Few-shot Lifelong Reinforcement Learning with Generalization Guarantees

Anonymous Authors[1]

Abstract

In lifelong reinforcement learning (RL), 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. Most existing lifelong RL or meta RL algorithms are based on heuristic methods, lacking theoretical guarantees. In this paper, we propose a new lifelong RL approach with theoretical performance guarantee, by extending the PAC-Bayes theory in supervised learning to the regime of RL. 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 learns better prior distributions of policies, which results in tighter generalization bounds and further improves future learning. We test our proposed algorithm on various OpenAI's Gym environments. Experimental results show that our proposed algorithms can efficiently adapt to new tasks by continuously distilling knowledge from past tasks and outperforms recent state-of-the-art lifelong RL algorithms.

1. Introduction

Deep reinforcement learning has demonstrated outstanding performance over many challenging tasks, such at playing games, visual navigation, robotic control and manipulation. However, while the current deep RL methods can learn individual policies for specific tasks, it remains very challenging to train a single network that generalizes across different tasks, which is more common in practice. Therefore, we consider the problem of lifelong reinforcement learning, where an agent is facing a series of different but related tasks. We want the policy learnt from the previous tasks can be fast and reliably transferred to new tasks, even in a few-shot learning 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 concrete goal locations. Inappropriate knowledge could result in "negative transfer", which not only does not help, but also harm 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.

Some lifelong learning methods or multi-task learning methods partially address these challenges by providing formal PAC guarantees (Brunskill & Li, 2013; Sun et al., 2020). However, their theory mainly focus on tabular RL, and can not adapt to large-scale environments. For large and complex environments, existing lifelong RL or meta RL works (Finn et al., 2017) use deep neural networks to achieve efficient adaptation. Although good empirical results are reported, these methods are usually based on heuristic, lacking theoretical guarantees.

Therefore, our goal is to design an algorithm that can learn common knowledge from past tasks and quickly adapt to new tasks, as well as provide a reliable theoretical guarantee to avoid negative transfer.

A recent work (Majumdar et al., 2018) propose to optimize a policy distribution over past tasks and directly use it for the following tasks. By applying the well-known PAC-Bayes theory, their algorithm has a nice theoretical guarantee and works for deep neural networks. 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 are heavily dependent on the selection of a prior policy distribution, and a bad selection of prior may cause significant drop of the performance.

Inspired by Majumdar et al. (2018), we propose an algorithm called PAC-Bayes Lifelong RL (PB-LRL). PB-LRL

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

seeks a *default policy distribution* that generalize 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. As justified by Finn et al. (2017), a good policy initialization can significantly improve the learning performance and achieve fast adaptation. We continuously update the default policy distribution by repeatedly applying the PAC-Bayes bound. More importantly, we propose to "learn" the prior policy distribution required in PAC-Bayes theory along with the default policy distribution. Thus, we do not require any expert knowledge and the prior can be automatically updated to improve the performance.

Our contribution mainly includes the following aspects:

1. We present a practical algorithms for lifelong reinforcement learning with theoretical guarantee, which learns a policy distribution that can adapt to new tasks within a small number of steps.

2. Our proposed 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.

3. Experimental results on diverse OpenAI's Gym environments show that our algorithm significantly outperforms 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) and Finn et al. (2017), which is mostly related to lifelong learning by an efficient meta-leaner. Schmidhuber (1992) and Bengio et al. have trained a meta-learner that learns how to update the parameters of the learner's model. Hochreiter et al. (2001), Andrychowicz et al., Li & Malik (2017) and Ha et al. (2016) applied this approach to learn to optimize deep networks for few-shot learning.

Koch, Ravi & Larochelle (2017), Vinyals et al. (2016), Reed et al. (2018) and Snell et al. (2017) have developed few-shot methods for supervised-learning. Due to the difference of problem settings of supervised-learning and reinforcement learning, which prevents these methods from extension to reinforcement learning settings.

Duan, Wang et al. (2017) has trained memory augmented recurrent learner to adapt to many tasks, and learned fast reinforcement learning agents. Saxe et al. (2014), Kirkpatrick et al. (2017), Krähenbühl et al. (2016), Salimans & Kingma (2016), Husken & Goerick (2000) and Maclaurin et al. have considered sensitivity in deep networks, often by 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 sensitivity on a given task distribution. Their method is agnostic to the form of the model and to the particular learning task, 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, many works have mentioned the generalization gap under the policy space for new tasks and new environments. Early works such Nichol et al. (2018), Mnih et al. (2013) and Song et al. (2019) have quantified the generalization of learned policies. An effective generalization can require an extremely large number of training environments Cobbe et al. (2019), Cobbe et al. (2020). And generalization gap always exists in the 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) and Goyal et al. (2019) have applied regularization approaches to improve the generalization, but they do not explicitly exploit the structure of the RL, or limited to a particular task which prevents learner to exploit causal relationships in the environment. Sinha et al. (2020) have done adversarial perturbations to the underlying data distribution to provides robustness guarantees but the amount of perturbations is not easy to learn.

PAC-Bayes Theory: In classical supervised learning works done 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) and Neyshabur et al. (2017), they utilized PAC-Bayes theory McAllester (1999b) to study the generalization bounds.

Neu et al. (2017), Kearns et al., Bagnell & Schneider (2001) and Bagnell (2004) studied "regularizing" policies to prevent over-fitting and lead to sample efficiency. So Schulman, Fard & Pineau (2010) and Fard et al. (2012) have applied PAC-Bayes theory to learn controlled policies for Markov Decision Processes with provable sample complexity bounds. However, these approaches required learner with multiple interactions with a given MDP which is less efficient.

Then Majumdar et al. (2018), Veer & Majumdar (2020) have made provable generalization guarantees under distributional shifts and learned policies that could zero-shot generalized well 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 multi-task lifelong settings which in general is more complicated.

In our work, we generated a PAC-Bayes bound as part of the objective 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 focused on the multi-task lifelong RL settings, and we used zero-shot generalization to novel tasks by directly operating on policy space to obtain upper bounds on the expected cost of meta policies, which functioned as a good policy initialization for new tasks.

Like Finn et al. (2017) 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 to the learner and achieve extremely efficient adaptation to new tasks in only a few gradient steps.

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 Markov Decision Process (MDP), which is denoted by a tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, P, R, \gamma, \mu \rangle$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, P is the transition kernel, R is the reward function, $\gamma \in (0, 1)$ is the discount factor, and μ 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 $s_1 \sim \mu$, $a_t \sim \pi(a|s_t)$, $s_{t+1} \sim P(s|s_t, a_t)$ 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 $\eta(\pi) = \mathbb{E}_{\tau\sim\pi}[r(\tau)] = \mathbb{E}_{s_1,a_1,\cdots\sim\mu,\pi,P,R}[\sum_{t=1}^{\infty} \gamma^{t-1} r_t]$.

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 $\mathcal{S}, \mathcal{A}, \gamma, \mu$, but may have different transition probabilities P 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:

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

and the true expected loss as:

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

The following theorem provides a useful upper bound for the true expected loss.

Theorem 1 (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:*

$$\underbrace{l_{\mathcal{D}}(P)}_{\text{True expected loss}} \leq \underbrace{l_S(P)}_{\text{Training loss}} + \underbrace{\sqrt{\frac{\mathbb{D}_{KL}\left(P \| P_0\right) + \log\left(\frac{2\sqrt{N}}{\delta}\right)}{2N}}}_{\text{"Regularizer"}} \tag{3}$$

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.

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. (2018). We carefully extend the work of Majumdar et al. (2018) 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

$$C(P; \mathcal{M}) = \mathop{\mathbb{E}}_{\pi\sim P}[C(\pi; \mathcal{M})] = \mathop{\mathbb{E}}_{\pi\sim P}[-\eta_{\mathcal{M}}(\pi)], \tag{4}$$

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 2 (PAC-Bayes Bound for Lifelong RL) *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*

$$\underbrace{\mathop{\mathbb{E}}_{\mathcal{M}\sim\mathcal{D}} C(P;\mathcal{M})}_{\textit{True cost}} \leq \underbrace{\frac{1}{N}\sum_{i=1}^{N} C(P;\mathcal{M}_i)}_{\textit{Training cost}} + \underbrace{\sqrt{\frac{\mathbb{D}_{KL}\left(P\|P_0\right) + \log\left(\frac{2\sqrt{N}}{\delta}\right)}{2N}}}_{\textit{Regularizer}} \tag{5}$$

with probability at least $1-\delta$.

Remarks. (1) Theorem 2 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 2 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.

Majumdar et al. (2018) 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 challenging 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. (2018) 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 2, we introduce our algorithm called PAC-Bayes Lifelong RL (PB-LRL), 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.

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 $\mu, \sigma \in \mathbb{R}^d$;
Initialize prior policy mean and derivation $\mu_0, \sigma_0 \in \mathbb{R}^d$;
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
  end for
  if $i \bmod N = 0$ **then**
    {Meta Update}
    $\mu \leftarrow \mu - \beta(\sum_{i=1}^N g_\mu[i] + \nabla_\mu L_r(\mu, \sigma, \mu_0, \sigma_0, N))$
    $\sigma \leftarrow \sigma - \beta(\sum_{i=1}^N g_\sigma[i] + \nabla_\sigma L_r(\mu, \sigma, \mu_0, \sigma_0, N)))$
    $\mu_0 \leftarrow (1-\lambda)\mu_0 + \lambda\mu$
    $\sigma_0 \leftarrow (1-\lambda)\sigma_0 + \lambda\sigma$
  end if
end for

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 2, 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.

The proposed algorithm PB-LRL is illustrated in Algorithm 1, where we use d-dimensional Gaussian distributions as the default and prior policy distribution, i.e., $\mathcal{N}(\mu, \sigma)$ and $\mathcal{N}(\mu_0, \sigma_0)$. 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 (5) 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. The regularization function L in line 18-19 is defined as

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

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). 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 learning each task separately and updating policy based on each tasks 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 w.r.t. 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 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. 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 episodes. Note that the number of per-task steps is much less than usually required in traditional learning, so that the agent has to achieve fast adaptation to gain high rewards.

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

All experimental results are averaged over 10 random seeds to reduce noise, and we plot the confidence interval by 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.

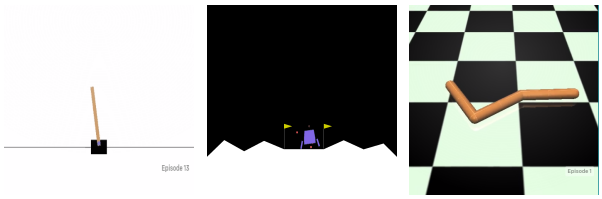
5.1. In the CartPole Environment

In the classic CartPole environment depicted in Figure 1(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. 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.

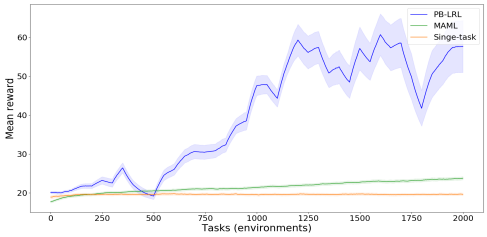
We implement a variant of CartPole, called CartPole-Goal, where the agent is supposed to reach a "goal position" on the track. The reward is defined by $r(x) = \exp|x - x_{goal}|$, where x is the agent's current position, and x_{goal} is the goal's position. In a lifelong setting, the agent interacts 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 standard deviation 0.1 are more similar with each other, but the goals with standard deviation 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 the goals with standard deviation 0.5 are sufficiently 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.



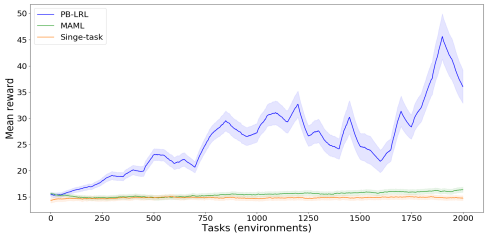
(a) (b) (c)

Figure 1. The illustration of three environments. (a) CartPole, (b) LunarLander, (c) Swimmer

A comparison of the performance of our PB-LRL and baselines is shown in Figure 6, on CartPole-Goal environments with standard deviations 0.1 and 0.5. We can see that PB-LRL significantly outperforms MAML and the single-task learner in both case. As it sees more and more tasks, PB-LRL gradually and constantly improves its performance on each of the single tasks. In Figure 6, because of the high divergence of goals, we observe some fluctuation of PB-LRL, but over 2000 tasks, PB-LRL successfully overcomes the divergence of tasks, and achieves fast adaptation to the later tasks. In contrast, the performance of MAML increases much slower than PB-LRL. In Figure 2(b), MAML achieves small progress compared with the naive single-task due to the high divergence of tasks.



(a)



(b)

Figure 2. Comparison between our algorithm PB-LRL and baselines on CartPole-Goal. (a) $x_{goal} \sim \mathcal{N}(0, 0.1)$, $N = 25$; (b) $x_{goal} \sim \mathcal{N}(0, 0.5)$, $N = 50$;

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.

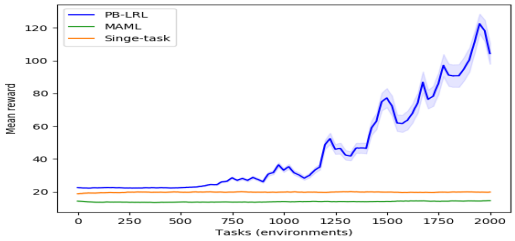
The intrinsic dynamics would affect the lifelong learning performance. For example, in the Cart-Pole environment (Figure 1(a)), different mass of the cart could generate different friction force between the cart and the track, which later affects the acceleration of the cart (the acceleration of the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

To help show that, 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.

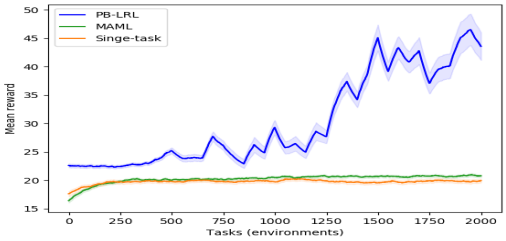
Figure 3 shows performance of our method with baselines under different single task dynamics when we set $\mu_c = 0.5$ or $\mu_c = 1$. Each single task has different m_i which results in 2000 different small dynamics.

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 is more sensitive to single 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.



(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) $\mu_c = 0.5$; (b) $\mu_c = 1.0$;

5.2. In the LunarLander Environment:

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

For the LunarLander environment, the agent (lunar module) needs to learn to land at different goal positions as its tasks. The goal position in a task is defined as the expected landing location x_{goal} of the lunar module. 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 4. 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 some baselines. As show in Figure 4, 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 efficiently in adapting to new tasks by learning from previously seen tasks comparing to the two baselines. The possible reason that our algorithm can outperform MAML is that we use PAC-Bayes theory in updating our policy.

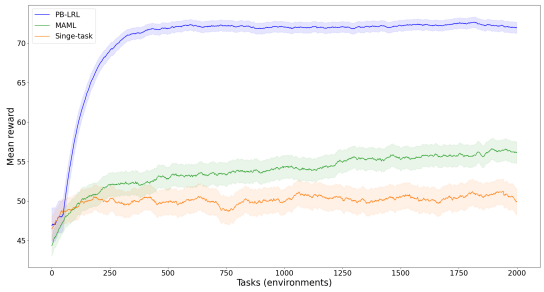


Figure 4. Comparison between our algorithm PB-LRL and baselines on LunarLander (with 0.1 standard deviation interval). The meta learning update frequence is $N = 50$.

5.3. In the Swimmer Environment

The swimmer 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. The goal of the original task is to make the robot swim forward as fast as possible by actuating the two joints.

In our experiment, we simulate the environments by generating random goal velocity for the robot, which is uniformly chosen from $[0.1, 0.2]$. The reward is defined as $r(s, a) = -1.5|v_x - v_{goal}| - 1e^{-4}||a||_2^2$, where $|v_x - v_{goal}|$ 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 5. Our PB-LRL algorithm can efficiently adapt to new tasks and steadily improve the performance, and constantly outperform MAML and the single-task baseline.

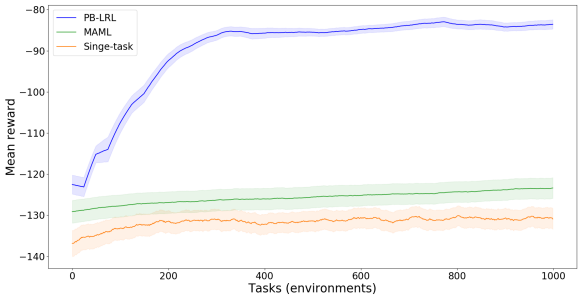
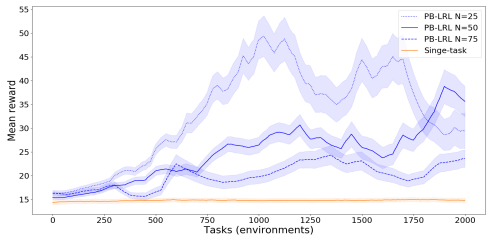
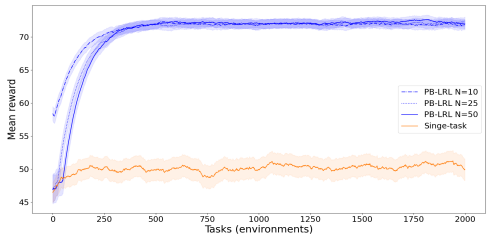


Figure 5. Comparison between our algorithm and baselines on Swimmer. The meta learning update frequence is $N = 25$.

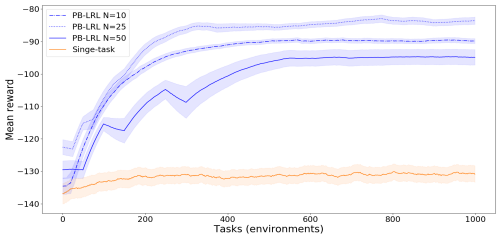
5.4. Investigating the Effects of N



(a)



(b)



(c)

Figure 6. Comparison of different update frequency N on (a) CartPole-Goal; (b) LunarLander; (c) LunarLander.

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 the faster update but will make the learning less stable. The effect of larger N is 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 environments.

In the CartPole environment as shown in Figure 6(a), $N = 25$ exhibits the fast and unstable learning behavior as in theory and $N = 50$ performs the best in the learning w.r.t final reward. In contrast, in LunarLander (as in Figure 6(b)) and Swimmer (as in Figure 6(c)) environments, for all N cases, learning curves are nearly stable. In LunarLander all three N cases reach to similar final rewards, in Swimmer $N = 25$ reaches to the highest reward. One common thing is in all three environments, the largest N leads to the slowest learning, which is also corresponding to 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 portion 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 samples of tasks, this actually imposes an implicit requirement on both efficiency and accuracy for learning, which leads to that the moderate N perforsm the best as in our results.

6. Discussion and Future Works

In this paper, we propose a novel algorithm named PB-LRL for lifelong RL that achieves fast adaptation with theoretical guarantees. We show via systematically designed experiments that PB-LRL significantly outperforms the naive single-task learning algorithm and a state-of-the-art meta-learning algorithm MAML.

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 a good N and maximize the average rewards.

More gradient descent steps for optimizing default policy. The current PB-LRL uses 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 achieves 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 are sufficient to build a good approximate model.

References

Andrychowicz, M., Denil, M., Gómez, S., Hoffman, M. W., Pfau, D., and Schaul, T. Learning to learn by gradient descent by gradient descent. pp. 9.

Bagnell, J. A. *Learning Decisions: Robustness, Uncertainty, and Approximation*. 2004.

Bagnell, J. A. and Schneider, J. G. Autonomous helicopter control using reinforcement learning policy search methods. In *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164)*, volume 2, pp. 1615–1620 vol.2, May 2001. doi: 10.1109/ROBOT.2001.932842.

Bengio, S., Bengio, Y., Cloutier, J., and Gecsei, J. On the Optimization of a Synaptic Learning Rule. pp. 29.

Bengio, Y., Bengio, S., and Cloutier, J. Learning a Synaptic Learning Rule. In *In Conference on Optimality in Biological and Artificial Networks*, 1991.

Brunskill, E. and Li, L. Sample complexity of multi-task reinforcement learning. In *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*, UAI'13, pp. 122–131, Arlington, Virginia, USA, 2013. AUAI Press.

Cobbe, K., Klimov, O., Hesse, C., Kim, T., and Schulman, J. Quantifying Generalization in Reinforcement Learning. In *International Conference on Machine Learning*, pp. 1282–1289. PMLR, May 2019.

Cobbe, K., Hesse, C., Hilton, J., and Schulman, J. Leveraging Procedural Generation to Benchmark Reinforcement Learning. *arXiv:1912.01588 [cs, stat]*, July 2020.

Duan, Y. Meta Learning for Control. pp. 125.

Dziugaite, G. K., Hsu, K., Gharbieh, W., Arpino, G., and Roy, D. M. On the role of data in PAC-Bayes bounds. *arXiv:2006.10929 [cs, stat]*, October 2020.

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

Fard, M. M., Pineau, J., and Szepesvari, C. PAC-Bayesian Policy Evaluation for Reinforcement Learning. *arXiv:1202.3717 [cs, stat]*, February 2012.

Finn, C., Abbeel, P., and Levine, S. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. March 2017.

Germain, P., Lacasse, A., Laviolette, F., and Marchand, M. 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.

Goyal, A., Islam, R., Strouse, D., Ahmed, Z., Botvinick, M., Larochelle, H., Bengio, Y., and Levine, S. InfoBot: Transfer and Exploration via the Information Bottleneck. *arXiv:1901.10902 [cs, stat]*, April 2019.

Ha, D., Dai, A., and Le, Q. V. HyperNetworks. *arXiv:1609.09106 [cs]*, December 2016.

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

Husken, M. and Goerick, C. 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.

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

Kearns, M. J., Mansour, Y., and Ng, A. Y. Approximate Planning in Large POMDPs via Reusable Trajectories. pp. 7.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D., and Hadsell, R. 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.

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

Krähenbühl, P., Doersch, C., Donahue, J., and Darrell, T. Data-dependent Initializations of Convolutional Neural Networks. *arXiv:1511.06856 [cs]*, September 2016.

Langford, J. and Shawe-Taylor, J. 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.

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

Maclaurin, D., Duvenaud, D., and Adams, R. P. Gradient-based Hyperparameter Optimization through Reversible Learning. pp. 10.

Majumdar, A., Farid, A., and Sonar, A. PAC-Bayes Control: Learning Policies that Provably Generalize to Novel Environments. June 2018.

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

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

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

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. Playing Atari with Deep Reinforcement Learning. *arXiv:1312.5602 [cs]*, December 2013.

Naik, D. K. and Mammone, R. J. 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.

Neu, G., Jonsson, A., and Gómez, V. A unified view of entropy-regularized Markov decision processes. *arXiv:1705.07798 [cs, stat]*, May 2017.

Neyshabur, B., Bhojanapalli, S., McAllester, D., and Srebro, N. Exploring Generalization in Deep Learning. *arXiv:1706.08947 [cs]*, July 2017.

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

Nichol, A., Pfau, V., Hesse, C., Klimov, O., and Schulman, J. Gotta Learn Fast: A New Benchmark for Generalization in RL. *arXiv:1804.03720 [cs, stat]*, April 2018.

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

Peng, X. B., Andrychowicz, M., Zaremba, W., and Abbeel, P. 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.

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

Reed, S., Chen, Y., Paine, T., van den Oord, A., Eslami, S. M. A., Rezende, D., Vinyals, O., and de Freitas, N. Few-shot Autoregressive Density Estimation: Towards Learning to Learn Distributions. *arXiv:1710.10304 [cs]*, February 2018.

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

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

Schmidhuber, J. 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.

Schulman, J. Trust Region Policy Optimization. pp. 9.

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

Sinha, A., Namkoong, H., Volpi, R., and Duchi, J. Certifying Some Distributional Robustness with Principled Adversarial Training. *arXiv:1710.10571 [cs, stat]*, May 2020.

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

Song, X., Jiang, Y., Tu, S., Du, Y., and Neyshabur, B. Observational Overfitting in Reinforcement Learning. *arXiv:1912.02975 [cs, stat]*, December 2019.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. 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.

Sun, Y., Yin, X., and Huang, F. Temple: Learning template of transitions for sample efficient multi-task rl, 2020.

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

Tan, J., Zhang, T., Coumans, E., Iscen, A., Bai, Y., Hafner, D., Bohez, S., and Vanhoucke, V. Sim-to-Real: Learning Agile Locomotion For Quadruped Robots. *arXiv:1804.10332 [cs]*, May 2018.

Thrun, S. and Pratt, L. Learning to Learn: Introduction and Overview. In Thrun, S. and Pratt, L. (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.

Tobin, J., Biewald, L., Duan, R., Andrychowicz, M., Handa, A., Kumar, V., McGrew, B., Ray, A., Schneider, J., Welinder, P., Zaremba, W., and Abbeel, P. 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.

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

Vinyals, O., Blundell, C., Lillicrap, T., Kavukcuoglu, K., and Wierstra, D. Matching Networks for One Shot Learning. *Advances in Neural Information Processing Systems*, 29:3630–3638, 2016.

Wang, Z., Bapst, V., Heess, N., Mnih, V., Munos, R., Kavukcuoglu, K., and de Freitas, N. Sample Efficient Actor-Critic with Experience Replay. *arXiv:1611.01224 [cs]*, July 2017.