FEW-SHOT LIFELONG REINFORCEMENT LEARNING WITH GENERALIZATION GUARANTEES: AN EMPIRICAL PAC-BAYES APPROACH

Anonymous authors
Paper under double-blind review

ABSTRACT

We propose a new empirical PAC-Bayes approach to develop lifelong reinforcement learning algorithms with theoretical guarantees. The main idea is to extend the PAC-Bayes theory in supervised learning to the reinforcement learning regime. More specifically, we train a distribution of policies, and gradually improve the distribution parameters via optimizing the generalization error bound using trajectories from each task. As the agent sees more tasks, it elicits better prior distributions of policies, resulting in tighter generalization bounds and improved future learning. To demonstrate the superior performance of our method compared to recent state-of-the-art methods, we test the proposed algorithms on various OpenAI's Gym and Mujuco environments and show that they adapt to new tasks more efficiently by continuously distilling knowledge from past tasks.

1 INTRODUCTION

Deep reinforcement learning (RL) has demonstrated outstanding performance in many challenging tasks, such as playing games, visual navigation, robotic control, and manipulation. However, most DRL algorithms are trained to perform well on specific tasks and are not generalizable to many different tasks. To handle this challenge, we resort to lifelong RL, in which an agent interacts with a sequence of different but related tasks. The goal of lifelong RL is to let the agent quickly adapt to any new task by distilling knowledge from past tasks, even in a few-shot learning (FSL) case where the agent can only interact with each task for a small number of steps.

There are two major challenges for few-shot lifelong learning. (1) It is difficult to confidently determine **what knowledge should be transferred**. For example, if an agent is deployed in a series of mazes, seeking rewards located at random states. The ideal knowledge to be transferred among tasks should be how to walk and how to navigate, instead of the specific goal locations. Inappropriate knowledge could result in "negative transfer", which harms the learning of future tasks. (2) It is challenging to distill useful knowledge when the samples gained from each task is limited. The agent needs to achieve **fast adaptation** in order to obtain both high rewards and meta knowledge.

To handle the above challenges, we propose a LRL algorithm named PB-LRL (PAC-Bayes Lifelong RL) which learns common knowledge from past tasks and quickly adapts to new tasks, as well as provide a reliable theoretical guarantee to avoid the negative transfer. Under lifelong PAC-Bayes framework, the main idea of PB-LRL is to seek a *default policy distribution* that generalizes well to all tasks in the underlying tasks distribution, and then for any new task, we start from a policy sampled from the default policy distribution and use it to continuously update the default policy distribution by repeatedly applying the PAC-Bayes bound. More specifically, we propose to "learn" the prior policy distribution required in PAC-Bayes theory along with the default policy distribution. Thus, our approach falls in an empirical Bayes framework which automatically upgrades the prior and does not require additional expert knowledge.

The main idea of PB-LRL builds upon the works from Finn et al. (2017) and Majumdar et al. (2021). In a lifelong reinforcement learning setting, Finn et al. (2017) exploits deep neural network policies to demonstrate a good policy initialization can significantly improve the learning performance and achieve fast adaptation. Though promising empirical results have been reported, their methods are heuristic and lack theoretical guarantees. In recent work, Majumdar et al. (2021) proposes to optimize a policy distribution over past tasks and directly use it for the following tasks. By applying the PAC-Bayes theory, their algorithm has a nice theoretical guarantee and also works for deep neural

network policies. However, their algorithm can not be used in lifelong learning, where the tasks can be very diverse, making the learnt policy from previous experiences hardly perform well in new tasks. Also, their algorithm and theory require the selection of a "good" prior policy distribution, and a bad prior selection may cause significant performance drop. In contrast, the PB-LRL algorithm can continuously evolve by repeatedly utilizing the PAC-Bayes theory to optimize the policy distribution and its prior, without any expert knowledge of environments. Other works Brunskill & Li (2013); Sun et al. (2020) also exploit the PAC-Bayes theory to handle lifelong or multitask RL problems. However, their theory mainly focuses on tabular settings, and can not easily adapt to large-scale environments.

To demonstrate the efficacy of the PB-LRL algorithm, we also conduct thorough numerical benchmarks on diverse OpenAI's Gym environments. Our results show that the PB-LRL algorithm outperforms the competing state-of-the-art lifelong RL algorithms.

2 Related Works

Meta-Learning: Agents' lifelong learning requires the generalization guarantee of policy to new tasks. Meta-learning has been studied by Bengio et al. (1991); Naik & Mammone (1992); Thrun & Pratt (1998); Finn et al. (2017), which is mostly related to lifelong learning through an efficient meta-leaner. Schmidhuber (1992); Bengio et al. have trained a meta-learner that learns how to update the parameters of the learners model. Hochreiter et al. (2001); Andrychowicz et al.; Li & Malik (2017); Ha et al. (2016) applied this approach to learn optimization of deep networks for FSL.

Koch; Ravi & Larochelle (2017); Vinyals et al. (2016); Reed et al. (2018); Snell et al. (2017) have developed few-shot methods for supervised-learning. The different problem settings of supervised learning than RL prevents the aforementioned methods from being applied to RL settings.

Duan; Wang et al. (2017) has learned fast RL agents to adapt to multiple tasks by training memory augmented recurrent learner. Saxe et al. (2014); Kirkpatrick et al. (2017); Krähenbühl et al. (2016); Salimans & Kingma (2016); Husken & Goerick (2000); Maclaurin et al. have considered sensitivity in deep networks using methods like good random initialization or data-dependent initialization. But these methods are often data-inefficient.

Finn et al. (2017) directly updated the meta learner weights using the gradient based on the sensitivity of a given task distribution. Their method is agnostic to the form of the model and to the particular learning task, furthermore, their methods are complementary to the approach presented here and could potentially be used as a baseline for our method. Our experiments show that our method outperforms their approach.

RL Generalization: For RL, early works such as Nichol et al. (2018); Mnih et al. (2013); Song et al. (2019) have quantified the generalization of learned policies. Subsequently, many works have suggested that an effective generalization require an extremely large number of training environments (Cobbe et al., 2019; 2020), and the generalization gap in the policy space for new tasks and environments always exists in sim-to-real problems (Tobin et al., 2018). Peng et al. (2018); Tan et al. (2018); Srivastava et al. (2014); Ioffe & Szegedy (2015); Pacelli & Majumdar (2020); Goyal et al. (2019) have applied regularization approaches to improve the generalization, but they do not explicitly exploit the structure of the RL, or these methods are limited to a particular task which prevents the learner to exploit causal relationships in the environment. Sinha et al. (2020) have done adversarial perturbations to the underlying data distribution to provide robustness guarantees, but the amount of perturbations is not easy to learn.

PAC-Bayes Theory: In classic supervised learning works by Langford & Shawe-Taylor (2002); Seeger (2002); Germain et al. (2009) and in deep learning works by Dziugaite et al. (2020); Neyshabur et al. (2018; 2017), PAC-Bayes theory (McAllester, 1999b) is utilized to study the generalization bounds. For RL, Schulman; Fard & Pineau (2010); Fard et al. (2012) have applied PAC-Bayes theory to learn controlled policies for MDP with provable sample complexity bounds. However, their approaches require the learner with multiple interactions in a given MDP, which lacks efficiency.

Then Majumdar et al. (2021); Veer & Majumdar (2020) have made provable generalization guarantees under distributional shifts and learned policies that could zero-shot generalize to novel environments by obtaining upper bounds on the expected cost of policies on novel environments. However, they have not shown whether or not the bound can be workable in generally more complicated multi-task lifelong settings.

In our work, we generate a PAC-Bayes bound as part of a lifelong learning algorithm to achieve provably data-efficient control on novel tasks. While reusing knowledge from past tasks is a crucial ingredient in making high-capacity scalable models, we focus on multi-task lifelong RL settings, and we make policies zero-shot generalize to novel tasks by directly operating on policy space to obtain upper bounds on the expected cost of meta policies. These policies function as a good policy initialization for new tasks.

While resembling Finn et al. (2017) in the following places: our method also does not induce additional parameters and uses the same gradient descent update for both learner and meta learner to provide a good initialization, the main difference between our work and theirs is that we also provide the theoretical guarantees, and empirically our work outperforms them in terms of achieving extremely efficient adaptation to new tasks in only a few gradient steps in different OpenAI's Gym environments.

3 PRELIMINARIES

In this section, we discuss the primary technical background that we leverage in this paper.

3.1 REINFORCEMENT LEARNING

In RL, an agent interacts with the environment by taking actions, observing states and receiving rewards. The environment is modeled by a MDP, which is denoted by a tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, T, R, \gamma, \nu \rangle$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, T is the transition kernel, R is the reward function, $\gamma \in (0, 1)$ is the discount factor, and ν is the initial state distribution.

A *trajectory* $\tau \sim \pi$ generated by *policy* π is a sequence $s_1, a_1, r_1, s_2, a_2, \cdots$, where $s_1 \sim \nu$, $a_t \sim \pi(a|s_t)$, $s_{t+1} \sim T(s|s_t, a_t)$ and $r_t = R(s_t, a_t)$. The goal of an RL agent is to find an optimal policy π^* that maximizes the *expected total rewards* η, which is defined as $\eta(\pi) = \mathbb{E}_{\tau\sim\pi}[r(\tau)] = \mathbb{E}_{s_1,a_1,\cdots\sim\nu,\pi,T,R}[\sum_{t=1}^{\infty} \gamma^{t-1} r_t]$.

3.2 META LEARNING AND LIFELONG LEARNING

In lifelong RL, the agent interacts with a (probably infinite) sequence of tasks, which are i.i.d coming from an underlying distribution of tasks (MDPs), denoted as $\mathcal{D}$. Suppose that these tasks share the same $\mathcal{S}, \mathcal{A}, \gamma, \nu$, but may have different transition probabilities T and rewards R. The learning process is:

1. initialize a policy π_0;
2. sample a task (MDP) $\mathcal{M}_i \sim \mathcal{D}$;
3. starting from π_0, learn a policy π_i for task M_i to maximize rewards.

3.3 PAC-BAYES THEORY IN SUPERVISED LEARNING

Suppose $\mathcal{Z}$ is an input space and $\mathcal{Z}'$ is a set of labels. Let $\mathcal{D}$ be the (unknown) true distribution on $\mathcal{Z}$. Let $\mathcal{H}$ be a hypothesis class consisting of functions $h_w : \mathcal{Z} \to \mathcal{Z}'$ parameterized by $w \in \mathbb{R}^d$ (e.g., neural networks parameterized by weights w). Let $l : \mathcal{H} \times \mathcal{Z} \to \mathbb{R}$ be a loss function. We will denote by $\mathcal{P}$ the space of probability distributions on the parameter space $\mathbb{R}^d$. Informally, we will refer to distributions on $\mathcal{H}$ when we mean distributions over the underlying parameter space. PAC-Bayes analysis then applies to learning algorithms that output a distribution over hypotheses. Generally, such algorithms will be given a "prior" distribution $P_0 \in \mathcal{P}$ in the beginning and learns a posterior distribution $P \in \mathcal{P}$ after observing training data samples.

Let us denote the training loss associated with the posterior distribution P as:

$$l_S(P) := \frac{1}{N} \sum_{z \in S} \mathop{\mathbb{E}}_{w\sim P} [l(h_w; z)] \tag{1}$$

and the true expected loss as:

$$l_{\mathcal{D}}(P) := \mathop{\mathbb{E}}_{z\sim\mathcal{D}} \mathop{\mathbb{E}}_{w\sim P} [l(h_w; z)] \tag{2}$$

We are interested in how "close" the training loss $l_s(P)$ and the true expected loss $l_{\mathcal{D}}(P)$, such metric can be obtained by using the KL divergence (Maurer, 2004; McAllester, 1999a). Based on that, the following theorem provides an upper bound on the true expected loss by computing the KL inverse.

Theorem 1 (PAC-Bayes Bound for Supervised Learning). *For any $\delta \in (0,1)$, with probability at least $1-\delta$ over samples $S \sim \mathcal{D}^N$, the following inequality holds:*

$$l_{\mathcal{D}}(P) \leq \mathbb{D}_{KL}^{-1}\left(l_S(P) \| \frac{\mathbb{D}_{KL}\left(P \| P_0\right) + \log\left(\frac{2\sqrt{N}}{\delta}\right)}{2N}\right) \tag{3}$$

$\mathbb{D}_{KL}^{-1}$, is called KL inverse, with its definition found in §A equation 9. Another practical upper bound for the true expected loss that is useful for the purpose of optimization is provided by the following theorem.

Theorem 2 (PAC-Bayes Upper Bound for Supervised Learning Maurer (2004); McAllester (1999a)). *For any $\delta \in (0,1)$, with probability at least $1-\delta$ over samples $S \sim \mathcal{D}^N$, the following inequality holds:*

$$\underbrace{l_{\mathcal{D}}(P)}_{\text{True expected loss}} \leq \underbrace{l_S(P)}_{\text{Training loss}} + \underbrace{\sqrt{\frac{\mathbb{D}_{KL}\left(P \| P_0\right) + \log\left(\frac{2\sqrt{N}}{\delta}\right)}{2N}}}_{\text{Regularizer"}} \tag{4}$$

4 Our Methods

In this paper, we propose a guaranteed lifelong RL algorithm that incrementally distills knowledge from previous tasks and quickly adapt to a new task. The algorithm is based on optimizing a PAC bound for the true expected cost for lifelong RL. We derived this bound based on the PAC-Bayes theory.

4.1 PAC-Bayes Bound for Lifelong RL

Our theoretical learning guarantee is mainly based on the PAC-Bayes theory, which was first applied to RL by Majumdar et al. (2021). We carefully extend the work of Majumdar et al. (2021) to the regime of lifelong RL.

Define Π as the whole policy space parameterized by $\theta \in \mathbb{R}^d$ (θ could be a neural network). Let P be any policy distribution over the policy space Π. Then the *cost* of P evaluated on task $\mathcal{M}$ is defined as

$$C(P;\mathcal{M}) = \mathop{\mathbb{E}}_{\pi\sim P}[C(\pi;\mathcal{M})] = \mathop{\mathbb{E}}_{\pi\sim P}[-\eta_{\mathcal{M}}(\pi)], \tag{5}$$

where $\eta_{\mathcal{M}}(\pi)$ is the total expected reward of policy π in MDP $\mathcal{M}$.

Then, the expected cost of P over the task distribution $\mathcal{D}$ can be bounded by the following theorem.

Theorem 3 (PAC-Bayes Bound for Lifelong RL). [1] *For any $\delta \in (0,1)$, over the lifelong task distribution $\mathcal{D}$ and N i.i.d. sampled tasks $\{\mathcal{M}_i\}_{i=1}^N$, for any distribution over policy space P and any prior distribution P_0, the following inequality holds*

$$\underbrace{\mathop{\mathbb{E}}_{\mathcal{M}\sim\mathcal{D}} C(P;\mathcal{M})}_{\text{True cost}} \leq \underbrace{\frac{1}{N}\sum_{i=1}^{N} C(P;\mathcal{M}_i)}_{\text{Training cost}} + \underbrace{\sqrt{\frac{\mathbb{D}_{KL}\left(P \| P_0\right) + \log\left(\frac{2\sqrt{N}}{\delta}\right)}{2N}}}_{\text{Regularizer}} \tag{6}$$

with probability at least $1-\delta$.

Remarks. (1) Theorem 3 guarantees that the expected loss/cost of the policy distribution over the underlying task distribution is upper bounded by the loss/cost computed on the past tasks and a regularization term.
(2) Theorem 3 holds for any policy distribution P and any prior distribution P_0 in the policy space Π.
(3) The regularizer involves the number of training tasks N, and the KL-divergence between P and P_0. Therefore, the tightness of the bound depends on the selection of prior P_0, and the number of training tasks.

[1] We assume that costs are bounded in the range $[0,1]$, if not, we can always normalize them.

Majumdar et al. (2021) use the PAC-Bayes bound to learn a policy distribution P over a set of similar but noisy tasks so that the expected cost of the policy on a new task is bounded. However, their method cannot be directly applied to our lifelong setting due to the following two reasons. **(1) Task divergence.** the tasks in lifelong learning can be divergent. For example, task A is to find an apple in the upper-right corner in a maze, and task B is to find an orange in the lower-left in the same maze; although the two tasks share the same transition dynamics, they require totally different policies, so that learning one single policy distribution for all tasks will not work well. This is similar to the first challenge we stated in the introduction about transferring proper knowledge. **(2) Choosing prior.** As pointed out by the remarks above, the tightness of the bound depend on the prior policy distribution P_0. However, in practice, it is hard to choose a good prior before learning starts. Majumdar et al. (2021) select the prior randomly, which is not optimal, or by expert knowledge, which is usually unrealistic.

Therefore, in the next section, we propose a new algorithm for lifelong RL that overcomes the aforementioned two problems.

4.2 Improve Lifelong Learning with PAC-Bayes Bound

Based on Theorem 3, we introduce our algorithm called PAC-Bayes Lifelong RL (PB-LRL, illustrated in §B Algorithm 1), which achieves efficient knowledge transfer and fast adaptation to novel tasks. Note that PB-LRL is a meta-learning algorithm that be combined with any single-task RL algorithm, provided that the base policy is a form of policy search defined on the same space of parameters that the PAC-Bayes bound uses.

The key to PB-LRL is to learn a policy distribution P as a policy initializer, called *default policy*. More specifically, we optimize the default policy P with Theorem 3, and whenever there is a new task, we sample a behavior policy from P and use any single-task learning method to fine-tune the policy. Hence, every task gets a "customized" policy, while the common knowledge is shared by the default policy, which solves the task divergence problem. Since in lifelong setting, the tasks are streaming in, we update the default policy every N tasks and estimate the training cost of P by the most recent N tasks.

To remedy the challenge of choosing prior, PB-LRL uses an "evolving" prior rather than a fixed prior. After every N tasks, we update P by minimizing the PAC-Bayes bound evaluated at the current prior P_0. Then, we update the prior policy distribution by making it closer to the new default policy. Because the default policy is updated to minimize the expected cost, a fixed prior may result in larger and larger $\mathbb{D}_{KL}(P||P_0)$, then looser and looser PAC-Bayes bound. In contrast, we slowly move the prior towards the default policy by $P_0 = (1-\lambda)P_0 + \lambda P$, where $\lambda \in [0, 1]$ is a hyper-parameter controlling the moving speed. In this way, even if the prior is initialized with a bad choice, we are still able to find a good prior during learning, and use it to improve the default policy.

In §B Algorithm 1, we use d-dimensional Gaussian distributions as the default and prior policy distribution, i.e., $\mathcal{N}(\mu, \sigma)$ and $\mathcal{N}(\mu_0, \sigma_0)$. It is noteworthy to mention that our algorithm and theoretical guarantees also work for other distributions. Before learning starts, the agent initializes a default policy distribution P and a prior policy distribution P_0 randomly or by domain knowledge (Line 2-3). For every task, the agent first samples a policy $\theta \sim P$ (Line 6-8), estimate the cost of θ (Line 9-11), and then fine-tune the policy with any single-task learning algorithm (Line 13-15). After every N tasks, the agent summarizes its recent experience, and updates default policy P by seeking to minimize the generalization error bound (6) evaluated at the current prior P_0 (Line 18-19). Then, the agent also updates the policy prior by $P_0 = (1-\lambda)P_0 + \lambda P$ (Line 20-21). Note that the prior P_0 is only used to construct the PAC-Bayes bound and optimize the default policy P.

The cost function $C(\pi, \mathcal{M})$ is approximated by $C(\mathcal{T}) = \frac{1}{|\mathcal{T}|}\sum_{\tau\in\mathcal{T}} r(\tau)$, where $\mathcal{T}$ is the trajectories generated by the current policy. More specifically, for each step, we sample $a \sim \pi(a|s;\theta)$, where $\theta = \mu + \sqrt{\sigma} \odot \epsilon$ and $\epsilon \sim \mathcal{N}(0, I_d)$, which is in line 7-9.

The regularization function L in line 18-19 is defined as

$$L(\mu, \sigma, \mu_0, \sigma_0, N) = \sqrt{\frac{\mathbb{D}_{KL}[\mathcal{N}(\mu, \sigma)||\mathcal{N}(\mu_0, \sigma_0)] + \log\frac{2\sqrt{N}}{\delta}}{2N}}$$

We note that while P is chosen by optimizing *PAC-Bayes Bound* (i.e., the RHS of inequality (6)), this is a valid not final upper bound. Upon having the optimal policy distribution $P^* := \mathcal{N}(\mu^*, \sigma^*)$ and

the corresponding policy function $\pi^* \sim P^*$, a tighter final upper bound on true cost $\mathbb{E}_{\mathcal{M}\sim\mathcal{D}} C(P;\mathcal{M})$ is available based on the inequality (3) and parameters sampled from P^*. Theoretically, we provide such bound in the following theorem.

Theorem 4 (Final PAC-Bayes Bound for Lifelong RL)**.** *After optimizing the PAC-Bayes Bound and outputting the θ^* from PB-LRL algorithm, we draw a large number of samples $(\theta_1, \cdots, \theta_K)$ from $\mathcal{N}(\mu^*, \sigma^*)$, for any $\delta \in (0,1)$, $\delta' \in (0,1)$, over the lifelong task distribution $\mathcal{D}$ and N i.i.d. sampled tasks $\{\mathcal{M}_i\}_{i=1}^N$, for the distribution under policy P^* and any prior distribution P_0, the following inequality holds*

$$\mathop{\mathbb{E}}_{\mathcal{M}\sim\mathcal{D}} C(P^*;\mathcal{M}) \leq \mathbb{D}_{KL}^{-1}\left(\widehat{C}(P^*;\mathcal{M}) \| \frac{\mathbb{D}_{KL}\left(P^* \| P_0\right) + \log\left(\frac{2\sqrt{N}}{\delta}\right)}{2N}\right) \tag{7}$$

with probability at least $1-\delta-\delta'$. Where the $\widehat{C}(P^*;\mathcal{M})$ is an upper bound on the training cost $C(P^*;\mathcal{M})$ with high probability $1-\delta'$: $C(P^*;\mathcal{M}) \leq \widehat{C}(P^*;\mathcal{M}) := \mathbb{D}_{KL}^{-1}\left(\frac{1}{NK}\sum_{i=1}^N\sum_{j=1}^K C(\pi^*;\mathcal{M}_i) \| \frac{1}{K}\log(\frac{2}{\delta'})\right)$

We defer all proofs of our theorems to the §A.

Importance of the update frequency N. The value of hyper-parameter N controls how quickly the meta learner – default policy and prior policy – are updated. The larger N is, the slower the meta learning learns. However, a large N also implies the PAC-Bayes upper bound is tighter and thus the update is more confident. Hence, there is an interesting trade-off between efficiency and accuracy. We empirically evaluate the influence of N in experiments in Section 5.4.

5 Experiments

In this section, we show the performance of our proposed algorithm on various environments for the lifelong learning tasks and compare it with the baseline algorithms.

Environments. We evaluate our proposed PB-LRL in multiple OpenAI's Gym and Mujuco environments, including discrete control (CartPole, LunarLander) and continous control (Swimmer) (§C.1). For each environment, we extend the original implementation to the lifelong setting, where the tasks have varying goals or dynamics. The agent needs to sequentially learn multiple tasks, each for a small number of steps. We introduce the detailed settings and results of the three environments respectively in Section 5.1, 5.2 and 5.3.

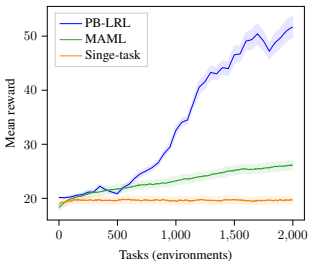
Baselines. Our PB-LRL is a meta RL method and can use any single-task RL algorithm as the base learner. For simplicity of comparison, we use Vanilla Policy Gradient (VPG) Sutton et al. (1998) as the base learner. And we compare our PB-LRL with the following two baselines, both of which use VPG as their base learning algorithm.

- Single-task learning. The Single-task learning baseline is to learn each task separately and update policy based on each task separately.
- MAML Finn et al. (2017). MAML is a state-of-the-art meta-learning method, which uses a heuristic method to optimize the meta policy. By simply aggregating the losses of a batch of tasks after one-step learning and taking gradient with respect to the meta policy, MAML aims to find model parameters that are valuable for all tasks.

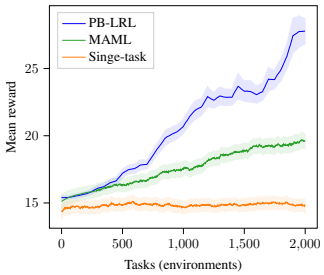
Note that in both baselines, they treat the policy as a single point in the parameter space rather than a distribution as in our algorithm.

Hyper-parameters. In each lifelong environment, we run the agent on 2000 tasks (or 1000 tasks for Swimmer). In every task, we only run 10 episodes with at most 300 steps per episode. Note that the number of per-task steps is much less than that usually required in traditional learning, so that the agent has to achieve fast adaptation to gain high rewards. More details are in §C.2.

The selection of update frequency N for our algorithm is specified in the captions of the figures. Similarly, MAML also updates its meta policy for every batch of tasks; for a fair comparison, we set MAML's update frequency to be the same with N in all experiments.



(a)



(b)

Figure 1: Comparison between our algorithm PB-LRL and baselines on CartPole-Goal. (a) $x_{goal} \sim \mathcal{N}(0, 0.1)$, $N = 25$; (b) $x_{goal} \sim \mathcal{N}(0, 0.5)$, $N = 50$; All results are averaged over 10 random seeds, with C.I. by a $0.32(10\%)$ standard deviation

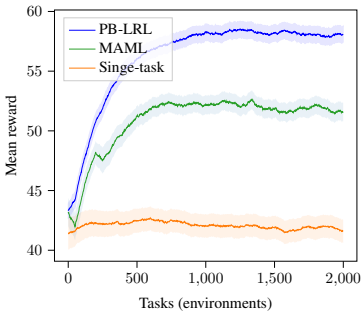
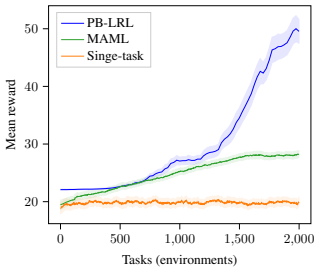
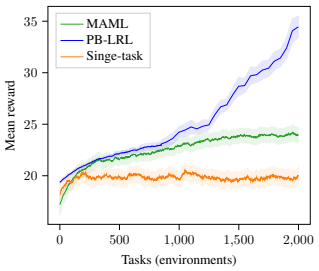


Figure 2: Comparison between our algorithm PB-LRL and baselines on LunarLander (with 0.1 standard deviation interval). The meta learning update frequency is $N = 50$.



(a)



(b)

Figure 3: Comparison between our algorithm PB-LRL and baselines on CartPole-Mass. The meta-learning update frequency $N = 50$, Different Cart Mass: (a) $\mu_c = 0.5$, $\sigma_c = 1.0$; (b) $\mu_c = 1.0$, $\sigma_c = 1.0$;

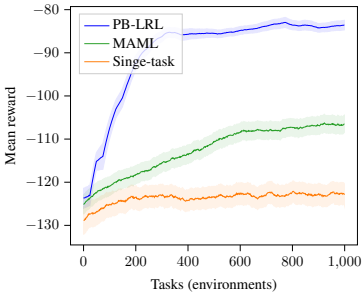


Figure 4: Comparison between our algorithm and baselines on Swimmer. The meta learning update frequency is $N = 25$.

5.1 In the CartPole Environment

We extend the original setting into a lifelong setting by defining two variants of CartPole, as respectively shown in Section 5.1.1 and Section 5.1.2.

5.1.1 Tasks with Varying Goals.

We implement a variant of CartPole, called CartPole-Goal, where the agent is supposed to reach a "goal position" on the track rather than keep the pole upright. The reward is defined by $r(x) = -\exp|x - x_{goal}|$, where x is the cart's current position, and x_{goal} is the goal's position. In a lifelong setting, the agent interacts with a series of tasks with different goal positions. For every task, the goal position is sampled from a Gaussian distribution centered at 0. The variance/standard deviation controls how divergent those random tasks are. We separately test two different standard deviations: 0.1 and 0.5. Goals with a standard deviation of 0.1 are more similar to each other, but the goals with a standard deviation of 0.5 are more divergent and harder to transfer knowledge. (Note that the active position range along the track is $[-2.4, 2.4]$, thus $> 99.9\%$ goals with a standard deviation of 0.5 are spread out over the track.) An ideal agent should distill the knowledge of "how to keep balance" from previous tasks, and quickly adapt to new tasks with different goals.

A comparison of the performance of our PB-LRL and baselines is shown in Figure 1, on CartPole-Goal environments with standard deviations of 0.1 and 0.5. We can see that PB-LRL significantly outperforms MAML and the single-task learner in both cases. As it sees more and more tasks, PB-LRL gradually and constantly improves its performance on each of the tasks. In Figure 1, because of the high divergence of goals, we observe some fluctuation of PB-LRL, but in the end, PB-LRL successfully overcomes the divergence of tasks, and achieves fast adaptation to the later tasks with the highest reward. In contrast, the performance of MAML increases much slower than PB-LRL. In

Figure 1(b), MAML achieves small progress compared with the naive single-task due to the high divergence of tasks.

5.1.2 TASKS WITH VARYING DYNAMICS.

In this section, we provide experimental results to show our algorithm outperforms baselines when the underlying tasks have different dynamics.

The intrinsic dynamics would affect lifelong learning performance. For example, in the Cart-Pole environment (§C.1 Figure 6(a)), different mass of the cart could generate different friction forces between the cart and the track, which later affects the acceleration of the cart (the acceleration of the cart is less with a heavier cart) and the rotation of the pole, then it will result in alternated MDP models so the knowledge transfer between meta learner and learner changed.

To help illustrate the above situation, we compared our algorithm with baselines by changing the single task mass, called CartPole-Mass (m_c). With other conditions of the environment fixed (friction force, controller force, etc.). Specifically, we set $m_c \sim \mathcal{N}(\mu_c, \sigma_c)$, and for every task, we sampled $\{m_{c_i}\}_1^{2000}$ from the above distribution.

Figure 3 shows performance of our method with baselines under different single task dynamics when we set $\mu_c = 0.5$ or $\mu_c = 1$. Every single task has different m_{c_i} which results in 2000 different small dynamics.

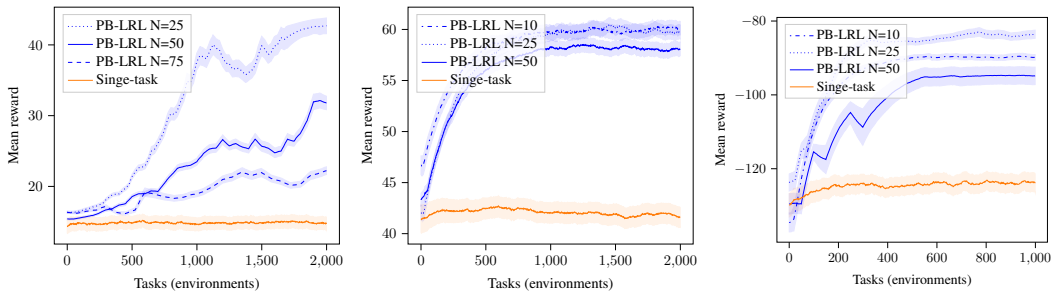
Figure 3(a) and 3(b) show that our method is able to learn the policy that leads to the highest reward compared to baselines. We also see that MAML outperforms Single-task learning under different task dynamics. The performance of our method is invariant to which single task dynamics it encountered, indicating its generalization ability to divergent dynamics for lifelong learning.

5.2 IN THE LUNARLANDER ENVIRONMENT:

For the LunarLander environment, the agent (lunar module) needs to learn to land at different goal positions. The goal position in a task is defined as the expected landing location x_{goal}. We sample uniformly to get different goals/landing locations. The reward is given by the distance between the landing position x and goal position as $r(x) = -\exp|x - x_{goal}|$.

The learning progress of PB-LRL is shown as in Figure 2. During the training, the agent is learning from different goals and samples, our algorithm can still learn to improve its performance throughout the time. This shows the ability of our algorithm can learn to quickly adapt to new tasks by distilling knowledge from previously seen tasks.

We also conduct the comparison between our algorithm and the two baselines. As shown in Figure 2, we can see that the MAML performs better than Single-task learning, this is not surprising since during training MAML takes additional consideration in performing well on the overall task distribution rather than only on a single task. Our algorithm is more efficient in adapting to new tasks by learning from previously seen tasks compared to the two baselines.



(a) (b) (c)

Figure 5: Comparison of different update frequency N on (a) CartPole-Goal; (b) LunarLander; (c) Swimmer.

5.3 IN THE SWIMMER ENVIRONMENT

The goal of the Swimmer is to make the robot swim forward as fast as possible by actuating the two joints. We simulate different environments by generating random goal velocity for the robot, which is uniformly chosen from $[0.1, 0.2]$. The reward is defined as $r(s, a) = -1.5||v_x - v_{goal}||_2 - 1e^{-4}||a||_2^2$, where $||v_x - v_{goal}||_2$ is the difference between the current velocity of the agent and the goal velocity.

We present the comparison results between our algorithm and the baselines in Figure 4. Our PB-LRL algorithm can efficiently adapt to new tasks and steadily improve the performance, and constantly outperform MAML and the single-task baseline.

5.4 INVESTIGATING THE EFFECTS OF N

As we discussed in Section 4.2, the hyper-parameter, update frequency N, balances the trade-off between efficiency and accuracy of updating policies. In theory, the smaller N leads to a faster update but will make the learning less stable. The effect of larger N results in a slower but more stable learning. Our experiment results have verified this theoretical interpretation. However, from our experiment results, we can also see that the practical effect of N on the performance of learning also depends on the environment.

In the CartPole environment as shown in Figure 5(a), $N = 25$ exhibits the fast and unstable learning behavior as in theory and performs the best in the learning in terms of the final reward. In contrast, in LunarLander (as in Figure 5(b)) and Swimmer (as in Figure 5(c)) environments, for all N cases, learning curves are nearly stable. In LunarLander all three N cases reach similar final rewards; in Swimmer $N = 25$ reaches the highest reward. One common thing is in all three environments, the largest N leads to the slowest learning, which is consistent with the theory.

The interpretation of the dependency on environments for learning behaviors, as shown in our results, can be intuitive. If tasks in a specific environment all share high portions of general features, then the effect of N on stability will be negligible. In addition, in practice when we test the outcome of learning, we only learn from a finite sample of tasks, this actually imposes an implicit requirement on both efficiency and accuracy for learning, which leads to that the moderate N performs the best as in our results.

6 DISCUSSION AND FUTURE WORKS

We would also like to point out that PB-LRL can be further extended and improved in a number of possible directions, which are exciting to explore in the future.

Automatic and adaptive selection of the meta update frequency N. As we discussed in Section 4.2 and 5.4, the selection of hyper-parameter N is essential for both the theory and the experiment. Moreover, the best selection of N usually depends on the underlying task distributions. In practice, one may need to try different N's manually to determine which N works better. However, we can get rid of the manual selection by designing an automatic and adaptive algorithm. Our algorithm 1 uses a fixed N, but it also works if N changes after each meta update. More specifically, one can increase N if the learning performance is unstable, and decrease N if the performance is stable but raises slowly. Based on our observation in experiments, it is better to start from a small N and increase it when the performance fluctuates. In this way, we can hopefully converge to an optimal N and maximize the average rewards.

More gradient descent steps for optimizing default policy. The current PB-LRL uses a one-step gradient descent to update the default policy in the "meta update" (Algorithm 1 line 16-22). The gradient step reduces the PAC-Bayes upper bound, but does not necessarily achieve the minimum. If one wants to really minimize the upper bound, more gradient steps are needed. The difficulty of doing so is that we already lost access to the past tasks, thus we cannot directly evaluate the cost of a new default policy on an old task. There are two approximate solutions: (1) using importance sampling to evaluate the new policy with the old trajectories, and (2) building a deep prediction model for each task $\hat{\mathcal{M}}_i$ during single-task learning and evaluate new policies on $\hat{\mathcal{M}}_i$, in the scenario where the interactions with each task suffice to build a good approximate model.

7 ETHICS STATEMENT

We have read the code of ethics carefully and ensured that our paper conforms to them.

8 REPRODUCIBILITY STATEMENT

We make our best effort to ensure the reproducibility of the paper's experiments and provide clear guidelines. More specifically, we include detailed experiment setup information in C.1. Besides, we give detailed description on the architecture of the neural network we use and we explain how we control and adjust different variables in C.2. We list all the important hyperparameters and their default values in C.2. There is no extra data processing steps to be conducted. All the work is done in the code in an end-to-end manner. We also give the link to the source code in the supplementary material.

We explain all assumptions of our theorems clearly, and provide the complete proof of the theorems in the appendix §A.

REFERENCES

Marcin Andrychowicz, Misha Denil, Sergio Gómez, Matthew W Hoffman, David Pfau, and Tom Schaul. Learning to learn by gradient descent by gradient descent. pp. 9.

Samy Bengio, Yoshua Bengio, Jocelyn Cloutier, and Jan Gecsei. On the Optimization of a Synaptic Learning Rule. pp. 29.

Yoshua Bengio, Samy Bengio, and Jocelyn Cloutier. Learning a Synaptic Learning Rule. In *In Conference on Optimality in Biological and Artificial Networks*, 1991.

Emma Brunskill and Lihong Li. Sample complexity of multi-task reinforcement learning. In *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*, UAI'13, pp. 122131, Arlington, Virginia, USA, 2013. AUAI Press.

Karl Cobbe, Oleg Klimov, Chris Hesse, Taehoon Kim, and John Schulman. Quantifying Generalization in Reinforcement Learning. In *International Conference on Machine Learning*, pp. 1282–1289. PMLR, May 2019.

Karl Cobbe, Christopher Hesse, Jacob Hilton, and John Schulman. Leveraging Procedural Generation to Benchmark Reinforcement Learning. *arXiv:1912.01588 [cs, stat]*, July 2020.

Yan Duan. Meta Learning for Control. pp. 125.

Gintare Karolina Dziugaite, Kyle Hsu, Waseem Gharbieh, Gabriel Arpino, and Daniel M. Roy. On the role of data in PAC-Bayes bounds. *arXiv:2006.10929 [cs, stat]*, October 2020.

M. Fard and Joelle Pineau. PAC-Bayesian Model Selection for Reinforcement Learning. *Advances in Neural Information Processing Systems*, 23:1624–1632, 2010.

Mahdi MIlani Fard, Joelle Pineau, and Csaba Szepesvari. PAC-Bayesian Policy Evaluation for Reinforcement Learning. *arXiv:1202.3717 [cs, stat]*, February 2012.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. March 2017.

Pascal Germain, Alexandre Lacasse, François Laviolette, and Mario Marchand. PAC-Bayesian learning of linear classifiers. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pp. 353–360, New York, NY, USA, June 2009. Association for Computing Machinery. ISBN 978-1-60558-516-1. doi: 10.1145/1553374.1553419.

Anirudh Goyal, Riashat Islam, Daniel Strouse, Zafarali Ahmed, Matthew Botvinick, Hugo Larochelle, Yoshua Bengio, and Sergey Levine. InfoBot: Transfer and Exploration via the Information Bottleneck. *arXiv:1901.10902 [cs, stat]*, April 2019.

David Ha, Andrew Dai, and Quoc V. Le. HyperNetworks. *arXiv:1609.09106 [cs]*, December 2016.

Sepp Hochreiter, Arthur Younger, and Peter Conwell. *Learning To Learn Using Gradient Descent*. September 2001. ISBN 978-3-540-42486-4. doi: 10.1007/3-540-44668-0_13.

M. Husken and C. Goerick. Fast learning for problem classes using knowledge based network initialization. In *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*, volume 6, pp. 619–624 vol.6, July 2000. doi: 10.1109/IJCNN.2000.859464.

Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *arXiv:1502.03167 [cs]*, March 2015.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, March 2017. ISSN 0027-8424, 1091-6490. doi: 10.1073/pnas.1611835114.

Gregory Koch. Siamese Neural Networks for One-Shot Image Recognition. pp. 30.

Philipp Krähenbühl, Carl Doersch, Jeff Donahue, and Trevor Darrell. Data-dependent Initializations of Convolutional Neural Networks. *arXiv:1511.06856 [cs]*, September 2016.

John Langford and Rich Caruana. (not) bounding the true error. *Advances in Neural Information Processing Systems*, 2:809–816, 2002.

John Langford and John Shawe-Taylor. PAC-Bayes & margins. In *Proceedings of the 15th International Conference on Neural Information Processing Systems*, NIPS'02, pp. 439–446, Cambridge, MA, USA, January 2002. MIT Press.

Ke Li and Jitendra Malik. Learning to Optimize Neural Nets. *arXiv:1703.00441 [cs, math, stat]*, November 2017.

Dougal Maclaurin, David Duvenaud, and Ryan P Adams. Gradient-based Hyperparameter Optimization through Reversible Learning. pp. 10.

Anirudha Majumdar, Alec Farid, and Anoopkumar Sonar. Pac-bayes control: learning policies that provably generalize to novel environments. *The International Journal of Robotics Research*, 40 (2-3):574–593, 2021.

Andreas Maurer. A note on the pac bayesian theorem. *arXiv preprint cs/0411099*, 2004.

David A McAllester. Some pac-bayesian theorems. *Machine Learning*, 37(3):355–363, 1999a.

David A. McAllester. Some PAC-Bayesian Theorems. *Machine Learning*, 37(3):355–363, December 1999b. ISSN 1573-0565. doi: 10.1023/A:1007618624809.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing Atari with Deep Reinforcement Learning. *arXiv:1312.5602 [cs]*, December 2013.

D. K. Naik and R. J. Mammone. Meta-neural networks that learn by learning. In *[Proceedings 1992] IJCNN International Joint Conference on Neural Networks*, volume 1, pp. 437–442 vol.1, June 1992. doi: 10.1109/IJCNN.1992.287172.

Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nathan Srebro. Exploring Generalization in Deep Learning. *arXiv:1706.08947 [cs]*, July 2017.

Behnam Neyshabur, Srinadh Bhojanapalli, and Nathan Srebro. A PAC-Bayesian Approach to Spectrally-Normalized Margin Bounds for Neural Networks. *arXiv:1707.09564 [cs]*, February 2018.

Alex Nichol, Vicki Pfau, Christopher Hesse, Oleg Klimov, and John Schulman. Gotta Learn Fast: A New Benchmark for Generalization in RL. *arXiv:1804.03720 [cs, stat]*, April 2018.

Vincent Pacelli and Anirudha Majumdar. Learning Task-Driven Control Policies via Information Bottlenecks. *arXiv:2002.01428 [cs, math, stat]*, February 2020.

X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel. Sim-to-Real Transfer of Robotic Control with Dynamics Randomization. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3803–3810, May 2018. doi: 10.1109/ICRA.2018.8460528.

Sachin Ravi and Hugo Larochelle. OPTIMIZATION AS A MODEL FOR FEW-SHOT LEARNING. pp. 11, 2017.

Scott Reed, Yutian Chen, Thomas Paine, Aäron van den Oord, S. M. Ali Eslami, Danilo Rezende, Oriol Vinyals, and Nando de Freitas. Few-shot Autoregressive Density Estimation: Towards Learning to Learn Distributions. *arXiv:1710.10304 [cs]*, February 2018.

Tim Salimans and Diederik P. Kingma. Weight Normalization: A Simple Reparameterization to Accelerate Training of Deep Neural Networks. *arXiv:1602.07868 [cs]*, June 2016.

Andrew M. Saxe, James L. McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv:1312.6120 [cond-mat, q-bio, stat]*, February 2014.

J. Schmidhuber. Learning to Control Fast-Weight Memories: An Alternative to Dynamic Recurrent Networks. *Neural Computation*, 4(1):131–139, January 1992. ISSN 0899-7667. doi: 10.1162/neco.1992.4.1.131.

John Schulman. Trust Region Policy Optimization. pp. 9.

Matthias Seeger. PAC-Bayesian Generalisation Error Bounds for Gaussian Process Classification. *Journal of Machine Learning Research*, 3(Oct):233–269, 2002. ISSN ISSN 1533-7928.

Aman Sinha, Hongseok Namkoong, Riccardo Volpi, and John Duchi. Certifying Some Distributional Robustness with Principled Adversarial Training. *arXiv:1710.10571 [cs, stat]*, May 2020.

Jake Snell, Kevin Swersky, and Richard S. Zemel. Prototypical Networks for Few-shot Learning. *arXiv:1703.05175 [cs, stat]*, June 2017.

Xingyou Song, Yiding Jiang, Stephen Tu, Yilun Du, and Behnam Neyshabur. Observational Overfitting in Reinforcement Learning. *arXiv:1912.02975 [cs, stat]*, December 2019.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, January 2014. ISSN 1532-4435.

Yanchao Sun, Xiangyu Yin, and Furong Huang. Temple: Learning template of transitions for sample efficient multi-task rl, 2020.

Richard S Sutton, Andrew G Barto, et al. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge, 1998.

Jie Tan, Tingnan Zhang, Erwin Coumans, Atil Iscen, Yunfei Bai, Danijar Hafner, Steven Bohez, and Vincent Vanhoucke. Sim-to-Real: Learning Agile Locomotion For Quadruped Robots. *arXiv:1804.10332 [cs]*, May 2018.

Sebastian Thrun and Lorien Pratt. Learning to Learn: Introduction and Overview. In Sebastian Thrun and Lorien Pratt (eds.), *Learning to Learn*, pp. 3–17. Springer US, Boston, MA, 1998. ISBN 978-1-4615-5529-2. doi: 10.1007/978-1-4615-5529-2_1.

J. Tobin, L. Biewald, R. Duan, M. Andrychowicz, A. Handa, V. Kumar, B. McGrew, A. Ray, J. Schneider, P. Welinder, W. Zaremba, and P. Abbeel. Domain Randomization and Generative Models for Robotic Grasping. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3482–3489, October 2018. doi: 10.1109/IROS.2018.8593933.

Sushant Veer and Anirudha Majumdar. Probably Approximately Correct Vision-Based Planning using Motion Primitives. *arXiv:2002.12852 [cs, eess, math]*, November 2020.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching Networks for One Shot Learning. *Advances in Neural Information Processing Systems*, 29: 3630–3638, 2016.

Ziyu Wang, Victor Bapst, Nicolas Heess, Volodymyr Mnih, Remi Munos, Koray Kavukcuoglu, and Nando de Freitas. Sample Efficient Actor-Critic with Experience Replay. *arXiv:1611.01224 [cs]*, July 2017.

APPENDICES TO FEW-SHOT LIFELONG REINFORCEMENT LEARNING WITH GENERALIZATION GUARANTEES: AN EMPIRICAL PAC-BAYES APPROACH

A DETAILED PROOFS

We first give some useful definitions and backgrounds.

Definition 1 (KL divergence between two Bernoulli distributions with probability of success a and probability of success b)**.** *For scalars $a, b \in [0, 1]$, we define*

$$\mathbb{D}_{KL}(a\|b) \coloneqq \mathbb{D}_{KL}(B(a)\|B(b)) = a \log \frac{a}{b} + (1-a) \log \frac{1-a}{1-b} \tag{8}$$

where $B(a)$ denotes the Bernoulli distribution on $\{0, 1\}$ with parameter (i.e. mean) a

Definition 2 (KL inverse)**.** *PAC-Bayes bounds are typically expressed as bounds on a quantity $b^* \in [0, 1]$ of the form $\mathbb{D}_{KL}(a\|b^*) \leq c$ (for some $a \in [0, 1]$ and $c \geq 0$). These bounds can then be used to upper bound b^* by the KL inverse as follows:*

$$b^* \leq \mathbb{D}_{KL}^{-1}(a\|c) := \sup \{b \in [0, 1] | \mathbb{D}_{KL}(a\|b) \leq c\} . \tag{9}$$

One can treat $\mathbb{D}_{KL}^{-1}(a\|c)$ as the supreme of all b such that $\mathbb{D}_{KL}(a\|b) \leq c$.

Now we provide the proofs of our main theorems. We first prove the theorem 2.

Proof. Based on the upper bound for the KL inverse in theorem 1, we applying the Pinsker's inequality: $\mathbb{D}^{-1}(a\|c) \leq a + \sqrt{c/2}$ to the RHS of equation (3), so we get

$$\mathbb{D}_{KL}^{-1}\left(l_S(P)\|\frac{\mathbb{D}_{KL}\left(P\|P_0\right) + \log\left(\frac{2\sqrt{N}}{\delta}\right)}{2N}\right) \leq \underbrace{l_S(P)}_{\text{Training loss}} + \underbrace{\sqrt{\frac{\mathbb{D}_{KL}\left(P\|P_0\right) + \log\left(\frac{2\sqrt{N}}{\delta}\right)}{2N}}}_{\text{Regularizer"}}$$

, combines the inequality (3), which proves the equation (4) □

We now prove the theorem 3.

Proof. The proof follows from theorem 2 given a reduction from the lifelong RL problem to the supervised learning setting.

In the supervised learning, we are provided with the input data, e.g. z, in the lifelong RL, we have the task $\mathcal{M}$. Choosing a hypothesis h_w of supervised learning corresponds to choosing a lifelong RL policy π. A hypothesis h_w sends z to a label z', while the τ is a roll-out trajectory from task $\mathcal{M}$'s MDP under the policy π. The label incurs a loss $l(h_w; z)$, while the τ has a total expected reward $C(\pi; \mathcal{M}) = -\eta_{\mathcal{M}}(\pi)$.

Let Π be the whole policy space parameterized by $\theta \in \mathbb{R}^d$. Let P be any policy distribution over the policy space Π. Let P_0 be any prior distribution. By the analogy of theorem 2, we have the training cost over multiple tasks $\frac{1}{N}\sum_{i=1}^{N} C(P; \mathcal{M}_i)$, and a regularizer term $\sqrt{\frac{\mathbb{D}_{KL}(P\|P_0)+\log\left(\frac{2\sqrt{N}}{\delta}\right)}{2N}}$, which together bounds the true expected loss/cost of the policy distribution over the underlying task distribution $\mathbb{E}_{\mathcal{M}\sim\mathcal{D}}\, C(P; \mathcal{M})$. □

Definition 3. *Follow the notations from section 3.3 for the supervised learning, we let S be a sample set of N pairs (z, z'), and we define true error $e(h) \coloneqq \mathbb{P}_{\mathcal{D}}(h(z) \neq z')$, and the empirical error rate $\hat{e}(h) \coloneqq \mathbb{P}_S(h(z) \neq z')$. Let P be a distribution over the hypotheses which can be found in an example dependent manner. Let P_0 be a distribution over the hypotheses which is chosen a prioriwithout dependence on the examples. Let $e_{P(h)} = \mathbb{E}_{h\sim P}\, e(h)$ be the true error rate of the stochastic hypothesis which, in any evaluation, draws a hypothesis h from P, and outputs $h(z)$. Let $\hat{e}_{P(h)} = \mathbb{E}_{h\sim P}\, \hat{e}(h)$ be the average empirical error rate of the same stochastic hypothesis. Let $\hat{e}(h)_{\hat{P}(h)} \coloneqq \mathbb{P}_{S,\hat{P}}(h(z) \neq z')$ be the observed rate of failure of a K random hypotheses drawn according to P and applied to a random training example.*

Theorem 5 (Sample Convergence Bound Langford & Caruana (2002)). *For all distributions, P, for all sample sets S, for any $\delta \in (0,1)$*

$$\mathbb{P}_P\left(\mathbb{D}_{KL}(\hat{e}(h)_{\hat{P}(h)} \| \hat{e}(h)_{P(h)}) \geq \frac{\ln \frac{2}{\delta}}{K}\right) \leq \delta \tag{10}$$

where K is the number of evaluations of the stochastic hypothesis.

Finally, we give the proof of the theorem 4.

Proof. Upon having the optimal policy distribution $P^* \coloneqq \mathcal{N}(\mu^*, \sigma^*)$ and the corresponding policy function $\pi^* \sim P^*$, we obtain the $\theta^* = \mu^* + \sqrt{\sigma^*} \odot \epsilon$ from PB-LRL algorithm. We are interested in the training cost

$$C(P^*; \mathcal{M}) = \mathop{\mathbb{E}}_{\pi \sim P^*}[C(\pi^*; \mathcal{M})],$$

so we draw large number of independent, identically distributed parameter samples $\theta_1, \cdots, \theta_K$ from $\mathcal{N}(\mu^*, \sigma^*)$ to estimate the above expectation. Thus, we denote this estimator as

$$\bar{C}(P^*; \mathcal{M}) \coloneqq \frac{1}{NK} \sum_{i=1}^{N} \sum_{j=1}^{K} C(\pi^*; \mathcal{M}_i).$$

For any $\delta' \in (0,1)$, by theorem 5, we have with high probability $1 - \delta'$

$$C(P^*; \mathcal{M}) \leq \widehat{C}(P^*; \mathcal{M}) \coloneqq \mathbb{D}_{KL}^{-1}\left(\bar{C}(P^*; \mathcal{M}) \| \frac{1}{K} \log(\frac{2}{\delta'})\right) \tag{11}$$

We then bound the difference between the true expected cost $\mathbb{E}_{\mathcal{M} \sim \mathcal{D}} C(P^*; \mathcal{M})$ and training cost $C(P^*; \mathcal{M})$. Followed from theorem 1, for any $\delta \in (0,1)$, with high probability $1 - \delta$,

$$\mathop{\mathbb{E}}_{\mathcal{M} \sim \mathcal{D}} C(P^*; \mathcal{M}) \leq \mathbb{D}_{KL}^{-1}\left(C(P^*; \mathcal{M}) \| \frac{\mathbb{D}_{KL}\left(P^* \| P_0\right) + \log\left(\frac{2\sqrt{N}}{\delta}\right)}{2N}\right) \tag{12}$$

holds. Since $C(P^*; \mathcal{M}) \leq \widehat{C}(P^*; \mathcal{M})$, using the definition from 8, for any $c >= 0$, we have

$$\sup\left\{b \in [0,1] \| \mathbb{D}_{KL}\left(\widehat{C}(P^*; \mathcal{M}) \| b\right) = c\right\} \geq \sup\left\{b \in [0,1] \| \mathbb{D}_{KL}\left(C(P^*; \mathcal{M}) \| b\right) = c\right\},$$

therefore, we have

$$\sup\left\{b \in [0,1] \| \mathbb{D}_{KL}\left(\widehat{C}(P^*; \mathcal{M}) \| b\right) \leq \frac{\mathbb{D}_{KL}\left(P^* \| P_0\right) + \log\left(\frac{2\sqrt{N}}{\delta}\right)}{2N}\right\}$$

$$\geq \sup\left\{b \in [0,1] \| \mathbb{D}_{KL}\left(C(P^*; \mathcal{M}) \| b\right) \leq \frac{\mathbb{D}_{KL}\left(P^* \| P_0\right) + \log\left(\frac{2\sqrt{N}}{\delta}\right)}{2N}\right\},$$

i.e.,

$$\mathbb{D}_{KL}^{-1}\left(\widehat{C}(P^*; \mathcal{M}) \| \frac{\mathbb{D}_{KL}\left(P^* \| P_0\right) + \log\left(\frac{2\sqrt{N}}{\delta}\right)}{2N}\right) \geq \mathbb{D}_{KL}^{-1}\left(C(P^*; \mathcal{M}) \| \frac{\mathbb{D}_{KL}\left(P^* \| P_0\right) + \log\left(\frac{2\sqrt{N}}{\delta}\right)}{2N}\right)$$

Let A_1 be the event of equation (11) not held, and A_2 be the event of equation (12) not held, each condition of two equations has probability $1 - \delta$ and $1 - \delta'$, by the union bound, the total event

$$\mathop{\mathbb{E}}_{\mathcal{M} \sim \mathcal{D}} C(P^*; \mathcal{M}) \leq \mathbb{D}_{KL}^{-1}\left(\widehat{C}(P^*; \mathcal{M}) \| \frac{\mathbb{D}_{KL}\left(P^* \| P_0\right) + \log\left(\frac{2\sqrt{N}}{\delta}\right)}{2N}\right) \tag{13}$$

is bounded with

$$\mathbb{P}(\text{the bounds satisfied}) := 1 - \mathbb{P}(\bigcup_{i=1}^{2} A_i) \geq 1 - \sum_{i=1}^{2} \mathbb{P}(A_i) = 1 - \delta - \delta'$$

□

B PB-LRL

Algorithm 1 PAC-Bayes Lifelong RL (PB-LRL)

Input: policy dimension d; learning rates α, β; update frequency N; failure probability δ; the number of steps allowed in each task T; prior evolving speed λ
Initialize default policy mean and derivation μ, $\sigma \in \mathbb{R}^d$;
Initialize prior policy mean and derivation μ_0, $\sigma_0 \in \mathbb{R}^d$;
Initialize cost gradients $g_\mu[i] = 0, \forall i = 1, 2, \cdots, N$, $g_\sigma[i] = 0, \forall i = 1, 2, \cdots, N$
for $i = 1, 2, 3, \cdots$ **do**
  Receive a new task $\mathcal{M}_i \sim \mathcal{D}$
  Sample $\epsilon \sim \mathcal{N}(0, I_d)$
  Set initial policy $\theta \leftarrow \mu + \sqrt{\sigma} \odot \epsilon$
  Use θ to rollout trajectories $\mathcal{T}$
  Compute cost $c = C(\mathcal{T})$
  Compute gradients $g_\mu[i] = \nabla_\mu c$, $g_\sigma[i] = \nabla_\sigma c$
  Make a copy of policy $\theta_i = \theta$
  for $t = 1, 2, \cdots, T$ **do**
    Optimize θ_i for $\mathcal{M}_i$ with any single-task method with learning rate α
  end for
  if $i \bmod N = 0$ **then**
    {Meta Update by using regularization loss function L}
    $\mu \leftarrow \mu - \beta(\sum_{k=i-N}^{N} g_\mu[k] + \nabla_\mu L(\mu, \sigma, \mu_0, \sigma_0, N))$
    $\sigma \leftarrow \sigma - \beta(\sum_{k=i-N}^{N} g_\sigma[k] + \nabla_\sigma L(\mu, \sigma, \mu_0, \sigma_0, N)))$
    $\mu_0 \leftarrow (1 - \lambda)\mu_0 + \lambda\mu$
    $\sigma_0 \leftarrow (1 - \lambda)\sigma_0 + \lambda\sigma$
  end if
end for

C ENVIRONMENT AND EXPERIMENT

C.1 OPENAI AND MAMUJOCO ENVIRONMENT

(a) (b) (c)

Figure 6: The illustration of three environments. (a) CartPole, (b) LunarLander, (c) Swimmer

C.1.1 CARTPOLE ENVIRONMENT

In the classic CartPole environment depicted in §C.1 Figure 6(a), a pole is attached by an un-actuated joint to a cart, which moves along a frictionless track. The system is controlled by applying a force of +1 or -1 to the cart. A reward of +1 is provided for every timestep that the pole remains upright. The episode ends when the pole falls or the cart moves far away from the center.

C.1.2 LUNARLANDER ENVIRONMENT

The general task in LunarLander environment (as shown in §C.1 Figure 6(b)) is to let lunar module land at a pre-defined goal location.

C.1.3 SWIMMER ENVIRONMENT

The swimmer (as shown in §C.1 Figure 6(c)) is a planar robot with 3 links and 2 actuated joints. Fluid is simulated through viscosity forces, which apply drag on each link, allowing the swimmer to move forward.

C.2 HYPER-PARAMETERS

In all experiments, we use a 2-layer MLP with 32 nodes per layer to parametrize the policy. The single-task learning rate α and meta learning rate β are both set to be 10^{-4}. The prior-update-speed parameter λ is set as 0.5. The default policy and the prior policy are both initialized as standard normal distributions.

C.3 EXPERIMENTS

All experimental results are averaged over 10 random seeds to reduce noise, and we plot the confidence interval by a 0.32 (10%) standard deviation. And to increase readability, we smooth all the curves in plots using the exponential moving average algorithm with smoothing parameter 0.99.

The code is located at https://www.dropbox.com/sh/5m91t7xvems7ncy/AACjZTYPOaJqDHWQYf6mKUDEa?dl=0.