Online Multi-task Learning with Hard Constraints

Gábor Lugosi[*]
ICREA and Universitat Pompeu Fabra
Barcelona, Spain
lugosi@upf.es

Omiros Papaspiliopoulos[†]
Universitat Pompeu Fabra
Barcelona, Spain
omiros.papaspiliopoulos@upf.edu

Gilles Stoltz[‡]
Ecole Normale Supérieure, CNRS
Paris, France
HEC Paris, CNRS,
Jouy-en-Josas, France
gilles.stoltz@ens.fr

Abstract

We discuss multi-task online learning when a decision maker has to deal simultaneously with M tasks. The tasks are related, which is modeled by imposing that the M–tuple of actions taken by the decision maker needs to satisfy certain constraints. We give natural examples of such restrictions and then discuss a general class of tractable constraints, for which we introduce computationally efficient ways of selecting actions, essentially by reducing to an on-line shortest path problem. We briefly discuss "tracking" and "bandit" versions of the problem and extend the model in various ways, including non-additive global losses and uncountably infinite sets of tasks.

1 Introduction

Multi-task learning has recently received considerable attention, see [DLS07, ABR07, Men07, CCBG08]. In multi-task learning problems, one simultaneously learns several tasks that are related in some sense. The relationship of the tasks has been modeled in different ways in the literature. In our setting, a decision maker chooses an action simultaneously for each of M given tasks, in a repeated manner. (To each of these tasks corresponds a game, and we will use interchangeably the concepts of game and task.) The relatedness is accounted for by putting some hard constraints on these simultaneous actions.

As a motivating example, consider a distance-selling company that designs several commercial offers for its numerous customers, and the customers are ordered (say) by age. The company has to choose whom to send which offer. A loss of earnings is suffered whenever a customer does not receive the commercial offer that would have been best for him. Basic marketing considerations suggest that offers given to customers with similar age should not be very different, so the company selects a batch of offers that satisfy such a constraint. Additional budget constraint may limit further the set of batches from which the company may select. After the offers are sent out, the customers' responses are observed (at least partially) and new offers are selected and sent. We model such situations by playing many repeated games simultaneously with the restriction that the vector of actions that can be selected at a time needs to belong to a previously given set. This set in determined beforehand by the budget and marketing constraints discussed above. The goal of the decision maker is to minimize the total accumulated regret (across the many games and through time), that is, perform, on the long run, almost as well as the best constant vector of actions satisfying the constraint.

The problem of playing repeatedly several games simultaneously has been considered by [Men07] who studies convergence to Nash equilibria but does not address the issue of computational feasibility when a large number of games is played. On-line multi-task learning problems were also studied by [ABR07] and [DLS07]. As the latter reference, we consider minimizing regret simultaneously in parallel, by enforcing however some hard constraints. As [ABR07], we measure the total loss as the sum of the losses suffered in each game but assume that all tasks have to be performed at each round. (This assumption is, however relaxed in Section 8, where we consider global losses more general than the sums of losses.) The main additional difficulty we face is the requirement that the decision maker chooses from a restricted subset of vectors of actions. In previous models restrictions were only considered on the comparison class, but not on the way the decision maker plays.

We formulate the problem in the framework of on-line regret minimization, see [CBL06] for a survey. The main challenge is to construct a strategy for playing the many games simultaneously with small regret such that the strategy has a manageable computational complexity. We show that in various natural examples the computational problem may be reduced to an online shortest path problem in an associated graph for which well-known efficient algorithms exist. (We however propose a specific scheme for implementation that is slightly more effective.)

The results can be extended easily to the "tracking" case in which the goal of the decision maker is to perform as

[*]Supported by the Spanish Ministry of Science and Technology grant MTM2006-05650 and by the PASCAL Network of Excellence under EC grant no. 506778

[†]Supported by the Spanish Ministry of Science and Technology under grant MTM2008-06660 and a "Ramon y Cajal" fellowship

[‡]Supported by the French National Research Agency (ANR) under grants JCJC06-137444 "From applications to theory in learning and adaptive statistics" and 08-COSI-004 "Exploration–exploitation for efficient resource allocation", and by the PASCAL Network of Excellence under EC grant no. 506778

well as the best strategy that can change the vector of actions (taken from the restricted set) at a limited number of times. We also consider the "bandit" version of the problem when the decision maker, instead of observing the losses of all actions in all games, only learns the sum of the losses of the chosen actions.

Finally, we also consider cases when there are infinitely many tasks, indexed by real numbers. In such cases the decision maker chooses a function from a certain restricted class of functions. We show examples that are natural extensions of the cases we consider for finitely many tasks and discuss the computational issues that are closely related to the theory of exact simulation of continuous-time Markov chains.

We concentrate on exponentially weighted average forecasters because, when compared to its most likely competitors, that is, follow-the-leader-type algorithms, they have better performance guarantees, especially in the case of bandit feedback. Besides, the two families of forecasters, as pointed out by [ABR07], usually have implementation complexities of the same order.

2 Setup and notation

In the simplest model studied in this paper, a decision maker deals simultaneously with M tasks, indexed by $j = 1, \ldots, M$. For simplicity, we assume that all games share the same finite action space $\mathcal{X} = \{x_1, \ldots, x_N\} \subset \mathbb{R}$. (Here, we do not identify actions with integers but with real numbers, for reasons that will be clear in Section 3.)

To each tasks $j = 1, \ldots, M$ there is an associated outcome space $\mathcal{Y}_j$ and a loss function $\ell^{(j)} : \mathcal{X} \times \mathcal{Y}_j \to [0, 1]$. We denote by $\mathbf{x} = \left(x_{k_1}, \ldots, x_{k_M}\right)$ the elements of $\mathcal{X}^M$ and call them vectors of simultaneous actions. Here and in the sequel, the indices k_j belong to $\{1, \ldots, N\}$. The tasks are played repeatedly and at each round $t = 1, 2, \ldots$, the decision maker chooses a vector $\mathbf{X}_t = (X_{1,t} \ldots, X_{M,t}) \in \mathcal{X}^M$ of simultaneous actions. (That is, he chooses indices $K_{1,t}, \ldots, K_{M,t} \in \{1, \ldots, N\}$ and $X_{j,t} = x_{K_{j,t}}$ for all $j = 1, \ldots, M$.) We assume that the choice of $\mathbf{X}_t$ can be made at random, according to a probability distribution over $\mathcal{X}^M$ which will usually be denoted by $\mathbf{p}_t$. The behavior of the opponent player among all tasks is described by the vector of outcomes $\mathbf{y}_t = (y_{1,t}, \ldots, y_{M,t})$.

We are interested in the loss suffered by the decision maker and we do not assume any specific probabilistic or strategic behavior of the environment. In fact, the outcome vectors $\mathbf{y}_t$, for $t = 1, 2, \ldots$, can be completely arbitrary and we measure the performance of the decision maker by comparing it to the best of a class of reference strategies. The total loss suffered by the decision maker at time t is just the sum of the losses over tasks:

$$\ell(\mathbf{X}_t, \mathbf{y}_t) = \sum_{j=1}^{M} \ell^{(j)}(X_{j,t}, y_{j,t}) \, .$$

The important point is that the decision maker has some restrictions to be obeyed in each round, which we also call hard constraints. They are modeled by a subset $\mathcal{A}$ of the set of possible simultaneous actions $\mathcal{X}^M$; the forecaster is only allowed to play vectors $\mathbf{X}_t$ in $\mathcal{A}$. This subset $\mathcal{A}$ captures the relatedness among the tasks.

The decision maker aims at minimizing his regret, defined by the difference of his cumulative loss with respect to the cumulative loss of the best constant vector of actions, determined in hindsight, among the set of allowed vectors $\mathcal{A}$. Formally, the regret is defined by

$$R_n = \sum_{t=1}^{n} \ell(\mathbf{X}_t, \mathbf{y}_t) - \min_{\mathbf{x} \in \mathcal{A}} \sum_{t=1}^{n} \ell(\mathbf{x}, \mathbf{y}_t) \, .$$

In the basic, *full information*, version of the problem the decision maker, after choosing $\mathbf{X}_t$, observes the vector of outcomes $\mathbf{y}_t$. In the *bandit* setting, only the total loss $\ell(\mathbf{X}_t, \mathbf{y}_t)$ becomes available to the decision maker.

Observe that in the case of $M = 1$ task, the problem reduces to the well-studied problem of "on-line prediction with expert advice" or "sequential regret minimization," see [CBL06] for the history and basic results. This is also the case when $M \geqslant 2$ but $\mathcal{A} = \mathcal{X}^M$, since the decision maker could then treat each task independently from others and maintain M parallel forecasting schemes, at least in the full-information setting. Under the bandit assumption the problem becomes the "multi-task bandit problem" discussed in [CBL09], which is also easy to solve by available techniques. However, when $\mathcal{A}$ is a proper subset of $\mathcal{X}^M$, interesting computational problems arise. The efficient implementation we propose requires a condition the set $\mathcal{A}$ of restrictions needs to satisfy. This structural condition, satisfied in several natural examples discussed below, permits us to reduce the problem to the well-studied problem of predicting as well as the best path between two fixed vertices of a graph.

In order to make the model meaningful, just like in the most basic versions of the problem, we allow the decision maker to randomize its decision in each period. More formally, at each round of the repeated game, the decision maker determines a distribution on $\mathcal{X}^M$ (restricted to the set $\mathcal{A}$) and draws the action vector $\mathbf{X}_t$ according to this distribution. Before determining the outcomes, the opponent may have access to the probability distribution the decision maker uses but not to the realizations of the random variables.

Structure of the paper

We start by stating some natural examples on which the proposed techniques will be illustrated. We then study the full-information version of the problem (when the decision maker observes all past outcomes before determining his probability distribution) by proposing first a hypothetical scheme with good performance and then stating an efficient implementation of it.

We also consider various extensions. One of them is the bandit setting, when only the sum of losses of the chosen simultaneous actions are observed. Another extension is the "tracking" problem when, instead of competing with the best constant vector of actions, the decision maker intends to perform as well as the best strategy that is allowed to switch a certain limited number of times (but always satisfying the restrictions). We also consider alternative global loss functions that do not necessarily sum the losses over the tasks. Finally, we describe a setting in which there are infinitely many tasks indexed by an interval. This is a natural extension of the main examples we work with and the algorithmic problem

has some interesting connections with exact simulation of continuous-time discrete Markov chains.

3 Motivating examples

We start by describing four examples that we will be able to handle with the proposed machinery. The examples are defined by their corresponding sets $\mathcal{A} \subset \mathcal{X}^M$ of permitted simultaneous actions.

Example 1 (Internal coherence) Assume that tasks are linearly ordered and any two consecutive tasks, though different, share some similarity. Therefore, it is a natural requirement that the actions taken in two consecutive games be not too far away from each other. One may also interpret this as a matter of internal coherence of the decision maker. To model this, we assume that the actions are ranked in the action set $\mathcal{X}$ according to some logic and impose some maximal dissimilarity $\gamma > 0$ between the actions of two consecutive tasks, that is,

$$\mathcal{A} = \left\{ \left(x_{k_1}, \ldots, x_{k_M}\right) : \ \forall j \leqslant M-1, \ \left|x_{k_j} - x_{k_{j+1}}\right| \leqslant \gamma \right\} .$$

Example 2 (Escalation constraint) Once again we assume that the tasks are linearly ordered and the actions are ranked. Imagine that tasks correspond to consumers and that the higher the index of the task, the more favorable the conditions for the consumer (and the higher the loss of earnings of the seller, who is the decision maker). The constraint decision maker has to satisfy is that higher-ranked costumers need to receive better conditions, at least within the same round of play. That is, the simultaneous actions must form a non-decreasing sequence in the following sense,

$$\mathcal{A} = \left\{ \left(x_{k_1}, \ldots, x_{k_M}\right) : \ \forall j \leqslant M-1, \ x_{k_j} \leqslant x_{k_{j+1}} \right\} .$$

Example 3 (Constancy constraint) Assume that tasks are ordered and that the decision maker should not vary its action too often. This is measured by the fact that the decision maker must stick to an action for several consecutive tasks and that he can shift to a new action only at a limited number m of tasks, which we model by

$$\mathcal{A} = \left\{ \left(x_{k_1}, \ldots, x_{k_M}\right) : \ \sum_{j=1}^{M-1} \mathbb{I}_{\{k_j \neq k_{j+1}\}} \leqslant m \right\} .$$

Example 4 (Budget constraint) Here we assume that the number x_{k_j} associated with action k in task j represents the cost of choosing this action. The freedom of the decision maker is limited by a budget constraint. For example, one may face a situation when the decision maker has a constant budget B to be used at each round, that is,

$$\mathcal{A} = \left\{ \left(x_{k_1}, \ldots, x_{k_M}\right) : \ \sum_{j=1}^{M} x_{k_j} \leqslant B \right\} .$$

To make things more concrete, we assume, in this example only, that $x_k = k$. One should then take for B as an integer between M and NM. For smaller values $\mathcal{A}$ becomes empty and for larger values $\mathcal{A} = \mathcal{X}^M$.

4 Exponentially weighted averages

By considering each element of $\mathcal{A}$ as a (meta-)expert, we can reduce the problem to the usual single-task setting and exhibit a forecaster with a good performance bound that, in its straightforward implementation, has a computational cost proportional to the cardinality of $\mathcal{A}$.

More precisely, for each round $n \geqslant 1$, we denote by

$$L_n(\mathbf{x}) = \sum_{t=1}^{n} \ell(\mathbf{x}, \mathbf{y}_t)$$

the cumulative loss of the simultaneous actions $\mathbf{x} \in \mathcal{X}$, and define an instance of the exponentially weighted average forecaster on these cumulative losses. That is, at round $t = 1$, the decision maker draws an element $\mathbf{X}_1$ uniformly at random in $\mathcal{A}$ and for each round $t \geqslant 2$, draws $\mathbf{X}_t$ at random according to the distribution $\mathbf{p}_t$ on $\mathcal{A}$ which puts the following mass on each $\mathbf{x} \in \mathcal{A}$,

$$\mathbf{p}_t(\mathbf{x}) = \frac{\exp\bigl(-\eta L_{t-1}(\mathbf{x})\bigr)}{\sum_{\mathbf{a}\in\mathcal{A}} \exp\bigl(-\eta L_{t-1}(\mathbf{a})\bigr)} , \tag{1}$$

where $\eta > 0$ is a parameter to be tuned. The bound follows from a direct application of well-known results, see, for instance, [CBL06, Corollary 4.2].

Proposition 1 *For all $n \geqslant 1$, the above instance of the exponentially weighted average forecaster, when run with $\eta = (1/M)\sqrt{8(\ln N)/n}$, ensures that for all $\delta > 0$, its regret is bounded, with probability at most $1 - \delta$, as*

$$R_n \leqslant M \left(\sqrt{\frac{n \ln |\mathcal{A}|}{2}} + \sqrt{\frac{n}{2} \ln \frac{1}{\delta}} \right)$$

where $|\mathcal{A}|$ denotes the cardinality of $\mathcal{A}$.

The computational complexity of this forecaster, in its naive implementation, is proportional to $|\mathcal{A}|$, which is prohibitive in all examples of Section 3 since the cardinality of $\mathcal{A}$ is exponentially large. For example, in Example 1, if we denote by

$$\rho = \min\left\{ \left| \left\{ x' \in \mathcal{X} : \ |x - x'| \leqslant \gamma \right\} \right| : \ x \in \mathcal{X} \right\}$$

a common lower bound on the number of γ–close actions to any action in $\mathcal{X}$, then

$$|\mathcal{A}| \geqslant N \rho^{M-1} .$$

In Example 2, by first choosing the m actions to be used (in increasing order) and the $m-1$ corresponding shift points, one gets

$$\begin{aligned} |\mathcal{A}| &= \sum_{m=1}^{N} \binom{N}{m} \binom{M+m-1}{m-1} \\ &\geqslant \sum_{m=1}^{N} \binom{N}{m} \frac{M^{m-1}}{(m-1)!} \geqslant \frac{(M+1)^N}{(N-1)!} . \end{aligned}$$

In the case of at most m shifts in the simultaneous actions, discussed in Example 3, we have

$$|\mathcal{A}| \geqslant \binom{M+m}{m} N(N-1)^m$$

(where the lower bound is obtained by considering only the simultaneous actions with exactly m shifts). That is, $|\mathcal{A}|$ is of the order of $(MN)^m/m!$. Finally, with the budget constraint of Example 4, the typical size of $\mathcal{A}$ is exponential in M, as $|\mathcal{A}| \geqslant \rho^M$ where $\rho = \lfloor B/M \rfloor$ is the lower integer part of B/M (recall that the minimal expense to complete each task is 1).

5 Efficient implementation with online shortest path

In this section we show how the computational problem of drawing a random vector of actions $\mathbf{X}_t \in \mathcal{A}$ according to the exponentially weighted average distribution can be reduced to the well-studied online shortest path problem. Recall that in the online shortest path problem (see, e.g., [TW04, GLL04, GLL05]) the decision maker selects, at each round of the game, a path between two given vertices (the *source* and the *sink*) in a given graph. A loss is assigned to each edge of the graph in every round of the game and the loss of a path is the sum of the losses of the edges. A path can be selected according to the exponentially weighted average distribution in a computationally efficient way by a dynamic programming-type algorithm, see [TW04] or [CBL06, Section 5.4]. The algorithm has complexity $O(|\mathcal{E}|)$ where $\mathcal{E}$ is the set of edges of the graph.

We first explain how the problem of drawing a joint action in the multi-task problem can be reduced to an online shortest path problem in all the examples presented above and then indicate how to efficiently sample from the distribution $\mathbf{p}_t$ defined in (1).

5.1 A Markovian description of the constraints

In order to define the corresponding graph in which the online shortest path problem is equivalent with our hard-constrained multi-task problem, we introduce a set $\mathcal{S}$ of hidden *states*. The value of the hidden state controls that the hard constraints are satisfied along the sequence of simultaneous actions. To this end, denote by S the state function, which, given a vector of actions (of length $\leqslant M$), outputs the corresponding state in $\mathcal{S}$.

We also consider an additional state $\star$ meaning that the hard constraint is not satisfied. We denote $\mathcal{S}^\star = \mathcal{S} \cup \{\star\}$. By definition,

$$\mathcal{A} = \left\{ \mathbf{x} \in \mathcal{X}^M : \, S(\mathbf{x}) \neq \star \right\} .$$

To make things more concrete we now describe $\mathcal{S}$ and S on all four examples introduced in Section 3.

The first two examples are the simplest as all the information is contained in the current action; their hidden state space $\mathcal{S}$ is reduced to a single state OK. For Example 1, for all sequences $\left(x_{k_1}, \ldots, x_{k_j}\right)$ of length $1 \leqslant j \leqslant M$, one defines

$$S\Big(\left(x_{k_1}, \ldots, x_{k_j}\right)\Big) = \begin{cases} \text{OK} & \text{if for all } i \leqslant j-1, \;\; \left|x_{k_i} - x_{k_{i+1}}\right| \leqslant \gamma, \\ \star & \text{otherwise,} \end{cases}$$

whereas for Example 2,

$$S\Big(\left(x_{k_1}, \ldots, x_{k_j}\right)\Big) = \begin{cases} \text{OK} & \text{if for all } i \leqslant j-1, \;\; x_{k_i} \leqslant x_{k_{i+1}}, \\ \star & \text{otherwise.} \end{cases}$$

In Example 3 the underlying hidden state counts the number of shifts seen so far in the sequence of actions, so $\mathcal{S} = \{0, \ldots, m\}$ and for all sequences $\left(x_{k_1}, \ldots, x_{k_j}\right)$ of length less or equal to M, we first define

$$S'\Big(\left(x_{k_1}, \ldots, x_{k_j}\right)\Big) = \sum_{j=1}^{M-1} \mathbb{I}_{\{k_j \neq k_{j+1}\}}$$

and then $S\Big(\left(x_{k_1}, \ldots, x_{k_j}\right)\Big)$

$$= \begin{cases} S'\Big(\left(x_{k_1}, \ldots, x_{k_j}\right)\Big) & \text{if } S'\Big(\left(x_{k_1}, \ldots, x_{k_j}\right)\Big) \leqslant m, \\ \star & \text{otherwise.} \end{cases}$$

Finally, in Example 4, the hidden state monitors the budget spent so far, that is, $\mathcal{S} = \{0, \ldots, B\}$,

$$S'\Big(\left(x_{k_1}, \ldots, x_{k_j}\right)\Big) = \sum_{i=1}^{j} x_{k_i} \, ,$$

and $S\Big(\left(x_{k_1}, \ldots, x_{k_j}\right)\Big)$

$$= \begin{cases} S'\Big(\left(x_{k_1}, \ldots, x_{k_j}\right)\Big) & \text{if } S'\Big(\left(x_{k_1}, \ldots, x_{k_j}\right)\Big) \leqslant B, \\ \star & \text{otherwise.} \end{cases}$$

In view of these examples, the following assumption on S is natural.

Assumption 1 *The state function is Markovian in the following sense. For all $j \geqslant 2$ and all vectors $\left(x_{k_1}, \ldots, x_{k_j}\right)$, the state $S\Big(\left(x_{k_1}, \ldots, x_{k_j}\right)\Big)$ only depends on the value of x_{k_j} and on the state $S\Big(\left(x_{k_1}, \ldots, x_{k_{j-1}}\right)\Big)$.*

Remark 1 For all problems, there exists always a state space $\mathcal{S}$ and a state function S such that the previous assumptions holds. However, the more complex the dependence on the past, the larger $\mathcal{S}$ is. As the algorithmic complexity of the proposed forecaster is at least proportional to the cardinality of $\mathcal{S}$ (see Section 5.3.1), this shows that the only interesting cases are those for which $\mathcal{S}$ is small.

We further assume that there exists a transition function T that, to each pair (x, s) (corresponding to some task j) formed by an action $x \in \mathcal{X}$ and a hidden state $s \in \mathcal{S}^\star$, associates pairs $(x', s') \in \mathcal{X} \times \mathcal{S}$ (to be used in task $j+1$). Put differently, $T\big((x,s)\big)$ is a subset of $\mathcal{X} \times \mathcal{S}^\star$ that indicates all legal transitions. We impose that when the prefix of a sequence is already in the dead end state $s = \star$, the whole sequence stays in $\star$, that is, for all $x \in \mathcal{X}$,

$$T\big((x, \star)\big) = \mathcal{X} \times \{\star\} \, .$$

Once again, to make things more concrete, we describe T for the four examples introduced in Section 3.

Example 1 relies on $\mathcal{S} = \{\text{OK}\}$ and the transitions

$$T\big((x, \text{OK})\big) = \big(\mathcal{X} \cap [x - \gamma,\, x + \gamma]\big) \times \{\text{OK}\}$$

for all $x \in \mathcal{X}$. Example 2 can be modeled with $\mathcal{S} = \{\text{OK}\}$ and the transitions

$$T\big((x, \text{OK})\big) = [x, x_N] \times \{\text{OK}\}\ .$$

for all $x \in \mathcal{X}$.

For Example 3, the transition function is given by

$$T\big((x, s)\big) = \{(x, s)\} \cup \Big(\big(\mathcal{X} \setminus \{x\}\big) \times \{s + 1\}\Big)$$

for all $s = 0, \dots, m - 1$ and

$$T\big((x, m)\big) = \{(x, m)\} \cup \Big(\big(\mathcal{X} \setminus \{x\}\big) \times \{\star\}\Big)\ .$$

Finally, the one of Example 4 is given by

$$T\big((x, s)\big) = \begin{cases} \mathcal{X} \times \{s + x\} & \text{if } s + x \leqslant B, \\ \mathcal{X} \times \{\star\} & \text{if } s + x > B. \end{cases}$$

5.2 Reduction to an online shortest path problem

We are now ready to describe the graph by which a constrained multi-task problem can be reduced to an online shortest path problem. Assume that $\mathcal{A}$ is such that there is a corresponding state space $\mathcal{S}$, a state function S satisfying Assumption 1, and a transition function T. Assumption 1 is needed below to construct a graph with few edges.

We define the cumulative losses $L_n^{(j)}$ suffered in each task $j = 1, \dots, M$ between rounds $t = 1$ and n as follows. For all $x \in \mathcal{X}$,

$$L_n^{(j)}(x) = \sum_{t=1}^{n} \ell^{(j)}(x, y_{j,t})\ .$$

Of course, with the notation above, for all $n \geqslant 1$ and all $\mathbf{x} = \big(x_{k_1}, \dots, x_{k_j}\big)$,

$$L_n(\mathbf{x}) = \sum_{j=1}^{M} L_n^{(j)}\big(x_{k_j}\big)\ .$$

In the sequel, we extend the notation by convention to $n = 0$, by $L_0 \equiv 0$ and $L_0^{(j)} \equiv 0$ for all j.

Then, for each round $t = 1, \dots, n$, we define a directed acyclic graph with at most $MN|\mathcal{S}|$ vertices. Each vertex corresponds to task-action-state triple (j, x_k, s), where $j = 1, \dots, M$, $k = 1, \dots, N$, and $s \in \mathcal{S}$. Two vertices $v = (j, x_k, s)$ and $v' = (j', x_{k'}, s')$ are connected with a directed edge if and only if $j' = j + 1$, and $(x_{k'}, s') \in T(x_k, s)$, that is, $(x_k, s) \to (x_{k'}, s')$ is a legal transition between tasks j and $j + 1$. The loss associated with such an edge equals $L_{t-1}^{(j')}(x_{k'})$, the cumulative loss of action $x_{k'}$ in task j' in the previous time rounds. We also add two vertices, the "source" node u_0 and the "sink" u_1 as follows. There is a directed edge between u_0 and every vertex of the form $(1, x_k, s)$ with $k = 1, \dots, N$ and $s \neq \star$. Its associated losses equal $L_{t-1}^{(1)}(x_k)$. Finally, every vertex of the form (M, x_k, s) with $k = 1, \dots, N$ and $s \neq \star$ is connected to the sink u_1 with edge loss 0.

In the graph defined above, choosing a path between the source and the sink is equivalent to choosing a legal M–tuple of actions in the multi-task problem. (Note that there is no path between u_0 and u_1 containing a vertex with $s = \star$.) The sum of the losses over the edges of a path is just the cumulative loss of the corresponding M–tuple of actions. Generating a legal random M–tuple according to the exponentially weighted average distribution is thus equivalent to generating a random path in this graph according to the exponentially weighted average distribution. This can be done with a computational complexity of the order of the number of edges defined above, see, e.g., [CBL06, Section 5.4]. In our case, since edges only connect two consecutive tasks, the number of edges is at most $1 + MN^2|\mathcal{S}|^2$. In Section 5.3.1 we discuss the number of edges and the related complexity on the examples of Section 3.

Since edges only exist between consecutive tasks, the above implementation by reduction to an online shortest path problem takes a simple form, which we detail below for concreteness. It will be useful to have it for Section 8.2.

5.3 Brief recall of the way the efficient implementation goes

In order to generate a random M–tuple of actions according to the distribution $\mathbf{p}_t$, we first rewrite the probability distribution $\mathbf{p}_t$ in terms of the state function S and the cumulative losses $L_{t-1}^{(j)}$ suffered in each task j. To do so, we denote by $\delta_{\mathbf{x}}$ the Dirac mass on $\mathbf{x} = \big(x_{k_1}, \dots, x_{k_M}\big)$, that is, the probability distribution over $\mathcal{X}$ that puts all probability mass on $\mathbf{x}$. The definition (1) then rewrites as

$$\mathbf{p}_t = \sum_{\mathbf{x} \in \mathcal{X}^M} \frac{\mathbb{I}_{\{S(\mathbf{x}) \neq \star\}} \exp\Big(-\eta \sum_{j=1}^{M} L_{t-1}^{(j)}\big(x_{k_j}\big)\Big)}{\sum_{\mathbf{a} \in \mathcal{X}^M} \mathbb{I}_{\{S(\mathbf{a}) \neq \star\}} \exp\Big(-\eta \sum_{j=1}^{M} L_{t-1}^{(j)}\big(a_{k_j}\big)\Big)}\, \delta_{\mathbf{x}}\ . \tag{2}$$

Before proceeding with the random generation of vectors $\mathbf{X}_t$ according to $\mathbf{p}_t$, we introduce an auxiliary sequence of weights and explain how to maintain it. For all rounds $t \geqslant 0$, tasks $j \in \{1, \dots, M\}$, actions $x \in \mathcal{X}$, and states $s \in \mathcal{S}$, we define

$$w_{t,j,x,s} = \sum_{x_{k_1}, \dots, x_{k_{j-1}} \in \mathcal{X}} \exp\left(-\eta \left(L_t^{(j)}(x) + \sum_{i=1}^{j-1} L_t^{(i)}\big(x_{k_i}\big)\right)\right) \times \mathbb{I}_{\left\{S(i_{k_1}, \dots, i_{k_{j-1}}, x) = s\right\}}\ .$$

Note that we do not consider the state $\star$ here.

Now, for all rounds $t \geqslant 0$, actions $x \in \mathcal{X}$, and states $s \in \mathcal{S}$, one simply has

$$w_{t,1,x,s} = \exp\Big(-\eta L_t^{(1)}(x)\Big)\ \mathbb{I}_{\{S(x) = s\}}\ .$$

Then, an induction (on j) using Assumption 1 shows that for all $1 \leqslant j \leqslant M - 1$, actions $x' \in \mathcal{X}$, and states $s' \in \mathcal{S}$,

$$w_{t,j+1,x',s'} = \sum_{x \in \mathcal{X},\, s \in \mathcal{S}} w_{t,j,x,s}\, \mathbb{I}_{\{(x',s') \in T((x,s))\}} \exp\Big(-\eta L_t^{(j+1)}(x')\Big)\ . \tag{3}$$

We now show how to use these weights to sample from the desired distribution $\mathbf{p}_t$, for $t \geqslant 1$. We proceed in a backwards manner, drawing first $X_{M,t}$, then, conditionally to the value of $X_{M,t}$, generating $X_{M-1,t}$, and so on, till $X_{1,t}$.

To draw $X_{M,t}$, we note that equation (2) shows that the M–th marginal induced by $\mathbf{p}_t$ is the distribution over $\mathcal{X}$ that puts a probability mass proportional to

$$\sum_{s \in \mathcal{S}} w_{t-1,M,x,s}$$

on each action $x \in \mathcal{X}$. It is therefore easy to generate a random element $X_{M,t}$ with the appropriate distribution. We actually need to draw a pair $(X_{M,t}, S_{M,t}) \in \mathcal{X} \times \mathcal{S}$ distributed according to the distribution on $\mathcal{X} \times \mathcal{S}$ proportional to the $w_{t-1,M,k,s}$.

We then aim at drawing the actions (and hidden states) corresponding to the previous tasks according to the (conditional) distribution $\mathbf{p}_t\big(\,\cdot \,|\, X_{M,t}, S_{M,t}\big)$ on $(\mathcal{X} \times \mathcal{S})^{M-1}$. Again by using the Markovian assumption on S, it turns out that the $(M-1)$–th marginal of this distribution on $\mathcal{X} \times \mathcal{S}$ is proportional, for all pairs $(x, s) \in \mathcal{X} \times \mathcal{S}$, to

$$w_{t,M-1,x,s}\ \mathbb{I}_{\{(X_{M,t},S_{M,t}) \in T((x,s))\}}\ .$$

This procedure, based on conditioning by the future, can be repeated to draw conditionally all the actions $X_{1,t}, X_{2,t}, \ldots,$ $X_{M,t}$ and hidden state spaces $S_{1,t}, S_{2,t}, \ldots, S_{M,t}$. In particular, we use, to draw $X_{j,t}$ and $S_{j,t}$, the distribution on $\mathcal{X} \times \mathcal{S}$ proportional to

$$w_{t,j,x,s}\ \mathbb{I}_{\{(X_{j+1,t},S_{j+1,t}) \in T((x,s))\}}\ . \tag{4}$$

The realization $\mathbf{X}_t = (X_{1,t}, X_{2,t}, \ldots, X_{M,t})$ obtained this way is indeed according to the distribution $\mathbf{p}_t$.

5.3.1 Complexity of this procedure for the considered examples

The space complexity is of the order of at most $O\big(MN|\mathcal{S}|\big)$, since weights have to be stored for all ask-action-state triples. The computational complexity, at a given task, for performing the updates (3) for all x' and s' is bounded by the number of pairs (x', s') times the maximal number of pairs (x, s) that lead to (x', s'). We denote by $T_{\max}$ this maximal number of transitions. Then, the complexity of performing (3) for all tasks is bounded by $O\big(MN|\mathcal{S}|T_{\max}\big)$. The complexity of the random generations (4) is negligible in comparison, since it is of the order of $O\big(MN|\mathcal{S}|\big)$.

We now compute $T_{\max}$ for the four examples described in Section 3 and summarize the complexity results (both for the efficient and the naive implementations) in the table below. In Example 1, in addition to the parameter ρ introduced in Section 4, we consider a common upper bound on the number of γ–close actions to any action in $\mathcal{X}$,

$$\vartheta = \max\left\{ \Big| \big\{ x' \in \mathcal{X} :\ |x - x'| \leqslant \gamma \big\} \Big| :\ x \in \mathcal{X} \right\} .$$

Then, $T_{\max} = \vartheta$. In Example 2, the value $T_{\max} = N$ is satisfactory. In Example 3, only $T_{\max} = N$ pairs (x, s), of the form $x = x'$ and $s = s'$ or $x \neq x'$ and $s' = s + 1$, can lead to (x', s'). A similar argument shows that in the case of Example 4, only $T_{\max} = N$ such transitions are possible also.

Ex.	Efficient	Naive
1.	$MN\vartheta$	$\geqslant N\rho^{M-1}$
2.	MN^2	$\geqslant (M+1)^N/(N-1)!$
3.	MN^2m	$\geqslant (MN)^m/m!$
4.	MN^2B	$\geqslant (B/M)^M$

6 Tracking

In the problem of *tracking the best expert* of [HW98, Vov99], the goal of the forecaster is, instead of competing with the best fixed action, to compete with the best sequence of actions that can switch actions a limited number of times. We may formulate the tracking problem in the framework of multi-task learning with hard constraints. In this case, just like before, at each time t, the decision maker chooses an M–tuple of actions from the set $\mathcal{A}$ of legal vectors. However, now regret is measured by comparing the cumulative loss of the forecaster $\sum_{t=1}^n \ell(\mathbf{X}_t.\mathbf{y}_t)$ with

$$\min_{(\mathbf{x}_1,\ldots,\mathbf{x}_n) \in \Sigma_K(\mathcal{A})} \sum_{t=1}^n \ell(\mathbf{x}_t, \mathbf{y}_t)$$

where $\Sigma_K(\mathcal{A})$ is the set of all sequences of vectors of $\mathcal{A}$ that may switch values at most K times (i.e., the time interval $1 \ldots, n$ can be divided into at most $K+1$ intervals such that over each interval the same M–tuple of actions). In this case it is well known that exponentially weighted average over the class $\Sigma_K(\mathcal{A})$ of meta-experts (see [CBL06, Sections 5.5 and 5.6] for a statement of the results and precise bibliographic references) yields a regret

$$\begin{aligned}
&\sum_{t=1}^n \ell(\mathbf{X}_t, \mathbf{y}_t) - \min_{(\mathbf{x}_1,\ldots,\mathbf{x}_n) \in \Sigma_K(\mathcal{A})} \sum_{t=1}^n \ell(\mathbf{x}_t, \mathbf{y}_t) \\
&\quad = O\left(M\sqrt{n\left(K \ln|\mathcal{A}| + K \ln\frac{n}{K}\right)} + M\sqrt{n \ln\frac{1}{\delta}} \right)
\end{aligned}$$

which holds with probability $1 - \delta$. Moreover, the complexity of the generation of the M–tuples of actions achieving the regret bound above is bounded, at round t, by $O\big(t^2 + MN^2|\mathcal{S}|^2\, Kt\big)$.

7 Multi-task learning in bandit problems

In this section we briefly discuss a more difficult version of the problem when the decision maker only observes the total loss $\ell(\mathbf{X}_t, \mathbf{y}_t)$ suffered though the M games but the sequence $\mathbf{y}_t$ of outcomes remains hidden. This may be considered as a "bandit" variant of the basic problem.

Then our problem becomes an instance of an *online linear optimization* problem studied by [AK04, MB04, GLLO07, DHK08, AHR08, BDH$^+$08, CBL09]. For example, since the dimension of the underlying space is given by the number of edges, in number always less than $1 + MN^2|\mathcal{S}|^2$, the results of [DHK08] imply that a variant of the exponentially weighted average predictor achieves an expected regret of the order

$$\begin{aligned}
&\mathbb{E}\left[\sum_{t=1}^n \ell(\mathbf{X}_t, \mathbf{y}_t)\right] - \min_{\mathbf{x} \in \mathcal{X}} \mathbb{E}\left[\sum_{t=1}^n \ell(\mathbf{x}, \mathbf{y}_t)\right] \\
&\quad = O\left(M\left(M^{3/2}N^3|\mathcal{S}|^3 + N|\mathcal{S}|\sqrt{M}\ln|\mathcal{A}| \right)\sqrt{n} \right) .
\end{aligned}$$

[BDH+08] proved that an appropriate modification of the forecaster satisfies this regret bound with high probability. As the predictor of [DHK08] requires exponentially weighted averages based on appropriate estimates of the losses, it can be implemented efficiently with the methods described in Section 5. More precisely, it first computes, at each round t, estimates of all losses $\ell^{(j)}(x, y_{j,t})$, when $x \in \mathcal{S}$ and $j = 1, \ldots, M$ and then can use the methods described in Section 5. The computationally most complex point is to compute these estimates, which essentially relies on computing and inverting an incidence matrix of size bounded by the number of edges. This can be done in time $O\big(M^2 N^4 |\mathcal{S}|^4\big)$. Details are omitted.

8 Other measures of loss

In this section we study two variations of the multi-task problem in which the loss of the decision maker in a round is computed in a way different from summing the losses over the tasks. [DLS07] measure losses by different norms of the loss vector across tasks but they do not consider the hard constraints introduced here.

8.1 Choosing a subset of the tasks

In our first example, at every round of the game, the forecaster chooses m out of the M tasks and only the losses over the chosen tasks count in the total loss. For simplicity we only consider the full-information case here when the decision maker has access to all losses (not only those that correspond to the chosen tasks).

Formally, we add an extra action $-$ which means that the decision maker does not play in this task. Of course, $\ell^{(j)}(-, y) = 0$ for all j and $y \in \mathcal{Y}_j$. We model this by

$$\mathcal{A} = \left\{ \big(x_{k_1}, \ldots, x_{k_M}\big) \in \big(\mathcal{X} \cup \{-\}\big)^M : \sum_{j=1}^{M} \mathbb{I}_{\{x_{k_j} \neq -\}} = m \right\} .$$

Since an element of $\mathcal{A}$ is characterized by the m tasks (out of M) in which it takes one among the N actions of $\mathcal{X}$, we have

$$|\mathcal{A}| = \binom{M}{m} N^m .$$

Here again, the bound of Proposition 1 applies and an efficient implementation is possible as in Section 5, at a cost of $O\big(MN^2 m^2\big)$.

Of course, additional hard constraints could be added in this example.

8.2 Choosing a different global loss

This paragraph is inspired by [DLS07] where a notion of a "global loss function" is introduced. The loss measured $\ell(\mathbf{X}_t, \mathbf{y}_t)$ in a round is now a given function ψ of the losses $\ell^{(j)}(X_{j,t}, y_{j,t})$ incurred in each task j, which may be different from their sum,

$$\ell(\mathbf{X}_t, \mathbf{y}_t) = \psi\Big(\ell^{(1)}(X_{1,t}, y_{1,t}), \ldots, \ell^{(M)}(X_{M,t}, y_{M,t})\Big) .$$

Examples include for instance the max-loss or the min loss,

$$\psi(u_1, \ldots, u_M) = \max\{u_1, \ldots, u_M\} \quad \text{or} \quad \psi(u_1, \ldots, u_M) = \min\{u_1, \ldots, u_M\} ,$$

whenever one thinks in terms of the best or worst performance.

We make a Markovian assumption on the losses. More precisely, we assume that they can be computed recursively as follows. There exists a function φ on $\mathbb{R}^2$ such that, defining the sequence $(v_2, \ldots, v_M)$ as

$$v_2 = \varphi(u_1, u_2) \quad \text{and} \quad v_t = \varphi(v_{t-1}, u_t) \ \text{ for } t \geqslant 3 ,$$

one has

$$v_M = \psi(u_1, \ldots, u_M) .$$

This means that if the values v_t are added as a hidden state space $\mathcal{V}$, and if the latter is not too big, computation of the distributions $\mathbf{p}_t$ defined, for all rounds $t \geqslant 0$ and all simultaneous actions $\mathbf{x} \in \mathcal{A}$, by

$$\mathbf{p}_t(\mathbf{x}) = \frac{\exp\Big(-\eta \sum_{s=1}^{t-1} \ell(\mathbf{x}, \mathbf{y}_t)\Big)}{\sum_{\mathbf{a} \in \mathcal{A}} \exp\Big(-\eta \sum_{s=1}^{t-1} \ell(\mathbf{a}, \mathbf{y}_t)\Big)} ,$$

can be done efficiently (a statement which we will be made more precise below). In addition, it is immediate, by reduction to the single-task setting, that a regret bound as in Proposition 1 holds, where one simply has to replace M with the supremum norm of ψ over the losses.

We only need to explain how and when the results of Section 5.3 extend to the case considered above. The state $\mathcal{V}$ of possible values for the possible sequences of v_t should not be too large and the update (3) has to be modified, in the sense that it is unnecessary to multiply by the exponential of the losses; the global loss will be taken care of at the last step only, its value being tracked by the additional hidden space. The complexity is of the order of at most $O\big(MN^2 |\mathcal{S}|^2 |\mathcal{V}|^2\big)$. Examples of small $|\mathcal{V}|$ include the case when the global loss is a max-loss or a min-loss and the case when all outcome spaces $\mathcal{Y}_j$ and loss functions $\ell^{(j)}$ are identical. In this case, $|\mathcal{V}| = N$.

Note that here, in addition to this change of the measure of the total incurred in a round, additional hard constraints can still be considered, since the base state space $\mathcal{S}$ is designed to take care of them.

9 Multi-task learning with a continuum of tasks and hard constraints

In this section we extend our model by considering infinitely many tasks. We focus on the case when tasks are indexed by the $[0, 1]$ interval. We start by describing and motivating the setup, then propose an ideal forecaster whose exact efficient implementation remains a challenge. We propose a discretization instead, which will take us back to the previously discussed case of a finite number of tasks.

9.1 Continuum of tasks with a constrained number of shifts

Assume that tasks are indexed by $g \in [0, 1]$. The decision maker has access to a finite set $\mathcal{X} = \{x_1, \ldots, x_N\}$ of actions. Taking simultaneous actions in all games at a given

round t is now modeled by choosing a measurable function

$$I_t : g \in [0,1] \mapsto I_t(g) \in \mathcal{X} .$$

The opponent chooses a bounded measurable loss function $\psi_t : [0,1] \times \mathcal{X} \to [0,1]$. The loss incurred by the decision maker is then given by

$$\ell_t(I_t) = \int_{[0,1]} \psi_t\big(g, I_t(g)\big)\, \mathrm{d}g = \sum_{x \in \mathcal{X}} \int_{\{I_t = x\}} \psi_t(g,x)\, \mathrm{d}g .$$

As before, we require that the action of the decision maker satisfies a hard constraint. One case that is easy to formulate is, that I_t must be right-continuous and the family of actions taken simultaneously, $\big(I_t(g)\big)_{g \in [0,1]}$ must contain at most a given number m of shifts, where by definition, there is a shift at g if for all $\varepsilon > 0$, the set $I_t\big([g-\varepsilon, g]\big)$ contains at least two actions. We denote by $\mathcal{A}$ the set of such simultaneous actions. Actually, any element of $\mathcal{A}$ can be described by its shifts (in number at most m), denoted by $g_1, \ldots, g_{m'}$, with $m' \leqslant m$, and the actions taken in the intervals $[g_j, g_{j+1}[$ for all $j = 0, \ldots, m'-1$ where $g_0 = 0$, and on $[g_{m'}, 1]$.

The aim of the decision maker is to minimize the cumulative regret

$$R_n = \sum_{t=1}^n \ell_t(I_t) - \inf_{I \in \mathcal{A}} \sum_{t=1}^n \ell_t(I) ,$$

where the I_t are picked from $\mathcal{A}$.

The setting above appears naturally as an alternative model when the number M of tasks is large: the decision maker has to determine actions for each individual in a large population. He does so by setting some thresholds. The population can be all the individuals of a country, they are ordered by their ages (or incomes), and some decisions are made according to this age categorization.

9.2 An ideal forecaster

We denote by μ the distribution on $\mathcal{A}$ induced by the uniform distribution on $\mathcal{X}^{m+1} \times [0,1]^m$ via the mesurable application

$$\begin{aligned} &\big(x_{k_1}, \ldots, x_{k_{m+1}}, g_1, \ldots, g_m\big) \\ &\mapsto \mathbb{I}_{[0, g_{(1)}[} x_{k_1} + \left(\sum_{j=2}^m \mathbb{I}_{[g_{(j-1)}, g_{(j)}[} x_{k_j} \right) + \mathbb{I}_{[g_{(m+1)}, 1]} x_{k_{m+1}} , \end{aligned} \tag{5}$$

where we denoted by $\big(g_{(1)}, \ldots, g_{(m)}\big)$ the order statistics of the $g_1, \ldots, g_m$. (It is useful to observe for later purposes that if $G_1, \ldots, G_m$ are i.i.d. uniform, then the vector

$$\begin{aligned} &V(G_1, \ldots, G_m) \\ &= \big(G_{(1)}, G_{(2)} - G_{(1)}, \ldots, G_{(m)} - G_{(m-1)}, 1 - G_{(m)}\big) \end{aligned} \tag{6}$$

is uniformly distributed over the simplex of probability distributions with $m+1$ elements.)

For all $t \geqslant 1$, the ideal forecaster uses probability distributions $\mathbf{p}_t$ over $\mathcal{A}$, defined below, and draws the application I_t giving the simultaneous actions to be taken at round t according to $\mathbf{p}_t$. For $t = 1$, we take $\mathbf{p}_1 = \mu$. For $t \geqslant 2$, we take $\mathbf{p}_t$ as the probability distribution absolutely continuous with respect to μ and with density

$$\mathrm{d}\mathbf{p}_t(I) = \frac{\exp\Big(-\eta \sum_{s=1}^{t-1} \ell_s(I)\Big)}{\int_{\mathcal{A}} \exp\Big(-\eta \sum_{s=1}^{t-1} \ell_s(J)\Big)\, \mathrm{d}\mu(J)}\, \mathrm{d}\mu(I) . \tag{7}$$

The performance of this forecaster may be bounded as follows. Note that no assumption of continuity or convexity on the functions ℓ_s is needed here.

Theorem 2 *For all $n \geqslant 1$, the above instance of the exponentially weighted average forecaster, when run with*

$$\eta = \sqrt{\frac{8(m+1)\ln(N\sqrt{n})}{n}} ,$$

ensures that for all $\delta > 0$, its regret is bounded, with probability at most $1-\delta$, as

$$R_n \leqslant \sqrt{n}\left(1 + \sqrt{\frac{(m+1)\ln(N\sqrt{n})}{2}}\right) + \sqrt{\frac{n}{2}\ln\frac{1}{\delta}} .$$

Proof: By the Hoeffding-Azuma inequality, since the ψ_t take bounded values in $[0,1]$, we have that with probability at least $1-\delta$,

$$R_n \leqslant \sum_{t=1}^n \int_{\mathcal{A}} \ell_t(I)\, \mathrm{d}\mathbf{p}_t(I) - \inf_{I \in \mathcal{A}} \sum_{t=1}^n \ell_t(I) + \sqrt{\frac{n}{2}\ln\frac{1}{\delta}} . \tag{8}$$

We denote, for all $t \geqslant 1$,

$$W_t = \int_{\mathcal{A}} \exp\left(-\eta \sum_{s=1}^t \ell_s(I)\right) \mathrm{d}\mu(I)$$

(with the convention $W_0 = 1$). The bound on the difference in the right-hand side of (8) can be obtained by upper bounding and lower bounding

$$\ln W_n = \sum_{t=1}^n \ln \frac{W_t}{W_{t-1}} .$$

The upper bound is obtained, as in [CBL06, Theorem 2.2], by Hoeffding's inequality,

$$\ln \frac{W_t}{W_{t-1}} \leqslant -\eta \int_{\mathcal{A}} \ell_t(I)\, \mathrm{d}\mathbf{p}_t(I) + \frac{\eta^2 M^2}{8} .$$

A lower bound can be proved with techniques similar to the ones appearing in [BK97], see also [CBL06, page 49]. We denote by I^* the element of $\mathcal{A}$ achieving the infimum in the definition of the regret (if it does not exist, then we take an element of $\mathcal{A}$ whose cumulative loss is arbitrarily close to the infimum). As indicated in Section 9.1, I^* can be described by the (ordered) shifting times $g_1^*, \ldots, g_m^*$ and the corresponding actions $x_{k_1^*}, \ldots, x_{k_{m+1}^*}$. We denote by λ the Lebesgue measure. We consider the set of the simultaneous actions I that differ from I^* on a union of intervals of total length at most $\varepsilon > 0$, for some parameter $\varepsilon > 0$,

$$\mathcal{A}_\varepsilon(I^*) = \big\{ I : \lambda\{I \neq I^*\} \leqslant \varepsilon \big\} .$$

$\mathcal{A}_\varepsilon(I^*)$ contains in particular the I that can be described with the same $m+1$ actions as I^* and for which the shifting times $g_1, \ldots, g_m$ are such that

$$\sum_{j=1}^{m} \bigl|g_{(j)} - g_j^*\bigr| \leqslant \varepsilon \,,$$

i.e., the I for which the corresponding probability distribution $V(g_1, \ldots, g_m)$ as defined in (6) is ε–close in ℓ^1–distance to $V\bigl(g_1^*, \ldots, g_m^*\bigr)$. Because μ induces by construction, via the application V, the uniform distribution over the simplex of probability distributions over $m+1$ elements, we get, by taking also into account the choice of the fixed $m+1$ actions of I^*,

$$\mu\bigl(\mathcal{A}_\varepsilon(I^*)\bigr) \geqslant \frac{\varepsilon^m}{N^{m+1}} \,.$$

Here, we used the same argument as in [BK97], based on observing the fact that the uniform measure of the ε–neighborhood of a point in the simplex of probability distributions over d elements equals ε^{d-1}. In addition, because the ψ_t take values in $[0,1]$, we have, for all $I \in \mathcal{A}_\varepsilon(I^*)$ and all $s \geqslant 1$,

$$\ell_s(I) \leqslant \ell_s(I^*) + \lambda\{I \neq I^*\} \leqslant \ell_s(I^*) + \varepsilon \,.$$

Putting things together, we have proved that

$$\begin{aligned}
\ln W_n &= \ln \int_{\mathcal{A}} \exp\left(-\eta \sum_{s=1}^{n} \ell_s(I)\right) \mathrm{d}\mu(I) \\
&\geqslant \ln \left(\mu\bigl(\mathcal{A}_\varepsilon(I^*)\bigr) \exp\left(-\eta\left(\varepsilon n + \sum_{s=1}^{n} \ell_s(I^*)\right)\right)\right) \\
&\geqslant -\eta \sum_{s=1}^{n} \ell_s(I^*) - \left(m \ln \frac{1}{\varepsilon} + (m+1)\ln N + \eta\varepsilon n\right) \,.
\end{aligned}$$

Combining the upper and lower bounds on $\ln W_n$ and substituting the proposed value for η concludes the proof. ■

Efficient implementation in this context requires exact simulation of a step function I according to (7), that is, from the distribution

$$\mathrm{d}\mathbf{p}_t(I) \propto \exp\left(-\eta \int_0^1 \varphi_{t-1}\bigl(g, I(g)\bigr)\mathrm{d}g\right) \mathrm{d}\mu(I) \qquad (9)$$

for the functions defined, for each $x \in \mathcal{X}$, as

$$\varphi_{t-1}(\,\cdot\,, x) = \sum_{s=1}^{t-1} \psi_s(\,\cdot\,, x) \,,$$

which take values in $[0, t-1]$. One could simulate from (9) by rejection sampling proposing from μ; the probability of acceptance is bounded below by something of the order of $e^{-\sqrt{t}}$, in view of the value of η. Therefore, the computational cost of such an algorithm, although only linear in m and N, would be typically exponential in t, hence unappealing.

Note that the problem (at each round t) can be represented as a discrete-time Markov model. The Markov chain Z is given by the pairs formed by the shifting times and their corresponding actions, $Z_j = \bigl(G_{(j)}, K_{j+1}\bigr)$, for $j = 0, \ldots, m$ and with the convention $G_{(0)} = 0$. Let π denote the law of this Markov chain when the times $G_1, \ldots, G_m$ are i.i.d. uniform over $[0,1]$ and the action indices $K_1, \ldots, K_{m+1}$ are taken i.i.d. uniform in $\{1, \ldots, N\}$. Then simulating I according to (9) is equivalent to simulating Z according to the distribution

$$\mathrm{d}\tilde{\pi}_{t-1}(Z) \propto \prod_{j=2}^{m+1} w_j\bigl(K_{j-1}, G_{(j-1)}, G_{(j)}\bigr)\, \mathrm{d}\pi(Z)$$

where, for $g \leqslant g'$,

$$w_j(k, g, g') = \exp\left(-\eta \int_g^{g'} \varphi_{t-1}(u, x_k)\,\mathrm{d}u\right) \,,$$

Exact simulation from $\tilde{\pi}_{t-1}$ is feasible when the state-space of Z is finite, and consists, e.g., in the same type of dynamic programming approach discussed in Section 5. However, this is not the case here, since the second component of Z_j takes values in $[0,1]$. Approximating the state-space of Z by a grid is a possibility for an approximate implementation, but it will be typically less efficient than the approximation we advocate in Section 9.3.

An interesting alternative is to resort to sequential Monte Carlo methods (broadly known as particle filters, see for example [DdFG01] for a survey). This is a class of methods ideally suited for approximating Feynman-Kac formulae; a concrete example is the computation of expectations of bounded functions with respect to the laws $\tilde{\pi}_{t-1}$ defined above. This is achieved by generating a swarm of a given large number of weighted particles. The generation of particles is done sequentially in $j = 1, \ldots, m+1$ by importance sampling, and it involves interaction of the particles at each step. This generates an interacting particle system whose stability properties are well studied (see, for instance, [DM04]). Resampling a single element from the particle population according to the weights gives as an approximate sample from $\tilde{\pi}_{t-1}$, hence from (9). The total variation distance between the approximation and the target is typically $C(m+1)/K$, for some constant C depending on the range of the integrands. In the most naive implementation in this context, one might thus have that C is exponentially small in t/m. The idea of an on-going work would be to make C independent of t by carefully designing the importance sampling at each step taking into account the characteristics of the φ_{t-1}.

Below we use a simple discretization and apply the techniques of previous sections to achieve approximate sampling from (7).

9.3 Approximate generation by discretization

Here we show how an approximate version of the forecaster described above can be implemented efficiently.

The argument works by partitioning $[0,1]$ into intervals $G^0 = [0, 1/\varepsilon[$, $G^1 = [1/\varepsilon, 2/\varepsilon[$, …, G^{M_ε} of length ε (except maybe for the last interval of the partition), for some fixed $\varepsilon > 0$, and using the same action for all tasks in each G^j. Here, we aggregate all tasks within an interval G^j into a super-task j. We have $M = M_\varepsilon = \lceil 1/\varepsilon \rceil$ of these super-tasks and will be able to apply the techniques of the finite case.

More precisely, we restrict our attention to the elements of $\mathcal{A}$ whose shifting times (in number less or equal to m) are

starting points of some G^j, that is, are of the form j/ε for $0 \leqslant j \leqslant M_\varepsilon$. We call them simultaneous actions compatible with the partitioning and denote by $\mathcal{B}_\varepsilon$ the set formed by them. The loss of super-task j at time t given the simultaneous actions described by the element $I \in \mathcal{B}_\varepsilon$ is denoted by

$$\ell_t^{(j)}(I) = \int_{G^j} \psi_t\big(g, I(j/\varepsilon)\big)\, \mathrm{d}g \;.$$

Note that these losses satisfy $\ell_t^{(j)}(I) \in [0, \varepsilon]$.

By the same argument as the one used in the proof of Theorem 2, we have

$$\inf_{I\in\mathcal{A}} \sum_{t=1}^n \ell_t(I) \leqslant \inf_{I\in\mathcal{B}_\varepsilon} \sum_{t=1}^n \ell_t(I) + \frac{mn\varepsilon}{2} \;.$$

This approximation argument, combined with Proposition 1 and the results of Section 5 leads to the following. (We use here the fact that there are not more than

$$\left(\begin{array}{c} M_\varepsilon \\ m \end{array}\right) N^m \leqslant \big(M_\varepsilon N\big)^m$$

elements in $\mathcal{B}_\varepsilon$.

Theorem 3 *For all $\varepsilon > 0$, the weighted average forecaster run on the M_ε super-tasks defined above, under the constraint of not more than m shifts, ensures that for a proper choice of η and with probability at least $1-\delta$, the regret is bounded as*

$$R_n \leqslant \sqrt{\frac{nm\ln\big(N\lceil 1/\varepsilon\rceil\big)}{2}} + \frac{mn\varepsilon}{2} + \sqrt{\frac{n}{2}\ln\frac{1}{\delta}}$$

In addition, its complexity of implementation is $O\big((Nm)^2/\varepsilon\big)$.

The choice of ε of the order of $1/\sqrt{n}$ yields a bound comparable to the one of Theorem 2, for a moderate computational cost of $O\big(\sqrt{n}(Nm)^2\big)$.

These results can easily be extended to the bandit setting, when ψ_t is only observed through I_t as

$$\ell_t(I_t) = \int_{[0,1]} \psi_t\big(g, I_t(g)\big)\, \mathrm{d}g \;.$$

This is because whenever I_t is compatible with the partitioning, the latter is also the sum of the losses of the actions taken in each of the super-tasks. The techniques of Section 7 can then be applied again.

References

[ABR07] J. Abernethy, P.L. Bartlett, and A. Rakhlin. Multitask learning with expert advice. In *Proceedings of the 20th Annual Conference on Learning Theory*, pages 484–498, New-York, 2007. Springer.

[AHR08] J. Abernethy, E. Hazan, and A. Rakhlin. Competing in the dark: An efficient algorithm for bandit linear optimization. In *Proceedings of the 21st Annual Conference on Learning Theory (COLT 2008)*, pages 263–274, 2008.

[AK04] B. Awerbuch and R.D. Kleinberg. Adaptive routing with end-to-end feedback: distributed learning and geometric approaches. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing*, pages 45–53, New York, 2004. ACM.

[BDH+08] P. Bartlett, V. Dani, T. Hayes, S.M. Kakade, A. Rakhlin, and A. Tewari. High-probability regret bounds for bandit online linear optimization. In *Proceedings of the 21st Annual Conference on Learning Theory (COLT 2008)*, pages 335–342, 2008.

[BK97] Avrim Blum and Adam Kalai. Universal portfolios with and without transaction costs. In *Proceedings of the 10th Annual Conference on Learning Theory*, pages 309–313. ACM Press, 1997.

[CBL06] N. Cesa-Bianchi and G. Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, New-York, 2006.

[CBL09] N. Cesa-Bianchi and G. Lugosi. Combinatorial bandits. Technical report, 2009.

[CCBG08] G. Cavallanti, N. Cesa-Bianchi, and C. Gentile. Linear algorithms for online multitask classification. In Omnipress, editor, *Proceedings of the 21st Annual Conference on Learning Theory*, 2008.

[DdFG01] A. Doucet, N. de Freitas, and N. Gordon, editors. *Sequential Monte Carlo Methods in Practice*. Statistics for Engineering and Information Science. Springer-Verlag, New York, 2001.

[DHK08] V. Dani, T. Hayes, and S.M. Kakade. The price of bandit information for online optimization. In *Proceedings of NIPS 2008*, 2008.

[DLS07] Ofer Dekel, Philip M. Long, and Yoram Singer. Online learning of multiple tasks with a shared loss. *Journal of Machine Learning Research*, 8:2233–2264, 2007.

[DM04] P. Del Moral. *Feynman-Kac formulae*. Probability and its Applications (New York). Springer-Verlag, New York, 2004. Genealogical and interacting particle systems with applications.

[GLL04] A. György, T. Linder, and G. Lugosi. Efficient algorithms and minimax bounds for zero-delay lossy source coding. *IEEE Transactions on Signal Processing*, 52:2337–2347, 2004.

[GLL05] A. György, T. Linder, and G. Lugosi. Tracking the best of many experts. In *Proceedings of the 18th Annual Conference on Learning Theory*, pages 204–216, 2005.

[GLLO07] A. György, T. Linder, G. Lugosi, and Gy. Ottucsák. The on-line shortest path problem under partial monitoring. *Journal of Machine Learning Research (JMLR)*, 8:2369–2403, 2007.

[HW98] M. Herbster and M. Warmuth. Tracking the best expert. *Machine Learning*, 32(2):151–178, 1998.

[MB04] H.B. McMahan and A. Blum. Online geometric optimization in the bandit setting against an adaptive adversary. In *Learning theory*, volume 3120 of *Lecture Notes in Computer Sciences*, pages 109–123. Springer, Berlin, 2004.

[Men07] F. Mengel. Learning across games. Technical report, IVIE-working paper AD 2007-05, 2007.

[TW04] E. Takimoto and M. Warmuth. Path kernels and multiplicative updates. *Journal of Machine Learning Research*, 4(5):773–818, 2004.

[Vov99] V. Vovk. Derandomizing stochastic prediction strategies. *Machine Learning*, 35(3):247–282, 1999.