Fast Policy Learning through Imitation and Reinforcement

Ching-An Cheng
Georgia Tech
Atlanta, GA 30332

Xinyan Yan
Georgia Tech
Atlanta, GA 30332

Nolan Wagener
Georgia Tech
Atlanta, GA 30332

Byron Boots
Georgia Tech
Atlanta, GA 30332

Abstract

Imitation learning (IL) consists of a set of tools that leverage expert demonstrations to quickly learn policies. However, if the expert is suboptimal, IL can yield policies with inferior performance compared to reinforcement learning (RL). In this paper, we aim to provide an algorithm that combines the best aspects of RL and IL. We accomplish this by formulating several popular RL and IL algorithms in a common mirror descent framework, showing that these algorithms can be viewed as a variation on a single approach. We then propose LOKI, a strategy for policy learning that first performs a small but random number of IL iterations before switching to a policy gradient RL method. We show that if the switching time is properly randomized, LOKI can learn to outperform a suboptimal expert and converge faster than running policy gradient from scratch. Finally, we evaluate the performance of LOKI experimentally in several simulated environments.

1 INTRODUCTION

Reinforcement learning (RL) has emerged as a promising technique to tackle complex sequential decision problems. When empowered with deep neural networks, RL has demonstrated impressive performance in a range of synthetic domains (Mnih et al., 2013; Silver et al., 2017). However, one of the major drawbacks of RL is the enormous number of interactions required to learn a policy. This can lead to prohibitive cost and slow convergence when applied to real-world problems, such as those found in robotics (Pan et al., 2017).

Imitation learning (IL) has been proposed as an alternate strategy for faster policy learning that works by leveraging additional information provided through expert demonstrations (Pomerleau, 1989; Schaal, 1999). However, despite significant recent breakthroughs in our understanding of imitation learning (Ross et al., 2011; Cheng and Boots, 2018), the performance of IL is still highly dependent on the quality of the expert policy. When only a suboptimal expert is available, policies learned with standard IL can be inferior to the policies learned by tackling the RL problem directly with approaches such as policy gradients.

Several recent attempts have endeavored to combine RL and IL (Ross and Bagnell, 2014; Chang et al., 2015; Nair et al., 2017; Rajeswaran et al., 2017; Sun et al., 2018). These approaches incorporate the cost information of the RL problem into the imitation process, so the learned policy can *both* improve faster than their RL-counterparts and outperform the suboptimal expert policy. Despite reports of improved empirical performance, the theoretical understanding of these combined algorithms are still fairly limited (Rajeswaran et al., 2017; Sun et al., 2018). Furthermore, some of these algorithms have requirements that can be difficult to satisfy in practice, such as state resetting (Ross and Bagnell, 2014; Chang et al., 2015).

In this paper, we aim to provide an algorithm that combines the best aspects of RL and IL. We accomplish this by first formulating first-order RL and IL algorithms in a common mirror descent framework, and show that these algorithms can be viewed as a single approach that only differs in the choice of first-order oracle. On the basis of this new insight, we address the difficulty of combining IL and RL with a simple, randomized algorithm, named LOKI (Locally Optimal search after K-step Imitation). As its name suggests, LOKI operates in two phases: picking K randomly, it first performs K steps of online IL and then improves the policy with a policy gradient method afterwards. Compared with previous methods that aim to combine RL and IL, LOKI is extremely straightforward to implement. Furthermore, it has stronger theoretical guarantees: by properly randomizing K, LOKI performs as if directly running policy gradient steps with the expert policy as the initial condition. Thus, not only can LOKI improve faster than common RL methods, but it can also significantly outperform a suboptimal expert. This is in contrast to previous methods, such as AGGREVATTE (Ross and Bagnell, 2014), which generally cannot learn a policy that is better than a one-step improvement over the expert policy. In addition to these theoretical contributions, we validate the performance of LOKI in multiple simulated environments. The empirical results corroborate our theoretical findings.

2 PROBLEM DEFINITION

We consider solving discrete-time γ-discounted infinite-horizon RL problems.[1] Let $\mathbb{S}$ and $\mathbb{A}$ be the state and the action spaces, and let Π be the policy class. The objective is to find a policy $\pi \in \Pi$ that minimizes an accumulated cost $J(\pi)$ defined as

$$\min_{\pi\in\Pi} J(\pi), \quad J(\pi) := \mathbb{E}_{\rho_\pi}\left[\sum_{t=0}^{\infty} \gamma^t c(s_t, a_t)\right], \quad (1)$$

in which $s_t \in \mathbb{S}$, $a_t \in \mathbb{A}$, c is the instantaneous cost, and ρ_π denotes the distribution of trajectories $(s_0, a_0, s_1, \dots)$ generated by running the stationary policy π starting from $s_0 \sim p_0(s_0)$.

We denote $Q_\pi(s,a)$ as the Q-function under policy π and $V_\pi(s) = \mathbb{E}_{a\sim\pi_s}[Q_\pi(s,a)]$ as the associated value function, where π_s denotes the action distribution given state s. In addition, we denote $d_{\pi,t}(s)$ as the state distribution at time t generated by running the policy π for the first t steps, and we define a joint distribution $d_\pi(s,t) = (1-\gamma)d_{\pi,t}(s)\gamma^t$ which has support $\mathcal{S} \times [0,\infty)$. Note that, while we use the notation $\mathbb{E}_{a\sim\pi}$, the policy class Π can be either deterministic or stochastic.

We generally will not deal with the objective function in (1) directly. Instead, we consider a surrogate problem

$$\min_{\pi\in\Pi} \mathbb{E}_{s,t\sim d_\pi}\mathbb{E}_{a\sim\pi_s}[A_{\pi'}(s,a)], \quad (2)$$

where $A_{\pi'} = Q_{\pi'} - V_{\pi'}$ is the (dis)advantage function with respect to some fixed reference policy π'. For compactness of writing, we will often omit the random variable in expectation; e.g., the objective function in (2) will be written as $\mathbb{E}_{d_\pi}\mathbb{E}_\pi[A_{\pi'}]$ for the remainder of paper.

By the performance difference lemma below, it is easy to see that solving (2) is equivalent to solving (1).

Lemma 1. *(Kakade and Langford, 2002) Let π and π' be two policies and $A_{\pi'}(s,a) = Q_{\pi'}(s,a) - V_{\pi'}(s)$ be the (dis)advantage function with respect to running π'. Then it holds that*

$$J(\pi) = J(\pi') + \frac{1}{1-\gamma}\mathbb{E}_{d_\pi}\mathbb{E}_\pi[A_{\pi'}]. \quad (3)$$

[1] LOKI can be easily adapted to finite-horizon problems.

3 FIRST-ORDER RL AND IL

We formulate both first-order RL and IL methods within a single mirror descent framework (Nemirovski et al., 2009), which includes common update rules (Sutton et al., 2000; Kakade, 2002; Peters and Schaal, 2008; Peters et al., 2010; Rawlik et al., 2012; Silver et al., 2014; Schulman et al., 2015b; Ross et al., 2011; Sun et al., 2017). We show that policy updates based on RL and IL mainly differ in first-order stochastic oracles used, as summarized in Table 1.

3.1 MIRROR DESCENT

We begin by defining the iterative rule to update policies. We assume that the learner's policy π is parametrized by some $\theta \in \Theta$, where Θ is a closed and convex set, and that the learner has access to a family of strictly convex functions $\mathcal{R}$.

To update the policy, in the nth iteration, the learner receives a vector g_n from a first-order oracle, picks $R_n \in \mathcal{R}$, and then performs a mirror descent step:

$$\theta_{n+1} = P_{n,g_n}(\theta_n) \quad (4)$$

where P_{n,g_n} is a prox-map defined as

$$P_{n,g_n}(\theta_n) := \arg\min_{\theta\in\Theta} \langle g_n, \theta\rangle + \frac{1}{\eta_n} D_{R_n}(\theta\|\theta_n). \quad (5)$$

η_n is the step size, and D_{R_n} is the Bregman divergence associated with R_n (Bregman, 1967): $D_{R_n}(\theta\|\theta_n) := R_n(\theta) - R_n(\theta_n) - \langle\nabla R_n(\theta_n), \theta - \theta_n\rangle$.

By choosing proper R_n, the mirror descent framework in (4) covers most RL and IL algorithms. Common choices of R_n include negative entropy (Peters et al., 2010; Rawlik et al., 2012), $\frac{1}{2}\|\theta\|_2^2$ (Sutton et al., 2000; Silver et al., 2014), and $\frac{1}{2}\theta^\top F(\theta_n)\theta$ with $F(\theta_n)$ as the Fisher information matrix (Kakade, 2002; Peters and Schaal, 2008; Schulman et al., 2015a).

3.2 FIRST-ORDER ORACLES

While both first-order RL and IL methods can be viewed as performing mirror descent, they differ in the choice of the first-order oracle that returns the update direction g_n. Here we show the vector g_n of both approaches can be derived as a stochastic approximation of the (partial) derivative of $\mathbb{E}_{d_\pi}\mathbb{E}_\pi[A_{\pi'}]$ with respect to policy π, but with a different reference policy π'.

Table 1: Comparison of First-Order Oracles

Method	First-Order Oracle
POLICY GRADIENT (Section 3.2.1)	$\mathbb{E}_{d_{\pi_n}} (\nabla_\theta \mathbb{E}_\pi) [A_{\pi_n}]$
DAGGERED (Section 3.2.2)	$\mathbb{E}_{d_{\pi_n}} (\nabla_\theta \mathbb{E}_\pi) [\mathbb{E}_{\pi^*}[d]]$
AGGREVATED (Section 3.2.2)	$\mathbb{E}_{d_{\pi_n}} (\nabla_\theta \mathbb{E}_\pi) [A_{\pi^*}]$
SLOLS (Section 6)	$\mathbb{E}_{d_{\pi_n}} (\nabla_\theta \mathbb{E}_\pi) [(1-\lambda)A_{\pi_n} + \lambda A_{\pi^*}]$
THOR (Section 6)	$\mathbb{E}_{d_{\pi_n}} (\nabla_\theta \mathbb{E}_\pi) [A^{H,\pi^*}_{\pi_n,t}]$

3.2.1 Policy Gradients

A standard approach to RL is to treat (1) as a stochastic nonconvex optimization problem. In this case, g_n in mirror descent (4) is an estimate of the policy gradient $\nabla_\theta J(\pi)$ (Williams, 1992; Sutton et al., 2000).

To compute the policy gradient in the nth iteration, we set the current policy π_n as the reference policy in (3) (i.e. $\pi' = \pi_n$), which is treated as constant in θ in the following policy gradient computation. Because $\mathbb{E}_{\pi_n}[A_{\pi_n}] = \mathbb{E}_{\pi_n}[Q_{\pi_n}] - V_{\pi_n} = 0$, using (3), the policy gradient can be written as[2]

$$\begin{aligned}
&(1-\gamma)\nabla_\theta J(\pi)|_{\pi=\pi_n} \\
&= \nabla_\theta \mathbb{E}_{d_\pi} \mathbb{E}_\pi [A_{\pi_n}]|_{\pi=\pi_n} \\
&= (\nabla_\theta \mathbb{E}_{d_\pi})[0] + \mathbb{E}_{d_\pi} (\nabla_\theta \mathbb{E}_\pi)[A_{\pi_n}]|_{\pi=\pi_n} \\
&= \mathbb{E}_{d_\pi} (\nabla_\theta \mathbb{E}_\pi)[A_{\pi_n}]|_{\pi=\pi_n} \qquad (6)
\end{aligned}$$

The above expression is unique up to a change of baselines: $(\nabla_\theta \mathbb{E}_\pi)[A_{\pi_n}]$ is equivalent to $(\nabla_\theta \mathbb{E}_\pi)[A_{\pi_n} + b]$, because $(\nabla_\theta \mathbb{E}_\pi)[b(s)] = \nabla_\theta b(s) = 0$, where $b : \mathbb{S} \to \mathbb{R}$ is also called a control variate (Greensmith et al., 2004).

The exact formulation of $(\nabla_\theta \mathbb{E}_\pi)[A_{\pi_n}]$ depends on whether the policy π is stochastic or deterministic. For stochastic policies,[3] we can compute it with the likelihood-ratio method and write

$$(\nabla_\theta \mathbb{E}_\pi)[A_{\pi_n}] = \mathbb{E}_\pi[A_{\pi_n} \nabla_\theta \log \pi] \qquad (7)$$

For deterministic policies, we replace the expectation as evaluation (as it is the expectation over a Dirac delta function, i.e. $a = \pi(s)$) and use the chain rule:

$$(\nabla_\theta \mathbb{E}_\pi)[A_{\pi_n}] = \nabla_\theta A_{\pi_n}(s, \pi) = \nabla_\theta \pi \nabla_a A_{\pi_n} \qquad (8)$$

Substituting (7) or (8) back into (6), we get the equation for stochastic policy gradient (Sutton et al., 2000) or deterministic policy gradient (Silver et al., 2014). Note that the above equations require the exact knowledge, or an unbiased estimate, of A_π. In practice, these terms are further approximated using function approximators, leading to biased gradient estimators (Konda and Tsitsiklis, 2000; Schulman et al., 2015b; Mnih et al., 2016).

[2]We assume the cost is sufficiently regular so that the order of differentiation and expectation can exchange.

[3]A similar equation holds for reparametrization (Grathwohl et al., 2017).

3.2.2 Imitation Gradients

An alternate strategy to RL is IL. In particular, we consider *online* IL, which interleaves data collection and policy updates to overcome the covariate shift problem of traditional batch IL (Ross et al., 2011). Online IL assumes that a (possibly suboptimal) expert policy π^* is available as a black-box oracle, from which demonstrations $a^* \sim \pi^*_s$ can be queried for any given state $s \in \mathbb{S}$. Due to this requirement, the expert policy in online IL is often an *algorithm* (rather than a human demonstrator), which is hard-coded or based on additional computational resources, such as trajectory optimization (Pan et al., 2017). The goal of IL is to learn a policy that can perform similar to, or better than, the expert policy.

Rather than solving the stochastic nonconvex optimization directly, online IL solves an online learning problem with per-round cost in the nth iteration defined as

$$l_n(\pi) = \mathbb{E}_{d_{\pi_n}} \mathbb{E}_\pi[\tilde{c}] \qquad (9)$$

where $\tilde{c} : \mathbb{S} \times \mathbb{A} \to \mathbb{R}$ is a surrogate loss satisfying the following condition: For all $s \in \mathbb{S}$ and $\pi \in \Pi$, there exists a constant $C_{\pi^*} > 0$ such that

$$C_{\pi^*} \mathbb{E}_\pi[\tilde{c}] \geq \mathbb{E}_\pi[A_{\pi^*}]. \qquad (10)$$

By Lemma 1, this implies $J(\pi_n) \leq J(\pi^*) + \frac{C_{\pi^*}}{1-\gamma} l_n(\pi_n)$. Namely, in the nth iteration, online IL attempts to minimize an online upper-bound of $J(\pi_n)$.

DAGGER (Ross et al., 2011) chooses $\tilde{c}$ to be a strongly convex function $\tilde{c}(s,a) = \mathbb{E}_{a^* \sim \pi^*_s}[d(a, a^*)]$ that penalizes the difference between the learner's policy and the expert's policy, where d is some metric of space $\mathbb{A}$ (e.g., for a continuous action space Pan et al. (2017) choose $d(a, a^*) = \|a - a^*\|^2$). More directly, AGGREVATTE simply chooses $\tilde{c}(s,a) = A_{\pi^*}(s,a)$ (Ross and Bagnell, 2014); in this case, the policy learned with online IL can potentially outperform the expert policy.

First-order online IL methods operate by updating policies with mirror descent (4) with g_n as an estimate of

$$\nabla_\theta l_n(\pi_n) = \mathbb{E}_{d_{\pi_n}} \left(\nabla_\theta \mathbb{E}_\pi\right)[\tilde{c}]|_{\pi=\pi_n} \qquad (11)$$

Similar to policy gradients, the implementation of (11) can be executed using either (7) or (8) (and with a control variate). One particular case of (11), with $\tilde{c} = A_{\pi^*}$, is known as AGGREVATED (Sun et al., 2017),

$$\nabla_\theta l_n(\pi_n) = \mathbb{E}_{d_{\pi_n}} \left(\nabla_\theta \mathbb{E}_\pi\right)[A_{\pi^*}]|_{\pi=\pi_n}. \qquad (12)$$

Similarly, we can turn DAGGER into a first-order method, which we call DAGGERED, by using g_n as an estimate of the first-order oracle

$$\nabla_\theta l_n(\pi_n) = \mathbb{E}_{d_{\pi_n}} \left(\nabla_\theta \mathbb{E}_\pi\right)\mathbb{E}_{\pi^*}[d]. \qquad (13)$$

A comparison is summarized in Table 1.

4 THEORETICAL COMPARISON

With the first-order oracles defined, we now compare the performance and properties of performing mirror descent with policy gradient or imitation gradient. We will see that while both approaches share the same update rule in (4), the generated policies have different behaviors: using policy gradient generates a *monotonically* improving policy sequence, whereas using imitation gradient generates a policy sequence that improves *on average*. Although the techniques used in this section are not completely new in the optimization literature, we specialize the results to compare performance and to motivate LOKI in the next section. The proofs of this section are included in Appendix B for completeness.

4.1 POLICY GRADIENTS

We analyze the performance of policy gradients with standard techniques from nonconvex analysis.

Proposition 1. *Let J be β-smooth and R_n be α_n-strongly convex with respect to norm $\|\cdot\|$. Assume $\mathbb{E}[g_n] = \nabla_\theta J(\pi_n)$. For $\eta_n \leq \frac{2\alpha_n}{\beta}$, it satisfies*

$$\mathbb{E}\left[J(\pi_{n+1})\right] \leq J(\pi_0) + \mathbb{E}\left[\sum_{n=1}^N \frac{2\eta_n}{\alpha_n}\|\nabla_\theta J(\pi_n) - g_n\|_*^2\right] + \frac{1}{2}\mathbb{E}\left[\sum_{n=1}^N \left(-\alpha_n\eta_n + \frac{\beta\eta_n^2}{2}\right)\|\hat{\nabla}_\theta J(\pi_n)\|^2\right]$$

where the expectation is due to randomness of sampling g_n, and $\hat{\nabla}_\theta J(\pi_n) := \frac{1}{\eta_n}\left(\theta_n - P_{n,\nabla_\theta J(\pi_n)}(\theta_n)\right)$ is a gradient surrogate.

Proposition 1 shows that monotonic improvement can be made under proper smoothness assumptions if the step size is small and noise is comparably small with the gradient size. However, the final policy's performance is sensitive to the initial condition $J(\pi_0)$, which can be poor for a randomly initialized policy.

Proposition 1 also suggests that the size of the gradient $\|\hat{\nabla}_\theta J(\pi_n)\|^2$ does not converge to zero on average. Instead, it converges to a size proportional to the sampling noise of policy gradient estimates due to the linear dependency of $\frac{2\eta_n}{\alpha_n}\|\nabla_\theta J(\pi_n) - g_n\|_*^2$ on η_n. This phenomenon is also mentioned by Ghadimi et al. (2016). We note that this pessimistic result is because the prox-map (5) is nonlinear in g_n for general R_n and Θ. However, when R_n is quadratic and Θ is unconstrained, the convergence of $\|\hat{\nabla}_\theta J(\pi_n)\|^2$ to zero on average can be guaranteed (see Appendix B.1 for a discussion).

4.2 IMITATION GRADIENTS

While applying mirror descent with a policy gradient can generate a monotonically improving policy sequence, applying the same algorithm with an imitation gradient yields a different behavior. The result is summarized below, which is a restatement of (Ross and Bagnell, 2014, Theorem 2.1), but is specialized for mirror descent.

Proposition 2. *Assume l_n is σ-strongly convex with respect to R_n.*[4] *Assume $\mathbb{E}[g_n] = \nabla_\theta l_n(\pi_n)$ and $\|g_n\|_* \leq G < \infty$ almost surely. For $\eta_n = \frac{1}{\hat{\sigma}n}$ with $\hat{\sigma} \leq \sigma$, it holds*

$$\frac{1}{N}\mathbb{E}\left[\sum_{n=1}^N J(\pi_n)\right] \leq J(\pi^*) + \frac{C_{\pi^*}}{1-\gamma}\left(\epsilon_{class} + \epsilon_{regret}\right)$$

where the expectation is due to randomness of sampling g_n, $\epsilon_{class} = \sup_{\{\pi_n\}} \inf_{\pi\in\Pi} \frac{1}{N}\sum_{n=1}^N l_n(\pi)$ and $\epsilon_{regret} = \frac{G^2(\log N + 1)}{2\hat{\sigma}N}$.

Proposition 2 is based on the assumption that l_n is strongly convex, which can be verified for certain problems (Cheng and Boots, 2018). Consequently, Proposition 2 shows that the performance of the policy sequence on average can converge close to the expert's performance $J(\pi^*)$, with additional error that is proportional to ϵ_{class} and ϵ_{regret}.

ϵ_{regret} is an upper bound of the average regret, which is less than $\tilde{O}(\frac{1}{N})$ for a *large* enough step size.[5] This characteristic is in contrast to policy gradient, which requires *small* enough step sizes to guarantee local improvement.

ϵ_{class} measures the expressiveness of the policy class Π. It can be *negative* if there is a policy in Π that outper-

[4]A function f is said to be σ-strongly convex with respect to R on a set $\mathcal{K}$ if for all $x, y \in \mathcal{K}$, $f(x) \geq f(y) + \langle\nabla f(y), x - y\rangle + \sigma D_R(x\|y)$.

[5]The step size should be large enough to guarantee $\tilde{O}(\frac{1}{N})$ convergence, where $\tilde{O}$ denotes Big-O but omitting log dependency. However, it should be bounded since $\epsilon_{\text{regret}} = \Theta\left(\frac{1}{\hat{\sigma}}\right)$.

Algorithm 1 LOKI

```
Parameters: $d$, $N_m$, $N_M$
Input: $\pi^*$
1: Sample $K$ with probability in (15).
2: for $t = 1 \dots K$ do    # Imitation Phase
3:     Collect data $\mathcal{D}_n$ by executing $\pi_n$
4:     Query $g_n$ from (11) using $\pi^*$
5:     Update $\pi_n$ by mirror descent (5) with $g_n$
6:     Update advantage function estimate $\hat{A}_{\pi_n}$ by $\mathcal{D}_n$
7: end for
8: for $t = K + 1 \dots$ do    # Reinforcement Phase
9:     Collect data $\mathcal{D}_n$ by executing $\pi_n$.
10:    Query $g_n$ from (6) f using $\hat{A}_{\pi_n}$
11:    Update $\pi_n$ by mirror descent (5) with $g_n$
12:    Update advantage function estimate $\hat{A}_{\pi_n}$ by $\mathcal{D}_n$
13: end for
```

forms the expert policy π^* in terms of $\tilde{c}$. However, since online IL attempts to minimize an online upper bound of the accumulated cost through a surrogate loss $\tilde{c}$, the policy learned with imitation gradients in general cannot be better than performing one-step policy improvement from the expert policy (Ross and Bagnell, 2014; Cheng and Boots, 2018). Therefore, when the expert is suboptimal, the reduction from nonconvex optimization to online convex optimization can lead to suboptimal policies.

Finally, we note that updating policies with imitation gradients does not necessarily generate a monotonically improving policy sequence, even for deterministic problems; whether the policy improves monotonically is completely problem dependent (Cheng and Boots, 2018). Without going into details, we can see this by comparing policy gradient in (6) and the special case of imitation gradient in (12). By Lemma 3, we see that

$$\begin{aligned}&\mathbb{E}_{d_{\pi_n}}(\nabla_\theta \mathbb{E}_\pi)[A_{\pi_n}] \\ &= (\nabla_\theta \mathbb{E}_{d_\pi})\mathbb{E}_{\pi_n}[A_{\pi^*}] + \mathbb{E}_{d_{\pi_n}}(\nabla_\theta \mathbb{E}_\pi)[A_{\pi^*}].\end{aligned}$$

Therefore, even with $\tilde{c} = A_{\pi^*}$, the negative of the direction in (12) is not necessarily a descent direction; namely applying (12) to update the policy is not guaranteed to improve the policy performance locally.

5 IMITATE-THEN-REINFORCE

To combine the benefits from RL and IL, we propose a simple randomized algorithm LOKI: first perform K steps of mirror descent with imitation gradient and then switch to policy gradient for the rest of the steps. Despite the algorithm's simplicity, we show that, when K is appropriately randomized, running LOKI has similar performance to performing policy gradient steps directly from the expert policy.

5.1 ALGORITHM: LOKI

The algorithm LOKI is summarized in Algorithm 1. The algorithm is composed of two phases: an imitation phase and a reinforcement phase. In addition to learning rates, LOKI receives three hyperparameters (d, N_m, N_M) which determine the probability of random switching at time K. As shown in the next section, these three hyperparameters can be selected fairly simply.

Imitation Phase Before learning, LOKI first randomly samples a number $K \in [N_m, N_M]$ according to the prescribed probability distribution (15). Then it performs K steps of mirror descent with imitation gradient. In our implementation, we set the per-round loss as[6]

$$l_n(\pi) = \mathbb{E}_{d_{\pi_n}}[\mathrm{KL}(\pi^* || \pi)], \tag{14}$$

which is the KL-divergence between the two policies. It can be easily shown that a proper constant C^* exists satisfying the requirement in (10) (Gibbs and Su, 2002). While using (14) does not guarantee learning a policy that outperforms the expert due to $\epsilon_{\text{class}} \geq 0$, with another reinforcement phase available, the imitation phase of LOKI is only designed to quickly bring the initial policy closer to the expert policy. Compared with choosing $\tilde{c} = A_{\pi^*}$ as in AGGREVATED, one benefit of choosing $\mathrm{KL}(\pi^* || \pi)$ (or its variants, e.g. $\mathbb{E}_{a\sim\pi}\mathbb{E}_{a^*\sim\pi^*}[\|a - a^*\|^2]$) is that it does not require learning a value function estimator. In addition, the imitation gradient can be calculated through *reparametrization* instead of a likelihood-ratio (Tucker et al., 2017), as now l_n is presented as a differentiable function. Consequently, the sampling variance of imitation gradient can be significantly reduced, for example, by using multiple samples of $a \sim \pi_n$ (with a single query from the expert policy) and then performing averaging.

Reinforcement Phase After the imitation phase, LOKI switches to the reinforcement phase. At this point, the policy π_K is much closer to the expert policy than the initial policy π_0. In addition, an estimate of A_{π_K} is also available. Because the learner's policies were applied to collect data in the previous *online* imitation phase, A_{π_n} can already be updated accordingly, for example, by minimizing TD error. Compared with other warm-start techniques, LOKI can learn *both* the policy and the advantage estimator in the imitation phase.

5.2 ANALYSIS

We now present the theoretical properties of LOKI. The analysis is composed of two steps. First, we show the

[6]While (14) does not conform with the classical choice $\mathbb{E}_\pi[\tilde{c}]$ in (9), a bound similar to (10) can be derived.

performance of $J(\pi_K)$ in Theorem 1, a generalization of Proposition 2 to consider the effects of non-uniform random sampling. Next, combining Theorem 1 and Proposition 1, we show the performance of LOKI in Theorem 2. The proofs are given in Appendix C.

Theorem 1. *Let $d \geq 0$, $N_m \geq 1$, and $N_M \geq 2N_m$. Let $K \in [N_m, N_M]$ be a discrete random variable such that*

$$P(K = n) = n^d / \textstyle\sum_{m=N_m}^{N_M} m^d. \tag{15}$$

Suppose l_n is σ-strongly convex with respect to R_n, $\mathbb{E}[g_n] = \nabla_\theta l_n(\pi_n)$, and $\|g_n\|^* \leq G < \infty$ almost surely. Let $\{\pi_n\}$ be generated by running mirror descent with step size $\eta_n = n^d / \hat{\sigma} \sum_{m=1}^n m^d$. For $\hat{\sigma} \leq \sigma$, it holds that

$$\mathbb{E}\left[J(\pi_K)\right] \leq J(\pi^*) + \Delta,$$

where the expectation is due to sampling K and g_n, $\Delta = \frac{C_{\pi^*}}{1-\gamma}\left(\epsilon^w_{class} + 2^{-d}\hat{\sigma} D_{\mathcal{R}} + G^2 C_{N_M}/\hat{\sigma} N_M\right)$, $D_{\mathcal{R}} = \sup_{R\in\mathcal{R}} \sup_{\pi,\pi'\in\Pi} D_R(\pi'||\pi)$, $\epsilon^w_{class} := \sup_{\{w_n\},\{\pi_n\}} \inf_{\pi\in\Pi} \frac{\sum_{n=1}^N w_n l_n(\pi)}{\sum_{n=1}^N w_n}$, and

$$C_{N_M} = \begin{cases} \log(N_M) + 1, & \text{if } d = 0 \\ \frac{8d}{3}\exp\left(\frac{d}{N_M}\right), & \text{if } d \geq 1 \end{cases}$$

Suppose $N_M \gg d$. Theorem 1 says that the performance of $J(\pi_K)$ in expectation converges to $J(\pi^*)$ in a rate of $\tilde{O}(d/N_M)$ when a proper step size is selected. In addition to the convergence rate, we notice that the performance gap between $J(\pi^*)$ and $J(\pi_K)$ is bounded by $O(\epsilon^w_{\text{class}} + 2^{-d} D_{\mathcal{R}})$. $\epsilon^w_{\text{class}}$ is a weighted version of the expressiveness measure of policy class Π in Proposition 2, which can be made small if Π is rich enough with respect to the *suboptimal* expert policy. $D_{\mathcal{R}}$ measures the size of the decision space with respect to the class of regularization functions $\mathcal{R}$ that the learner uses in mirror descent. The dependency on $D_{\mathcal{R}}$ is because Theorem 1 performs a suffix random sampling with $N_m > 0$. While the presence of $D_{\mathcal{R}}$ increases the gap, its influence can easily made small with a slightly large d due to the factor 2^{-d}.

In summary, due to the sublinear convergence rate of IL, N_M does not need to be large (say less than 100) as long as $N_M \gg d$; on the other hand, due to the 2^d factor, d is also small (say less than 5) as long as it is large enough to cancel out the effects of $D_{\mathcal{R}}$. Finally, we note that, like Proposition 2, Theorem 1 encourages using larger step sizes, which can further boost the convergence of the policy in the imitation phase of LOKI.

Given Proposition 1 and Theorem 1, now it is fairly easy to understand the performance of LOKI.

Theorem 2. *Running LOKI holds that*

$$\begin{aligned} &\mathbb{E}\left[J(\pi_N)\right] \leq J(\pi^*) + \Delta \\ &+ \mathbb{E}\left[\textstyle\sum_{n=K+1}^N \frac{2\eta_n}{\alpha_n} \|\nabla_\theta J(\pi_n) - g_n\|_*^2\right] \\ &+ \tfrac{1}{2}\mathbb{E}\left[\textstyle\sum_{n=K+1}^N \left(-\alpha_n \eta_n + \frac{\beta\eta_n^2}{2}\right) \|\hat{\nabla}_\theta J(\pi_n)\|^2\right], \end{aligned}$$

where the expectation is due to sampling g_n and K.

Firstly, Theorem 2 shows that π_N can perform better than the expect policy π^*, and, in fact, it converges to a locally optimal policy on average under the same assumption as in Proposition 1. Compare with to running policy gradient steps directly from the expert policy, running LOKI introduces an additional gap $O(\Delta + K\|\hat{\nabla}_\theta J(\pi)\|^2)$. However, as discussed previously, Δ and $K \leq N_M \ll N$ are reasonably small, for usual N in RL. Therefore, performing LOKI almost has the same effect as using the expert policy as the initial condition, which is the best we can hope for when having access to an expert policy.

We can also compare LOKI with performing usual policy gradient updates from a randomly initialized policy. The performance difference can be easily shown as $O(J(\pi^*) - J(\pi_0) + \Delta + K\|\hat{\nabla}_\theta J(\pi)\|^2)$. Therefore, if performing K steps of policy gradient from π_0 gives a policy with performance worse than $J(\pi^*) + \Delta$, then LOKI is favorable.

6 RELATED WORK

We compare LOKI with some recent attempts to incorporate the loss information c of RL into IL so that it can learn a policy that outperforms the expert policy. As discussed in Section 4, when $\tilde{c} = A_{\pi^*}$, AGGREVATE(D) can potentially learn a policy that is better than the expert policy (Ross and Bagnell, 2014; Sun et al., 2017). However, implementing AGGREVATE(D) exactly as suggested by theory can be difficult and inefficient in practice. On the one hand, while A_{π^*} can be learned off-policy using samples collected by running the expert policy, usually the estimator quality is unsatisfactory due to covariate shift. On the other hand, if A_{π^*} is learned on-policy, it requires restarting the system from any state, or requires performing $\frac{1}{1-\gamma}$-times more iterations to achieve the same convergence rate as other choices of $\tilde{c}$ such as KL$(\pi^*||\pi)$ in LOKI; both of which are impractical for usual RL problems.

Recently, Sun et al. (2018) proposed THOR (Truncated HORizon policy search) which solves a truncated RL problem with the expert's value function as the terminal loss to alleviate the strong dependency of AGGREVATED on the quality of A_{π^*}. Their algorithm uses an H-step truncated advantage function defined as $A_{\pi_n,t}^{H,\pi^*} = \mathbb{E}_{\rho_{\pi_n}}[\sum_{\tau=t}^{t+H-1} \gamma^{\tau-t} c(s_\tau, a_\tau) + \gamma^H V_{\pi^*}(s_{t+H}) - V_{\pi^*}(s_t)]$. While empirically the authors show that the learned policy can improve over the expert policy, the theoretical properties of THOR remain somewhat unclear.[7] In addition, THOR is more convoluted to implement and relies on multiple advantage function estimators. By contrast, LOKI has clearer theoretical properties, while being straightforward to implement with off-the-shelf learning algorithms.

Finally, we compare LOKI with LOLS (Locally Optimal Learning to Search), proposed by Chang et al. (2015). LOLS is an online IL algorithm which sets $\tilde{c} = Q_{\hat{\pi}_n^\lambda}$, where $\lambda \in [0,1]$ and $\hat{\pi}_n^\lambda$ is a mixed policy that at each time step chooses to run the current policy π_n with probability $1-\lambda$ and the expert policy π^* with probability λ. Like AGGREVATED, LOLS suffers from the impractical requirement of estimating $Q_{\hat{\pi}_n^\lambda}$, which relies on the state resetting assumption.

Here we show that such difficulty can be addressed by using the mirror descent framework with g_n as an estimate of $\nabla_\theta l_n^\lambda(\pi_n)$, where $l_n^\lambda(\pi) := \mathbb{E}_{d_{\pi_n}} \mathbb{E}_\pi[(1-\lambda)A_{\pi_n} + \lambda A_{\pi^*}]$. That is, the first-order oracle is simply a convex combination of policy gradient and AGGREVATED gradient. We call such linear combination SLOLS (simple LOLS) and we show it has the same performance guarantee as LOLS.

Theorem 3. *Under the same assumption in Proposition 2, running* SLOLS *generates a policy sequence, with randomness due to sampling g_n, satisfying*

$$\frac{1}{N}\mathbb{E}\left[\sum_{n=1}^{N} J(\pi_n) - ((1-\lambda)J_{\pi_n}^* + \lambda J(\pi^*))\right] \leq \frac{\epsilon_{class}^\lambda + \epsilon_{regret}^\lambda}{1-\gamma}$$

where $(1-\gamma)J_{\pi_n}^* = \min_{\pi\in\Pi} \mathbb{E}_{d_{\pi_n}} \mathbb{E}_\pi[Q_{\pi_n}] =: \mathbb{E}_{d_{\pi_n}}[V_{\pi_n}^*]$ *and* $\epsilon_{class}^\lambda = \min_{\pi\in\Pi} \frac{1}{N}(\sum_{n=1}^N \mathbb{E}_{d_{\pi_n}} \mathbb{E}_\pi[(1-\lambda)Q_{\pi_n} + \lambda Q_{\pi^*}]) - \frac{1}{N}(\sum_{n=1}^N \mathbb{E}_{d_{\pi_n}}[(1-\lambda)V_{\pi_n}^* + \lambda V_{\pi^*}])$.

In fact, the performance in Theorem 3 is actually a lower bound of Theorem 3 in (Chang et al., 2015).[8] Theorem 3 says that on average π_n has performance between the expert policy $J(\pi^*)$ and the intermediate cost $J_{\pi_n}^*$, as long as $\epsilon_{\text{class}}^\lambda$ is small (i.e., there exists a single policy in Π that is better than the expert policy or the local improvement from any policy in Π). However, due to the presence of $\epsilon_{\text{class}}^\lambda$, despite $J_{\pi_n}^* \leq J(\pi_n)$, it is not guaranteed that $J_{\pi_n}^* \leq J(\pi^*)$. As in Chang et al. (2015), either LOLS or SLOLS can necessarily perform on average better than the expert policy π^*. Finally, we note that recently both Nair et al. (2017) and Rajeswaran et al. (2017) propose a scheme similar to SLOLS, but with the AGGREVATE(D) gradient computed using offline batch data collected by the expert policy. However, there is no theoretical analysis of this algorithm's performance.

[7]The algorithm actually implemented by Sun et al. (2018) does not solve precisely the same problem analyzed in theory.

[8]The main difference is due to technicalities. In Chang et al. (2015), $\epsilon_{\text{class}}^\lambda$ is compared with a time-varying policy.

7 EXPERIMENTS

We evaluate LOKI on several robotic control tasks from OpenAI Gym (Brockman et al., 2016) with the DART physics engine (Lee et al., 2018)[9] and compare it with several baselines: TRPO (Schulman et al., 2015a), TRPO from expert, DAGGERED (the first-order version of DAGGER (Ross et al., 2011) in (13)), SLOLS (Section 6), and THOR (Sun et al., 2018).

7.1 TASKS

We consider the following tasks. In all tasks, the discount factor of the RL problem is set to $\gamma = 0.99$. The details of each task are specified in Table A in Appendix A.

Inverted Pendulum This is a classic control problem, and its goal is to swing up an pendulum and to keep it balanced in a upright posture. The difficulty of this task is that the pendulum cannot be swung up directly due to a torque limit.

Locomotion The goal of these tasks (Hopper, 2D Walker, and 3D Walker) is to control a walker to move forward as quickly as possible without falling down. In Hopper, the walker is a monoped, which is subjected to significant contact discontinuities, whereas the walkers in the other tasks are bipeds. In 2D Walker, the agent is constrained to a plane to simplify balancing.

Robot Manipulator In the Reacher task, a 5-DOF (degrees-of-freedom) arm is controlled to reach a random target position in 3D space. The reward consists of the negative distance to the target point from the finger tip plus a control magnitude penalty. The actions correspond to the torques applied to the 5 joints.

7.2 ALGORITHMS

We compare five algorithms (LOKI, TRPO, DAGGERED, THOR, SLOLS) and the idealistic setup of performing policy gradient steps directly from the expert policy (Ideal). To facilitate a fair comparison, all the algorithms are implemented based on a publicly available TRPO implementation (Dhariwal et al., 2017). Furthermore, they share the same parameters except for those

[9]The environments are defined in DartEnv, hosted at https://github.com/DartEnv.

that are unique to each algorithm as listed in Table A in Appendix A. The experimental results averaged across 25 random seeds are reported in Section 7.3.

Policy and Value Networks Feed-forward neural networks are used to construct the policy networks and the value networks in all the tasks (both have two hidden layers and 32 tanh units per layer). We consider Gaussian stochastic policies, i.e. for any state $s \in \mathbb{S}$, $\pi_s(a)$ is Gaussian distributed. The mean of the Gaussian $\pi_s(a)$, as a function of state, is modeled by the policy network, and the covariance matrix of Gaussian is restricted to be diagonal and independent of state. The policy networks and the value function networks are initialized randomly, except for the ideal setup (TRPO from expert), which is initialized as the expert.

Expert Policy The same sub-optimal expert is used by all algorithms (LOKI, DAGGERED, SLOLS, and THOR). It is obtained by running TRPO and stopping it before convergence. The estimate of the expert value function V_{π^*} (required by SLOLS and THOR) is learned by minimizing the sum of squared TD(0) error on a large separately collected set of demonstrations of this expert. The final explained variance for all the tasks is more than 0.97 (see Appendix A).

First-Order Oracles The on-policy advantage A_{π_n} in the first-order oracles for TRPO, SLOLS, and LOKI (in the reinforcement phase) is implemented using an on-policy value function estimator and Generalized Advantage Estimator (GAE) (Schulman et al., 2015b). For DAGGERED and the imitation phase of LOKI, the first-order oracle is calculated using (14). For SLOLS, we use the estimate $A_{\pi^*}(s_t, a_t) \approx c(s_t, a_t) + \gamma \hat{V}_{\pi^*}(s_{t+1}) - \hat{V}_{\pi^*}(s_t)$. And for THOR, $A^{H,\pi^*}_{\pi_n,t}$ of the truncated-horizon problem is approximated by Monte-Carlo samples with an on-policy value function baseline estimated by regressing on these Monte-Carlo samples. Therefore, for *all* methods, an on-policy component is used in constructing the first-order oracle. The exponential weighting in GAE is 0.98; the mixing coefficient λ in SLOLS is 0.5; N_M in LOKI is reported in Table A in Appendix A, and $N_m = \lfloor \frac{1}{2} N_M \rfloor$, and $d = 3$.

Mirror Descent After receiving an update direction g_n from the first-order oracle, a KL-divergence-based trust region is specified. This is equivalent to setting the strictly convex function R_n in mirror descent to $\frac{1}{2}\theta^\top F(\theta_n)\theta$ and choosing a proper learning rate. In our experiments, a larger KL-divergence limit (0.1) is selected for imitation gradient (14) (in DAGGERED and in the imitation phase of LOKI), and a smaller one (0.01) is set for all other algorithms. This decision follows the guideline provided by the theoretical analysis in Section 3.2.2 and is because of the low variance in calculating the gradient of (14). Empirically, we observe using the larger KL-divergence limit with policy gradient led to high variance and instability.

7.3 EXPERIMENTAL RESULTS

We report the performance of these algorithms on various tasks in Figure 1. The performance is measured by the accumulated *rewards*, which are directly provided by OpenAI Gym.

We first establish the performance of two baselines, which represent standard RL (TRPO) and standard IL (DAGGERED). TRPO is able to achieve considerable and almost monotonic improvement from a randomly initialized policy. DAGGERED reaches the performance of the suboptimal policy in a relatively very small number of iterations, e.g. 15 iterations in 2D Walker, in which the suboptimal policy to imitate is TRPO at iteration 100. However, it fails to outperform the suboptimal expert.

Then, we evaluate the proposed algorithm LOKI and Ideal, the performance of which we wish to achieve in theory. LOKI consistently enjoys the best of both TRPO and DAGGERED: it improves as fast as DAGGERED at the beginning, keeps improving, and then finally matches the performance of Ideal after transitioning into the reinforcement phase. Interestingly, the on-policy value function learned, though not used, in the imitation phase helps LOKI transition from imitation phase to reinforcement phase smoothly.

Lastly, we compare LOKI to the two other baselines (SLOLS and THOR) that combine RL and IL. LOKI outperforms these two baselines by a considerably large margin in Hopper, 2D Walker, and 3D Walker; but surprisingly, the performance of SLOLS and THOR are inferior even to TRPO on these tasks. The main reason is that the first-order oracles of both methods is based on an estimated expert value function $\hat{V}_{\pi^*}$. As $\hat{V}_{\pi^*}$ is only regressed on the data collected by running the expert policy, large covariate shift error could happen if the dimension of the state and action spaces are high, or if the uncontrolled system is complex or unstable. For example, in the low-dimensional Pendulum task and the simple Reacher task, the expert value function can generalize better. As a result, in these two cases, LOLS and THOR achieve super-expert performance. However, in more complex tasks, where the effects of covariant shift amplifies exponentially with the dimension of the state space, THOR and SLOLS start to suffer from the inaccuracy of $\hat{V}_{\pi^*}$, as illustrated in the 2D Walker and 3D Walker tasks.

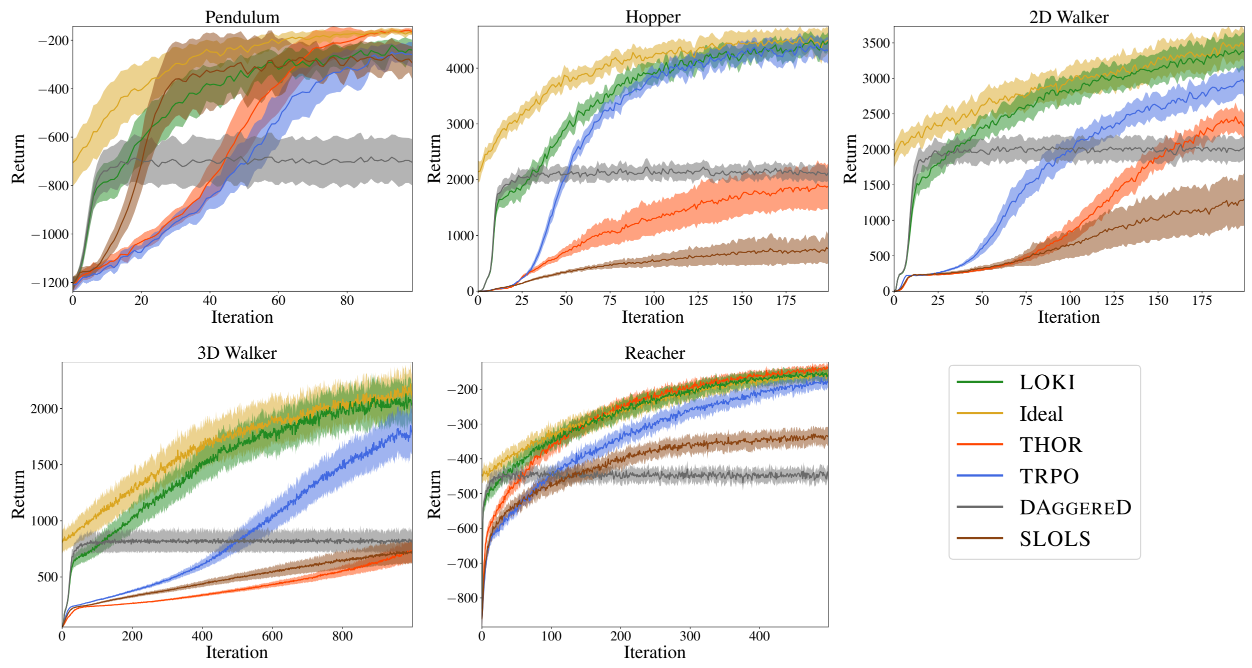

Figure 1: Learning curves. Shaded regions correspond to $\pm\frac{1}{2}$-standard deviation.

8 CONCLUSION

We present a simple, elegant algorithm, LOKI, that combines the best properties of RL and IL. Theoretically, we show that, by randomizing the switching time, LOKI can perform as if running policy gradient steps directly from the expert policy. Empirically, LOKI demonstrates superior performance compared with the expert policy and more complicated algorithms that attempt to combine RL and IL.

Acknowledgements

This work was supported in part by NSF NRI Award 1637758, NSF CAREER Award 1750483, and NSF Graduate Research Fellowship under Grant No. 2015207631.

References

Bregman, L. M. (1967). The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR Computational Mathematics and Mathematical Physics*, 7(3):200–217.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. (2016). Openai gym.

Chang, K.-W., Krishnamurthy, A., Agarwal, A., Daume III, H., and Langford, J. (2015). Learning to search better than your teacher.

Cheng, C.-A. and Boots, B. (2018). Convergence of value aggregation for imitation learning. In *International Conference on Artificial Intelligence and Statistics*, pages 1801–1809.

Dhariwal, P., Hesse, C., Klimov, O., Nichol, A., Plappert, M., Radford, A., Schulman, J., Sidor, S., and Wu, Y. (2017). Openai baselines.

Ghadimi, S., Lan, G., and Zhang, H. (2016). Minibatch stochastic approximation methods for nonconvex stochastic composite optimization. *Mathematical Programming*, 155(1-2):267–305.

Gibbs, A. L. and Su, F. E. (2002). On choosing and bounding probability metrics. *International Statistical Review*, 70(3):419–435.

Grathwohl, W., Choi, D., Wu, Y., Roeder, G., and Duvenaud, D. (2017). Backpropagation through the void: Optimizing control variates for black-box gradient estimation. *arXiv preprint arXiv:1711.00123*.

Greensmith, E., Bartlett, P. L., and Baxter, J. (2004). Variance reduction techniques for gradient estimates in reinforcement learning. *Journal of Machine Learning Research*, 5(Nov):1471–1530.

Juditsky, A., Nemirovski, A., et al. (2011). First order methods for nonsmooth convex large-scale optimization, i: general purpose methods. *Optimization for Machine Learning*, pages 121–148.

Kakade, S. and Langford, J. (2002). Approximately optimal approximate reinforcement learning. In *International Conference on Machine Learning*, volume 2, pages 267–274.

Kakade, S. M. (2002). A natural policy gradient. In *Advances in Neural Information Processing Systems*, pages 1531–1538.

Konda, V. R. and Tsitsiklis, J. N. (2000). Actor-critic algorithms. In *Advances in Neural Information Processing Systems*, pages 1008–1014.

Lee, J., Grey, M. X., Ha, S., Kunz, T., Jain, S., Ye, Y., Srinivasa, S. S., Stilman, M., and Liu, C. K. (2018). DART: Dynamic animation and robotics toolkit. *The Journal of Open Source Software*, 3(22):500.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, pages 1928–1937.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Nair, A., McGrew, B., Andrychowicz, M., Zaremba, W., and Abbeel, P. (2017). Overcoming exploration in reinforcement learning with demonstrations. *arXiv preprint arXiv:1709.10089*.

Nemirovski, A., Juditsky, A., Lan, G., and Shapiro, A. (2009). Robust stochastic approximation approach to stochastic programming. *SIAM Journal on optimization*, 19(4):1574–1609.

Pan, Y., Cheng, C.-A., Saigol, K., Lee, K., Yan, X., Theodorou, E., and Boots, B. (2017). Agile off-road autonomous driving using end-to-end deep imitation learning. *arXiv preprint arXiv:1709.07174*.

Peters, J., Mülling, K., and Altun, Y. (2010). Relative entropy policy search. In *AAAI*, pages 1607–1612. Atlanta.

Peters, J. and Schaal, S. (2008). Natural actor-critic. *Neurocomputing*, 71(7-9):1180–1190.

Pomerleau, D. A. (1989). Alvinn: An autonomous land vehicle in a neural network. In *Advances in Neural Information Processing Systems*, pages 305–313.

Rajeswaran, A., Kumar, V., Gupta, A., Schulman, J., Todorov, E., and Levine, S. (2017). Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087*.

Rawlik, K., Toussaint, M., and Vijayakumar, S. (2012). On stochastic optimal control and reinforcement learning by approximate inference. In *Robotics: Science and Systems*, volume 13, pages 3052–3056.

Ross, S. and Bagnell, J. A. (2014). Reinforcement and imitation learning via interactive no-regret learning. *arXiv preprint arXiv:1406.5979*.

Ross, S., Gordon, G., and Bagnell, D. (2011). A reduction of imitation learning and structured prediction to no-regret online learning. In *International Conference on Artificial Intelligence and Statistics*, pages 627–635.

Schaal, S. (1999). Is imitation learning the route to humanoid robots? *Trends in Cognitive Sciences*, 3(6):233–242.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. (2015a). Trust region policy optimization. In *International Conference on Machine Learning*, pages 1889–1897.

Schulman, J., Moritz, P., Levine, S., Jordan, M., and Abbeel, P. (2015b). High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., et al. (2017). Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*.

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., and Riedmiller, M. (2014). Deterministic policy gradient algorithms. In *International Conference on Machine Learning*.

Sun, W., Bagnell, J. A., and Boots, B. (2018). Truncated horizon policy search: Deep combination of reinforcement and imitation. *International Conference on Learning Representations*.

Sun, W., Venkatraman, A., Gordon, G. J., Boots, B., and Bagnell, J. A. (2017). Deeply aggrevated: Differentiable imitation learning for sequential prediction. *arXiv preprint arXiv:1703.01030*.

Sutton, R. S., McAllester, D. A., Singh, S. P., and Mansour, Y. (2000). Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, pages 1057–1063.

Tucker, G., Mnih, A., Maddison, C. J., Lawson, J., and Sohl-Dickstein, J. (2017). Rebar: Low-variance, unbiased gradient estimates for discrete latent variable models. In *Advances in Neural Information Processing Systems*, pages 2624–2633.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Reinforcement Learning*, pages 5–32. Springer.