Lifelong Learning with Branching Experts

Yi-Shan Wu YSWU@DI.KU.DK
Department of Computer Science, University of Copenhagen, Denmark

Yi-Te Hong TED0504@IIS.SINICA.EDU.TW
Institute of Information Science, Academia Sinica, Taiwan
Department of Computer Science and Information Engineering, National Taiwan University, Taiwan

Chi-Jen Lu CJLU@IIS.SINICA.EDU.TW
Institute of Information Science, Academia Sinica, Taiwan

Editors: Vineeth N Balasubramanian and Ivor Tsang

Abstract

The problem of branching experts is an extension of the experts problem where the set of experts may grow over time. We compare this problem in different learning settings along several axes: adversarial versus stochastic losses; a fixed versus a growing set of experts (branching experts); and single-task versus lifelong learning with expert advice. First, for the branching experts problem, we achieve tight regret bounds in both adversarial and stochastic setting with a single algorithm. While it was known that the adversarial branching experts problem is strictly harder than the non-branching one, the stochastic branching experts problem is in fact no harder. Next, we study the extension to the lifelong learning with expert advice in which one has to make online predictions with a sequence of tasks. For this problem, we provide a single algorithm which works for both adversarial and stochastic setting, and our bounds when specialized to the case without branching recover the regret bounds previously achieved separately via different algorithms. Furthermore, we prove a regret lower bound which shows that in the lifelong learning scenario, the case with branching experts now becomes strictly harder than the non-branching case in the stochastic setting.

Keywords: Lifelong Learning; Branching Experts; Best of Both Worlds

1. Introduction

In the classical problem of prediction with expert advice, referred to as the experts problem, a learner makes online decisions over some T steps, by consulting advice from a fixed set of N experts. At each step, the learner makes decisions with the advices from a set of experts. Afterward, both the experts and the learner receive their losses and proceed to the next round. Here we focus on the full-information setting in which the learner gets to know the losses of all the experts as feedback. The goal of the learner is to minimize the regret, which is the difference between his total loss and that of the best offline expert. More information can be found in standard textbooks such as (Cesa-Bianchi and Lugosi, 2006). While the original formulation of the problem already captures a broad range of natural scenarios, there are still many which require modification or generalization of the problem.

© 2021 Y.-S. Wu, Y.-T. Hong & C.-J. Lu.

We would like to study the effect of encountering a changing number of experts on the online decision problem. This theme reoccurs in many different works in online learning in the form of, for instance, branching experts, sleeping experts, etc. Here we will follow the framework of the branching experts (Gofer et al., 2013), where the number of experts only increases.

As an illustration, we imagine the classical experts problem with an enormously huge set of experts. Recall that the classical experts problem has a regret bound of $\mathcal{O}(\sqrt{T \log N})$, which is bad when N is extremely large. This bound is known to be tight in the adversarial case when the experts have no relation with each other. However, one can imagine scenarios when experts share similar losses and may be beneficial to cluster these experts into sub-groups. Within each sub-group, there is a representative expert that approximates with enough accuracy the remaining experts, in terms of their losses being approximately equal, up to the current time-step. Splitting a sub-group into smaller ones occurs at a certain time-step when the losses of the experts in this sub-group are no longer approximately similar. In this way, new representative experts emerge for the newly-split sub-groups. Whenever a representative expert can approximate the rest of the group well, the learning problem may be simplified by considering only the advices of those representatives: their number may hopefully be much smaller and in turn gives a smaller regret bound guarantee. The number of the set of the (representative) experts grows. In this sense, our notion of a growing number of experts is that of the representative experts. Furthermore, the newly-emerged experts at various time-steps are not unrelated to the pre-existing ones. They emerge from their predecessors (the latter we call parents and the former children) and inherit approximately from their parents the losses in previous rounds, before their creation. This gives a tree structure to the losses of the experts. We consider two different settings: the adversarial and the stochastic setting, which we make this distinction in terms of how the losses are generated. For the stochastic setting, what we have in mind conceptually (albeit we only tackle the full-information case) may be the continuum-armed bandit problem where there are possibly infinitely many arms. While the total number of arms is large, neighboring arms do share approximately equal mean rewards. For this example, branching of the experts (or the splitting of the sub-groups) may correspond to when we differentiate apart the clustered arms according to some approximation error level .

1.1. Related Works

1.1.1. Low-Rank Experts

A related line of work is the low-rank experts problem by Hazan et al. (2016). The learner makes decisions from N experts, where N may possibly be a large number. However, there is an underlying structure for the experts, namely, the N-dimensional loss functions ℓ_t only span a k-dimensional subspace. That is, the loss matrix L composed of ℓ_t as the column vectors is of rank k, where k may be much smaller than N. The goal is to obtain smaller regret bound guarantees in terms of k, rather than N. Note that the experts problem with losses that can be clustered into k sub-groups may be viewed as a low-rank experts problem such that the loss matrix $L = [\ell_1, \ldots, \ell_t, \ldots, \ell_T]$ can be approximated within some error ε by a matrix of rank k. In (Hazan et al., 2016), the authors consider both the adversarial and stochastic low-rank experts setting and obtain regret bound $\mathcal{O}(k\sqrt{T})$ and pseudo-regret bound $\mathcal{O}(\sqrt{kT})$, respectively, by applying separately two types of algorithms: the online-mirror-descent and the follow-the-leader algorithm. In particular, applied to the stochastic branching experts problem, Hazan et al. (2016) observed that suppose only k representative experts are seen, using the follow-the-leader algorithm, the pseudo-regret bound is $\mathcal{O}(\sqrt{T \log k})$ plus the approximation error term. On the other hand, however, for the adversarial branching experts problem with k clusters, this approach does not give a tight regret bound, since Gofer et al. (2013) give a tighter bound $\Theta(\sqrt{kT})$.

For the adversarial branching experts setting with experts gradually revealed, a regret lower bound of $\Omega(\sqrt{TM})$ turns out to be unavoidable in the adversarial setting, where M is the number of branching steps which can be as large as $\Omega(N)$ (Gofer et al., 2013) . On the other hand, in the stochastic setting, as is mentioned in (Hazan et al., 2016), the regret can be lowered to $\mathcal{O}(\sqrt{T \log \tilde{N}})$, where $\tilde{N}$ is the number of representative experts, via the follow-the-leader algorithm, which is different from the Hedge-based algorithms of (Gofer et al., 2013; Cohen and Mannor, 2017). We would like to know if it is possible to have one single algorithm which can work in both settings, just as in the non-branching case in works such as (Mourtada and Gaïffas, 2019).

1.1.2. Construction of sleeping experts

Another line of work for tackling the experts problem with varying number of experts is by following the route of constructing sleeping experts on the fly. (Luo and Schapire, 2015; Mourtada and Maillard, 2017). Notice a main difference from our setting is that they didn't impose the tree structure condition of the branching experts. Thus, in these works, the newly-emerged experts may have no relations at all with the current set of experts. However, in our setting, the emerged experts when created have already in their past suffered losses inherited from their parents. This make adopting their approach with certain difficulties in contrast to the on-the-fly construction and computation in the aforementioned papers.

In the paper (Luo and Schapire, 2015), for the non-branching case, the AdaNormal-Hedge.TV algorithm has small adaptive regrets for both adversarial and stochastic setting. The tracking regret can then be derived for adversarial setting with M shifts, it is bounded by $\hat{O}(\sqrt{MT \ln(NT)} + M)$. In the stochastic setting, the bound is $\hat{O}(M \ln(NT)/\Delta + M)$. $(M = 1)$ In both bounds, the $\hat{O}$ notation hides a $\log\log(T)$ factor. At each time-step t, the algorithm creates N number of sleeping experts, indexed by (t, i) for $i \in [N]$, who is asleep pior to time-step t, wakes up at time-step t and suffers the same loss of expert i from then on. In total there will be NT number of sleeping experts. At time $t + 1$, the learner plays the distribution which is obtained by aggregating the awake experts. Suppose the number of the experts may vary, and no tree structure condition imposed for the losses, then the above results extend with N supplanted by N_T, the number of distinct experts seen until time-step T.

Now in order to apply this method to our branching expert setting, we make the following modifications and then we discuss the differences between the two. At each time step t, with N_t denoting the current number of representative experts, (Step A) the algorithm first creates N_t number of sleeping experts, indexed by (t, i) for $i \in [N_t]$, who is asleep prior to time-step t, awakes at time-step t and suffers losses as expert i from then on; (Step B) for the newly-emerged experts (indexed from $N_{t-1} + 1$ to N_t), the algorithm creates

$(N_t - N_{t-1}) \times (t-1)$ additional sleeping experts, each indexed by $(s, j), s \in [t-1]$, and $j \in [N_{t-1}+1, N_t]$, who is asleep prior to time-step s, awakes at time-step s and suffers losses of its parent (with small approximation errors) from then on up to $t-1$.

For the sleeping experts (s, j) created in (Step B), who awakes at time $s < t$ with $j \in [N_{t-1}+1, N_t]$, it suffers approximately the same losses as the representatives of expert j before time t and as expert j hereafter time t. (The losses of a branching expert can be approximated by those of its representatives, for the time steps when it has not yet been created.) At time $t+1$, the algorithm predicts by aggregating the awake experts using AdaNormalHedge type algorithms. In total, there are TN_T number of sleeping experts. In the adversarial setting, the shifting regret with M shifts is bounded by $\hat{O}(\sqrt{MT\ln(TN_T)} + M)$ and in the stochastic setting, by $\hat{O}(M\ln(TN_T)/\Delta + M)$. This modification of AdaNormalHedge-type algorithm may apply to the single-task branching experts, where M now denotes the number of branching steps. We remark that the bounds contain an additional log(T) term and an extra factor of log log T in the $\hat{O}$-notation, which we manage to remove in our bound for Algorithm 1.

So far nothing seems to be different for the branching expert setting; however, although this gives a similar on-the-fly construction of the sleeping experts, the corresponding computation is more involved as we will see and the time complexity of AdaNormalHedge.TV for the branching expert setting is not the same as in (Luo and Schapire, 2015). When a branching expert $i \in [N_{t-1}+1, N_t]$ emerges at time t, it inherits the losses from its representatives in previous rounds. Contrary to the setting stated in page 7 in (Luo and Schapire, 2015), $R_{t-1,(s,j)}$ and $C_{t-1,(s,j)}$ (which denote respectively, the cumulative regret and the the cumulative magnitude of the instantaneous regrets up to time $t-1$) are no longer zeros for the sleeping experts $(s, j) \in [t-1] \times [N_{t-1}+1, N_t]$ created in (Step B) (because there are loss inheritance from the past). Thus, in order to compute the distribution of play at time $t+1$ the learner has to start from time-step $\tau = 1$ to $t+1$ to recompute iteratively the modified distributions of play aggregated from the now awake experts, by Eqn.(3) in (Luo and Schapire, 2015): $p_{\tau,(s,j)} \propto q_{(s,j)} w(R_{\tau-1,(s,j)}, C_{\tau-1,(s,j)})$, for $(s, j) \in W_\tau$ (the awake set at time τ). This recomputation needs to take place when there occurs a branching of the experts (that is, when $N_t > N_{t-1}$).

1.1.3. Lifelong Learning

Besides the comparison of stochastic versus adversarial losses, another direction for extension is to go from a single branching experts task to multiple ones. Now, instead of learning each task separately, the hope is that knowledge learned from one task could be transferred for other tasks, which would make learning easier and more efficient later on. For this to be possible, tasks should be related with some commonality. (Baxter et al., 2000; Maurer, 2009; Pentina and Lampert, 2014; Denevi et al., 2019) considered a task environment, from which all tasks are generated according to a common distribution. A different model considered by (Ruvolo and Eaton, 2013; Pentina and Ben-David, 2015; Alquier et al., 2017; Wu et al., 2019), which is natural for tasks such as classification, is that there is some feature representation shared by all tasks, while tasks differ by requiring different predictors on top of it. The feature representation usually requires a large and complex network, so we wish to learn it continuously across tasks. With a common feature representation shared among tasks, learning the predictor on top of it can be relatively simple, which we can afford to re-learn for each different task. Following prior works, we will use the terms representation and predictor for convenience, although we are not restricted to classification tasks.

There is also the issue of batch versus online learning. (Baxter et al., 2000; Maurer, 2005, 2009; Maurer et al., 2013) considered the fully batch setting, while (Balcan et al., 2015; Pentina and Ben-David, 2015; Alquier et al., 2017) considered a batch-within-online one. In contrast, Alquier et al. (2017); Denevi et al. (2019); Wu et al. (2019) considered the online-within-online setting, also called lifelong learning, in which tasks arrive sequentially and so do the samples of each task. In this paper, we study this lifelong learning setting, and measure the regret against an offline algorithm which must use a fixed representation for all tasks but can use different predictors for different tasks. Alquier et al. (2017); Denevi et al. (2019) among others considered a notion of regret which treats each task equally and their algorithms only update the representation at the end of each task. Consequently, a large number of tasks is required to make their regret small. On the other hand, Wu et al. (2019) introduced a different regret notion which treats each step equally, and proposed algorithms to update the representation at each step which can achieve small regret with few tasks. Although the regret notion in (Alquier et al., 2017) is appropriate for several natural applications, here we adopt the regret notion in (Wu et al., 2019). While Wu et al. (2019) achieve good regret bounds in both adversarial and stochastic settings, different algorithms are used for different settings. Again, we would like to know if one algorithm can work in both worlds: adversarial and stochastic. Moreover, we would like to generalize the branching experts problem to the lifelong learning scenario, because the many situations it models also have natural generalization to lifelong learning.

There are other regret notion such as adaptive regret or shifting regret which measures how well the online learner's performance in a particular interval or against the best among sequences of shifting off-line experts up to a fixed number. They are different from that we used in lifelong learning, as these do not model the hindsight that different tasks share some commonality (a shared representation). In our lifelong learning with branching experts, an action is decomposed into a representation and a predictor, and there is a common best representation shared by all tasks in hindsight. Both the number of representations, N_G, and the number of predictors, N_H, grow over time. Our algorithms decouple the learning of representations, which continues across tasks, from that of predictors, which resets in each task. Works on shifting regret such as (Luo and Schapire, 2015) do not make such a distinction. Notice that a regret bound of the order $\sqrt{T \log N_G + KT \log N_H}$, where the number of tasks K does not interact with the number of representations N_G is desirable in lifelong learning as N_G is typically much larger than N_H so that we do not want to relearn the representation in each task.

1.2. Roadmap

We start from the simpler branching experts problem. Recall that having gradually revealed experts makes the problem harder in the adversarial setting, but not so in the stochastic setting. While Gofer et al. (2013); Cohen and Mannor (2017) achieve nearly tight regret bounds in the adversarial setting, a lower regret bound in the stochastic setting is achieved via a different algorithm (Hazan et al., 2016). Although it may appear necessary to have different algorithms tailored for different settings, we show that one single algorithm suffices to achieve almost tight bounds in both settings. That is, we can run our algorithm without knowing whether the world is adversarial or stochastic. If it is adversarial, our algorithm achieves a regret of about $\mathcal{O}(\sqrt{TM \log N})$, where M is the number of branching steps. If stochastic, the regret becomes $\mathcal{O}(\frac{1}{\Delta} \log N)$, when suboptimal experts have some gap Δ, and about $\mathcal{O}(\sqrt{T \log N})$ otherwise, both independent of M. We also provide a second algorithm which achieves slightly worse regret but works better for learning representations in lifelong learning discussed next.

Based on the second algorithm for learning representations and the first for learning predictors, we build our algorithm for lifelong learning which can again work simultaneously in both adversarial and stochastic settings. When specialized to the non-branching case, we recover the previous bounds in (Wu et al., 2019) for both settings. Let us stress that while different algorithms are needed for different settings in (Wu et al., 2019), here we use one single algorithm which can be run without needing to know the world it is in. Furthermore, we show that the regret bounds achieved by our algorithm are almost tight. An interesting phenomenon we discover is the following. When there are large loss gaps in the stochastic setting, we can still achieve small regret bounds independent of the number M of branching steps. On the other hand, when the gaps becomes smaller, the regret approaches the adversarial one, which depends on M. That is, in contrast to the one-task case, having branching now provably makes the problem harder in the stochastic setting, and the dependence on M can no longer be avoided.

In addition, we also extend our algorithms to the case with extremely large or even infinite numbers of representations and predictors, and have the regrets bounded in terms of their covering numbers, which may be much smaller. This is our initial motivation for studying branching experts. In the infinite case, our algorithms are not efficient in general, and our results should be seen as giving minimax regret upper bounds in terms of covering numbers.

1.3. Contribution

We consider the contribution of our work as follows. Conceptually, we answer the question of whether a single algorithm can work simultaneously in multiple settings. Moreover, we discover the interesting phenomenon of how branching and multi-task have different effects in the adversarial and stochastic settings. Technically, we find appropriate generalization of previous works so that simple modification to their algorithms can work simultaneously for multiple settings. Moreover, for lifelong learning, it becomes much more difficult to have a single algorithm for both settings. In particular, showing that it also works in the stochastic setting requires substantial technical effort, since the combination of branching and multi-task could make the world look adversarial.

2. Preliminaries

Let us first introduce some notations. For any $n \in \mathbb{N}$, let $[n]$ denote the set $\{1, \ldots, n\}$. For a loss vector $\ell \in [0,1]^n$ and an index $i \in [n]$, let ℓ_i denote its i-th coordinate. Later we will often have a time-indexed vector $\ell_t \in [0,1]^n$, and we will write $\ell_{t,i}$ for its i-th coordinate. The branching experts problem is a modification of the online learning with expert advice problem. The difference is that the full set of experts, denoted as H, is not initially given, but only gradually revealed to the learner. The game starts with only one known expert, and over the time, more experts are revealed and become active. Once a new expert is revealed and becomes active, the learner can start to take the expert's advice, with some probability. Each newly revealed expert branches out from a previously revealed one, called its parent, under the assumption of having similar previous losses (each within some distance ε). From that moment on, the newly revealed expert can have losses completely different from his parent in the adversarial setting. Following (Gofer et al., 2013; Cohen and Mannor, 2017), we assume that all the previous losses of a revealed expert are given, although our algorithms actually work under a weaker assumption that only the loss of the previous step be given. The branching process results in a tree structure, where at each time t, each active expert corresponds to a leaf node of the current tree, and let $H_t \subseteq H$ denote this set of active experts. As in the traditional experts problem, the game lasts for T steps, and at each time t, the learner plays according to some distribution p_t over H_t, and suffers expected loss $\langle p_t, \ell_t \rangle = \mathbb{E}_{i\sim p_t}[\ell_{t,i}]$, according to some loss vector $\ell_t \in [0,1]^{H_t}$. We will focus on the full-information setting, in which the whole ℓ_t is revealed after his play.

We consider the adversarial setting in which the loss vectors can be arbitrary, as well as the stochastic setting in which they are sampled i.i.d. according to some fixed but unknown distribution with mean μ. Following the standard approach, we evaluate the learner by its regret, defined as $\sum_{t\in[T]} \mathbb{E}_{i\sim p_t}\left[\ell_{t,i}\right] - \sum_{t\in[T]} \ell_{t,i^*}$, where i^* is the best expert in the whole set H which minimizes the second sum. In the stochastic setting, a related notion called pseudo regret is often considered, defined as $\sum_{t\in[T]} \mathbb{E}_{i\sim p_t}\left[\mu_i\right] - \sum_{t\in[T]} \mu_{i^*} = \sum_{t\in[T]} \mathbb{E}_{i\sim p_t}\left[\Delta_i\right]$, with $i^* = \arg\min_{i\in H} \mu_i$ and $\Delta_i = \mathbb{E}[\ell_{t,i} - \ell_{t,i^*}]$, called the gap of expert i. Let $\Delta = \min_{i\neq i^*} \Delta_i$.

We will assume that the set H_t at each time t covers or represents all others well as follows.

Assumption 2.1 *At any time t, H_t forms an ε-cover in the sense that for any $h \in H$, there is some expert in H_t, denoted as $\pi_t(h)$ and called his representative, such that $|\ell_\tau(h) - \ell_\tau(\pi_t(h))| \le \varepsilon$ for any $\tau < t$. Moreover, if a new expert h branches out at the start of time $t+1$, it comes from $\pi_t(h)$.*

Based on this assumption with a small ε, it makes sense to use only the active set H_t for predictions at time t, as any other expert looks similar to his representative so far. Here we use a fixed ε over all steps mainly to simplify our presentation; it is straightforward to generalize our results to allow a different ε_t for a different time t.

Regarding how the new experts are revealed, there are two natural possibilities. The first is that it is controlled by an adversary, who can do this in an arbitrary way as long as the ε-cover assumption is not violated. The second is that the learner has the full control. This is particularly relevant in the scenario when there are an extremely large, possibly infinite, number of experts, so that it may be beneficial to represent them with only a small subset. Then, the learner also needs to decide how to do the branching to minimize the regret.

Next, let us turn to the lifelong learning problem, and an expert now becomes a pair of representation and predictor, as discussed in the introduction. Here, we adopt the formulation of (Wu et al., 2019) and extend it to the branching setting, in which the whole representation set G and predictor set H are not given initially, but revealed gradually.

In the problem, there are K tasks which arrive sequentially, with task k lasting for some T_k steps, for a total of T time steps. For each time t which corresponds to step s of task k, we will write either t or (k, s) when there is no confusion. For each $t = (k, s)$, we need to choose a representation g_t with a predictor h_t from some growing sets $G_t \subseteq G$ and $H_t^{g_t} \subseteq H$, respectively, which jointly provide a decision. After this decision, we suffer some loss $\ell_t(g_t, h_t)$, and get to know $\ell_t(g, h)$ for any $g \in G_t$ and $h \in H_t^g$, before proceeding to the next step. Similar to the branching experts problem, we also assume that G_t grows with a tree structure, and so does H_t^g for each $g \in G_t$. In addition, we assume that if g branches out at time t from its parent g', then H_t^g grows from $H_{t-1}^{g'}$, with $H_t^g \supseteq H_{t-1}^{g'}$.

To model that different tasks are related in lifelong learning, we follow previous works and assume the existence of a good representation which is shared by all tasks, although they can still use different predictors. We would like to compare an online algorithm with such an offline one, which motivates the following notion of regret:

$$\sum_{k,s} \mathbb{E}\left[\ell_{k,s}(g_{k,s}, h_{k,s})\right] - \sum_{k,s} \ell_{k,s}(g^*, h_k^*),$$

where the expectation is taken over the sampling of $(g_{k,s}, h_{k,s})$ from the algorithm, while g^* denotes the optimal representation in G and $h_k^* \in H$ its optimal predictor in task k of the best offline algorithm, which minimize the second sum above. We also consider the stochastic setting, in which for each task k, the loss function $\ell_{k,s}$, for each step s, is sampled i.i.d. from some distribution with mean $\mathbb{E}[\ell_{k,s}(g, h)] = \mu_k(g, h)$ for any (g, h). Let $\mu_k(g) = \min_h \mu_k(g, h)$, and we assume the existence of some $g^* \in G$ which remains the best in every task, so that $\mu_k(g^*) < \mu_k(g)$ for any k and $g \neq g^*$. A related notion of pseudo regret is defined as

$$\sum_{k,s} \mathbb{E}\left[\mu_k(g_{k,s}, h_{k,s})\right] - \sum_{k,s} \mu_k(g^*),$$

where the expectation is again over the sampling of $(g_{k,s}, h_{k,s})$. As in the one-task case, here we also have similar notions of gaps:

$$\Delta_G = \min_k \min_{g \neq g^*} (\mu_k(g) - \mu_k(g^*)) \text{ and } \Delta_H = \min_k \min_{h \neq h_k^*} (\mu_k(g^*, h) - \mu_k(g^*)).$$

Finally, as in the single-task scenario, now we will also assume that G_t and H_t^g form some covers as follows.

Assumption 2.2 *At any time t, G_t forms an ε_G-cover in the sense that for any $g \in G$, there is some element in G_t, denoted as $\pi_t(g)$, such that $|\ell_\tau(g, h) - \ell_\tau(\pi_t(g), h)| \leq \varepsilon_G$ for any $\tau < t$ and $h \in H$. Moreover, if some g branches out at the start of time $t + 1$, it comes from $\pi_t(g)$.*

Assumption 2.3 *At any time t, for any $g \in G_t$, H_t^g forms an ε_H-cover in the sense that for any $h \in H$, there is some element in H_t^g, denoted as $\pi_t^g(h)$, such that $|\ell_\tau(g, h) - \ell_\tau(g, \pi_t^g(h))| \leq \varepsilon_H$ for any $\tau < t$. Moreover, if some h branches out at the start of time $t + 1$, it comes from $\pi_t^g(h)$.*

3. Single-task learning

In this section, we consider the single-task setting with branching experts, and we focus on the case that there is a finite number of experts and the branching of experts is controlled by the environment. We will present two algorithms. The first is simpler and achieves tight regret bounds in both adversarial and stochastic setting. However, when used in the lifelong learning setting later, we are unable to prove a tight regret bound in the stochastic setting when suboptimal representations have large gaps. Thus, we provide another algorithm which achieves slightly worse regret bounds in the single-task case but works better for the case of lifelong learning.

Algorithm 1. Our first algorithm is based on the Hedge algorithm, but to deal with the case of branching experts, we need to make some changes. In particular, at a step when there are new experts branching out, we need to decide the distribution for playing them as well as existing ones. While there are previous algorithms which work in the adversarial setting, our goal is to have a single algorithm which works simultaneously in both the adversarial and stochastic setting.

The algorithm works as follows. At any step t, let m be the number of branching steps before, let $H_t = [N_t]$ be the current active set of experts, and similarly to Hedge algorithm (with time-varying learning rates), we define the probability of playing each expert $i \in H_t$ as

$$p_{t,i} = \frac{e^{-\eta_t \bar{L}_{t-1,i}}}{\sum_{j \in H_t} e^{-\eta_t \bar{L}_{t-1,j}}}, \text{ for } \eta_t = \sqrt{\frac{m \log N_t}{t}}, \tag{1}$$

where $\bar{L}_{t-1} = \sum_{\tau=1}^{t-1} \bar{\ell}_\tau$, with $\bar{\ell}_{\tau,i} = \ell_{\tau,\pi_{\tau+1}(i)}$ for each τ. Recall that for each τ, $\pi_{\tau+1}(i)$ is the representative of i in $H_{\tau+1}$, with $\pi_{\tau+1}(i) = i$ if $i \in H_{\tau+1}$. That is, we update each expert using the losses of his representatives until he is revealed. Note that such surrogate losses are close to the true ones based on Assumption 2.1.

Note that compared to the algorithm of (Cohen and Mannor, 2017) which resets the distribution each time branching occurs, our algorithm updates distributions in a smooth way so that the learning continues across the branching steps, which is beneficial in the stochastic setting as the loss distributions remain fixed through time. On the other hand, although the algorithm of (Gofer et al., 2013) also has a smooth update rule, it allows suboptimal experts to keep taking probability measure from the optimal expert, which keeps it from achieving tight regret in the stochastic setting. Our update rule avoids this problem.

Let T be the total number of time steps, M the total number of branching steps, and $N = N_T$ the final number of experts. Assume that Assumption 2.1 holds. Then the regret of Algorithm 1 in the adversarial setting is guaranteed by the following, which we prove in Appendix A.1.

Theorem 1 *For the adversarial branching experts problem, the expected regret of Algorithm 1 is at most* $\mathcal{O}\left(\sqrt{TM \log N} + \varepsilon T\right)$.

Note that with $M = O(1)$ and $\varepsilon = 0$, we recover the existing bound of $\mathcal{O}(\sqrt{T \log N})$ in the non-branching case. However, when there is a super-constant number of branching steps, the regret bound above becomes higher with an extra $\sqrt{M}$ factor. This factor is in fact unavoidable, as will be shown later in Theorem 8. This means that in the adversarial setting, allowing branching makes the problem strictly harder.

On the other hand, our next theorem shows that the hardness disappears in the stochastic setting, and in fact the same algorithm, our Algorithm 1, can achieve regret bounds independent of M.

Theorem 2 *For the stochastic branching experts problem, the pseudo-regret of Algorithm 1 is at most* $\min\{\mathcal{O}(\frac{1}{\Delta}\log N), \mathcal{O}(\sqrt{T\log N})\}$ *if* $\varepsilon \leq \frac{\Delta}{8}$ *and* $\mathcal{O}(\sqrt{T\log N} + \varepsilon T)$ *otherwise.*

We will prove the theorem in Appendix A.2. Note that with $\varepsilon = 0$, we recover previous bounds in the non-branching case.

Algorithm 2. Our second algorithm is inspired by works of Gofer et al. (2013); Gaillard et al. (2014); van Erven and Koolen (2016) and as a variant of the Prod algorithm. To motivate our algorithm, let us start from that of (Gofer et al., 2013) which is also based on the Hedge algorithm, but now at each branching step, each parent splits his weight to all his newly revealed children. Here, we further readjust the weights slightly to keep the weight of the (unknown) optimal expert from becoming too small. Moreover, as that algorithm uses a fixed learning rate, which needs to be set appropriately with some prior knowledge of the world, we borrow ideas from Gaillard et al. (2014); van Erven and Koolen (2016) for handling this issue. As the appropriate learning rate now also depends on the number of branching steps, we find it easier to take the approach of creating for each expert several artificial experts with different learning rates.

Formally, at step t, we maintain a set $Q_t = \{2^{-\tau} : \text{integer } \tau \in [0, \log_2 t]\}$ of learning rates, and for each active expert $i \in [N_t]$, we create $|Q_t|$ artificial experts (i, η), for $\eta \in Q_t$, and we work on this set of $\hat{N}_t = N_t \times |Q_t|$ artificial experts instead. Note that in addition to the branching caused by original experts, we now also have branching caused by the learning rates (when $\lfloor \log_2 t \rfloor \geq \log_2(t-1)$), which happens at most $\mathcal{O}(\log T)$ times. Each time a new learning rate η is added to Q_t, each active expert $(i, 2\eta)$ branches out a new one (i, η).

Next, we define how the weights are updated. At each time t, each active expert (i, η) has a weight $W^{\eta}_{t,i}$, with $W^{\eta}_{0,i} = 1$ initially, and we play (i, η) with probability $p^{\eta}_{t,i} = \eta W^{\eta}_{t,i}/Z_t$ for a normalization factor Z_t. Then we update the weights in the following way. First, we compute auxiliary weights:

$$\hat{W}^{\eta}_{t+1,i} = W^{\eta}_{t,i}(1 + \eta r_{t,i}) \text{ and } \tilde{W}^{\eta}_{t+1,i} = \alpha_{t+1} \cdot \frac{1}{\hat{N}_{t+1}} + (1 - \alpha_{t+1}) \cdot \hat{W}^{\eta}_{t+1,i},$$

where $r_{t,i} = \langle p_t, \bar{\ell}_t \rangle - \bar{\ell}_{t,i}$, with $\bar{\ell}_{t,i}$ defined as in Algorithm 1, and $\alpha_{t+1} = 1/(t+1)^2$. Next, for each active (i', η') at step t, let $C_{i',\eta'}$ denote the set consisting of himself and his newly revealed children if any, and compute for each $(i, \eta) \in C_{i',\eta'}$ the weight

$$W^{\eta}_{t+1,i} = \tilde{W}^{\eta'}_{t+1,i'}/|C_{i',\eta'}|.$$

Here again, we assume the ε-cover assumption. Then the following theorem guarantees the regret of Algorithm 2 in the adversarial setting, which we prove in Appendix A.3. Let us remark that although our algorithm creates artificial experts, the regret is still measured against original experts, and it depends on the number of branching steps of i^*'s representatives, denoted as M_{i^*}.

Theorem 3 *For the adversarial branching experts problem, the expected regret of Algorithm 2 is at most* $\mathcal{O}(\sqrt{T(M_{i^*}\log N + \log T)} + \varepsilon T)$.

Note that the bound above is worse than that of Algorithm 1, with an additional dependency on $\log T$. On the other hand, it has the advantage of replacing the factor M of total number of branching steps by the smaller M_{i^*}, which is crucial for our lifelong learning algorithm later. The following theorem shows that Algorithm 2 also works in the stochastic setting, but again with slightly worse regret bounds than those of Algorithm 1. We prove the theorem in Appendix A.4.

Theorem 4 *For the stochastic branching experts problem, the pseudo-regret of Algorithm 2 is at most* $\min\{\mathcal{O}(\frac{1}{\Delta}\log(NT)), \mathcal{O}(\sqrt{T\log(NT)})\}$ *if* $\varepsilon \le \frac{\Delta}{8}$ *and* $\mathcal{O}(\sqrt{T\log(NT)} + \varepsilon T)$ *otherwise.*

4. Lifelong learning with branching experts

In this section, we consider the lifelong learning problem, in which there is a sequence of tasks arriving sequentially. Following (Wu et al., 2019) which works for the non-branching case, we would like to apply our single-task algorithms to the lifelong learning problem in the more general branching case. The key here is also to design appropriate loss functions to update the algorithms, but now we face an additional challenge of having branching representations which also have branching predictors, and needing to make sure that the ε-cover condition holds so that we can apply our previous algorithms.

Algorithm 3. Our initial attempt, based on (Wu et al., 2019), was to use Algorithm 1 to learn both representations and predictors. While it achieves a tight adversarial regret bound, it does not seem to guarantee a good bound for the stochastic setting. Thus, we opt to use Algorithm 2 for learning representations instead. More precisely, we will run a single copy of Algorithm 2, denoted as $\texttt{alg}_G$, for learning representations, using some designed loss functions defined later in (2), and we will update it continuously across different tasks. On the other hand, for each active representation $g \in G_t$ at step t, we will run a separate copy of Algorithm 1, denoted as $\texttt{alg}_H^g$ for learning its predictors, and we will reset it and redo its learning when entering a new task. In more details, our algorithm works as follows at any step t.

If this is the start of a new task, we reset $\texttt{alg}_H^g$ and redo its learning. Otherwise, we update $\texttt{alg}_H^g$ based on the surrogate loss functions $\bar{\ell}_\tau^g$, for $\tau \le t-1$, defined as $\bar{\ell}_\tau^g(h) = \ell_\tau(\bar{g}_\tau, \bar{h}_\tau)$ where $\bar{g}_\tau = \pi_{\tau+1}(g)$ and $\bar{h}_\tau = \pi_{\tau+1}^{\bar{g}_\tau}(h)$. Let $\mathcal{H}_t^g$ denote the resulting distribution of $\texttt{alg}_H^g$.

Next, we update $\texttt{alg}_G$ for learning representations based on the artificial loss functions $\hat{\ell}_\tau$ for $\tau \le t-1$, defined as

$$\hat{\ell}_\tau(g) = \mathop{\mathbb{E}}_{h\sim\mathcal{H}_\tau^{\bar{g}_\tau}} \left[\ell_\tau(\bar{g}_\tau, h)\right]. \tag{2}$$

Note that for any t, with $\mathcal{G}_t$ denoting the distribution of $\texttt{alg}_G$ at step t, $\mathbb{E}_{g\sim\mathcal{G}_t}[\hat{\ell}_t(g)]$ equals the expected loss of our algorithm at that step.

Regret bounds. Let T be the total number of steps, K the number of tasks, N_G the final number of representations, and N_H the maximal number of predictors over tasks. Moreover, let M_{g^*} be the total number of branching steps of g^*'s representatives and M_H that of their predictors. Assume that for some ε_G and ε_H, Assumptions 2.2 & 2.3 hold, and let $\varepsilon = \max\{\varepsilon_H, \varepsilon_G\}$. Then in the adversarial setting, we have the following regret bound, which we prove in Appendix B.1.

Theorem 5 *For the adversarial lifelong learning problem, the expected regret of our algorithm is at most $\mathcal{O}(R_G + \sqrt{TM_H \log N_H} + \varepsilon T)$, where $R_G = \sqrt{T(M_{g^*} \log N_G + \log T)}$.*

Note that $M_H \geq K$ as it sums the total number of predictor-branching over tasks. As one can see, in the non-branching case, with $M_{g^*}, M_H = \mathcal{O}(1)$ and $\varepsilon = 0$, we recover the previous bound in (Wu et al., 2019). On the other hand, for a super-constant M_{g^*}, the bound becomes larger. As we will show in Theorem 8, this extra M_{g^*} factor is in fact unavoidable. Next, we show that the same algorithm also works in the stochastic setting, which we prove in Appendix B.2.

Theorem 6 *Let $\Delta = \min\{\Delta_G, \Delta_H\}$, $R_H = \sqrt{TK \log N_H}$ and recall R_G from Theorem 5. Then for the stochastic lifelong learning problem, the pseudo-regret of Algorithm 3 is at most $\min\{\mathcal{O}(\frac{1}{\Delta}(\log(N_G T) + K \log N_H)), \mathcal{O}(R_G + R_H)\}$ when $\varepsilon \leq \frac{\Delta}{8}$ and $\mathcal{O}(R_G + R_H + \varepsilon T)$ otherwise.*

Let us remark that for the case with large Δ and small ε, the regret bound matches that in the non-branching case of (Wu et al., 2019). It does not have the M_{g^*} factor in the adversarial setting, which shows that with large Δ, branching again does not make the problem harder in the stochastic setting. On the other hand, when Δ is small enough, the regret increases to that of the adversarial setting and the factor M_{g^*} appears (in R_G), unlike in the one-task case in Theorem 2. Such a dependence on M_{g^*} is in fact unavoidable as shown later in Theorem 9. This interesting phenomenon demonstrates some fundamental difference caused by having multiple tasks. In particular, a suboptimal representation can have different gaps in different tasks, ranging from very small, which determines Δ, to very large, and this makes the problem of lifelong learning much more challenging. In contrast, the gap of a suboptimal expert in the one-task case stays the same for the whole time, and this nice property is commonly used for guaranteeing small regret in previous works.

5. Infinite case

In this section, we consider the case in which the online algorithm has full control of the branching, and we will only discuss the case of lifelong learning, as the single-task case follows by restricting to $K = 1$ and $N_G = 1$. Now the online algorithm has the whole G and H available at the beginning, but as their sizes may be extremely large, possibly infinite, it would be better to use small subsets to cover them at each time. The goal is to bound the regret in terms of their potentially much smaller covering numbers, a standard notion which we define in Appendix B.3.

We would like to apply our algorithm for the finite case, but now we also need to decide how to do the branching. Note that in order to apply our results in the previous section, we have to do the branching whenever the ε-cover assumption is violated. However, to avoid branching out too many representations or predictors, we follow the approach of (Cohen and Mannor, 2017) and only branch out a minimal number of them so that some packing conditions are also met. More precisely, at each time t, after seeing the loss function ℓ_t, we form G_{t+1} by initializing it with G_t and iteratively adding any $g \in G$ which is not ε_G-covered by the current G_{t+1}, in the sense that for any $g' \in G_{t+1}$, $|\ell_\tau(g,h) - \ell_\tau(g',h)| > \varepsilon_G$ for some $h \in H$ and $\tau \leq t$. We will stop growing G_t as soon as there is no more such uncovered g. After we finish forming G_{t+1}, we form H^g_{t+1} for each $g \in G_{t+1}$ in the following way. First, we initialize H^g_{t+1} with $H^{\pi_t(g)}_t$ where $\pi_t(g)$ is the representative of g at step t if $g \notin G_t$ and is itself otherwise. Then we iteratively add any $h \in H$ which is not ε_H-covered by the current H^g_{t+1}, in the sense that for any $h' \in H^g_{t+1}$, $|\ell_\tau(g,h) - \ell_\tau(g,h')| > \varepsilon_H$ for some $\tau \leq t$.

Assume that the representatives of g^* branch M_{g^*} times while their predictors branch M_H times during the T time steps. Then we have the following, which we prove in Appendix B.3.

Corollary 7 *For the lifelong learning problem, assume that the representation set G has a finite ε_G-covering of size N_G, and the predictor set of the best representation has a finite ε_H-covering of size N_H, with $\varepsilon_G \leq \varepsilon_H/4$. Then our Algorithm 3 achieves a regret bound of $\mathcal{O}(\sqrt{T(M_{g^*}\log N_G + \log T)} + \sqrt{TM_H \log N_H} + \varepsilon T)$ in the adversarial setting, and with M_H replaced by K in the stochastic setting.*

Let us remark that we do not provide a bound for the stochastic setting with large gaps as it does not seem natural to assume the existence of large gaps in the infinite case. Note that by setting $K = 1$ and $N_G = 1$, we obtain a stochastic regret bound of $\mathcal{O}(\sqrt{T \log N} + \varepsilon T)$ for the one-task case, where N is the covering number of a possibly much larger set of experts.

6. Lower Bounds

In Appendix C.1, we discuss why previous algorithms for the branching experts problem do not work well simultaneously in both adversarial and stochastic setting, which motivates us to design different algorithms. Next, we provide lower bounds for the lifelong learning problem in both settings, which show the tightness of the upper bounds achieved by our algorithms.

6.1. Adversarial setting

Recall the notation that T denotes the total number of steps, K the number of tasks, N_G the final number of representations, and N_H the maximal number of predictors over tasks, while M_G and M_H are the total numbers of branching steps for representations and predictors respectively. Then we have the following lower bound which we prove in Appendix C.2.

Theorem 8 *In the adversarial setting, the lifelong learning problem with branching has an expected regret lower bound of $\Omega(\sqrt{TM_G \log(N_G/M_G)} + \sqrt{TM_H \log(N_H K/M_H)})$.*

The lower bound shows that our upper bounds in Theorem 1 and Theorem 5 are almost tight.

6.2. Stochastic setting

Next, we consider the lifelong learning problem in the stochastic setting. Note that having a growing number of predictors does not make the problem harder in the stochastic setting, as shown by our algorithm. On the other hand, having a growing number of representations does make the problem harder, as shown in the following, which we prove in Appendix C.3.

Theorem 9 *Let $K' = \min\{K, M_G\}$. Assume that $N_G \geq 4K'$ and there are at least 4 predictors in each task. Then in the stochastic setting, the lifelong learning problem has a pseudo-regret lower bound of $\Omega(\sqrt{TK' \log(N_G/K')} + \sqrt{TK \log N_H})$.*

Let us remark that the first term in the regret bound above comes from that of learning representations, and it has a $\sqrt{K'}$ factor. This shows that the learning problem becomes more difficult, compared to that with a fixed number of representations ($M_G = 1$) and that with only one task ($K = 1$). In fact, for learning representations, the combination of $M_G \geq 2$ and $K \geq 2$ pushes the stochastic setting close to the adversarial one. This is because an optimal representation can branch out a suboptimal one with a high expected loss in a new task (even though it was optimal previously). This can force a large regret lower bound just as in the adversarial setting, as shown in the proof.

Acknowledgments

The main part of this project was done while YSW was a research assistant at Institute of Information Science, Academia Sinica, Taiwan. YSW acknowledge the support by European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 801199 and the support by the Independent Research Fund Denmark, grant number 0135-00259B.

References

Pierre Alquier, The Tien Mai, and Massimiliano Pontil. Regret bounds for lifelong learning. In *Proceedings on the International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 261–269, 2017.

Maria-Florina Balcan, Avrim Blum, and Santosh Vempala. Efficient representations for lifelong learning and autoencoding. In *Proceedings of the Conference on Learning Theory (COLT)*, pages 191–210, 2015.

Jonathan Baxter et al. A model of inductive bias learning. *Journal Of Artificial Intelligence Research*, 12(149-198):3, 2000.

Nicolo Cesa-Bianchi and Gabor Lugosi. *Prediction, Learning, and Games.* Cambridge University Press, 2006.

Alon Cohen and Shie Mannor. Online learning with many experts. *arXiv preprint arXiv:1702.07870*, 2017.

Giulia Denevi, Dimitris Stamos, Carlo Ciliberto, and Massimiliano Pontil. Online-within-online meta-learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 13089–13099, 2019.

Pierre Gaillard, Gilles Stoltz, and Tim van Erven. A second-order bound with excess losses. In *Proceedings of the Conference on Learning Theory (COLT)*, pages 176–196, 2014.

Eyal Gofer, Nicolo Cesa-Bianchi, Claudio Gentile, and Yishay Mansour. Regret minimization for branching experts. In *Proceedings of the Conference on Learning Theory (COLT)*, pages 618–638, 2013.

Elad Hazan, Tomer Koren, Roi Livni, and Yishay Mansour. Online learning with low rank experts. In *Proceedings of the Conference on Learning Theory (COLT)*, pages 1096–1114, 2016.

Haipeng Luo and Robert E Schapire. Achieving all with no parameters: AdaNormalHedge. In *Proceedings of the Conference on Learning Theory (COLT)*, pages 1286–1304, 2015.

Andreas Maurer. Algorithmic stability and meta-learning. *Journal of Machine Learning Research (JMLR)*, 6(Jun):967–994, 2005.

Andreas Maurer. Transfer bounds for linear feature learning. *Machine Learning*, 75(3): 327–350, 2009.

Andreas Maurer, Massi Pontil, and Bernardino Romera-Paredes. Sparse coding for multitask and transfer learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 343–351, 2013.

Jaouad Mourtada and Stéphane Gaïffas. On the optimality of the hedge algorithm in the stochastic regime. *Journal of Machine Learning Research (JMLR)*, 20(83):1–28, 2019.

Jaouad Mourtada and Odalric-Ambrym Maillard. Efficient tracking of a growing number of experts. *arXiv preprint arXiv:1708.09811*, 2017.

Anastasia Pentina and Shai Ben-David. Multi-task and lifelong learning of kernels. In *Proceedings of the International Conference on Algorithmic Learning Theory (ALT)*, pages 194–208, 2015.

Anastasia Pentina and Christoph Lampert. A pac-bayesian bound for lifelong learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 991–999, 2014.

Paul Ruvolo and Eric Eaton. Ella: An efficient lifelong learning algorithm. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 507–515, 2013.

Tim van Erven and Wouter M Koolen. Metagrad: Multiple learning rates in online learning. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3666–3674. 2016.

Yi-Shan Wu, Po-An Wang, and Chi-Jen Lu. Lifelong optimization with low regret. In *Proceedings on the International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 89, pages 448–456, 2019.