Lifelong Optimization with Low Regret

Yi-Shan Wu
Institute of Information Science
Academia Sinica, Taiwan

Po-An Wang
Institute of Information Science
Academia Sinica, Taiwan

Chi-Jen Lu
Institute of Information Science
Academia Sinica, Taiwan

Abstract

In this work, we study a problem arising from two lines of works: online optimization and lifelong learning. In the problem, there is a sequence of tasks arriving sequentially, and within each task, we have to make decisions one after one and then suffer corresponding losses. The tasks are related as they share some common representation, but they are different as each requires a different predictor on top of the representation. As learning a representation is usually costly in lifelong learning scenarios, the goal is to learn it continuously through time across different tasks, making the learning of later tasks easier than previous ones. We provide such learning algorithms with good regret bounds which can be seen as natural generalization of prior works on online optimization.

1 Introduction

State-of-the-art machine learning algorithms can now solve many problems even better than humans. However, machines are still far from being intelligent, and they typically require a large amount of training data in order to do well for a task. On the other hand, humans are able to learn a new task very efficiently with little training, perhaps by utilizing knowledge accumulated from previously learned tasks. This motivates the study called lifelong learning (Thrun and Pratt, 1998), which aims to perform better over time by transferring information learned from previously tasks to later ones, under the belief that there are some commonalities across tasks.

In order to benefit from the learning of previous tasks, the new task must have something in common with previous ones. To model such commonalities among tasks, Baxter et al. (2000) considered a task environment, from which all the tasks are generated according to a common probability distribution. This was followed by several subsequent works such as those of Maurer (2005, 2009); Pentina and Lampert (2014, 2015). A different model which is natural for classification/regression problems, studied by e.g. Ruvolo and Eaton (2013); Pentina and Ben-David (2015); Alquier et al. (2017), is that different tasks share some common feature representation. The tasks are different in the sense that they need different predictors built on top the representation in order to make the predictions. As a concrete example, one typical way to classify a raw data x is to first extract a feature representation $g(x)$ and then build a predictor h on top of it to make the prediction $h(g(x))$. The feature extractor g is usually large and complex, but with such a useful feature representation, the predictor can be very simple. Following previous works, we will use the terms "representation" and "predictor", although there are other forms of commonality/difference among tasks. Let us remark that while most prior works focused on learning with fixed architectures, there were also empirical studies (Rusu et al., 2016; Lee et al., 2017) on the possibility of evolving the network structures over different tasks.

Several settings have been considered in the general theme of lifelong learning which differ in how samples are given. The first one, often referred to as learning-to-learn, is a batch setting in which the training samples of all the tasks are available at the beginning (Baxter et al., 2000; Maurer, 2005, 2009; Maurer et al., 2013). The second is a batch-within-online setting in which tasks arrive sequentially but at the start of each task, all its training samples are available (Balcan et al., 2015; Pentina and Ben-David, 2015; Pentina and Urner, 2016; Alquier et al., 2017). The third is known as online multi-task learning (Agarwal et al., 2008; Abernethy et al., 2007; Dekel et al., 2007; Cavallanti et al., 2010; Saha et al., 2011; Lugosi et al., 2009; Murugesan et al., 2016). In this setting, learning also proceeds sequentially in rounds, but in each round, all the tasks are present and

Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics (AISTATS) 2019, Naha, Okinawa, Japan. PMLR: Volume 89. Copyright 2019 by the author(s).

there is a new sample for each task. The last setting is a fully online one, in which both the tasks and their samples arrive sequentially so that in each round, we only see a sample from one single task (Alquier et al., 2017). To measure the performance of a learning algorithm, different settings have their own natural choices. For the first two settings, in which each task has all its training examples at the beginning, a natural choice is to measure the number of samples or tasks needed to guarantee some notion of generalization error. For the last two settings, in which the samples of each task arrive one after one, an often adopted measure is the regret.

A different line of works which is related to (especially the last setting of) lifelong learning is on the topic known as online optimization (see e.g. Cesa-Bianchi and Lugosi (2006)). While most prior works on lifelong learning focussed on classification/regression problems, works on online optimization have touched upon a broader range of optimization problems, including making discrete decisions and optimizing general convex functions. In the problem of online optimization, we have to make repeated decisions and then suffer corresponding losses, according to some loss functions, for a number of iterations. A standard goal is to minimize the regret, which is the difference between the total loss we suffer and that of an offline algorithm using a fixed decision. Thus, one can see such an online optimization problem as a special case of the lifelong learning problem with only one task. Some works on online optimization, such as Herbster and Warmuth (1998); Zinkevich (2003), have also considered the generalization of measuring regret against an offline algorithm which can change decisions for a number of times. This generalization is used to model concept shifts in learning, but these works do not assume any relationship among these different concepts (offline decisions). Again, one can see this variant as a special case of the lifelong learning problem, which does have more than one task but these tasks have nothing in common.

In this work, we would like to put the study of online optimization and lifelong learning in the same framework. To avoid confusion with different settings of lifelong learning, we call ours the lifelong optimization problem. In this problem, we have a sequence of tasks arriving sequentially and for each task, we need to make a sequence of decisions one after one. After making each decision, a loss function is revealed and we suffer a corresponding loss. Each decision is made by combining a representation with a predictor. The relationship among tasks comes from sharing a common representation, while the difference among tasks comes from needing different predictors. To capture this, we measure the regret by comparing against an offline algorithm which must use a fixed representation for all the tasks but is allowed to use different predictors for different tasks. Note that Alquier et al. (2017) considered a similar setting as ours, but using a different regret measure. More precisely, our regret treats each decision as equally important, just as is usually considered in online optimization, while the regret measure of Alquier et al. (2017) treats each task equally important. Although their regret definition may be useful in some settings, such as that of learning-to-learn considered in (Maurer et al., 2016), a large number of tasks and a large number of steps in each task are necessary for making their regret bounds small.

We believe that in natural scenarios of lifelong learning, we usually have no control over the number of tasks we will see and how long each task is. Therefore, we would like to understand if it is possible to achieve good regret bounds in general without relying on such assumptions. Moreover, as learning the representations is typically much more costly than learning predictors in lifelong learning, we would like to understand if it is possible to learn them continuously through time across different tasks, instead of relearning them in different tasks. If so, this would allow one to learn new tasks faster, by saving the time for learning the representation. We answer these questions affirmatively.

We first consider the full-information setting in which we get to know the whole loss function after our decision at each step. In Section 3, we focus on the adversarial setting, and we start from the case that the representations and predictors are finite, with size G and H respectively, but the loss functions can be arbitrary. For this, we provide an efficient algorithm achieving a regret of $\mathcal{O}\left(\sqrt{T\log G}+\sqrt{KT\log H}\right)$, where T is the total number of steps and K is the number of tasks. Compared to relearning the representation for each task, which has a regret of $\mathcal{O}\left(\sqrt{KT\log G}+\sqrt{KT\log H}\right)$, our bound prevents the number of tasks from affecting the learning of representations. This makes our algorithm appealing when $G \gg H$ which is natural in scenarios of lifelong learning. In addition, we also consider cases with either infinite representations or infinite predictors, and we show that good regret bounds can still be achieved under natural assumptions, although with some achieved via inefficient algorithms. Then in Section 4, we consider the stochastic setting in which the loss functions in each task are drawn i.i.d. from some unknown but fixed distribution, although different tasks can have different distributions. We focus on the case with finite G and H, as regret bounds in other cases usually do not get better than those in the adversarial setting. We show that when the optimal representation and their accompanying predictors are considerably better than suboptimal ones, a regret in the order of $\log T$ can be achieved.

In Section 5, we consider the bandit setting in which the feedback we receive at each step is only the loss value instead of the whole loss function. Such a restriction causes a more serious problem in our lifelong optimization scenario. Now, in addition to worrying about exploration of predictors, we also have to worry about the exploration of representations. In particular, more aggressive exploration of representations seems needed, because otherwise we would not receive enough feedbacks to learn their predictors well, and without learning the predictors well, we cannot choose the representations appropriately as we do not know how good they are. On the other hand, do more exploration and less exploitation would likely lead to larger regret bounds. Our results in the bandit setting are in fact weaker than those in the full-information one. We show that in the adversarial setting, a regret of $\mathcal{O}(\sqrt{TGH\log(GH^K)})$ can be achieved. On the other hand, we show that in the stochastic setting, a regret in the order of $\log T$ is still possible, but with slightly worse dependency on other parameters.

Our results show that indeed one can learn the representation continuously which helps make the learning of new tasks more efficiently. Finally, let us remark that some of our results can be seen as natural generalization of previous results on online optimization. Take our result for finite G and H as an example. When there is only one task ($K = 1$), our bound recovers the known regret bound of $\mathcal{O}(\sqrt{T\log G})$ for the experts problem. When there are K tasks but they share no commonality ($G = 1$), our bound recovers the known regret bound of $\mathcal{O}(\sqrt{KT\log H})$ when measured against an offline algorithm which can change actions K times. We hope that more works on online optimization can also be generalized to our setting of lifelong optimization, and we see our work as providing initial steps towards this direction.

2 Preliminaries

To motivate the abstract problem we will formulate next, let us first consider the online classification problem as a concrete example. In the problem, there is a sequence of data arriving one at a time for us to make classification. For the data x_t in iteration t, we choose a feature representation g_t and a predictor h_t to make the classification $h_t(g_t(x_t))$. Then we suffer some corresponding loss $\ell_t(g_t, h_t)$, where ℓ_t depends on x_t and its true label. This problem belongs to the more general problem of online optimization, and the problem of lifelong optimization we consider is a natural generalization of online optimization. In our problem, we also have to make repeated decisions and then suffer corresponding losses for a number of iterations, but these iterations are now divided into tasks which are related in a certain way, so that we need a more general regret notion.

Formally, suppose there are K tasks which arrive one after one, with the k-th task lasting for T_k steps, for a total of T steps. To avoid possible confusion, we will refer to these T steps as T iterations, with each iteration corresponding to some step s in some task k. For each task k and each step s in it, we need to choose a representation $g_{k,s}$ and an accompanying predictor $h_{k,s}$ from some sets $\mathcal{G}$ and $\mathcal{H}$, respectively, which jointly provide a decision for us. After making this decision, we suffer some loss $\ell_{k,s}(g_{k,s}, h_{k,s})$ according to some loss function $\ell_{k,s}$, receive some feedback information, and then proceed to the next iteration. In the full-information setting, the feedback information we receive is the whole loss function $\ell_{k,s}$, while in the bandit setting, we only receive the loss value $\ell_{k,s}(g_{k,s}, h_{k,s})$. For simplicity of presentation, we assume that each loss value falls in the interval $[0, 1]$; the generalization is straightforward. In addition, we will assume that we know exactly when each task starts, as were usually assumed in prior works in lifelong learning, although most of our results can be extended to the general case without this assumption. Moreover, in the scenario of lifelong learning, one usually has in mind that learning representations is much harder than learning individual predictors. Thus, we will assume that either the size of $\mathcal{G}$, denoted as G, is much larger than that of $\mathcal{H}$, denoted as H, when both are finite, or the dimensionality of $\mathcal{G}$ is much larger than that of $\mathcal{H}$ when both are infinite, or the loss functions on $\mathcal{H}$ have special structures.

Similarly to the traditional online optimization problem, our goal here is also to minimize some notion of regret. To model that different tasks are related, we follow previous works and assume the existence of some good representation which is shared by all tasks, although different tasks still have their own predictors. This motivates us to consider the following version of (expected) regret:

$$\mathbb{E}\left[\sum_{k,s}\ell_{k,s}(g_{k,s},h_{k,s}) - \min_{g,h_1,\cdots,h_K}\sum_{k,s}\ell_{k,s}(g,h_k)\right] \quad (1)$$

where the expectation is taken over the algorithm's own randomness. The first sum in the expectation corresponds to the total loss of our online algorithm while the second sum corresponds to that of the best offline algorithm we compare to. We will use g^* to denote the optimal representation and h_k^* the corresponding optimal predictors in task k of the best offline algorithm. Note that in the definition above, each step contributes equally, just as in the standard regret notion, so that longer tasks matter more, which differs from the notion of regret adopted by Alquier et al. (2017).

One naive attempt to solve the problem is to treat each pair (g, h) as a joint action and apply existing online algorithms. However, the regret guarantee given in this way is weaker than what we want, as it is compared against a single pair (g^*, h^*) fixed through all the tasks, instead of allowing a different h^*_k for a different task k. To deal with this issue, a simple way around is to redo the learning for each task, but this unfortunately leads to weaker results as discussed in the introduction, where the dominating term has the number K of tasks interacting with the number G of representations. To avoid this, we would like to have learning algorithms which can learn the representations continuously through time, using all the data across different tasks.

Due to the page limit, we will move all our proofs to the appendix in the supplementary material.

3 Full-Information Adversarial Setting

In this section, we consider the adversarial setting in which the whole loss fuction at each step is revealed after our decision and the loss functions have no pattern, possibly chosen by an adversary.

Recall that we hope to learn the representations continuously through time using all the data across different tasks, while we still have to relearn predictors for different tasks. To do that, we would like to decouple the learning of representations from that of predictors, for them to have different learning algorithms as well as different learning schedules. However, it is not clear how to learn the representations, and a technical challenge is to construct appropriate loss functions to guide their learning. Recall that the actual loss functions depend on both the representation and the predictor, so how good a representation is actually depends on the predictor we choose to accompany it. This means that a good representation may look bad if we choose a bad predictor to go with it. Then which predictor should we use? A sensible choice seems to be its best predictor in a task, as it is what we measure regret against to, but we do not know what it is as the predictor which looks best so far may turn out to be bad in the end in the adversarial setting. This is perhaps one reason why Alquier et al. (2017) considered a weaker regret notion and chose to update their representations only at the end of each task, but consequently requiring a large number of tasks in order to have a good regret bound.

In Subsection 3.1, we will consider the case with finite G for which we have efficient algorithms achieving good regret bounds. We will consider the cases with infinite G and finite H in Subsection 3.2, and then infinite G and infinite H in Subsection 3.3, for which we have good regret bounds but via inefficient algorithms.

3.1 Finite representations

In this subsection, we consider the case in which G is finite. First, we describe our solution via a generic algorithm, which can use any algorithm $\mathtt{alg}_G$ for learning representations and any algorithm $\mathtt{alg}_H$ for learning predictors, with the resulting regret bound guaranteed by those of these two algorithms. After that, we will consider two applications in which there are efficient algorithms for $\mathtt{alg}_G$ and $\mathtt{alg}_H$ with good regret bounds.

Algorithm. For learning the representation, we take a single copy of $\mathtt{alg}_G$ and have it update continuously through time, across different tasks. For each possible representation g, we have a separate copy of $\mathtt{alg}_H$, denoted as $\mathtt{alg}_H^{(g)}$, for learning the accompanying predictors. When starting a new task k, we reset each copy $\mathtt{alg}_H^{(g)}$ and redo its learning, while we update the algorithm $\mathtt{alg}_G$ continuously for the whole T iterations. At step s in task k, we sample a representation $g_{k,s}$ according to some distribution $\mathcal{G}_{k,s}$ of $\mathtt{alg}_G$, followed by sampling a predictor $h_{k,s}$ according to some distribution $\mathcal{H}_{k,s}^{(g_{k,s})}$ of $\mathtt{alg}_H^{(g_{k,s})}$ (each distribution can focus all the measure on one element if the algorithm is deterministic). The joint action we play is $(g_{k,s}, h_{k,s})$, and the loss we suffer is $\ell_{k,s}(g_{k,s}, h_{k,s})$. Then we update the distribution of $\mathtt{alg}_G$ using the loss function on g defined as

$$\hat{\ell}_{k,s}(g) = \mathop{\mathbb{E}}_{h \sim \mathcal{H}_{k,s}^{(g)}} [\ell_{k,s}(g, h)],$$

and update the distribution of each copy $\mathtt{alg}_H^{(g)}$ using $\ell_{k,s}(g, \cdot)$ as the loss function on predictors.

The regret of our algorithm is guaranteed by the following, which we prove in Appendix A.1.

Theorem 3.1. *Suppose the t-step regret bounds of $\mathtt{alg}_G$ and $\mathtt{alg}_H$ are $\mathtt{reg}_G(t)$ and $\mathtt{reg}_H(t)$, respectively. Then the T-step regret bound of our algorithm is at most $\mathtt{reg}_G(T) + \sum_{k=1}^{K} \mathtt{reg}_H(T_k)$, where T_k denotes the number of steps in task k.*

Next, we instantiate the theorem in two scenarios, with the following two results, based on the multiplicative update (MU) algorithm of Littlestone and Warmuth (1994); Freund and Schapire (1997), and the online gradient-descent (OGD) algorithm of Zinkevich (2003), which we will prove in Appendix A.2 and A.3 respectively.

Corollary 3.2. *Suppose both G and H are finite and the loss functions are arbitrary. Then there is an efficient algorithm which achieves a regret of $\mathcal{O}(\sqrt{T\log G}+\sqrt{KT\log H})$.*

Corollary 3.3. *Suppose G is finite, H is infinite, each $\ell_{k,s}(g,\cdot)$ is convex given any g, and there exist $R, D > 0$ such that $\|h-h'\|_2 \le R$ and $\|\partial\ell_{k,s}(g,h)/\partial h\|_2 \le D$ for any $h,h' \in \mathcal{H}$, $g \in \mathcal{G}$, task k and step s. Then there is an efficient algorithm achieving a regret of $\mathcal{O}(\sqrt{T\log G}+DR\sqrt{KT})$.*

Remark. Let us remark that although our results work under the assumption that we know when each task starts, they can be easily extended to the general case without this assumption, by choosing an appropriate algorithm for $\texttt{alg}_H$ and running it continuously across tasks (instead of rerunning it for each task). In fact, Corollary 3.3 still holds as the OGD algorithm used there can work in this way. To replace Corollary 3.2, we can use e.g. the fixed-share algorithm of Herbster and Warmuth (1998) for $\texttt{alg}_H$ and have a slightly larger regret bound of $\mathcal{O}(\sqrt{T\log G}+\sqrt{KT\log(TH)})$.

Note that the regret bounds in both Corollaries 3.2 & 3.3 have a term which is related to learning predictors and increases with the number of tasks K. This seems unavoidable as each task requires us to relearn a new predictor. On the other hand, the term involving G in each theorem, which is related to learning representations, does not depend on K. Our algorithms make this possible by managing to learn representations continuously through time across different tasks. Compared to relearning the representation for each task, we save a factor of $\sqrt{K}$ from the term involving G, which makes our algorithms appealing when having a large number of representations with many tasks. In fact, one can show that our regret upper bounds are tight, based on existing lower bounds in the traditional settings. As an example, the following theorem shows a matching lower bound for the case of finite G and H, which we prove in A.4.

Theorem 3.4. *The problem with finite G and H and arbitrary loss functions in the full-information setting has a regret lower bound of $\Omega(\sqrt{T\log G}+\sqrt{KT\log H})$.*

3.2 Infinite representations and finite predictors

Next, let us consider the case in which G is infinite but H is finite. Let us make a similar assumption that each $\ell_{k,s}(\cdot,h)$ is convex given any h, and there exist positive R and D such that $\|g-g'\|_2 \le R$ and $\|\partial\ell_{k,s}(g,h)/\partial g\|_2 \le D$ for any $g,g' \in \mathcal{G}$, any $h \in \mathcal{H}$, any task k and any step s. Although one may also hope to apply the OGD algorithm for learning the representations, it is not clear how to construct appropriate loss functions for them. This is because each loss function $\ell_{k,s}(g,h)$ is convex in g if h is fixed. On the other hand, if we consider a natural choice by taking minimization over h, the resulting function would no longer be guaranteed to be convex in g.

Our result is the following which we prove in Appendix A.5. Let us remark that our algorithm is not efficient as it runs in time proportional to H^K, so our contribution here is to understand how small a regret bound can be achieved in this setting.

Theorem 3.5. *Under the assumption stated above, there is an algorithm which achieves a regret of $\mathcal{O}(\sqrt{TK\log H}+DR\sqrt{T})$.*

3.3 Infinite representations and infinite predictors with Lipschitz loss functions

Finally, we consider the case with both G and H being infinite. More precisely, suppose that $\mathcal{G} \subseteq \mathbb{R}^n$ and $\mathcal{H} \subseteq \mathbb{R}^d$ are both closed and bounded, with diameters R_1 and R_2, respectively, so that $\|g-g'\|_2 \le R_1$ for any $g,g' \in \mathcal{G}$ and $\|h-h'\|_2 \le R_2$ for any $h,h' \in \mathcal{H}$. To reflect that learning representations is harder than learning predictors, we assume that $n \gg d$. Moreover, in order to make small regret possible, some assumption must be made on the loss functions. Here we assume that they are Lipschitz continuous, in the sense that there exist constants C_1 and C_2 such that for any k, s, any $g,g' \in \mathcal{G}$, and any $h,h' \in \mathcal{H}$, $|\ell_{k,s}(g,h)-\ell_{k,s}(g',h)| \le C_1\|g-g'\|_2$, and $|\ell_{k,s}(g,h)-\ell_{k,s}(g,h')| \le C_2\|h-h'\|_2$. With only such a weak assumption on loss functions, one cannot hope to have an efficient algorithm, so our goal here again is to understand how small a regret bound one can achieve.

Our result is summarized in the following theorem, which we prove in Appendix A.6.

Theorem 3.6. *Under the assumptions stated above, there is an algorithm which achieves a regret of $\mathcal{O}(\sqrt{nT\log T}+\sqrt{dKT\log T})$.*

4 Finite Full-Information Stochastic Setting

In this section, we consider the full-information stochastic setting with both G and H being finite, aiming for a better regret bound. Formally, for each task k, there is some fixed but unknown distribution such that at any step s in the task, the loss function $\ell_{k,s}$ is sampled i.i.d. from it, with mean $\mathbb{E}[\ell_{k,s}(g,h)] = \mu_k(g,h)$ for any (g,h). Let $\mu_k(g) = \min_h \mu_k(g,h)$, which measures how good a representation g is in task k. We assume that the best representation in every task is the same, denoted as g^*, so that $\mu_k(g^*) < \mu_k(g)$ for any k and $g \neq g^*$. This is what makes the tasks related. On the other hand, for any representation g, its optimal predictors may differ in different tasks, so that further learning is still needed when going into a new task, even if g^* has been learned.

As common in prior works, we consider in this section the so-called pseudo-regret, defined as

$$\mathbb{E}\left[\sum_{k,s} \ell_{k,s}\left(g_{k,s}, h_{k,s}\right)\right] - \min_{h_1,\ldots,h_K} \mathbb{E}\left[\sum_{k,s} \ell_{k,s}\left(g^*, h_k\right)\right],$$

which is at most the regret defined in Eq.(1) and equals

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

where the expectation above is over the sampling of $g_{k,s}$ and $h_{k,s}$ if the algorithm is randomized.

Following previous works for the stochastic setting, we hope that the optimal representation g^* can be determined within a small number of iterations. In the traditional setting with only one task, we can simply follow the leader by choosing the arm which looks best empirically so far, since the empirical mean is likely to be close to the true mean with an error proportional to $1/\sqrt{t}$ after seeing its t previous loss values. On the other hand, in our lifelong optimization setting, the mean loss of a representation g may keep changing when going into new tasks. Consequently, its average mean loss so far may be considerably different from its mean loss of the next iteration, so that a representation which looks good on average so far may turn out much worse than we expect. Moreover, even when we somehow choose the optimal representation g^*, it may not appear good if we do not choose a good predictor to accompany it, which makes the job of identifying the optimal g^* even harder.

Nevertheless, we will show that when g^* is considerably better than others with some gap, it can still be identified within a reasonable number of iterations. After identifying g^*, the problem becomes much easier as we can then focus on learning the predictors of g^* only. Before that, we still need to search through a large space of representations, which may result in a large regret if not done properly. Fortunately, we will show that our adversarial algorithm in the previous section can actually provide a good enough regret bound.

Algorithm. Now we describe our algorithm, which works in two phases. In the first phase, which we call exploration phase, we run our adversarial algorithm in Theorem 3.2, but in addition, we also maintain some statistics and look for the optimal representation. More precisely, in iteration t corresponding to step s in task k, we compute the average empirical loss of each representation g as

$$\bar{L}_t(g) = \frac{1}{t-1} \sum_{i=1}^{k} \min_{h_i} \sum_{j \in I_i} \ell_{i,j}(g, h_i)$$

where I_i denotes the collection of steps so far in task i. We stop the adversarial algorithm in some iteration $t = \hat{T}$ when there is some representation $\hat{g}$ which dominates others: with

$$\bar{L}_t(\hat{g}) < \bar{L}_t(g) - 2\sigma_t \text{ for any } g \neq \hat{g},$$

where

$$\sigma_t = \sqrt{(c/t)\log(t^2 G H^K/\delta)},$$

for a large enough constant c (which can be determined in the proof of Theorem 4.1). Then we enter the second phase, which we call the exploitation phase. In this phase, for any remaining task k and any step s in it, we always choose $g_{k,s} = \hat{g}$, and accompany with it the predictor $h_{k,s} = \arg\min_h \sum_{j<s} \ell_{k,j}(\hat{g}, h)$, which is the best predictor empirically so far in the task.

Similarly to previous works in the stochastic setting, our regret bound depends on some notion of gaps between arms. Here, we consider the following two:

$$\Delta = \min_k \min_{g \neq g^*} \left(\mu_k(g) - \mu_k(g^*)\right) \text{ and } \quad (2)$$

$$\triangle_* = \min_k \min_{h \neq h_k^*} \left(\mu_k(g^*, h) - \mu_k(g^*)\right). \quad (3)$$

Note that Δ is the smallest gap between the mean loss of g^* and those of others over tasks, which determines how hard it is to distinguish the optimal g^* from suboptimal ones. On the other hand, $\triangle_*$ is the smallest gap of mean losses from g^*'s suboptimal predictors over tasks, which determines how hard it is to distinguish the optimal predictors of g^* from suboptimal ones. The regret of our algorithm is guaranteed by the following theorem.

Theorem 4.1. *For any $\delta > 0$, with probability at least $1 - \frac{3}{2}\delta$, the pseudo-regret of our algorithm is at most*

$$\mathcal{O}\left(\frac{1}{\Delta}\log\frac{G}{\Delta\delta} + \frac{1}{\Delta}K\log H + \frac{1}{\triangle_*}K\log\frac{TH}{\delta}\right).$$

In Appendix B, we provide the proof of the theorem and summarize our algorithm as Algorithm 1.

5 Bandit Setting

In this section, we consider the bandit setting, in which the feedback information we receive at each step is the loss value $\ell_{k,s}(g_{k,s}, h_{k,s})$ of our action $(g_{k,s}, h_{k,s})$, instead of the whole loss function $\ell_{k,s}(\cdot)$. Thus, unlike in the full information setting, now we do not have the whole loss function in each step to update our distribution. Following previous works, our approach is to construct appropriate estimators of the true loss functions and feed these estimators to update appropriate full-information algorithms. Next, we focus on the case with finite G and H, and consider the adversarial setting as well as the stochastic setting.

5.1 Finite Adversarial bandit setting

First, we consider the adversarial case. To design a bandit algorithm in this case, a natural attempt is to replace the full-information MU algorithms used in Theorem 3.2 by the EXP3 bandit algorithms of Auer et al. (2002b). However, this would result in a large regret bound by following the regret analysis in the proof. This is because the analysis there relies on being able to learn the predictors of each representation well, which is possible in the full-information setting. However, in the bandit setting, if we do not play a representation often enough, we can not receive enough feedback information to learn its predictors well, but making more aggressive exploration leads to worse regret bound.

Instead, we take a different approach, by reducing our problem to the following "experts over actions" problem. In this new problem, there is a set $\mathcal{G} \times \mathcal{H}^K$ of experts. Each expert is indexed by some $(g, \vec{h})$, with $g \in \mathcal{G}$ and $\vec{h} = (h_1, h_2, \cdots, h_K) \in \mathcal{H}^K$, who in every step s of task k plays the action (g, h_k), which has the loss value $\ell_{k,s}(g, h_k)$. Now what an online algorithm can do at each step is to choose an expert and play his/her action, and the regret is measured against the total loss of the best expert, which in fact is the same as the regret defined in Eq.(1). Therefore, we can use any online algorithm for this new problem to solve our initial problem. In particular, we would like to run the EXP3 algorithm on the experts and apply its regret analysis.

However, there is an apparent efficiency issue as the number of experts is GH^K, and we would like to avoid maintaining such a large number of probability values as needed by EXP3 when implemented naively. The idea is that it suffices to be able to sample from the distribution of actions played by the experts at each step. Since there are only GH possible actions at each step, the distribution at each step can actually be specified by only GH values. Therefore, our algorithm at each step s of task will maintain such a distribution $\mathcal{P}_{k,s}(\cdot, \cdot)$ over the actions in $\mathcal{G} \times \mathcal{H}$, and update it in the following way.

Algorithm. Initially, we have $\mathcal{P}_{1,1}(g, h) = 1/(GH)$ for every $(g, h) \in \mathcal{G} \times \mathcal{H}$. Then, after step s of task k, we make the following update according to two cases. If step s is not the last step of task k, we update the next distribution as

$$\mathcal{P}_{k,s+1}(g, h) = \mathcal{P}_{k,s}(g, h) \cdot e^{-\eta \bar{\ell}_{k,s}(g,h)} / Z_{k,s},$$

where η is some learning rate, $\bar{\ell}_{k,s}$ is some estimator for the true loss function $\ell_{k,s}$, to be specified later, and $Z_{k,s}$ is some normalization factor for making $\mathcal{P}_{k,s+1}$ a probability distribution. Otherwise, if task k ends at step s, we update the next distribution as

$$\mathcal{P}_{k+1,1}(g, h) = \sum_{h'} \mathcal{P}_{k,s}(g, h') \cdot e^{-\eta \bar{\ell}_{k,s}(g,h')} / \bar{Z}_{k,s},$$

where $\bar{Z}_{k,s}$ is again some normalization factor for making $\mathcal{P}_{k+1,1}$ a probability distribution. Note that when we start at task $k+1$, we relearn the predictors for each representation g, as the probability $\mathcal{P}_{k+1,1}(g, h)$ defined above does not depend on h so that all the predictors of g start with the same conditional probability.

Our estimator for $\ell_{k,s}(g, h)$ is defined as the following:

$$\bar{\ell}_{k,s}(g, h) = \frac{\ell_{k,s}(g, h)}{\mathcal{P}_{k,s}(g, h)} \mathbb{1}_{g = g_{k,s}, h = h_{k,s}}. \tag{4}$$

Note that conditioned on all previous randomness before step s of task k, the expected value of $\bar{\ell}_{k,s}(g, h)$, over the sampling of $g_{k,s}$ and $h_{k,s}$, is exactly $\ell_{k,s}(g, h)$, for any g and h. This means that $\bar{\ell}_{k,s}(\cdot)$ is indeed an unbiased estimator for the true loss function $\ell_{k,s}(\cdot)$.

The following shows that our distributions indeed match EXP3's, which we prove in Appendix C.1,

Lemma 5.1. *For any step s of task k, the distribution of actions played by the EXP3 algorithm based on the loss estimator $\bar{\ell}_{k,s}$ is the same as our distribution $\mathcal{P}_{k,s}$.*

Using the learning rate $\eta = \sqrt{(\log(GH^K))/(TGH)}$, the regret achieved by our algorithm is guaranteed by the following theorem.

Theorem 5.2. *For the problem with finite G and H and arbitrary loss functions, our bandit algorithm achieves a regret of $\mathcal{O}(\sqrt{TGH \log(GH^K)})$.*

In Appendix C.2, we provide the proof of the theorem and summarize our algorithm in Algorithm 2.

The following theorem provides a lower bound for the problem, which we prove in Appendix C.3.

Theorem 5.3. *The problem with finite G and H and arbitrary loss functions in the bandit setting has a regret lower bound of $\Omega\left(\sqrt{TGH} + \sqrt{TKH}\right)$.*

5.2 Finite stochastic bandit setting

Next, we consider the stochastic setting studied in Section 4 with gaps Δ and Δ_* defined in Eq.(2) and Eq.(3), respectively, but now having only bandit feedbacks.

A natural attempt is to use the UCB algorithm of Auer et al. (2002a) to learn both representations and predictors. However, the standard regret analysis of UCB relies crucially on the assumption that the mean of each arm's loss does not change over time, as this guarantees that a suboptimal arm with gap Δ is likely to be distinguished after being played for about $1/\Delta^2$ iterations, and each such iteration contributes Δ to the total regret. Unfortunately, such an argument can no longer work in our lifelong learning setting because the mean of each representation's loss (with respect to its best predictor in a task) here is not fixed over time. In fact, in the bandit setting, a suboptimal representation g may have a lower average mean loss over the iterations g is played, compared to the average mean loss of the optimal g^* over the iterations g^* is played, because g and g^* are played at different iterations (although we assume that $\mu_k(g^*) < \mu_k(g)$ for any k, we allow $\mu_k(g^*) > \mu_{k'}(g)$ for $k \neq k'$).

To avoid such an issue, we would like to design our bandit algorithm based on our full-information algorithm in Section 4. In particular, we would like to replace the full-information adversarial algorithm used in the first phase there by the adversarial bandit algorithm in Subsection 5.1, and then use the UCB bandit algorithm to learn the predictors in the second phase. However, an apparent difficulty is that there is no guarantee that each representation is played often enough by our EXP3-based adversarial bandit algorithm, and we may never have the confidence intervals of all the representations small enough in order to identify the optimal representation g^* reliably. This requires us to find a different way for identifying g^*. Our key observation is that in the stochastic setting, the optimal arm must be played often enough in order to guarantee a small regret. Therefore, our solution is to run our adversarial bandit algorithm, which has a small regret, for a sufficient number of iterations, and then take the most often played representation as g^*.

Algorithm. Our algorithm runs in two phases. In the first phase, which lasts for $\tilde{T} = c(\frac{GH \log(GH^K)}{\Delta^2})$ iterations, for a large enough constant c, we use our adversarial bandit algorithm in Subsection 5.1 to choose our actions. The purpose of this phase is to identify the optimal representation, and we take the representation $\hat{g}$ which was played most often during this phase. Then we enter the second phase, in which we always choose the representation $\hat{g}$. As there is no representation to learn, the problem of learning each task then reduces to the traditional multi-armed bandit problem, and we simply use the UCB algorithm to relearn the predictors in each remaining task. The regret achieved by our algorithm is guaranteed by the following theorem.

Theorem 5.4. *In the finite stochastic bandit setting, the pseudo-regret of our algorithm is at most* $\mathcal{O}\left(\frac{GH \log(GH^K)}{\Delta} + \frac{KH \log T}{\Delta_*}\right)$ *with high probability.*

In Appendix C.4, we provide the proof of the theorem and summarize our algorithm in Algorithm 3.

References

Abernethy, J., Bartlett, P., and Rakhlin, A. (2007). Multitask learning with expert advice. In *International Conference on Computational Learning Theory*, pages 484–498. Springer.

Agarwal, A., Rakhlin, A., and Bartlett, P. (2008). Matrix regularization techniques for online multitask learning. *EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2008-138*.

Alquier, P., Mai, T. T., and Pontil, M. (2017). Regret bounds for lifelong learning. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, pages 261–269.

Auer, P., Cesa-Bianchi, N., and Fischer, P. (2002a). Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2-3):235–256.

Auer, P., Cesa-Bianchi, N., Freund, Y., and Schapire, R. E. (2002b). The nonstochastic multiarmed bandit problem. *SIAM journal on computing*, 32(1):48–77.

Balcan, M.-F., Blum, A., and Vempala, S. (2015). Efficient representations for lifelong learning and autoencoding. In *Conference on Learning Theory*, pages 191–210.

Baxter, J. et al. (2000). A model of inductive bias learning. *J. Artif. Intell. Res.(JAIR)*, 12(149-198):3.

Cavallanti, G., Cesa-Bianchi, N., and Gentile, C. (2010). Linear algorithms for online multitask classification. *Journal of Machine Learning Research*, 11(Oct):2901–2934.

Cesa-Bianchi, N. and Lugosi, G. (2006). *Prediction, Learning, and Games*. Cambridge University Press, New York, NY, USA.

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

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

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

Lee, J., Yun, J., Hwang, S., and Yang, E. (2017). Lifelong learning with dynamically expandable networks. *arXiv preprint arXiv:1708.01547*.

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

Lugosi, G., Papaspiliopoulos, O., and Stoltz, G. (2009). Online multi-task learning with hard constraints. *arXiv preprint arXiv:0902.3526*.

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

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

Maurer, A., Pontil, M., and Romera-Paredes, B. (2013). Sparse coding for multitask and transfer learning. In *International Conference on Machine Learning*, pages 343–351.

Maurer, A., Pontil, M., and Romera-Paredes, B. (2016). The benefit of multitask representation learning. *The Journal of Machine Learning Research*, 17(1):2853–2884.

Murugesan, K., Liu, H., Carbonell, J., and Yang, Y. (2016). Adaptive smoothed online multi-task learning. In *Advances in Neural Information Processing Systems*, pages 4296–4304.

Pentina, A. and Ben-David, S. (2015). Multi-task and lifelong learning of kernels. In *International Conference on Algorithmic Learning Theory*, pages 194–208. Springer.

Pentina, A. and Lampert, C. (2014). A pac-bayesian bound for lifelong learning. In *International Conference on Machine Learning*, pages 991–999.

Pentina, A. and Lampert, C. H. (2015). Lifelong learning with non-iid tasks. In *Advances in Neural Information Processing Systems*, pages 1540–1548.

Pentina, A. and Urner, R. (2016). Lifelong learning with weighted majority votes. In *Advances in Neural Information Processing Systems*, pages 3612–3620.

Rusu, A. A., Rabinowitz, N. C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., and Hadsell, R. (2016). Progressive neural networks. *arXiv preprint arXiv:1606.04671*.

Ruvolo, P. and Eaton, E. (2013). Ella: An efficient lifelong learning algorithm. In *International Conference on Machine Learning*, pages 507–515.

Saha, A., Rai, P., DaumÃ, H., Venkatasubramanian, S., et al. (2011). Online learning of multiple tasks and their relationships. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 643–651.

Thrun, S. and Pratt, L., editors (1998). *Learning to Learn*. Kluwer Academic Publishers, Norwell, MA, USA.

Zinkevich, M. (2003). Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 928–936.