Online Multitask Learning with Long-Term Memory

Mark Herbster, Stephen Pasteris, Lisa Tse
Department of Computer Science
University College London
London
United Kingdom
(m.herbster|s.pasteris|l.tse)@cs.ucl.ac.uk

Abstract

We introduce a novel online multitask setting. In this setting each task is partitioned into a sequence of segments that is unknown to the learner. Associated with each segment is a hypothesis from some hypothesis class. We give algorithms that are designed to exploit the scenario where there are many such segments but significantly fewer associated hypotheses. We prove regret bounds that hold for any segmentation of the tasks and any association of hypotheses to the segments. In the single-task setting this is equivalent to *switching with long-term memory* in the sense of [1]. We provide an algorithm that predicts on each trial in time linear in the number of hypotheses when the hypothesis class is finite. We also consider infinite hypothesis classes from reproducing kernel Hilbert spaces for which we give an algorithm whose per trial time complexity is cubic in the number of cumulative trials. In the single-task special case this is the first example of an efficient regret-bounded switching algorithm with long-term memory for a non-parametric hypothesis class.

1 Introduction

We consider a model of online prediction in a non-stationary environment with multiple interrelated tasks. Associated with each task is an asynchronous data stream. As an example, consider a scenario where a team of drones may need to decontaminate an area of toxic waste. In this example, the tasks correspond to drones. Each drone is receiving a data stream from its sensors. The data streams are non-stationary but interdependent as the drones are travelling within a common site. At any point in time, a drone receives an instance x and is required to predict its label y. The aim is to minimize mispredictions. As is standard in regret-bounded learning we have no statistical assumptions on the data-generation process. Instead, we aim to predict well relative to some hypothesis class of predictors. Unlike a standard regret model, where we aim to predict well in comparison to a single hypothesis, we instead aim to predict well relative to a completely unknown sequence of hypotheses in each task's data stream, as illustrated by the "*coloring*" in Figure 1. Each *mode* (color) corresponds to a distinct hypothesis from the hypothesis class. A *switch* is said to have occurred whenever we move between modes temporally within the same task.

Thus in task 1, there are three modes and four switches. We are particularly motivated by the case that a mode once present will possibly recur multiple times even within different tasks, i.e., "modes" $\ll$ "switches." We will give algorithms and regret bounds for finite hypothesis classes (the "experts" model [2, 3, 4]) and for infinite non-parametric Reproducing Kernel Hilbert Space (RKHS) [5] hypothesis classes.

The paper is organized as follows. In the next section, we introduce our formal model for online switching multitask learning. In doing so we provide a brief review of some related online learning results which enable us to provide a prospectus for attainable regret bounds. This is done by

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

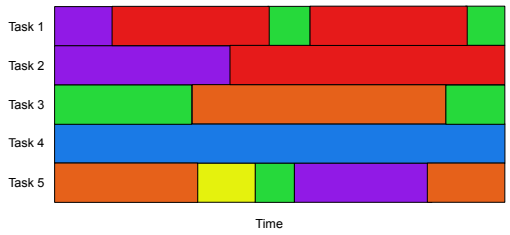

Figure 1: A Coloring of Data Streams (5 tasks, 6 modes, and 11 switches).

For $\tau = 1$ to T do
  Receive task $\ell^\tau \in [s]$.
  Set $i \leftarrow \ell^\tau$; $t \leftarrow \boldsymbol{\sigma}(\tau)$.
  Receive instance $x^\tau \equiv x_t^i \in \mathcal{X}$.
  Predict $\hat{y}^\tau \equiv \hat{y}_t^i \in \{-1, 1\}$.
  Receive label $y^\tau \equiv y_t^i \in \{-1, 1\}$.
  Incur Loss $\mathcal{L}_{01}(y^\tau, \hat{y}^\tau)$.

Figure 2: The Switching Multitask Model

considering the bounds achievable by non-polynomial time algorithms. We then provide a brief survey of related work as well as our notational conventions. In Sections 3 and 4 we provide algorithms and bounds for finite hypothesis classes and RKHS hypothesis classes, respectively. Finally, we provide a few concluding remarks in Section 5. The supplementary appendices contain our proofs.

2 Online Learning with Switching, Memory, and Multiple Tasks

We review the models and regret bounds for online learning in the single-task, switching, and switching with memory models as background for our multitask switching model with memory.

In the single-task online model a *learner* receives data sequentially so that on a trial $t = 1, \ldots, T$: 1) the learner receives an instance $x_t \in \mathcal{X}$ from the *environment*, then 2) predicts a label $\hat{y}_t \in \{-1, 1\}$, then 3) receives a label from the environment $y_t \in \{-1, 1\}$ and then 4) incurs a *zero-one* loss $\mathcal{L}_{01}(y_t, \hat{y}_t) := [y_t \neq \hat{y}_t]$. There are no probabilistic assumptions on how the environment generates its instances or their labels; it is an arbitrary process which in fact may be adversarial. The only restriction on the environment is that it does not "see" the learner's $\hat{y}_t$ until after it reveals y_t. The learner's aim will be to compete with a hypothesis class of predictors $\mathcal{H} \subseteq \{-1, 1\}^{\mathcal{X}}$ so as to minimize its *expected regret*, $R_T(h) := \sum_{t=1}^T \mathbb{E}[\mathcal{L}_{01}(y_t, \hat{y}_t)] - \mathcal{L}_{01}(y_t, h(x_t))$ for every hypothesis $h \in \mathcal{H}$, where the expectation is with respect to the learner's internal randomization.

In this paper we will consider two types of hypothesis classes: a finite set of hypotheses $\mathcal{H}_{\text{fin}}$, and a set $\mathcal{H}_K$ induced by a kernel K. A "multiplicative weight" (MW) algorithm [6] that achieves a regret bound[1] of the form

$$R_T(h) \in \mathcal{O}\left(\sqrt{\log(|\mathcal{H}_{\text{fin}}|)T}\right) \quad (\forall h \in \mathcal{H}_{\text{fin}}) \tag{1}$$

was given in [7] for finite hypothesis classes. This is a special case of the framework of "prediction with expert advice" introduced in [2, 3]. Given a reproducing kernel $K : \mathcal{X} \times \mathcal{X} \to \Re$ we denote the induced norm of the reproducing kernel Hilbert space (RKHS) $\mathcal{H}_K$ as $\|\cdot\|_K$ (for details on RKHS see [5] also Appendix C.2). Given an instance sequence $\boldsymbol{x} := (x_1, \ldots, x_T)$, we let $\mathcal{H}_K^{(\boldsymbol{x})} := \{h \in \mathcal{H}_K : h(x_t) \in \{-1, 1\}, \forall t \in [T]\}$ denote the functions in $\mathcal{H}_K$ that are binary-valued on the sequence. An analysis of online gradient descent (OGD_K) with the hinge loss, kernel K and randomized prediction [8, see e.g., Ch. 2 & 3] (proof included in Appendix C.3 for completeness) gives an expected regret bound of

$$R_T(h) \in \mathcal{O}\left(\sqrt{\|h\|_K^2 X_K^2 T}\right) \quad (\forall h \in \mathcal{H}_K^{(\boldsymbol{x})}), \tag{2}$$

where $X_K^2 \geq \max_{t \in [T]} K(x_t, x_t)$.

In the switching single-task model the hypothesis becomes a sequence of hypotheses $\boldsymbol{h} = (h_1, h_2, \ldots, h_T) \in \mathcal{H}^T$ and the regret is $R_T(\boldsymbol{h}) := \sum_{t=1}^T \mathbb{E}[\mathcal{L}_{01}(y_t, \hat{y}_t)] - \mathcal{L}_{01}(y_t, h_t(x_t))$. Two parameters of interest are the number of *switches* $k := \sum_{t=1}^{T-1} [h_t \neq h_{t+1}]$ and the number of *modes* $m := |\cup_{t=1}^T \{h_t\}|$, i.e., the number of the distinct hypotheses in the sequence. In this work we are

[1] Technically, when we say that an algorithm *achieves a bound*, it may be that the algorithm depends on a small set of parameters which we have then assumed are "tuned" optimally.

interested in *long-term memory*, that is, algorithms and bounds that are designed to exploit the case of $m \ll k$.

The methodology of [9] may be used to derive an expected regret bound for $\mathcal{H}_{\text{fin}}$ in the switching single-task model of the form $R_T(\boldsymbol{h}) \in \mathcal{O}(\sqrt{(k\log(|\mathcal{H}_{\text{fin}}|) + k\log(T/k))T})$. Freund in [10] posed an open problem to improve the results of [9] in the case of long-term memory ($m \ll k$). Freund gave counting arguments that led to an exponential-time algorithm with a regret bound of $R_T(\boldsymbol{h}) \in \mathcal{O}(\sqrt{(m\log(|\mathcal{H}_{\text{fin}}|) + k\log m + k\log(T/k))T})$. In [1] an efficient algorithm was given with nearly this bound, except for a small additional additive "$T\log\log T$" term under the square root. For the hypothesis class $\mathcal{H}_K^{(\boldsymbol{x})}$ we may give non-memory bounds of the form $R_T(\boldsymbol{h}) \in \mathcal{O}(\sqrt{k\max_t \|h_t\|_K^2 X_K^2 T})$ by using a simple modification [11] of OGD_K (see Appendix C.3). To the best of our knowledge there are no previous long-term memory bounds for $\mathcal{H}_K^{(\boldsymbol{x})}$ (however see the discussion of [12] in Section 2.2); these will be a special case of our multitask model, to be introduced next.

2.1 Switching Multitask Model

In Figure 2 we illustrate the protocol for our multitask model. The model is essentially the same as the switching single-task model, except that we now have s tasks. On each (global) trial τ the environment reveals the active task $\ell^\tau \in [s]$. The ordering of tasks chosen by the environment is arbitrary, and therefore the active task may change on every (global) trial τ. We use the following notational convention: (global time) $\tau \equiv {}^i_t$ (local time) where $i = \ell^\tau$, $t = \boldsymbol{\sigma}(\tau)$ and $\boldsymbol{\sigma}(\tau) := \sum_{j=1}^{\tau}[\ell^j = \ell^\tau]$. Thus $x^\tau \equiv x_t^i$, $y^\tau \equiv y_t^i$, etc., where the mapping is determined implicitly by the task vector $\boldsymbol{\ell} \in [s]^T$. Each task $i \in [s]$ has its own data pair (instance, label) sequence $(x_1^i, y_1^i), \ldots, (x_{T^i}^i, y_{T^i}^i)$ where $T = T^1 + \ldots + T^s$. The *multitask hypotheses multiset* is denoted as $\boldsymbol{h}^* = (h^1, \ldots, h^T) \equiv (h_1^1, \ldots, h_{T^1}^1, \ldots, h_1^s, \ldots, h_{T^s}^s) \in \mathcal{H}^T$. In the multitask model we denote the number of switches as $k(\boldsymbol{h}^*) := \sum_{i=1}^{s}\sum_{t=1}^{T^i-1}[h_t^i \neq h_{t+1}^i]$, the set of modes as $m(\boldsymbol{h}^*) := \cup_{i=1}^{s}\cup_{t=1}^{T^i}\{h_t^i\}$ and the multitask regret as $R_T(\boldsymbol{h}^*) := \sum_{i=1}^{s}\sum_{t=1}^{T^i}\mathbb{E}[\mathcal{L}_{01}(y_t^i, \hat{y}_t^i)] - \mathcal{L}_{01}(y_t^i, h_t^i(x_t^i))$. In the following, we give motivating upper bounds based on exponential-time algorithms induced by "meta-experts." We provide a lower bound with respect to $\mathcal{H}_K^{(\boldsymbol{x})}$ in Proposition 4.

The idea of "meta-experts" is to take the base class of hypotheses and to construct a class of "meta-hypotheses" by combining the original hypotheses to form new ones, and then apply an MW algorithm to the constructed class; in other words, we reduce the "meta-model" to the "base-model." In our setting, the base class is $\mathcal{H}_{\text{fin}} \subseteq \{-1,1\}^{\mathcal{X}}$ and our meta-hypothesis class will be some $\mathcal{H}' \subseteq \{-1,1\}^{\mathcal{X}'}$ where $\mathcal{X}' := \{(x,t,i) : x \in \mathcal{X}, t \in [T^i], i \in [s]\}$. To construct this set we define $\bar{\mathcal{H}}(k, m, s, \mathcal{H}_{\text{fin}}, T^1, \ldots, T^s) := \{(h_1^1, \ldots, h_{T^s}^s) = \bar{\boldsymbol{h}} \in \mathcal{H}_{\text{fin}}^T : k = k(\bar{\boldsymbol{h}}), m = |m(\bar{\boldsymbol{h}})|\}$ and then observe that for each $\bar{\boldsymbol{h}} \in \bar{\mathcal{H}}$ we may define an $h' : \mathcal{X}' \to \{-1,1\}$ via $h'((x,t,i)) := h_t^i(x)$, where h_t^i is an element of $\bar{\boldsymbol{h}}$. We thus construct $\mathcal{H}'$ by converting each $\bar{\boldsymbol{h}} \in \bar{\mathcal{H}}$ to an $h' \in \mathcal{H}'$. Hence we have reduced the switching multitask model to the single-task model with respect to $\mathcal{H}'$. We proceed to obtain a bound by observing that the cardinality of $\bar{\mathcal{H}}$ is bounded above by $\binom{T-s}{k}\binom{n}{m}m^s(m-1)^k$ where $n = |\mathcal{H}_{\text{fin}}|$. If we then substitute into (1) and then further upper bound we have

$$R_T(\boldsymbol{h}^*) \in \mathcal{O}\left(\sqrt{(m\log(n/m) + s\log m + k\log m + k\log((T-s)/k))T}\right), \tag{3}$$

for any $\boldsymbol{h}^* \in \mathcal{H}_{\text{fin}}^T$ such that $k = k(\boldsymbol{h}^*)$ and $m = |m(\boldsymbol{h}^*)|$. The drawback is that the algorithm requires exponential time. In Section 3 we will give an algorithm whose time to predict per trial is $\mathcal{O}(|\mathcal{H}_{\text{fin}}|)$ and whose bound is equivalent up to constant factors.

We cannot directly adapt the above argument to obtain an algorithm and bound for $\mathcal{H}_K^{(\boldsymbol{x})}$ since the cardinality, in general, is infinite, and additionally we do not know $\boldsymbol{x}$ in advance. However, the structure of the argument is the same. Instead of using hypotheses from $\mathcal{H}_K^{(\boldsymbol{x})}$ as building blocks to construct meta-hypotheses, we use multiple instantiations of an online algorithm for $\mathcal{H}_K^{(\boldsymbol{x})}$ as our building blocks. We let $\mathcal{A}_K := \{a[1], \ldots, a[m]\}$ denote our set of m instantiations that will act as a surrogate for the hypothesis class $\mathcal{H}_K^{(\boldsymbol{x})}$. We then construct the set, $\bar{\mathcal{A}}_K(k, m, s, T^1, \ldots, T^s) := \{\bar{\boldsymbol{a}} \in \mathcal{A}_K^T : k = k(\bar{\boldsymbol{a}}), m = |m(\bar{\boldsymbol{a}})|\}$. Each $\bar{\boldsymbol{a}} \in \bar{\mathcal{A}}_K$ now defines a meta-algorithm for the multitask

setting. That is, given an online multitask data sequence $(x_1^i, y_1^i), \ldots, (x_{T^j}^j, y_{T^j}^j)$, each element of $\bar{\boldsymbol{a}}$ will "color" the corresponding data pair with one of the m instantiations (we will use the function $\alpha : \{(t,i) : t \in [T^i], i \in [s]\} \to [m]$ to denote this mapping with respect to $\bar{\boldsymbol{a}}$). Each instantiation will *receive as inputs only the online sequence of the data pairs corresponding to its "color"*; likewise, the prediction of meta-algorithm $\bar{\boldsymbol{a}}$ will be that of the instantiation active on that trial. We will use as our base algorithm OGD_K. Thus for the meta-algorithm $\bar{\boldsymbol{a}}$ we have from (2),

$$\sum_{i=1}^{s}\sum_{t=1}^{T^i}\mathbb{E}[\mathcal{L}_{01}(y_t^i, \hat{y}_t^i)] \le \sum_{i=1}^{s}\sum_{t=1}^{T^i}\mathcal{L}_{01}(y_t^i, h[\alpha(_t^i)](x_t^i)) + \sum_{j=1}^{m}\mathcal{O}\left(\sqrt{\|h[j]\|_K^2 X^2 T^j}\right) \tag{4}$$

for any received instance sequence $\boldsymbol{x} \in \mathcal{X}^T$ and for any $h[1], \ldots, h[m] \in \mathcal{H}_K^{(\boldsymbol{x})}$. The MW algorithm [3, 2, 4] does not work just for hypothesis classes; more generally, it works for collections of algorithms. Hence we may run the MW as a meta-meta-algorithm to combine all of the meta-algorithms $\bar{\boldsymbol{a}} \in \bar{\mathcal{A}}_K$. Thus by substituting the loss for each meta-algorithm $\bar{\boldsymbol{a}}$ (the R.H.S. of (4)) into (1) and using the upper bound $\binom{T-s}{k} m^s (m-1)^k$ for the cardinality of $\bar{\mathcal{A}}_K$, we obtain (using upper bounds for binomial coefficients and the inequality $\sum_i \sqrt{p_i q_i} \le \sqrt{(\sum_i p_i)(\sum_i q_i)}$),

$$R_T(\boldsymbol{h}^*) \in \mathcal{O}\left(\sqrt{(\textstyle\sum_{h \in m(\boldsymbol{h}^*)} \|h\|_K^2 X_K^2 + s\log m + k\log m + k\log((T-s)/k))T}\right), \tag{5}$$

for any received instance sequence $\boldsymbol{x} \in \mathcal{X}^T$ and for any $\boldsymbol{h}^* \in \mathcal{H}_K^{(\boldsymbol{x})^T}$ such that $k = k(\boldsymbol{h}^*)$ and $m = |m(\boldsymbol{h}^*)|$.

The terms $m\log(n/m)$ (assuming $m \ll n$) and $\sum_{h \in m(\boldsymbol{h}^*)} \|h\|_K^2 X_K^2$ may be viewed as *learner complexities*, i.e., the price we "pay" for identifying the hypotheses that fit the modes. A salient feature of long-term memory bounds is that although the data pairs associated with each hypothesis are intermixed in the multitask sequence, we pay the learner complexity only modestly in terms of potentially leading multiplicative constants. A switching algorithm without long-term memory "forgets" and pays the full price for a mode on every switch or new task. We gave exponential-time algorithms for $\mathcal{H}_{\mathrm{fin}}$ and $\mathcal{H}_K^{(\boldsymbol{x})}$ with $\mathcal{O}(1)$ leading multiplicative constants in the discussion leading to (3) and (5). We give efficient algorithms for finite hypothesis classes and RKHS hypothesis classes in Sections 3 and 4, with time complexities of $\mathcal{O}(n)$ and $\mathcal{O}(T^3)$ per trial, and in terms of learner complexities they gain only leading multiplicative constants of $\mathcal{O}(1)$ and $\mathcal{O}(\log T)$.

2.2 Related Work

In this section we briefly describe other related work in the online setting that considers either *switching* or *multitask* models.

The first result for switching in the experts model was the WML algorithm [3] which was generalized in [9]. There is an extensive literature building on those papers, with some prominent results including [1, 13, 14, 15, 16, 17, 15, 18, 19, 20, 21, 22]. Relevant for our model are those papers [1, 14, 17, 15, 20, 21, 22] that address the problem of long-term memory ($m \ll k$), in particular [1, 14, 17].

Analogous to the problem of long-term memory in online learning is the problem of catastrophic forgetting in artificial neural network research [23, 24]. That is the problem of how a system can adapt to new information without forgetting the old. In online learning that is the problem of how an algorithm can both quickly adapt its prediction hypothesis and recall a previously successful prediction hypothesis when needed. In the experts model this problem was first addressed by [1], which gave an algorithm that stores each of its past state vectors, and then at each update mixes these vectors into the current state vector. In [14], an algorithm and bounds were given that extended the base comparision class of experts to include Bernoulli models. An improved algorithm with a Bayesian intepretation based on the idea of "circadian specialists" was given for this setting in [17]. Our construction of Algorithm 1 was based on this methodology.

The problem of linear regression with long term memory was posed as an open problem in [17, Sec. 5]. Algorithm 2 gives an algorithm for linear interpolation in a RKHS with a regret bound that reflects long-term memory. Switching linear prediction has been considered in [11, 25, 26, 12]. Only [12] addresses the issue of long-term memory. The methodology of [12] is a direct inspiration for Algorithm 2. We significantly extend the result of [12, Eq. (1)]. Their result was i) restricted to a

mistake as opposed to a regret bound, ii) restricted to finite positive definite matrices and iii) in their mistake bound the term analogous to $\sum_{h \in m(\boldsymbol{h}^*)} \|h\|_K^2 X_K^2$ was increased by a multiplicative factor of $\tilde{\mathcal{O}}(|m(\boldsymbol{h}^*)|)$, a significantly weaker result.

Multitask learning has been considered extensively in the batch setting, with some prominent early results including [27, 28, 29]. In the online multitask *expert* setting [30, 31, 32, 17] considered a model which may be seen as a special case of ours where each task is associated only with a single hypothesis, i.e., no internal switching within a task. Also in the expert setting [33, 34] considered models where the prediction was made for all tasks simultaneously. In [34] the aim was to predict well relative to a set of possible predefined task interrelationships and in [33] the interrelationships were to be discovered algorithmically. The online multitask *linear* prediction setting was considered in [35, 36, 37]. The models of [36, 37] are similar to ours, but like previous work in the expert setting, these models are limited to one "hypothesis" per task. In the work of [35], the predictions were made for all tasks simultaneously through a joint loss function.

2.3 Preliminaries

For any positive integer m, we define $[m] := \{1, 2, \ldots, m\}$. For any predicate $[\text{PRED}] := 1$ if PRED is true and equals 0 otherwise, and for any $x \in \Re$, $[x]_+ := x[x > 0]$. We denote the inner product of vectors as both $\boldsymbol{x}, \boldsymbol{w} \in \Re^n$ as $\langle \boldsymbol{x}, \boldsymbol{w} \rangle = \boldsymbol{x} \cdot \boldsymbol{w} = \sum_{i=1}^n x_i w_i$, component-wise multiplication $\boldsymbol{x} \odot \boldsymbol{w} := (x_1 w_1, \ldots, x_n w_n)$ and the norm as $\|\boldsymbol{w}\| = \sqrt{\langle \boldsymbol{w}, \boldsymbol{w} \rangle}$. If $f : \Re \to \Re$ and $\boldsymbol{x} \in \Re^n$ then $f(\boldsymbol{x}) := (f(x_1), \ldots, f(x_n))$. The xth-coordinate vector is denoted $\boldsymbol{e}_X^x := ([x = z])_{z \in X}$; we commonly abbreviate this to $\boldsymbol{e}^x$. We denote the probability simplex as $\Delta_{\mathcal{H}} := \{h \in [0,1]^{\mathcal{H}}\} \cap \{h : \sum_{h \in \mathcal{H}} = 1\}$ and set $\Delta_n := \Delta_{[n]}$. We denote the binary entropy as $H(p) := p \log \frac{1}{p} + (1-p) \log \frac{1}{1-p}$. If $\boldsymbol{v} \in \Delta_{\mathcal{H}}$ then $h \sim \boldsymbol{v}$ denotes that h is a random sample from the probability vector $\boldsymbol{v}$ over the set $\mathcal{H}$. For vectors $\boldsymbol{p} \in \Re^m$ and $\boldsymbol{q} \in \Re^n$ we define $[\boldsymbol{p}; \boldsymbol{q}] \in \Re^{m+n}$ to be the concatenation of $\boldsymbol{p}$ and $\boldsymbol{q}$, which we regard as a column vector. Hence $[\boldsymbol{p}; \boldsymbol{q}]^\top [\bar{\boldsymbol{p}}; \bar{\boldsymbol{q}}] = \boldsymbol{p}^\top \bar{\boldsymbol{p}} + \boldsymbol{q}^\top \bar{\boldsymbol{q}}$.

The notation $\boldsymbol{M}^+$ and $\sqrt{\boldsymbol{M}}$ denotes the pseudo-inverse and the unique positive square root, respectively, of a positive semi-definite matrix $\boldsymbol{M}$. The trace of a square matrix is denoted by $\text{tr}(\boldsymbol{Y}) := \sum_{i=1}^n Y_{ii}$ for $\boldsymbol{Y} \in \Re^{n \times n}$. The $m \times m$ identity matrix is denoted $\boldsymbol{I}^m$. A function $K : \mathcal{X} \times \mathcal{X} \to \Re$ is a strictly positive definite (SPD) kernel iff for every finite $X \subseteq \mathcal{X}$ the matrix $K(x, x')_{x,x' \in X}$ is symmetric and strictly positive definite, for example, the Gaussian kernel. In addition, we define $\boldsymbol{S}^m$ to be the set of $m \times m$ symmetric matrices and let $\boldsymbol{S}_+^m$ and $\boldsymbol{S}_{++}^m$ be the subset of positive semidefinite and strictly positive definite matrices, respectively. We define the squared radius of $\boldsymbol{M} \in \boldsymbol{S}_+^m$ as $\mathcal{R}_{\boldsymbol{M}} := \max_{i \in [m]} M_{ii}^+$. The (undirected) graph Laplacian matrix is defined by $\boldsymbol{L} := \boldsymbol{D} - \boldsymbol{A}$ where $\boldsymbol{D}$ is the degree matrix and $\boldsymbol{A}$ is the adjacency matrix. The corresponding (strictly) positive definite *PDLaplacian* of an m-vertex connected graph is $\boldsymbol{L}^\circ := \boldsymbol{L} + \mathcal{R}_L^{-1} \left(\frac{\boldsymbol{1}}{m}\right) \left(\frac{\boldsymbol{1}}{m}\right)^\top$.

3 Finite Hypothesis Classes

In this section we present the algorithm and the regret bound for finite hypothesis classes, with proofs given in Appendix A. The design and analysis of the algorithm is inspired by [17], which considers a Bayesian setting where, on each trial, each hypothesis h gives an estimated probability $P(y^\tau = \hat{y}|h)$ of the outcome y^τ. The idea is for the learner to predict a probability $\hat{P}(y^\tau = \hat{y})$ and the loss incurred is the log loss, $\log(1/\hat{P}(y^\tau))$. Our algorithm, on the other hand, is framed in the well known "Allocation" setting [38] where the learner must play, on trial τ, a vector $\boldsymbol{v}^\tau \in \Delta_n$ and incurs a loss of $\boldsymbol{c}^\tau \cdot \boldsymbol{v}^\tau$ where all components of $\boldsymbol{c}^\tau$ are in $[0, 1]$.

To gain some intuition about the algorithm we observe the following. The algorithm maintains and updates the following vectors: a "global" probability vector $\boldsymbol{\pi}^\tau \in \Delta_{\mathcal{H}_{\text{fin}}}$ and the "local" task weight vectors $\boldsymbol{w}_t^1, \ldots, \boldsymbol{w}_t^s \in [0,1]^{\mathcal{H}_{\text{fin}}}$. Given an hypothesis $h \in \mathcal{H}_{\text{fin}}$, the scalar π_h^τ represents our "confidence", on trial τ, that hypothesis h is in $m(\boldsymbol{h}^*)$. For a given task i, hypothesis $h \in \mathcal{H}_{\text{fin}}$, and local time t, the scalar $w_{t,h}^i$ represents our confidence that $h = h_t^i$ if we knew that h was in $m(\boldsymbol{h}^*)$. Putting together, $\pi_h^\tau w_{\boldsymbol{\sigma}(\tau),h}^i$ represents our confidence, on trial τ, that $h = h_t^i$. The weights $\boldsymbol{\pi}^\tau$ and $\boldsymbol{w}_t^i$ (for tasks i) are designed in such a way that, not only do they store all the information required by the algorithm, but also on each trial τ we need only update $\boldsymbol{\pi}^\tau$ and $\boldsymbol{w}_t^{\ell^\tau}$. Thus the algorithm

Algorithm 1 Predicting $\mathcal{H}_{\text{fin}}$ in a switching multitask setting.

Parameters: $\mathcal{H}_{\text{fin}} \subseteq \{-1,1\}^{\mathcal{X}}$; $s, m, k, T \in \mathbb{N}$

Initialization: $n := |\mathcal{H}_{\text{fin}}|$; $\boldsymbol{\pi}^1 \leftarrow \frac{\mathbf{1}}{n}$; $\mu := \frac{1}{m}$; $\boldsymbol{w}_1^1 = \cdots = \boldsymbol{w}_1^s \leftarrow \mu\mathbf{1}$; $\theta := 1 - \frac{k}{T-s}$; $\phi := \frac{k}{(m-1)(T-s)}$ and

$$\eta := \sqrt{\left(m \log\left(\tfrac{n}{m}\right) + smH\left(\tfrac{1}{m}\right) + (T-s)H\left(\tfrac{k}{T-s}\right) + (m-1)(T-s)H\left(\tfrac{k}{(m-1)(T-s)}\right)\right)\frac{2}{T}}$$

For $\tau = 1, \ldots, T$

- Receive task $\ell^\tau \in [s]$.
- Receive $x^\tau \in \mathcal{X}$.
- Set $i \leftarrow \ell^\tau$; $t \leftarrow \boldsymbol{\sigma}(\tau)$.
- Predict

$$\boldsymbol{v}^\tau \leftarrow \frac{\boldsymbol{\pi}^\tau \odot \boldsymbol{w}_t^i}{\boldsymbol{\pi}^\tau \cdot \boldsymbol{w}_t^i}, \qquad \hat{h}^\tau \sim \boldsymbol{v}^\tau, \qquad \hat{y}^\tau \leftarrow \hat{h}^\tau(x^\tau).$$

- Receive $y^\tau \in \{-1, 1\}$.
- Update:

$$\begin{aligned}
&i) \quad \forall h \in \mathcal{H}_{\text{fin}},\ c_h^\tau = \mathcal{L}_{01}(h(x^\tau), y^\tau) && ii)\ \boldsymbol{\delta} \leftarrow \boldsymbol{w}_t^i \odot \exp(-\eta \boldsymbol{c}^\tau)\\
&iii)\ \beta \leftarrow (\boldsymbol{\pi}^\tau \cdot \boldsymbol{w}_t^i)/(\boldsymbol{\pi}^\tau \cdot \boldsymbol{\delta}) && iv)\ \boldsymbol{\epsilon} \leftarrow \mathbf{1} - \boldsymbol{w}_t^i + \beta\boldsymbol{\delta}\\
&v) \quad \boldsymbol{\pi}^{\tau+1} \leftarrow \boldsymbol{\pi}^\tau \odot \boldsymbol{\epsilon} && vi)\ \boldsymbol{w}_{t+1}^i \leftarrow (\phi(\mathbf{1} - \boldsymbol{w}_t^i) + \theta\beta\boldsymbol{\delta}) \odot \boldsymbol{\epsilon}^{-1}
\end{aligned}$$

predicts in $\mathcal{O}(n)$ time per trial and requires $\mathcal{O}(sn)$ space. We bound the regret of the algorithm in the following theorem.

Theorem 1. *The expected regret of Algorithm 1 with parameters $\mathcal{H}_{\text{fin}} \subseteq \{-1,1\}^{\mathcal{X}}$; $s, m, k, T \in \mathbb{N}$ and*

$$C := m \log\left(\frac{n}{m}\right) + smH\left(\frac{1}{m}\right) + (T-s)H\left(\frac{k}{T-s}\right) + (m-1)(T-s)H\left(\frac{k}{(m-1)(T-s)}\right)$$

is bounded above by

$$\sum_{i=1}^{s}\sum_{t=1}^{T^i} \mathbb{E}[\mathcal{L}_{01}(y_t^i, \hat{y}_t^i)] - \mathcal{L}_{01}(y_t^i, h_t^i(x_t^i)) \leq \sqrt{2CT}$$

for any $\boldsymbol{h}^* \in \mathcal{H}_{\text{fin}}^T$ such that $k = k(\boldsymbol{h}^*)$, $m \geq |m(\boldsymbol{h}^*)|$, $m > 1$. Furthermore,

$$C \leq m \log\left(\frac{n}{m}\right) + s(\log(m) + 1) + k\left(\log(m-1) + 2\log\left(\frac{T-s}{k}\right) + 2\right).$$

In further comparison to [17] we observe that we can obtain bounds for the log loss with our algorithm by defining $\hat{P}(y^\tau = \hat{y}) := \sum_h v_h^\tau P(y^\tau = \hat{y}|h)$ and redefining $c_h^\tau := -\frac{1}{\eta}\log(P(y^\tau = \hat{y}|h))$ in the update. The resultant theorem then matches the bound of [17, Thm. 4] for single-task learning with long-term memory ($s = 1$) and the bound of [17, Thm. 6] for multitask learning with no switching ($k = 0$).

4 RKHS Hypothesis Classes

Our algorithm and its analysis builds on the algorithm for online inductive matrix completion with side-information (IMCSI) from [39, Theorem 1, Algorithm 2 and Proposition 4]. IMCSI is an example of a matrix multiplicative weight algorithm [40, 6]. We give notation and background from [39] to provide insight.

The max-norm (or γ_2 norm [41]) of a matrix $\boldsymbol{U} \in \Re^{m\times n}$ is defined by

$$\|\boldsymbol{U}\|_{\max} := \min_{\boldsymbol{P}\boldsymbol{Q}^\top = \boldsymbol{U}} \left\{ \max_{1\leq i\leq m} \|\boldsymbol{P}_i\| \times \max_{1\leq j\leq n} \|\boldsymbol{Q}_j\| \right\}, \tag{6}$$

where the minimum is over all matrices $\boldsymbol{P} \in \Re^{m\times d}$ and $\boldsymbol{Q} \in \Re^{n\times d}$ and every integer d. We denote the class of $m \times d$ *row-normalized* matrices as $\mathcal{N}^{m,d} := \{\hat{\boldsymbol{P}} \subset \Re^{m\times d} : \left\|\hat{\boldsymbol{P}}_i\right\| = 1,\ i \in [m]\}$. The quasi-dimension of a matrix is defined as follows.

Definition 2 ([39, Equation (3)]). *The* quasi-dimension *of a matrix* $\boldsymbol{U} \in \Re^{m \times n}$ *with respect to* $\boldsymbol{M} \in \boldsymbol{S}^m_{++}$, $\boldsymbol{N} \in \boldsymbol{S}^n_{++}$ *at* γ *as*

$$\mathcal{D}^{\gamma}_{\boldsymbol{M},\boldsymbol{N}}(\boldsymbol{U}) := \min_{\hat{\boldsymbol{P}}\hat{\boldsymbol{Q}}^\top = \gamma \boldsymbol{U}} \operatorname{tr}\left(\hat{\boldsymbol{P}}^\top \boldsymbol{M} \hat{\boldsymbol{P}}\right) \mathcal{R}_{\boldsymbol{M}} + \operatorname{tr}\left(\hat{\boldsymbol{Q}}^\top \boldsymbol{N} \hat{\boldsymbol{Q}}\right) \mathcal{R}_{\boldsymbol{N}}\,, \tag{7}$$

where the infimum is over all row-normalized matrices $\hat{\boldsymbol{P}} \in \mathcal{N}^{m,d}$ *and* $\hat{\boldsymbol{Q}} \in \mathcal{N}^{n,d}$ *and every integer* d*. If the infimum does not exist then* $\mathcal{D}^{\gamma}_{\boldsymbol{M},\boldsymbol{N}}(\boldsymbol{U}) := +\infty$ *(The infimum exists iff* $\|\boldsymbol{U}\|_{max} \leq 1/\gamma$*).*

The algorithm IMCSI addresses the problem of the online prediction of a binary comparator matrix $\boldsymbol{U}$ with side information. The side information is supplied as a pair of kernels over the row indices and the column indices. In [39, Theorem 1] a regret bound $\tilde{\mathcal{O}}(\sqrt{(\widehat{\mathcal{D}}/\gamma^2)T})$ is given, where $1/\gamma^2 \geq \|\boldsymbol{U}\|^2_{\max}$ and $\widehat{\mathcal{D}} \geq \mathcal{D}^{\gamma}_{\boldsymbol{M},\boldsymbol{N}}(\boldsymbol{U})$ are parameters of the algorithm that serve as upper estimates on $\|\boldsymbol{U}\|^2_{\max}$ and $\mathcal{D}^{\gamma}_{\boldsymbol{M},\boldsymbol{N}}(\boldsymbol{U})$. The first estimate $1/\gamma^2$ is an upper bound on the squared max-norm (Eq. (6)) which like the trace-norm may be seen as a proxy for the rank of the matrix [42]. The second estimate $\widehat{\mathcal{D}}$ is an upper bound of the *quasi-dimension* (Eq. (7)) which measures the quality of the side-information. The quasi-dimension depends upon the "best" factorization $(1/\gamma)\hat{\boldsymbol{P}}\hat{\boldsymbol{Q}}^\top = \boldsymbol{U}$, which will be smaller when the row $\hat{\boldsymbol{P}}$ (column $\hat{\boldsymbol{Q}}$) factors are in congruence with the row (column) kernel. We bound the quasi-dimension in Theorem 47 in Appendix B as a key step to proving Theorem 3.

In the reduction of our problem to a matrix completion problem with side information, the row indices correspond to the domain of the learner-supplied kernel K and the column indices correspond to the temporal dimension. On each trial we receive an x^τ (a.k.a. x^i_t). Thus the column of the comparator matrix (now $\boldsymbol{H}$) corresponding to time τ will contain the entries $H^\tau = (h^\tau(x^\upsilon))_{\upsilon \in [T]}$. Although we are predicting functions that are changing over time, the underlying assumption is that the change is sporadic; otherwise it is infeasible to prove a non-vacuous bound. Thus we expect $H^i_t \approx H^i_{t+1}$ and as such our column side-information kernel should reflect this expectation. Topologically we would therefore expect a kernel to present as s separate time *paths*, where nearness in time is nearness on the path. In the following we introduce the *path-tree-kernel* (the essence of the construction was first introduced in [43]), which satisfies this expectation in the single-task case. We then adapt this construction to the multitask setting.

A *path-tree* kernel $P : [T] \times [T] \to \Re$, is formed via the Laplacian of a fully complete binary *tree* with $N := 2^{\lceil \log_2 T \rceil + 1} - 1$ vertices. The *path* corresponds to the first T leaves of the tree, numbered sequentially from the leftmost to the rightmost leaf of the first T leaves. Denote this Laplacian as $\boldsymbol{L}$ where the path is identified with $[T]$ and the remaining vertices are identified with $[N] \setminus [T]$. Then using the definition $\boldsymbol{L}^\circ := \boldsymbol{L} + \left(\frac{\boldsymbol{1}}{N}\right)\left(\frac{\boldsymbol{1}}{N}\right)^\top \mathcal{R}_{\boldsymbol{L}}^{-1}$ we define $P(\tau, \upsilon) := (\boldsymbol{L}^\circ)^+_{\tau\upsilon}$ where τ, $\upsilon \in [T]$. We extend the path-tree kernel to a *multitask-path-tree* kernel by dividing the path into s contiguous segments, where segment i is a path of length T^i, and the task vector $\boldsymbol{\ell} \in [s]^T$ determines the mapping from global trial τ to task ℓ^τ and local trial $\boldsymbol{\sigma}(\tau)$. We define $\tilde{P}^{\boldsymbol{\ell},T^1,\ldots,T^s} : [T] \times [T] \to \Re$ as $\tilde{P}^{\boldsymbol{\ell},T^1,\ldots,T^s}(\tau, \upsilon) := P\left(\sum_{i=1}^{\ell^\tau - 1} T^i + \boldsymbol{\sigma}(\tau), \sum_{i=1}^{\ell^\upsilon - 1} T^i + \boldsymbol{\sigma}(\upsilon)\right)$. Observe we do not need to know the task vector $\boldsymbol{\ell}$ in advance; we only require upper bounds on the lengths of the tasks to be able to use this kernel. Finally, we note that it is perhaps surprising that we use a tree rather than a path directly. We discuss this issue following Lemma 49 in Appendix B.

Algorithm 2 requires $\mathcal{O}(t^3)$ time per trial t since we need to compute the eigendecomposition of three $\mathcal{O}(t) \times \mathcal{O}(t)$ matrices as well as sum $\mathcal{O}(t) \times \mathcal{O}(t)$ matrices up to t times. We bound the regret of the algorithm as follows.

Theorem 3. *The expected regret of Algorithm 2 with upper estimates,* $k \geq k(\boldsymbol{h}^*)$, $m \geq |m(\boldsymbol{h}^*)|$,

$$\hat{C} \geq C(\boldsymbol{h}^*) := \left(\sum_{h \in m(\boldsymbol{h}^*)} \|h\|^2_K X^2_K + 2(s + k - 1)m\lceil \log_2 T \rceil^2 + 2m^2\right),$$

$\hat{X}^2_K \geq \max_{\tau \in [T]} K(x^\tau, x^\tau)$*, and learning rate* $\eta = \sqrt{\frac{\hat{C}\log(2T)}{2Tm}}$ *is bounded by*

$$\sum_{i=1}^{s}\sum_{t=1}^{T^i} \mathbb{E}[\mathcal{L}_{01}(y^i_t, \hat{y}^i_t)] - \mathcal{L}_{01}(y^i_t, h^i_t(x^i_t)) \leq 4\sqrt{2\hat{C}\,T\log(2T)} \tag{8}$$

with received instance sequence $\boldsymbol{x} \in \mathcal{X}^T$ *and for any* $\boldsymbol{h}^* \in \mathcal{H}_K^{(\boldsymbol{x})^T}$.

Algorithm 2 Predicting $\mathcal{H}_K^{(\boldsymbol{x})}$ in a switching multitask setting.

Parameters: Tasks $s \in \mathbb{N}$, task lengths $T^1, \ldots, T^s \in \mathbb{N}$, $T := \sum_{i=1}^s T^i$, learning rate: $\eta > 0$, complexity estimate: $\hat{C} > 0$, modes: $m \in [T]$, SPD Kernel $K : \mathcal{X} \times \mathcal{X} \to \Re$, $\tilde{P} := \tilde{P}^{\boldsymbol{\ell}, T^1, \ldots, T^s} : [T] \times [T] \to \Re$, with $\max_{\tau \in [T]} K(x^\tau, x^\tau) \leq \hat{X}_K^2$, and $\hat{X}_P^2 := 2\lceil \log_2 T \rceil$.

Initialization: $\mathbb{U} \leftarrow \emptyset$, $\mathcal{X}^1 \leftarrow \emptyset$, $\mathcal{T}^1 \leftarrow \emptyset$.

For $\tau = 1, \ldots, T$

- Receive task $\ell^\tau \in [s]$.
- Receive $x^\tau \in \mathcal{X}$.
- Set $i \leftarrow \ell^\tau$; $t \leftarrow \boldsymbol{\sigma}(\tau)$; $x_t^i \equiv x^\tau$.
- Define

$$\boldsymbol{K}^\tau := (K(x,z))_{x,z \in \mathcal{X}^\tau \cup \{x^\tau\}} ; \quad \boldsymbol{P}^\tau := (\tilde{P}(\tau, \upsilon))_{\tau, \upsilon \in \mathcal{T}^\tau \cup \{\tau\}} ,$$

$$\tilde{\boldsymbol{X}}^\tau(\upsilon) := \left[\frac{\sqrt{\boldsymbol{K}^\tau} \boldsymbol{e}^{x^\upsilon}}{\sqrt{2\hat{X}_K^2}} ; \frac{\sqrt{\boldsymbol{P}^\tau} \boldsymbol{e}^\upsilon}{\sqrt{2\hat{X}_P^2}} \right] \left[\frac{\sqrt{\boldsymbol{K}^\tau} \boldsymbol{e}^{x^\upsilon}}{\sqrt{2\hat{X}_K^2}} ; \frac{\sqrt{\boldsymbol{P}^\tau} \boldsymbol{e}^\upsilon}{\sqrt{2\hat{X}_P^2}} \right]^\top ,$$

$$\tilde{\boldsymbol{W}}^\tau \leftarrow \exp\left(\log\left(\frac{\hat{C}}{2Tm} \right) \boldsymbol{I}^{|\mathcal{X}^\tau| + |\mathcal{T}^\tau| + 2} + \sum_{\upsilon \in \mathbb{U}} \eta y_\upsilon \tilde{\boldsymbol{X}}^\tau(\upsilon) \right) .$$

- Predict

$$Y^\tau \sim \textsc{Uniform}(-\gamma, \gamma) ; \quad \bar{y}^\tau \leftarrow \operatorname{tr}\left(\tilde{\boldsymbol{W}}^\tau \tilde{\boldsymbol{X}}^\tau \right) - 1 ; \quad \hat{y}_t^i := \hat{y}^\tau \leftarrow \operatorname{sign}(\bar{y}^\tau - Y^\tau) .$$

- Receive label $y_t^i := y^\tau \in \{-1, 1\}$.
- If $y^\tau \bar{y}^\tau \leq \frac{1}{\sqrt{m}}$ then

$$\mathbb{U} \leftarrow \mathbb{U} \cup \{t\} , \quad \mathcal{X}^{\tau+1} \leftarrow \mathcal{X}^\tau \cup \{x^\tau\}, \text{ and } \mathcal{T}^{\tau+1} \leftarrow \mathcal{T}^\tau \cup \{\tau\} .$$

- Else $\mathcal{X}^{\tau+1} \leftarrow \mathcal{X}^\tau$ and $\mathcal{T}^{\tau+1} \leftarrow \mathcal{T}^\tau$.

Comparing roughly to the bound of the exponential-time algorithm (see (5)), we see that the $\log m$ term has been replaced by an m term and we have gained a multiplicative factor of $\log 2T$. From the perspective of long-term memory, we note that the potentially dominant learner complexity term $\sum_{h \in m(\boldsymbol{h}^*)} \|h\|_K^2 X_K^2$ has only increased by a slight $\log 2T$ term. To gain more insight into the problem we also have the following simple lower bound.

Proposition 4. *For any (randomized) algorithm and any $s, k, m, \Gamma \in \mathbb{N}$, with $k + s \geq m > 1$ and $\Gamma \geq m \log_2 m$, there exists a kernel K and a $T_0 \in \mathbb{N}$ such that for every $T \geq T_0$:*

$$\sum_{\tau=1}^T \mathbb{E}[\mathcal{L}_{01}(y^\tau, \hat{y}^\tau)] - \mathcal{L}_{01}(y^\tau, h^\tau(x^\tau)) \in \Omega\left(\sqrt{(\Gamma + s \log m + k \log m) T} \right) ,$$

for some multitask sequence $(x^1, y^1), \ldots, (x^T, y^T) \in (\mathcal{X} \times \{-1, 1\})^T$ and some $\boldsymbol{h}^* \in [\mathcal{H}_K^{(\boldsymbol{x})}]^T$ such that $m \geq |m(\boldsymbol{h}^*)|$, $k \geq k(\boldsymbol{h}^*)$, $\sum_{h \in m(\boldsymbol{h}^*)} \|h\|_K^2 X_K^2 \geq |m(\boldsymbol{h}^*)| \log_2 m$, where $X_K^2 = \max_{\tau \in [T]} K(x^\tau, x^\tau)$.

Comparing the above proposition to the bound of the exponential-time algorithm (see (5)), the most striking difference is the absence of the $\log T$ terms. We conjecture that these terms are not necessary for the 0-1 loss. A proof of Theorem 3 and a proof sketch of Proposition 4 are given in Appendix B.

5 Discussion

We have presented a novel multitask setting which generalizes single-task switching under the long-term memory setting. We gave algorithms for finite hypothesis classes and for RKHS hypothesis classes with per trial prediction times of $\mathcal{O}(n)$ and $\mathcal{O}(T^3)$. We proved upper bounds on the regret for both cases as well as a lower bound in the RKHS case. An open problem is to resolve the gap in the RKHS case. On the algorithmic side, both algorithms depend on a number of parameters. There is extensive research in online learning methods to design parameter-free methods. Can some of these methods be applied here (see e.g., [44])? For a non-parametric hypothesis class, intuitively it

seems we must expect some time complexity dependence on T. However can we perhaps utilize decay methods such as [45, 46] or sketching methods [47] that have had success in simpler models to improve running times? More broadly, for what other infinite hypothesis classes can we give efficient regret-bounded algorithms in this switching multitask setting with long-term memory?

6 Acknowledgements

This research was supported by the U.S. Army Research Laboratory and the U.K. Ministry of Defence under Agreement Number W911NF-16-3-0001. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defence or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon. This research was further supported by the Engineering and Physical Sciences Research Council grant number EP/L015242/1.

Broader Impact

In general this work does not present any specific foreseeable societal consequence in the authors' joint opinion.

This is foundational research in *regret-bounded online learning*. As such it is not targeted towards any particular application area. Although this research may have societal impact for good or for ill in the future, we cannot foresee the shape and the extent.

Funding Transparency Statement

Funding

The authors were supported by the U.S. Army Research Laboratory and the U.K. Ministry of Defence under agreement number W911NF-16-3-0001 and by the Engineering and Physical Sciences Research Council grant number EP/L015242/1.

Competing Interests

The authors assert no competing interests.

References

[1] O. Bousquet and M.K. Warmuth. Tracking a small set of experts by mixing past posteriors. *Journal of Machine Learning Research*, 3:363–396, 2003.

[2] Volodimir G. Vovk. Aggregating strategies. In *Proceedings of the Third Annual Workshop on Computational Learning Theory*, COLT '90, pages 371–386, 1990.

[3] Nick Littlestone and Manfred K. Warmuth. The weighted majority algorithm. *Inf. Comput.*, 108(2):212–261, February 1994.

[4] Nicolo Cesa-Bianchi and Gabor Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, New York, NY, USA, 2006.

[5] N. Aronszajn. Theory of reproducing kernels. *Trans. Amer. Math. Soc.*, 68:337–404, 1950.

[6] Sanjeev Arora, Elad Hazan, and Satyen Kale. The multiplicative weights update method: a meta-algorithm and applications. *Theory of Computing*, 8(6):121–164, 2012.

[7] Nicolò Cesa-Bianchi, Yoav Freund, David Haussler, David P. Helmbold, Robert E. Schapire, and Manfred K. Warmuth. How to use expert advice. *J. ACM*, 44(3):427–485, May 1997.

[8] S. Shalev-Shwartz. Online Learning and Online Convex Optimization. *Foundations and Trends® in Machine Learning*, 4(2):107–194, 2011.

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

[10] Y. Freund. Private communication, 2000. Also posted on http://www.learning-theory.org.

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

[12] M. Herbster, S. Pasteris, and S. Pontil. Predicting a switching sequence of graph labelings. *Journal of Machine Learning Research*, 16:2003–2022, 2015.

[13] A. György, T. Linder, and G. Lugosi. Tracking the best of many experts. In *Proceedings 18th Annual Conference on Learning Theory*, pages 204–216, 2005.

[14] Wouter M. Koolen and Tim van Erven. Freezing and sleeping: Tracking experts that learn by evolving past posteriors, 2010.

[15] N. Cesa-Bianchi, P. Gaillard, G. Lugosi, and G. Stoltz. Mirror descent meets fixed share (and feels no regret). In *Advances in Neural Information Processing Systems 24*, pages 989–997, 2012.

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

[17] Wouter M. Koolen, Dmitry Adamskiy, and Manfred K. Warmuth. Putting bayes to sleep. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS 12, pages 135–143, Red Hook, NY, USA, 2012. Curran Associates Inc.

[18] D. Adamskiy, W. M. Koolen, A. Chernov, and V. Vovk. A closer look at adaptive regret. In *Proceedings of the 23rd International Conference on Algorithmic Learning Theory*, ALT'12, pages 290–304, 2012.

[19] A. Daniely, A. Gonen, and S. Shalev-Shwartz. Strongly adaptive online learning. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 1405–1411, 2015.

[20] Jaouad Mourtada and Odalric-Ambrym Maillard. Efficient tracking of a growing number of experts. In *Proceedings of the 28th International Conference on Algorithmic Learning Theory (ALT)*, volume 76 of *Proceedings of Machine Learning Research*, pages 517–539, 2017.

[21] Maria-Florina Balcan, Travis Dick, and Dravyansh Sharma. Online optimization of piecewise lipschitz functions in changing environments. *CoRR*, abs/1907.09137, 2019.

[22] Kai Zheng, Haipeng Luo, Ilias Diakonikolas, and Liwei Wang. Equipping experts/bandits with long-term memory. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d. Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 5929–5939. Curran Associates, Inc., 2019.

[23] Michael Mccloskey and Neil J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. *The Psychology of Learning and Motivation*, 24:104–169, 1989.

[24] Robert M. French. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3(4):128 – 135, 1999.

[25] J. Kivinen, A. J. Smola, and R. C. Williamson. Online learning with kernels. *IEEE Transactions on Signal Processing*, 52:2165–2176, 2004.

[26] N. Cesa-Bianchi and C. Gentile. Tracking the best hyperplane with a simple budget perceptron. In *Proceedings of the 18th Conference on Learning Theory*, pages 483–498, 2006.

[27] Jonathan Baxter. Learning internal representations. In *Proceedings of the Eighth Annual Conference on Computational Learning Theory*, COLT ?95, page 311?320, New York, NY, USA, 1995. Association for Computing Machinery.

[28] Rich Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, 1997.

[29] Theodoros Evgeniou and Massimiliano Pontil. Regularized multi–task learning. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 109–117, New York, NY, USA, 2004. Association for Computing Machinery.

[30] J. Abernethy, P. Bartlett, and A. Rakhlin. Multitask learning with expert advice. In *Proceedings 20th Annual Conference on Learning Theory*, pages 484–498, 2007.

[31] Alekh Agarwal, Alexander Rakhlin, and Peter Bartlett. Matrix regularization techniques for online multitask learning. Technical Report UCB/EECS-2008-138, EECS Department, University of California, Berkeley, Oct 2008.

[32] S. Avishek, R. Piyush, H. Daumé III, and S. Venkatasubramanian. Online learning of multiple tasks and their relationships. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*, pages 643–651, 2011.

[33] Alexander Rakhlin, Jacob D. Abernethy, and Peter L. Bartlett. Online discovery of similarity mappings. In Zoubin Ghahramani, editor, *Machine Learning, Proceedings of the Twenty-Fourth International Conference (ICML 2007), Corvallis, Oregon, USA, June 20-24, 2007*, volume 227 of *ACM International Conference Proceeding Series*, pages 767–774. ACM, 2007.

[34] Gábor Lugosi, Omiros Papaspiliopoulos, and Gilles Stoltz. Online multi-task learning with hard constraints. In *COLT 2009 - The 22nd Conference on Learning Theory, Montreal, Quebec, Canada, June 18-21, 2009*, 2009.

[35] O. Dekel, P.M. Long, and Y. Singer. Online learning of multiple tasks with a shared loss. *Journal of Machine Learning Research*, 8(10):2233–2264, 2007.

[36] G. Cavallanti, N. Cesa-Bianchi, and C. Gentile. Linear algorithms for online multitask classification. *Journal of Machine Learning Research*, 1:2901–2934, 2010.

[37] Christoph Hirnschall, Adish Singla, Sebastian Tschiatschek, and Andreas Krause. Coordinated online learning with applications to learning user preferences, 2017.

[38] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci.*, 55(1):119?139, August 1997.

[39] Mark Herbster, Stephen Pasteris, and Lisa Tse. Online matrix completion with side information. In *Advances in Neural Information Processing Systems 33*. 2020.

[40] K. Tsuda, G. Rätsch, and M.K. Warmuth. Matrix exponentiated gradient updates for on-line learning and bregman projection. *Journal of Machine Learning Research*, 6:995–1018, 2005.

[41] N. Linial, S. Mendelson, G. Schechtman, and A. Shraibman. Complexity measures of sign matrices. *Combinatorica*, 27(4):439–463, 2007.

[42] Jason D Lee, Ben Recht, Nathan Srebro, Joel Tropp, and Russ R Salakhutdinov. Practical large-scale optimization for max-norm regularization. In J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 1297–1305. Curran Associates, Inc., 2010.

[43] M. Herbster, G. Lever, and M. Pontil. Online prediction on large diameter graphs. In *Advances in Neural Information Processing Systems 21*, pages 649–656, 2008.

[44] Francesco Orabona and Dávid Pál. Coin betting and parameter-free online learning. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS16, pages 577–585, Red Hook, NY, USA, 2016. Curran Associates Inc.

[45] Ofer Dekel, Shai Shalev-shwartz, and Yoram Singer. The forgetron: A kernel-based perceptron on a fixed budget. In Y. Weiss, B. Schölkopf, and J. C. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 259–266. MIT Press, 2006.

[46] Giovanni Cavallanti, Nicolò Cesa-Bianchi, and Claudio Gentile. Tracking the best hyperplane with a simple budget perceptron. *Mach. Learn.*, 69(2-3):143–167, 2007.

[47] Yair Carmon, John C. Duchi, Aaron Sidford, and Kevin Tian. A rank-1 sketch for matrix multiplicative weights. In Alina Beygelzimer and Daniel Hsu, editors, *Conference on Learning Theory, COLT 2019, 25-28 June 2019, Phoenix, AZ, USA*, volume 99 of *Proceedings of Machine Learning Research*, pages 589–623. PMLR, 2019.

[48] Yoav Freund, Robert E. Schapire, Yoram Singer, and Manfred K. Warmuth. Using and combining predictors that specialize. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, STOC 97, pages 334–343, New York, NY, USA, 1997. Association for Computing Machinery.

[49] M. Herbster and M. Pontil. Prediction on a graph with a perceptron. In *Advances in Neural Information Processing Systems 19*, pages 577–584, 2006.

[50] Douglas Klein and Milan Randic. Resistance distance. *Journal of Mathematical Chemistry*, 12:81–95, 12 1993.

[51] M. Herbster, M. Pontil, and S. Rojas-Galeano. Fast prediction on a tree. In *Advances in Neural Information Processing Systems*, pages 657–664, 2009.

[52] A.B. Novikoff. On convergence proofs on perceptrons. In *Proceedings of the Symposium on the Mathematical Theory of Automata*, pages 615–622, 1962.

[53] J. Forster, N. Schmitt, and H.U. Simon. Estimating the optimal margins of embeddings in euclidean half spaces. In *Proceedings Computational Learning Theory*, pages 402–415, 2001.

[54] N. Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2:285–318, April 1988.

[55] Shai Ben-David, Dávid Pál, and Shai Shalev-Shwartz. Agnostic online learning. In *COLT 2009 - The 22nd Conference on Learning Theory, Montreal, Quebec, Canada, June 18-21, 2009*, 2009.

[56] Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, 2000.

[57] Martin Zinkevich. Online Convex Programming and Generalized Infinitesimal Gradient Ascent. In *Proceedings, Twentieth International Conference on Machine Learning*, volume 2, pages 928–935, 2003.

[58] N. Cesa-Bianchi, P. M. Long, and M. K. Warmuth. Worst-case quadratic loss bounds for on-line prediction of linear functions by gradient descent. *IEEE Transactions on Neural Networks*, 7(3):604–619, 1996. Earlier version in 6th COLT, 1993.