of a small pool of size $m := |\cup_{t=1}^{T} \{i_t\}| \ll k$. Using counting arguments Freund gave an exponential-time algorithm with the information-theoretic ideal regret bound of $\mathcal{R}(i_{1:T}) \leq c \ln\left(\binom{n}{m}\binom{T-1}{k} m(m-1)^k\right)$, which is upper-bounded by

$$c\left(m \ln n + k \ln \frac{T-1}{k} + (k-m+1)\ln m + k + m\right). \tag{4}$$

The first efficient algorithm solving Freund's problem was presented in the seminal paper [4]. This work introduced the notion of a *mixing scheme*, which is a distribution $\boldsymbol{\gamma}^{t+1}$ with support $\{0, \ldots, t\}$. Given $\boldsymbol{\gamma}^{t+1}$, the algorithm's update on each trial is the *mixture* over all past weight vectors,

$$\boldsymbol{w}^{t+1} = \sum_{q=0}^{t} \gamma_q^{t+1} \dot{\boldsymbol{w}}^q\,, \tag{5}$$

where $\dot{\boldsymbol{w}}^0 := \frac{1}{n}\mathbf{1}$, and $\gamma_0^1 := 1$. Intuitively, by mixing all "past posteriors" (MPP) the weights of previously well-performing experts can be prevented from vanishing and recover quickly. An efficient mixing scheme requiring $\mathcal{O}(n)$-time per trial is the "*uniform*" mixing scheme given by $\gamma_t^{t+1} = 1-\alpha$ and $\gamma_q^{t+1} = \frac{\alpha}{t}$ for $0 \leq q < t$. A better regret bound was proved with a "*decaying*" mixing scheme, given by

$$\gamma_q^{t+1} = \begin{cases} 1-\alpha & q = t \\ \alpha \frac{1}{(t-q)^\gamma} \frac{1}{Z_t} & 0 \leq q < t\,, \end{cases} \tag{6}$$

where $Z_t = \sum_{q=0}^{t-1} \frac{1}{(t-q)^\gamma}$ is a normalizing factor, and $\gamma \geq 0$. With a tuning of $\alpha = \frac{k}{T-1}$ and $\gamma = 1$ this mixing scheme achieves a regret bound of[2]

$$\mathcal{R}(i_{1:T}) \leq c\left(m \ln n + 2k \ln \frac{T-1}{k} + k \ln(m-1) + k + k \ln\ln(eT)\right). \tag{7}$$

It appeared that to achieve the best regret bounds, the mixing scheme needed to decay towards the past. Unfortunately, computing (6) exactly requires the storage of all past weights, at a cost of $\mathcal{O}(nt)$-time and space per trial. Observe that these schemes set $\gamma_t^{t+1} = 1-\alpha$, where typically α is small, since intuitively switches are assumed to happen infrequently. All updates using such schemes are of the form

$$\boldsymbol{w}^{t+1} = (1-\alpha)\dot{\boldsymbol{w}}^t + \alpha \tilde{\boldsymbol{v}}^t\,, \tag{8}$$

which we will call the *generalized share update* (see [7]). Fixed-Share is a special case when $\tilde{\boldsymbol{v}}^t = \frac{\mathbf{1}}{n}$ for all t. This generalized share update features heavily in this paper.

[1] Technically in the original Fixed-Share update each expert shares weight to all *other* experts, i.e., $w_i^{t+1} = (1-\alpha)\dot{w}_i^t + \frac{\alpha}{n-1}\sum_{j \neq i} \dot{w}_j^t$. The two updates achieve essentially the same regret bound and are equivalent up to a scaling of α.

[2] (7) is a simplified upper bound of the bound given in [4, Corollary 9], using $\ln(1+x) \leq x$.

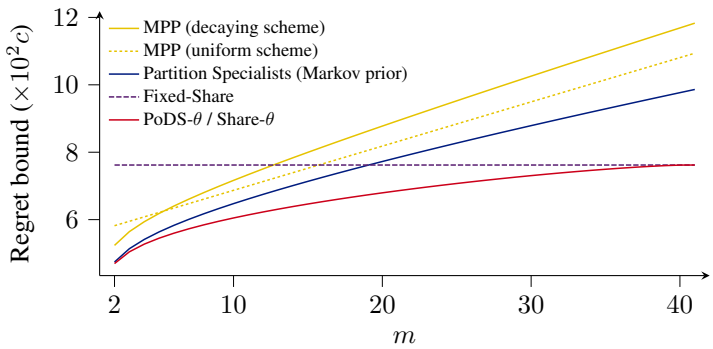

Figure 1: A comparison of the regret bounds discussed in this paper for $m \in [2, k+1]$ with $n = 500000$, $k = 40$, and $T = 4000$. Previous "memory" bounds (blue & yellow) are much worse than Fixed-Share for larger values of m while our bound (red) improves on Fixed-Share for all $m \in [2, k]$.

For a decade it remained an open problem to give the MPP update a Bayesian interpretation. This was finally solved in [26] with the use of *partition specialists*. Here on each trial t, a specialist (first introduced in [13]) is either *awake* and predicts in accordance with a prescribed base expert, or is *asleep* and abstains from predicting. For n base experts and finite time horizon T there are $n2^T$ partition specialists. For Freund's problem an assembly of m partition specialists can predict exactly as the comparison sequence of experts. The Bayesian interpretation of the MPP update given in [26, Theorem 2] was simple: to define a mixing scheme $\boldsymbol{\gamma}^{t+1}$ was to induce a prior over this set of partition specialists. The authors of [26] proposed a simple Markov chain prior over the set of partition specialists, giving an efficient $\mathcal{O}(n)$-time per trial algorithm with the regret bound

$$\mathcal{R}(i_{1:T}) \leq c\left[m \ln \frac{n}{m} + m\mathcal{H}\left(\frac{1}{m}\right) + (T-1)\mathcal{H}\left(\frac{k}{T-1}\right) + (m-1)(T-1)\mathcal{H}\left(\frac{k}{(m-1)(T-1)}\right)\right] \quad (9)$$

$$\leq c\left(m \ln n + 2k \ln \frac{T-1}{k} + (k-m+1)\ln m + 2(k+1)\right), \quad (10)$$

which is currently the best known regret bound for Freund's problem. It is not known which MPP mixing scheme corresponds to this Markov prior. In this work we improve on the bound (9) for tracking experts with memory (Theorem 3), and also show that this Markov prior on partition specialists corresponds to a geometrically-decaying mixing scheme for MPP (Proposition 5).

Adaptive online learning algorithms with memory have been shown to have better empirical performance than those without memory [14], and to be effective in real-world applications such as intrusion detection systems [35]. While considerable research has been done on switching with memory in online learning (see e.g., [4, 7, 19, 20, 26, 41]), there remain several open problems. Firstly, there remains a gap between the best known regret bound for an efficient algorithm and the information-theoretic ideal bound (4). Present in both bounds (7) and (10) is the factor of 2 in the second term, which does not appear in (4). In [26] this was interpreted as the cost of co-ordination between specialists, essentially one "pays" twice per switch as one specialist falls asleep and another awakens. In this paper we make progress in closing this gap by avoiding such additional costs the first time each expert is learned by the algorithm. That is, we pay to *remember* but not to *learn*.

Secondly, unless n is very large the current best known bound (9) beats Fixed-Share's bound (3) only when $m \ll k$, but suffers when m is even a moderate fraction of k. A natural question is can we improve on Fixed-Share when we relax the assumption that $m \ll k$, and only a few members of a sequence of experts need remembering (consider for instance, $m > k/2$)? In this paper we prove a regret bound that is not only tighter than (9) for all m, but under mild assumptions on n improves on Fixed-Share for all $m \leq k$. See Figure 1 where we show this behavior for several existing regret bounds and our regret bound.

Our regret bound will hold for two algorithms; one utilizes a weight-sharing update in the sense of (8), and the other utilizes a projection update. Why should we consider projections? Consider for example

a large model consisting of many weights, and to update these weights costs time and/or money. Alternatively consider the application of regret-bounded adaptive algorithms in online portfolio selection (see e.g., [37, 30]). Here each "expert" corresponds to a single asset, and the weight vector $\boldsymbol{w}^t$ corresponds to a portfolio. If ℓ_i^t is the negative log return of stock i after trading period t, then the loss function $\ell^t := -\ln \sum_{i=1}^n w_i^t e^{-\ell_i^t}$ is the negative log return of the portfolio. This loss is $(1,1)$-realizable by definition (although there is no prediction function [1]). The algorithm's update corresponds to actively re-balancing the portfolio after each trading period, but the investor may incur transaction costs proportional to the amount bought or sold (see e.g., [2, 30]). Online portfolio selection with transaction costs is an active area of research [9, 28, 30, 31]. In Section 3.2 we motivate the use of projection updates over weight-sharing with a guarantee in terms of such costs.

2.1 Related work

Switching (without memory) in online learning was first introduced in [32], and extended with the Fixed-Share algorithm [22]. An extensive literature has built on these works, including but not limited to [1, 4, 7, 8, 15, 16, 20, 21, 23, 26, 27, 34, 36, 41]. Relevant to this work are the results for switching with memory [4, 7, 20, 26, 27, 41]. The first was the seminal work of [4]. The best known result is given in [26], which we improve on. In [41] a reduction of switching with memory to switching without memory is given, although with a slightly worse regret bound than [4]. Related to the experts model is the *bandits* setting, which was addressed in the memory setting in [41]. In [7] a unified analysis of both Fixed-Share and MPP was given in the context of online convex optimization. They observed the generalized share update (8) and slightly improved the bounds of [4]. Adaptive regret [1, 8, 18, 32] has been used to prove regret bounds for switching but unfortunately does not generalize to the memory setting. This paper primarily builds on the work of [4] with a new geometrically-decaying mixing scheme, and on [23] with a new relative entropy projection algorithm.

3 Projection onto dynamic sets

In this section we give a relative entropy projection-based algorithm for tracking experts with memory. Given a non-empty set $\mathcal{C} \subseteq \Delta_n$ and a point $\boldsymbol{w} \in \text{ri}\, \Delta_n$ we define

$$\mathcal{P}(\boldsymbol{w}; \mathcal{C}) := \operatorname*{arg\,min}_{\boldsymbol{u} \in \mathcal{C}} D(\boldsymbol{u}, \boldsymbol{w})$$

to be the projection with respect to the relative entropy of $\boldsymbol{w}$ onto $\mathcal{C}$ [6]. Such projections were first introduced for switching (without memory) in online learning in [23], in which after every trial the weight vector $\dot{\boldsymbol{w}}^t$ is projected onto $\mathcal{C} = [\frac{\alpha}{n}, 1]^n \cap \Delta_n$, that is, the simplex with uniform box constraints. For prediction with expert advice this projection algorithm has the regret bound (3) (see [7]). Indeed, we will refer to $\boldsymbol{w}^{t+1} = \mathcal{P}(\dot{\boldsymbol{w}}^t; [\frac{\alpha}{n}, 1]^n \cap \Delta_n)$ as the "projection analogue" of (2).

Given $\boldsymbol{\beta} \in (0,1)^n$ such that $\|\boldsymbol{\beta}\|_1 \leq 1$, let

$$\mathcal{C}(\boldsymbol{\beta}) := \{\boldsymbol{x} \in \Delta_n : x_i \geq \beta_i, i = 1, \ldots, n\}$$

be a subset of the simplex which is convex and non-empty. Given $\boldsymbol{w} \in \text{ri}\, \Delta_n$, intuitively $\mathcal{P}(\boldsymbol{w}; \mathcal{C}(\boldsymbol{\beta}))$ is the projection of $\boldsymbol{w}$ onto the simplex with (non-uniform) lower box constraints $\boldsymbol{\beta}$. Relative entropy projection updates for tracking experts with memory were first suggested in [4, Section 5.2]. The authors observed that for any MPP mixing scheme $\boldsymbol{\gamma}^{t+1}$, the update (5) can be replaced with

$$\boldsymbol{w}^{t+1} = \mathcal{P}(\dot{\boldsymbol{w}}^t; \{\boldsymbol{w} \in \Delta_n : \boldsymbol{w} \succeq \gamma_q^{t+1} \dot{\boldsymbol{w}}^q, q = 0, \ldots, t\}), \tag{11}$$

and achieve the same regret bound. We build on this concept in this paper. Observe that for any choice of $\boldsymbol{\gamma}^{t+1}$ the set $\{\boldsymbol{w} \in \Delta_n : \boldsymbol{w} \succeq \gamma_q^{t+1} \dot{\boldsymbol{w}}^q, q = 0, \ldots, t\}$ corresponds to the set $\mathcal{C}(\boldsymbol{\beta})$ where

$$\beta_i = \max_{0 \leq q \leq t} \gamma_q^{t+1} \dot{w}_i^q \qquad i = 1, \ldots, n\,. \tag{12}$$

In this work we give an algorithm to compute $\mathcal{P}(\boldsymbol{w}; \mathcal{C}(\boldsymbol{\beta}))$ exactly for any $\mathcal{C}(\boldsymbol{\beta})$ in $\mathcal{O}(n)$ time. With this algorithm and the mapping (12), one immediately obtains the projection analogue of MPP for any mixing scheme $\boldsymbol{\gamma}^{t+1}$ at essentially no additional computational cost. We point out however that for arbitrary mixing schemes computing $\boldsymbol{\beta}$ from (12) takes $\mathcal{O}(nt)$-time on trial t, improving only

when some structure of the scheme can be exploited. We therefore propose the following method for tracking experts with memory *efficiently* using projection onto dynamic sets ("PoDS").

Just as (8) generalizes the Fixed-Share update (2), we propose PoDS as the analogous generalization of the update $\boldsymbol{w}^{t+1} = \mathcal{P}(\dot{\boldsymbol{w}}^t; \mathcal{C}(\alpha \frac{\mathbf{1}}{n}))$ (the projection analogue of Fixed-Share). PoDS maintains a vector $\boldsymbol{\beta}^t \in \Delta_n^\alpha$, and on each trial updates the weights by setting $\boldsymbol{w}^{t+1} = \mathcal{P}(\dot{\boldsymbol{w}}^t; \mathcal{C}(\boldsymbol{\beta}^t))$. Intuitively PoDS is the projection analogue of (8) with $\boldsymbol{\beta}^t$ corresponding simply to $\alpha \tilde{\boldsymbol{v}}^t$. In some cases $\boldsymbol{\beta}^t = \alpha \tilde{\boldsymbol{v}}^t$ for all t (e.g., for Fixed-Share), but in general equality may not hold since $\boldsymbol{\beta}^t$ and $\tilde{\boldsymbol{v}}^t$ can be functions of past weights, which may differ for weight-sharing and projection algorithms. Recall that (8) describes all MPP mixing schemes that set $\gamma_t^{t+1} = 1 - \alpha$. PoDS implicitly captures all such mixing schemes. This simple formulation of PoDS allows us to define new updates, which will correspond to new mixing schemes. In Section 3.2 we give a simple update and prove the best known regret bound.

3.1 Computing $\mathcal{P}(\boldsymbol{w}; \mathcal{C}(\boldsymbol{\beta}))$

Before we consider PoDS further, we first discuss the computation of $\mathcal{P}(\boldsymbol{w}; \mathcal{C}(\boldsymbol{\beta}))$. In [23] the authors showed that computing relative entropy projection onto the simplex with *uniform* box constraints is non-trivial, but gave an algorithm to compute it in $\mathcal{O}(n)$ time. We give a generalization of their algorithm to compute $\mathcal{P}(\boldsymbol{w}; \mathcal{C}(\boldsymbol{\beta}))$ exactly for any non-empty set $\mathcal{C}(\boldsymbol{\beta})$ in $\mathcal{O}(n)$ time. As far as we are aware our method to compute exact relative entropy projection onto the simplex with non-uniform (lower) box constraints in linear time is the first, and may be of independent interest (see e.g., [29]).

We first give an intuition into the form of $\mathcal{P}(\boldsymbol{w}; \mathcal{C}(\boldsymbol{\beta}))$, and then describe how Algorithm 3 computes this projection efficiently. Firstly consider the case that $\boldsymbol{w} \in \mathcal{C}(\boldsymbol{\beta})$, then trivially $\mathcal{P}(\boldsymbol{w}; \mathcal{C}(\boldsymbol{\beta})) = \boldsymbol{w}$, due to the non-negativity of $D(\boldsymbol{u}, \boldsymbol{w})$ and the fact that $D(\boldsymbol{u}, \boldsymbol{w}) = 0$ iff $\boldsymbol{u} = \boldsymbol{w}$ (see e.g., [6]). For the case that $\boldsymbol{w} \notin \mathcal{C}(\boldsymbol{\beta})$, this implies that the set $\{i \in [n] : w_i < \beta_i\}$ is non-empty. For each index i in this set the projection of $\boldsymbol{w}$ onto $\mathcal{C}(\boldsymbol{\beta})$ must set the component w_i to its corresponding constraint value β_i. The remaining components are then normalized, such that $\sum_{i=1}^n w_i = 1$. However, doing so may cause one (or more) of these components w_j to drop below its constraint β_j. The projection algorithm therefore finds the set of components Ψ of least cardinality to set to their constraint values such that when the remaining components are normalized, no component lies below its constraint.

Consider the following inefficient approach to finding Ψ. Given $\boldsymbol{w}$ and $\mathcal{C}(\boldsymbol{\beta})$, let $\boldsymbol{r} = \boldsymbol{w} \odot \frac{1}{\boldsymbol{\beta}}$ be a "ratio vector". Then sort $\boldsymbol{r}$ in ascending order, and sort $\boldsymbol{w}$ and $\boldsymbol{\beta}$ according to the ordering of $\boldsymbol{r}$. If $r_1 \geq 1$ then $\Psi = \emptyset$ and we are done ($\Rightarrow \boldsymbol{w} \in \mathcal{C}(\boldsymbol{\beta})$). Otherwise for each $k = 1, \ldots, n$: 1) let the candidate set $\Psi' = [k]$, 2) let $\boldsymbol{w}' = \boldsymbol{w}$ except for each $i \in \Psi'$ set $w_i' = \beta_i$, 3) re-normalize the remaining components of $\boldsymbol{w}'$, and 4) let $\boldsymbol{r}' = \boldsymbol{w}' \odot \frac{1}{\boldsymbol{\beta}}$. The set Ψ is then the candidate set Ψ' of least cardinality such that $\boldsymbol{r}' \succeq \mathbf{1}$. This approach requires sorting $\boldsymbol{r}$ and therefore even an efficient implementation takes $\mathcal{O}(n \log n)$ time. Algorithm 3 finds Ψ without having to sort $\boldsymbol{r}$. It instead specifies Ψ uniquely with a threshold, ϕ, such that $\Psi = \{i : r_i < \phi\}$. Algorithm 3 finds ϕ through repeatedly bisecting the set $\mathcal{W} = [n]$ by finding the median of the set $\{r_i : i \in \mathcal{W}\}$ (which can be done in $\mathcal{O}(|\mathcal{W}|)$ time [3]), and efficiently testing this value as the candidate threshold on each iteration. The smallest valid threshold then specifies the set Ψ. The following theorem states the time complexity of the algorithm and the form of the projection, which we will use in proving our regret bound (the proof is in Appendix A, where we give a more detailed description of the algorithm).

Theorem 1. *For any $\boldsymbol{\beta} \in (0,1)^n$ such that $\|\boldsymbol{\beta}\|_1 \leq 1$, and for any $\boldsymbol{w} \in \text{ri}\, \Delta_n$, let $\boldsymbol{p} = \mathcal{P}(\boldsymbol{w}; \mathcal{C}(\boldsymbol{\beta}))$, where $\mathcal{C}(\boldsymbol{\beta}) = \{\boldsymbol{x} \in \Delta_n : x_i \geq \beta_i, i = 1, \ldots, n\}$. Then $\boldsymbol{p}$ is such that for all $i = 1, \ldots, n$,*

$$p_i = \max \left\{ \beta_i ; \frac{1 - \sum_{j \in \Psi} \beta_j}{1 - \sum_{j \in \Psi} w_j} w_i \right\} , \tag{13}$$

where $\Psi := \{i \in [n] : p_i = \beta_i\}$. Furthermore, Algorithm 3 computes $\boldsymbol{p}$ in $\mathcal{O}(n)$ time.

The following corollary will be used in the proof of our regret bound.

Corollary 2. *Let $0 < \alpha < 1$. Then for any $\boldsymbol{u} \in \Delta_n$, $\boldsymbol{w} \in \text{ri}\, \Delta_n$, and $\boldsymbol{\beta} \in \text{ri}\, \Delta_n^\alpha$, let $\boldsymbol{p} = \mathcal{P}(\boldsymbol{w}; \mathcal{C}(\boldsymbol{\beta}))$. Then,*

$$D(\boldsymbol{u}, \boldsymbol{w}) - D(\boldsymbol{u}, \boldsymbol{p}) \geq \ln(1 - \alpha) . \tag{14}$$

Algorithms 1&2 PoDS-θ / Share-θ

Input: $n > 0$, $\eta = \frac{1}{c} > 0$, $\alpha \in [0,1]$, $\theta \in [0,1]$

▷ PoDS-θ

init: $\boldsymbol{w}^1 \leftarrow \frac{\mathbf{1}}{n}$; $\boldsymbol{\beta}^1 \leftarrow \alpha\frac{\mathbf{1}}{n}$

▷ Share-θ

init: $\boldsymbol{w}^1 \leftarrow \frac{\mathbf{1}}{n}$; $\boldsymbol{v}^1 \leftarrow \frac{\mathbf{1}}{n}$

▷ PoDS-θ & Share-θ

for $t \leftarrow 1$ to T **do**
  receive $\boldsymbol{x}^t \in \mathcal{D}^n$
  predict $\hat{y}^t = \texttt{pred}(\boldsymbol{w}^t, \boldsymbol{x}^t)$
  receive $y^t \in \mathcal{Y}$
  for $i \leftarrow 1$ to n **do**
    $\dot{w}_i^t \leftarrow \frac{w_i^t e^{-\eta \ell_i^t}}{\sum_{j=1}^n w_j^t e^{-\eta \ell_j^t}}$

▷ PoDS-θ

$$\boldsymbol{w}^{t+1} \leftarrow \mathcal{P}(\dot{\boldsymbol{w}}^t; \mathcal{C}(\boldsymbol{\beta}^t)) \tag{16}$$
$$\boldsymbol{\beta}^{t+1} \leftarrow (1-\theta)\boldsymbol{\beta}^t + \theta\alpha\dot{\boldsymbol{w}}^t$$

▷ Share-θ

$$\boldsymbol{w}^{t+1} \leftarrow (1-\alpha)\dot{\boldsymbol{w}}^t + \alpha\boldsymbol{v}^t \tag{17}$$
$$\boldsymbol{v}^{t+1} \leftarrow (1-\theta)\boldsymbol{v}^t + \theta\dot{\boldsymbol{w}}^t \tag{18}$$

Algorithm 3 $\mathcal{P}(\boldsymbol{w}; \mathcal{C}(\boldsymbol{\beta}))$ in $\mathcal{O}(n)$ time

Input: $\boldsymbol{w} \in \text{ri}\,\Delta_n$; $\boldsymbol{\beta} \in (0,1)^n$ s.t. $\|\boldsymbol{\beta}\|_1 \le 1$
Output: $\boldsymbol{w}' = \mathcal{P}(\boldsymbol{w}; \mathcal{C}(\boldsymbol{\beta}))$

init: $\mathcal{W} \leftarrow [n]$; $\boldsymbol{r} \leftarrow \boldsymbol{w} \odot \frac{1}{\boldsymbol{\beta}}$; $S_{\boldsymbol{w}} \leftarrow 0$; $S_{\boldsymbol{\beta}} \leftarrow 0$
while $\mathcal{W} \neq \emptyset$ **do**
  $\phi \leftarrow \texttt{median}(\{r_i : i \in \mathcal{W}\})$
  $\mathcal{L} \leftarrow \{i \in \mathcal{W} : r_i < \phi\}$
  $L_{\boldsymbol{\beta}} \leftarrow \sum_{i\in\mathcal{L}} \beta_i$; $L_{\boldsymbol{w}} \leftarrow \sum_{i\in\mathcal{L}} w_i$
  $\mathcal{M} \leftarrow \{i \in \mathcal{W} : r_i = \phi\}$
  $M_{\boldsymbol{\beta}} \leftarrow \sum_{i\in\mathcal{M}} \beta_i$; $M_{\boldsymbol{w}} \leftarrow \sum_{i\in\mathcal{M}} w_i$
  $\mathcal{H} \leftarrow \{i \in \mathcal{W} : r_i > \phi\}$
  $\lambda \leftarrow \frac{1-S_{\boldsymbol{\beta}}-L_{\boldsymbol{\beta}}}{1-S_{\boldsymbol{w}}-L_{\boldsymbol{w}}}$
  if $\phi\lambda < 1$ **then**
    $S_{\boldsymbol{w}} \leftarrow S_{\boldsymbol{w}} + L_{\boldsymbol{w}} + M_{\boldsymbol{w}}$
    $S_{\boldsymbol{\beta}} \leftarrow S_{\boldsymbol{\beta}} + L_{\boldsymbol{\beta}} + M_{\boldsymbol{\beta}}$
    if $\mathcal{H} = \emptyset$ **then**
      $\phi \leftarrow \texttt{min}(\{r_i : r_i > \phi, i \in [n]\})$
    $\mathcal{W} \leftarrow \mathcal{H}$
  else
    $\mathcal{W} \leftarrow \mathcal{L}$
$\lambda \leftarrow \frac{1-S_{\boldsymbol{\beta}}}{1-S_{\boldsymbol{w}}}$
$\forall i : 1, \ldots, n : w_i' \leftarrow \begin{cases} \beta_i & r_i < \phi \\ \lambda w_i & r_i \ge \phi \end{cases}$

3.2 A simple update rule for PoDS

We now suggest a simple update rule for $\boldsymbol{\beta}^t$ in PoDS for tracking experts with memory. The bound for this algorithm is given in Theorem 3. We first set $\boldsymbol{\beta}^1 = \alpha\frac{\mathbf{1}}{n}$ to be uniform, and with a parameter $0 \le \theta \le 1$ update $\boldsymbol{\beta}^t$ on subsequent trials by setting

$$\boldsymbol{\beta}^{t+1} = (1-\theta)\boldsymbol{\beta}^t + \theta\alpha\dot{\boldsymbol{w}}^t\,. \tag{15}$$

We refer to PoDS with this update as PoDS-θ. Intuitively the constraint vector $\boldsymbol{\beta}^t$ is updated in (15) by mixing in a small amount of the current weight vector, $\dot{\boldsymbol{w}}^t$, scaled such that $\|\boldsymbol{\beta}^{t+1}\|_1 = \alpha$. If expert i predicted well in the past, then its constraint β_i^t will be relatively large, preventing the weight from vanishing even if that expert suffers large losses locally. Using Algorithm 3 in its projection step, PoDS-θ has $\mathcal{O}(n)$ per-trial time complexity.

As discussed, the vector $\boldsymbol{\beta}^t$ of PoDS is conceptually equivalent to the vector $\alpha\tilde{\boldsymbol{v}}^t$ of the generalized share update (8). If PoDS has a simple update rule such as (15) then it is straightforward to recover the weight-sharing equivalent by simply "pretending" equality holds on all trials. We now do this for PoDS-θ. Clearly we have $\tilde{\boldsymbol{v}}^1 = \frac{\mathbf{1}}{n}$, and if $\boldsymbol{\beta}^t = \alpha\tilde{\boldsymbol{v}}^t$ and $\boldsymbol{\beta}^{t+1} = \alpha\tilde{\boldsymbol{v}}^{t+1}$, then $\tilde{\boldsymbol{v}}^{t+1} = \frac{1}{\alpha}\boldsymbol{\beta}^{t+1} = \frac{1}{\alpha}(1-\theta)\boldsymbol{\beta}^t + \theta\dot{\boldsymbol{w}}^t = (1-\theta)\tilde{\boldsymbol{v}}^t + \theta\dot{\boldsymbol{w}}^t$. This then leads to an efficient sharing algorithm, which we call Share-θ. In Section 4 we show this algorithm is in fact a new MPP mixing scheme, which surprisingly corresponds to the previous best known algorithm for this problem. Both PoDS-θ and Share-θ use the same parameters (α and θ), differing only in the final update (see Algorithms 1&2). We now give the regret bound which holds for both algorithms.

Theorem 3. *For any comparison sequence $i_1, \ldots, i_T$ containing k switches and consisting of m unique experts from a set of size n, if $\alpha = \frac{k}{T-1}$ and $\theta = \frac{k-m+1}{(m-1)(T-2)}$, the regret of both PoDS-θ and Share-θ with any prediction function and loss function which are $(c, \frac{1}{c})$-realizable is*

$$\mathcal{R}(i_{1:T}) \le c\left(m\ln n + (T-1)\mathcal{H}\left(\frac{k}{T-1}\right) + (m-1)(T-2)\mathcal{H}\left(\frac{k-m+1}{(m-1)(T-2)}\right)\right). \tag{19}$$

The regret bound (19) is at least $c((m-1)\ln\frac{T-1}{k} - (k-m+1)\ln\frac{k}{k-m+1}))$ tighter than the currently best known bound (9). Thus if $m \ll k$ then the improvement is $\approx cm\ln\frac{T}{k}$, and as $m \to k+1$ then

the improvement is $\approx ck \ln \frac{T}{k}$. Additionally note that if $m = k+1$ (i.e., every switch we track a *new* expert) the optimal tuning of θ is zero, and PoDS-θ reduces to setting $\boldsymbol{\beta}^t = \alpha \frac{\mathbf{1}}{n}$ on every trial. That is, we recover the projection analogue of Fixed-Share. This is also reflected in the regret bound since (19) reduces to (3). Since $x\mathcal{H}(\frac{y}{x}) \leq y \ln(\frac{x}{y}) + y$, the regret bound (19) is upper-bounded by

$$\mathcal{R}(i_{1:T}) \leq c\left[m \ln n + k \ln \frac{T-1}{k} + (k-m+1) \ln \frac{T-2}{k-m+1} + (k-m+1) \ln (m-1) + 2k - m + 1\right].$$

Comparing this to (10), we see that instead of paying $c \ln \frac{T-1}{k}$ *twice* on every switch, we pay $c \ln \frac{T-1}{k}$ once per switch and $c \ln \frac{T-2}{k-m+1}$ for every switch we *remember* an old expert ($k - m + 1$ times). Unlike previous results for tracking experts with memory, PoDS-θ and its regret bound (19) smoothly interpolate between the two switching settings. That is, it is capable of exploiting memory when necessary and on the other hand does not suffer when memory is not necessary (see Figure 1).

Projection vs. sharing in online learning. We now briefly consider the two types of updates discussed in this paper (projection and weight-sharing) when updating weights may incur costs. Recall the motivating example introduced in Section 2 was in online portfolio selection with transaction costs. It is straightforward to show that in this model transaction costs are proportional to the 1-norm of the difference in the weight vectors before and after re-balancing. In Theorem 4 we give a result which in this context guarantees the "cost" of projecting is less than that of weight-sharing.

To compare the update of PoDS and the generalized share update (8), we must consider for a set of weights $\dot{\boldsymbol{w}}^t$, the point $\mathcal{P}(\dot{\boldsymbol{w}}^t; \mathcal{C}(\boldsymbol{\beta}^t))$ and the point $(1-\alpha)\dot{\boldsymbol{w}}^t + \alpha\tilde{\boldsymbol{v}}^t$. However these points depend on $\boldsymbol{\beta}^t$ and $\tilde{\boldsymbol{v}}^t$ respectively, which may themselves be functions of previous weight vectors $\dot{\boldsymbol{w}}^1, \ldots, \dot{\boldsymbol{w}}^{t-1}$, which as discussed are generally not the same for each of the two algorithms. To compare the two updates equally we therefore assume that the current weights are the same (i.e., they must both update the same weights $\dot{\boldsymbol{w}}^t$), and additionally that $\boldsymbol{\beta}^t = \alpha\tilde{\boldsymbol{v}}^t$. The following theorem states that under mild conditions, PoDS is strictly less "expensive" than its weight-sharing counterpart.

Theorem 4. *Let $0 < \alpha < 1$. Then for any $\boldsymbol{v} \in \operatorname{ri} \Delta_n$, let $\boldsymbol{\beta} = \alpha\boldsymbol{v}$, and for any $\boldsymbol{w} \in \operatorname{ri} \Delta_n$, let $\boldsymbol{w}' = (1-\alpha)\boldsymbol{w} + \alpha\boldsymbol{v}$. Then,*

$$\|\mathcal{P}(\boldsymbol{w}; \mathcal{C}(\boldsymbol{\beta})) - \boldsymbol{w}\|_1 < \|\boldsymbol{w}' - \boldsymbol{w}\|_1 .$$

Thus if one has to pay to update weights, projection is the economical choice.

4 A geometrically-decaying mixing scheme for MPP

In this section we look more closely at Share-θ. We show that it is in fact a new type of *decaying* MPP mixing scheme which corresponds to the partition specialist algorithm with Markov prior.

Recall that the previous best known mixing scheme for MPP is the decaying scheme (6). Observe that in (6) the decay (with the "distance" to the current trial t) follows a power-law, and that computing (6) exactly takes $\mathcal{O}(nt)$ time per trial. We now derive an explicit MPP mixing scheme from the updates (17) and (18) of Share-θ. Observe that if we define $\dot{\boldsymbol{w}}^0 := \frac{\mathbf{1}}{n}$, then an iterative expansion of (18) on any trial t gives $\boldsymbol{v}^t = \sum_{q=0}^{t-1} \theta^{[q \neq 0]}(1-\theta)^{t-q-1}\dot{\boldsymbol{w}}^q$, from which (17) implies $\boldsymbol{w}^{t+1} = (1-\alpha)\dot{\boldsymbol{w}}^t + \alpha\boldsymbol{v}^t = \sum_{q=0}^{t} \gamma_q^{t+1}\dot{\boldsymbol{w}}^q$, where

$$\gamma_q^{t+1} = \begin{cases} 1-\alpha & q = t \\ \theta(1-\theta)^{t-q-1}\alpha & 1 \leq q < t \\ (1-\theta)^{t-1}\alpha & q = 0 . \end{cases} \tag{20}$$

Note that (20) is a valid mixing scheme since for all t, $\sum_{q=0}^{t} \gamma_q^{t+1} = 1$. The Share-$\theta$ update is therefore a new kind of decaying mixing scheme. In this new scheme the decay is *geometric*, and can therefore be computed efficiently, requiring only $\mathcal{O}(n)$ time and space per trial as we have shown. Furthermore MPP with this scheme has the improved regret bound (19).

Another interesting difference between the decaying schemes (20) and (6) is that when θ is small then (20) keeps γ_0^{t+1} relatively large initially and slowly decays this value as t increases. Intuitively

by heavily weighting the initial uniform vector $\dot{\boldsymbol{w}}^0$ on each trial early on, the algorithm can "pick up" the weights of new experts easily. Finally as in the case of PoDS-θ, if $m = k + 1$, then with the optimal tuning of $\theta = 0$, this update reduces to the Fixed-Share update (2).

Revisiting partition specialists. We now turn our attention to the previous best known result for tracking experts with memory (the partition specialists algorithm with a Markov prior [26]).

For sleep/wake patterns $(\chi_1 \dots \chi_T)$ the Markov prior is a Markov chain on states $\{w, s\}$, defined by the initial distribution $\boldsymbol{\pi} = (\pi_w, \pi_s)$ and transition probabilities $P_{ij} := P(\chi_{t+1} = j | \chi_t = i)$ for $i, j \in \{w, s\}$. The algorithm with these inputs efficiently collapses one weight per specialist down to two weights per expert. These two weight vectors, which we denote $\boldsymbol{a}_t$ and $\boldsymbol{s}_t$, represent the total weight of all awake and sleeping specialists associated with each expert, respectively. Note that the vectors $\boldsymbol{a}_t$ and $\boldsymbol{s}_t$ are not in the simplex, but rather the vector $(\boldsymbol{a}_t, \boldsymbol{s}_t) \in \Delta_{2n}$ and the "awake vector" $\boldsymbol{a}_t$ gets normalized upon prediction. The weights are initialized by setting $\boldsymbol{a}_1 = \pi_w \frac{\mathbf{1}}{n}$, and $\boldsymbol{s}_1 = \pi_s \frac{\mathbf{1}}{n}$. The update[3] of these weights after receiving the true label y^t is given by $a_i^{t+1} = P_{ww} \frac{a_i^t e^{-\eta \ell_i^t} (\sum_{j=1}^n a_j^t)}{\sum_{j=1}^n a_j^t e^{-\eta \ell_j^t}} + P_{sw} s_i^t$, and $s_i^{t+1} = P_{ws} \frac{a_i^t e^{-\eta \ell_i^t} (\sum_{j=1}^n a_j^t)}{\sum_{j=1}^n a_j^t e^{-\eta \ell_j^t}} + P_{ss} s_i^t$ for $i = 1, \dots, n$. Recall that the authors of [26] proved that an MPP mixing scheme implicitly induces a prior over partition specialists. The following states that the Markov prior is induced by (20).

Proposition 5. *Let $0 < \alpha < 1$, and $0 < \theta < 1$. Then the partition specialists algorithm with Markov prior parameterized with $P_{sw} = \theta$, $P_{ws} = \alpha$, $\pi_w = \frac{\theta}{\alpha + \theta}$, and $\pi_s = \frac{\alpha}{\alpha + \theta}$ is equivalent to Share-θ parameterized with α and θ.*

The proof (given in Appendix D) amounts to showing for all t that $\frac{\boldsymbol{a}_t}{\pi_w} = \boldsymbol{w}^t$ and $\frac{\boldsymbol{s}_t}{\pi_s} = \boldsymbol{v}^t$. The Markov prior on partition specialists therefore corresponds to a geometrically-decaying MPP mixing scheme! Note however that we have proved a better regret bound for this algorithm in Theorem 3.

5 Discussion

We gave an efficient projection-based algorithm for tracking experts with memory for which we proved the best known regret bound. We also gave an algorithm to compute relative entropy projection onto the simplex with non-uniform (lower) box constraints exactly in $\mathcal{O}(n)$ time, which may be of independent interest. We showed that the weight-sharing equivalent of our projection-based algorithm is in fact a geometrically-decaying mixing scheme for *Mixing Past Posteriors* [4]. Furthermore we showed that this mixing scheme corresponds exactly to the previous best known result (the partition specialists algorithm with Markov prior [26]), and we therefore improved their bound. We proved a guarantee favoring projection updates over weight-sharing when updating weights may incur costs, such as in portfolio optimization with proportional transaction costs. We are currently applying PoDS-θ to this problem, primarily extending the work of [37] in the sense of incorporating both the assumption of "memory" and transaction costs.

In this work we focused on proving good regret bounds, which naturally required optimally-tuned parameters. A limitation of our work is that in practice the optimal parameters are unknown. This is a common issue in online learning, and one may employ standard techniques to address this such as the "doubling trick", or by using a Bayesian mixture over parameters [39]. For a prominent recent result in this area see [25].

Finally, the work of [26] gave a Bayesian interpretation to MPP, however this is lost when one uses the projection update of PoDS. We ask: Is there also a Bayesian interpretation to these projection-based updates?

Ethical considerations. While the scope of applicability of online learning algorithms is wide, this research in regret-bounded online learning is foundational in nature and we therefore cannot foresee the extent of any societal impacts (positive or negative) this research may have.

[3] In [26] the algorithm is presented in terms of probabilities with the log loss. Here we give the update generalized to (c, η)-realizable losses.

References

[1] D. Adamskiy, W. M. Koolen, A. Chernov, and V. Vovk. A closer look at adaptive regret. *The Journal of Machine Learning Research*, 17(1):706–726, 2016.

[2] A. Blum and A. Kalai. Universal portfolios with and without transaction costs. *Machine Learning*, 35(3):193–205, 1999.

[3] M. Blum, R. W. Floyd, V. R. Pratt, R. L. Rivest, and R. E. Tarjan. Time bounds for selection. *J. Comput. Syst. Sci.*, 7(4):448–461, 1973.

[4] O. Bousquet and M. K. Warmuth. Tracking a small set of experts by mixing past posteriors. *Journal of Machine Learning Research*, 3(Nov):363–396, 2002.

[5] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[6] L. M. Bregman. The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR computational mathematics and mathematical physics*, 7(3):200–217, 1967.

[7] N. Cesa-Bianchi, P. Gaillard, G. Lugosi, and G. Stoltz. Mirror descent meets fixed share (and feels no regret). In *Conference on Neural Information Processing Systems*, volume 2, pages 989–997, 2012.

[8] A. Daniely, A. Gonen, and S. Shalev-Shwartz. Strongly adaptive online learning. In *International Conference on Machine Learning*, pages 1405–1411. PMLR, 2015.

[9] P. Das, N. Johnson, and A. Banerjee. Online lazy updates for portfolio selection with transaction costs. In *AAAI*. Citeseer, 2013.

[10] R. M. French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135, 1999.

[11] Y. Freund. Private Communication, 2000. Also posted on `http://www.learning-theory.org/`.

[12] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

[13] Y. Freund, R. E. Schapire, Y. Singer, and M. K. Warmuth. Using and combining predictors that specialize. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 334–343, 1997.

[14] R. B. Gramacy, M. K. Warmuth, S. Brandt, and I. Ari. Adaptive caching by refetching. *Advances in Neural Information Processing Systems*, 15:1489–1496, 2002.

[15] A. György, T. Linder, and G. Lugosi. Tracking the best of many experts. In *International Conference on Computational Learning Theory*, pages 204–216. Springer, 2005.

[16] A. Gyorgy, T. Linder, and G. Lugosi. Efficient tracking of large classes of experts. *IEEE Transactions on Information Theory*, 58(11):6709–6725, 2012.

[17] D. Haussler, J. Kivinen, and M. K. Warmuth. Sequential prediction of individual sequences under general loss functions. *IEEE Transactions on Information Theory*, 44(5):1906–1925, 1998.

[18] E. Hazan and C. Seshadhri. Efficient learning algorithms for changing environments. In *Proceedings of the 26th annual international conference on machine learning*, pages 393–400, 2009.

[19] M. Herbster, S. Pasteris, and M. Pontil. Predicting a switching sequence of graph labelings. *J. Mach. Learn. Res.*, 16:2003–2022, 2015.

[20] M. Herbster, S. Pasteris, and L. Tse. Online multitask learning with long-term memory. In *Advances in Neural Information Processing Systems*, volume 33, pages 17779–17791, 2020.

[21] M. Herbster and J. Robinson. Online prediction of switching graph labelings with cluster specialists. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[22] M. Herbster and M. K. Warmuth. Tracking the best expert. *Machine learning*, 32(2):151–178, 1998.

[23] M. Herbster and M. K. Warmuth. Tracking the best linear predictor. *Journal of Machine Learning Research*, 1(Sep):281–309, 2001.

[24] D. Hoeven, T. Erven, and W. Kotłowski. The many faces of exponential weights in online learning. In *Conference On Learning Theory*, pages 2067–2092. PMLR, 2018.

[25] K.-S. Jun, F. Orabona, S. Wright, and R. Willett. Improved strongly adaptive online learning using coin betting. In *Artificial Intelligence and Statistics*, pages 943–951. PMLR, 2017.

[26] W. M. Koolen, D. Adamskiy, and M. K. Warmuth. Putting bayes to sleep. In *NIPS*, pages 135–143, 2012.

[27] W. M. Koolen and T. van Erven. Freezing and sleeping: Tracking experts that learn by evolving past posteriors. *CoRR*, abs/1008.4654, 2010.

[28] S. S. Kozat and A. C. Singer. Universal switching portfolios under transaction costs. In *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 5404–5407. IEEE, 2008.

[29] W. Krichene, S. Krichene, and A. Bayen. Efficient bregman projections onto the simplex. In *2015 54th IEEE Conference on Decision and Control (CDC)*, pages 3291–3298. IEEE, 2015.

[30] B. Li and S. C. Hoi. Online portfolio selection: A survey. *ACM Computing Surveys (CSUR)*, 46(3):1–36, 2014.

[31] B. Li, J. Wang, D. Huang, and S. C. Hoi. Transaction cost optimization for online portfolio selection. *Quantitative Finance*, 18(8):1411–1424, 2018.

[32] N. Littlestone and M. K. Warmuth. The weighted majority algorithm. *Information and computation*, 108(2):212–261, 1994.

[33] M. McCloskey and N. J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier, 1989.

[34] J. Mourtada and O.-A. Maillard. Efficient tracking of a growing number of experts. In *International Conference on Algorithmic Learning Theory*, pages 517–539. PMLR, 2017.

[35] H. T. Nguyen and K. Franke. Adaptive intrusion detection system via online machine learning. In *2012 12th International Conference on Hybrid Intelligent Systems (HIS)*, pages 271–277. IEEE, 2012.

[36] D. Sharma, M.-F. Balcan, and T. Dick. Learning piecewise lipschitz functions in changing environments. In *International Conference on Artificial Intelligence and Statistics*, pages 3567–3577. PMLR, 2020.

[37] Y. Singer. Switching portfolios. *International Journal of Neural Systems*, 8(04):445–455, 1997.

[38] V. Vovk. A game of prediction with expert advice. *Journal of Computer and System Sciences*, 56(2):153–173, 1998.

[39] V. Vovk. Derandomizing stochastic prediction strategies. *Machine Learning*, 35(3):247–282, 1999.

[40] V. G. Vovk. Aggregating strategies. *Proc. of Computational Learning Theory, 1990*, 1990.

[41] K. Zheng, H. Luo, I. Diakonikolas, and L. Wang. Equipping experts/bandits with long-term memory. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[42] M. Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the 20th international conference on machine learning (icml-03)*, pages 928–936, 2003.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] All claims made in the abstract and introduction are established in the paper's contributions and scope (Sections 2-5). All proofs are contained in the appendices.
   (b) Did you describe the limitations of your work? [Yes] See Section 5 where we discuss the limitations of our algorithm relying on parameters which require tuning.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Section 5.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] The model of prediction with expert advice and its assumptions are defined in Section 2. For each theoretical result all assumptions are stated in theorem statements.
   (b) Did you include complete proofs of all theoretical results? [Yes] Complete proofs of all theoretical results are given in the appendices (please see the supplementary material).
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [N/A]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [N/A]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [N/A]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [N/A]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]