INCREMENTAL POLICY GRADIENT ESTIMATION FOR ONLINE CONTROL IN REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Policy gradient methods are built on the policy gradient theorem, which involves a term representing the complete sum of rewards into the future: the return. Due to this, one usually either waits until the end of an episode before performing updates, or learns an estimate of what the return will be–a so-called critic. Our emphasis is on the first approach in this work, detailing an incremental policy gradient update which neither waits until the end of the episode, nor relies on learning estimates of the return. We provide on-policy and off-policy variants of our algorithm. Theoretically, we draw a connection between the traces our methods use and the discounted future state visitation distribution. We conclude with an experimental evaluation of our methods on both simple-to-understand and complex domains.

1 POLICY GRADIENT METHODS

Policy gradient methods form a branch of reinforcement learning (Sutton & Barto, 2018) where behavior is learned directly, in contrast with *value-based* methods which derive behavior from predicted long-term outcomes. One perceived advantage of policy gradient methods is that they extend easily to high-dimensional action spaces, as opposed to value-based methods which tend to require expensive search during the selection of actions. Policy gradient methods are not new (Sutton, 1984; Williams, 1992; Konda & Tsitsiklis, 2000; Kakade & Langford, 2002; Kakade, 2002; Perkins & Pendrith, 2002; Perkins & Precup, 2003; Kober & Peters, 2009; Neumann et al., 2011) but recent interest in this approach has exploded with the widespread use of deep neural network function approximation in reinforcement learning (Lillicrap et al.; Schulman et al., 2015; 2017; Abdolmaleki et al., 2018; Haarnoja et al., 2018; Ahmed et al., 2019; Nachum et al., 2019; Fellows et al., 2019).

The most popular policy gradient methods (Lillicrap et al.; Schulman et al., 2015; 2017; Abdolmaleki et al., 2018; Haarnoja et al., 2018) learn an estimate of the return–the critic, which is used to improve behaviour–the actor; such methods are therefore known as *actor-critic* methods (Sutton & Barto, 2018). The hope is that the critics enhance data-efficiency and reduce the variance of updates.

Yet, the community has devoted relatively little attention to approaches that do not learn a critic. Indeed, one of the first policy gradient methods, REINFORCE, relies on Monte Carlo estimates of the return (Williams, 1992). Although such an approach can suffer from high variance, we note that critics tend to be extremely poor estimates of the return in practice (Ilyas et al., 2020). Indeed, PPO (Schulman et al., 2017) tends to perform better when its critic is learned through a method that is arguably closer to Monte Carlo estimation (i.e., setting $\lambda = 0.995$ in the λ-return). At the same time, however, REINFORCE can be undesirable in environments with long time horizons as updates may only take place at the end of environment interaction. This difficulty also precludes a REINFORCE-type algorithm for *continuing tasks*. Perhaps surprisingly, despite having estimates of the return readily available, little attention has been given to online actor-critic updates (Degris et al., 2012). It is clear that inadequacies exist in the foundations of current policy gradient methods.

We aim to close the gap between policy gradient methods that can operate online and those that are only suitable for episodic tasks with Monte Carlo estimates of the return. In particular, we propose an incremental policy gradient update which does not require a critic, but may still estimate the policy gradient for continuing tasks. The update accumulates feedback over time, and permits updates to the policy at every time step. Our approach is similar in spirit to the backward-view of

TD(λ) (Sutton, 1988). While we **do not** by any means claim that our method produces state-of-the-art results, we hope that it inspires future work in online policy gradient methods.

We highlight the following contributions of our work. **1)** We derive an incremental policy gradient update. **2)** We show that the traces used in our method are connected to the future state visitation distributions. **3)** We empirically evaluate properties of our method on both simple-to-understand and more complex environments, answering a series of precise experimental questions.

2 MDPs AND THE REINFORCE ALGORITHM

The reinforcement learning problem is often formalized with *Markov Decision Processes* (MDPs). An MDP (Puterman, 2014) is a tuple $(\mathcal{S}, \mathcal{A}, p, r, \gamma)$: $\mathcal{S}$ is the state space, $\mathcal{A}$ is the set of available actions, p is the transition probability function, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, and $\gamma \in [0, 1)$ is the discount factor. We will assume that both $\mathcal{S}$ and $\mathcal{A}$ are finite; the general case follows by considering measure-theoretic arguments (for instance, Konda & Tsitsiklis (2000)). We further assume that the reward function is bounded. A (stationary) policy is a mapping $\pi : \mathcal{S} \to \Delta_{\mathcal{A}}$, where $\Delta_{\mathcal{A}}$ is the simplex over $\mathcal{A}$. Usually, we parameterize π with parameters $\theta \in \mathbb{R}^d$ for some $d \in \mathbb{N}$.

The discounted total reward setting is typically considered in episodic tasks, and occasionally considered in continuing tasks. In this setting, given a starting state distribution p_0, the agent maximizes

$$J_\gamma(\pi_\theta) \doteq \mathbb{E}_{p_0}[v_\gamma^{\pi_\theta}(s)], \tag{1}$$

where $v_\gamma^{\pi_\theta}(s) \doteq \mathbb{E}[R_{t+1} + \gamma R_{t+2} + \cdots | S_t = s, \pi_\theta]$ is the *state-value function* representing the expected return (under π_θ) from state s. The policy gradient theorem (Sutton et al., 2000) yields

$$\nabla_\theta J_\gamma(\pi_\theta) = \sum_{s,a} d_\gamma^{\pi_\theta}(s) \pi_\theta(a \mid s) q_\gamma^{\pi_\theta}(s, a) \nabla_\theta \log \pi_\theta(a \mid s), \tag{2}$$

where $d_\gamma^{\pi_\theta}(s) \doteq \sum_{t=0}^{\infty} \gamma^t \Pr(S_t = s \mid p_0, \pi_\theta)$ is the (unnormalized, discounted) *future state visitation distribution*, and $q_\gamma^{\pi_\theta}(s, a) \doteq \mathbb{E}[R_{t+1} + \gamma R_{t+2} + \cdots | S_t = s, A_t = a]$ is the *action-value function*- the expected return conditioned on taking action a in state s, and following π_θ thereafter.

The REINFORCE (Williams, 1992) algorithm sampling one or several trajectories from an environment according to the policy π_θ, after which Monte Carlo estimates are formed of Equation 2 to be used with stochastic gradient ascent. A limitation of REINFORCE lies in having to wait until the agent has completed a (possibly lengthy) trajectory before an update to the policy can be performed.

3 INCREMENTAL POLICY GRADIENT

In this section, we derive our incremental policy gradient update. All proofs are in Appendix A.2.

We consider the *off-policy* setting, where the behavior policy may differ from the one being learned, as it generalizes the on-policy setting. Let μ be a behaviour policy. If $s > t$, $\rho_{s:t} \doteq 1$, otherwise

$$\rho_{s:t} \doteq \prod_{i=s}^{t} \frac{\pi_\theta(A_i \mid S_i)}{\mu(A_i \mid S_i)}.$$

To understand where our algorithm comes from, it will be helpful to write Equation 2 in an equivalent form. For completeness, we provide the full calculation in Appendix A.1.

$$\nabla_\theta J_\gamma(\pi_\theta) = \mathbb{E}_{p_0, \pi_\theta} \left[\sum_{t=0}^{\infty} \gamma^t R_{t+1} \sum_{j=0}^{t} \nabla_\theta \log \pi_\theta(A_j \mid S_j) \right]. \tag{3}$$

Let us first add importance-sampling corrections to Equation 3.

$$\nabla_\theta J_\gamma(\pi_\theta) = \mathbb{E}_{p_0, \mu} \left[\sum_{t=0}^{\infty} \gamma^t \rho_{0:t} R_{t+1} \left(\sum_{j=0}^{t} \nabla_\theta \log \pi_\theta(A_j \mid S_j) \right) \right]. \tag{4}$$

Grouping non-reward terms, this expression lets us define the following *eligibility trace* recursively:

$$\mathbf{e}_0 \doteq \rho_{0:0}\nabla_\theta \log \pi_\theta(A_0 \mid S_0),$$
$$\forall j > 0,\ \mathbf{e}_j \doteq \gamma\rho_{j:j}\mathbf{e}_{j-1} + \rho_{0:j}\gamma^j\nabla_\theta \log \pi_\theta(A_j \mid S_j). \tag{5}$$

Hence, we can rewrite the policy gradient as

$$\nabla_\theta J_\gamma(\pi_\theta) = \mathbb{E}_{p_0,\mu}\left[\sum_{t=0}^{\infty} R_{t+1}\mathbf{e}_t\right] \tag{6}$$

This leads to an algorithm which updates π_θ at each time step t with $R_{t+1}\mathbf{e}_t$. While changing π_θ technically invalidates Equation 6, it is reasonable to suspect that small learning rates result in small deviations from the policy gradient. This compromise may not be ideal, but we note that TD(λ)'s incremental implementation is similarly justified. In Section 5, we characterize the error that such policy updates introduce, and detail our algorithm in Algorithm 1, which we denote the Incremental Policy Gradient (IPG) algorithm.

One can observe that in a tabular setting, for identical trajectories, this algorithm eventually performs the same update that REINFORCE would have done. Such scenarios occur if states are never revisited in a trajectory, or if the policy changes slow enough that the same sequence of actions are sampled. But what does IPG do in cases where it's not equivalent? Because the same update is performed regardless of episode termination, the algorithm can be seen as performing a fixed-horizon policy gradient update to preceding states (as if the return ended there), and adds later terms to the gradient as they become available. That is, if an action leads to an immediate poor outcome, it readily reduces the odds of the state-action pair, but if it eventually leads to a long-term good outcome, the odds are increased accordingly through the eligibility trace. This emphasizes the algorithm's online adaptation, and how deviations from REINFORCE may occur when states are revisited.

Of note, despite the policy updates invalidating Equation 6, each reward is under the current time step's policy. Coupled with a reasonably slowly changing policy, the return for the current state-action pair will at least have correct, on-policy near-term information. with the later potentially off-policy information being attenuated by discounting. This results in multiple policy changes where the highest-weighted rewards of the return (from each state-action pair) are correct. In contrast, with REINFORCE, all of the information across an episode is batched to perform one correct change to the policy. Should one wish to perform more REINFORCE updates with the same information, none of the rewards are sampled under the current policy.

Due to sometimes better empirical performance, many policy gradient methods omit the γ^t term in Equation 3, resulting in a semi-gradient update (Nota & Thomas, 2019). The corresponding semi-gradient IPG update simply drops the γ^j term in the eligibility trace update in Equation 5.

Algorithm 1 Incremental Policy Gradient (IPG)

```
Given: policy $\pi_\theta$ (parameters $\theta$); trace $\mathbf{e}$; behaviour policy $\mu$ (optional; could be equal to $\pi_\theta$);
learning rate $\alpha$.
Draw starting state: $S_0 \sim p_0$.
Initialize trace: $\mathbf{e} \leftarrow 0, \rho \leftarrow 1$.
for $t = 0, 1, \ldots$ do
    Draw action: $A_t \sim \mu(\cdot \mid S_t)$
    Observe transition $(S_t, A_t, S_{t+1}, R_{t+1})$.
    Accumulate the importance-sampling ratio $\rho \leftarrow \rho\frac{\pi_\theta(A_t|S_t)}{\mu(A_t|S_t)}$.
    Update trace: $\mathbf{e} \leftarrow \gamma\frac{\pi_\theta(A_t|S_t)}{\mu(A_t|S_t)}\mathbf{e} + \rho\gamma^t\nabla_\theta \log \pi_\theta(A_t \mid S_t)$.    $\triangleright$ Omit $\gamma^t$ for semi-gradient
    Update policy: $\theta \leftarrow \theta + \alpha R_{t+1}\mathbf{e}$.
end for
```

4 Interpretation of the Traces

In this section, we draw a connection between the eligibility traces proposed above and the stationary distribution under the target policy.

We will need the following regularity assumption on the gradients of π_θ.

Assumption 1. *There is some $C \in \mathbb{R}$ such that for all $(s, a) \in \mathcal{S} \times \mathcal{A}$, $\|\nabla_\theta \log \pi_\theta(a \mid s)\|_\infty \leq C$, where $\|\cdot\|_\infty$ is the infinity norm.*

First, we provide a connection between $d_\gamma^{\pi_\theta}$ and a particular sum of traces.

Proposition 1 (Expected Sum of Traces, Discounted Setting)**.** *Under Assumption 1, the following holds for every state s.*

$$\nabla_\theta d_\gamma^{\pi_\theta}(s) = \gamma \sum_{t=0}^{\infty} \Pr(S_t = s \mid p_0, \pi_\theta)\mathbb{E}[\mathbf{e}_{t-1} \mid S_t = s] \tag{7}$$

That the traces are related to the gradient of $d_\gamma^{\pi_\theta}$ is not as surprising as it may appear. Indeed, since the policy gradient expression in Equation 6 only involves multiplication by a reward term and a summing of the resulting eligibility traces, the gradient of $d_\gamma^{\pi_\theta}$ must somehow be related to the traces. It is also interesting that every term of $\Pr(S_t \mid p_0, \pi_\theta)$ is multiplied by $\mathbb{E}[\mathbf{e}_{t-1} \mid S_t = s]$, rather than by $\mathbb{E}[\mathbf{e}_t \mid S_t = s]$. One intuition is that $\Pr(S_t \mid p_0, \pi_\theta)$ does not take into account the next action A_t, the log probability of which is included in $\mathbf{e}_t$ but not in $\mathbf{e}_{t-1}$.

Unfortunately, Proposition 1 does not provide a simple way to calculate $\nabla_\theta d_\gamma^{\pi_\theta}(s)$. To use Proposition 1 in this way would require sampling $\mathbf{e}_{t-1}$ for every state encountered along a trajectory, having access to $\Pr(S_t = s \mid p_0, \pi_\theta)$, and summing the resulting series.

5 Error Analysis

Here, we analyse the errors that policy changes introduce into the trace-based estimate of the policy gradient. We will assume linear function approximation: in particular, there exist features $\mathbf{x}(s, a) \in \mathbb{R}^d$ for each state-action pair such that policies are Boltzmann distributions over linear preferences:

$$\pi_\theta(a \mid s) \propto \exp(\theta^\top \mathbf{x}(s, a)),$$

for $\theta \in \mathbb{R}^d$. In the following, $\|\cdot\|$ will denote any norm, and $\|\cdot\|_1$ will denote the ℓ_1-norm, and $|\cdot|$ will denote the absolute value function.

Proposition 2. *Assume we have a trajectory $(s_0, a_0, r_1, \cdots, s_t, a_t, r_{t+1})$. Let $\theta, \theta_j \in \mathbb{R}^d$ be arbitrary, for $0 \leq j \leq t$. Then,*

$$\left\|\gamma^t r_{t+1} \sum_{j=0}^{t} \nabla_\theta \log \pi_{\theta_j}(a_j \mid s_j) - \gamma^t r_{t+1} \sum_{j=0}^{t} \nabla_\theta \log \pi_\theta(a_j \mid s_j)\right\| \tag{8}$$

$$\leq \gamma^t |r_{t+1}| \sum_{j=0}^{t} \|\pi(\cdot \mid s_j) - \pi_j(\cdot \mid s_j)\|_1 \max_a \|\mathbf{x}(s_j, a)\|, \tag{9}$$

$$\leq \gamma^t |r_{t+1}|(t+1) \max_j \left(\|\pi(\cdot \mid s_j) - \pi_j(\cdot \mid s_j)\|_1 \max_a \|\mathbf{x}(s_j, a)\|\right) \tag{10}$$

The interpretation of Proposition 2 is as follows. We imagine that the $\{\theta_j\}_{j=0}^t$ are a sequence of parameters obtained through successive policy updates. We set $\theta \doteq \theta_t$ and ask, what would my update at time t have been if I had followed the same trajectory, but had instead started with the policy π_θ and refrained from performing updates? Proposition 2 tells us that the difference between these two updates (Equation 8) decreases with the horizon length ($\gamma^t(t+1)$) and is smaller if the policy probabilities of π and π_j are close to each other ($\|\pi(\cdot \mid s_j) - \pi_j(\cdot \mid s_j)\|_1$). Note that because this result focuses on a particular trajectory, the inequality depends only on the policy probabilities on the actual trajectory, and not on parts of the state space the agent has not yet seen.

A limitation of Proposition 2 is its dependence on the maximum norm of the features, $\max_a \|\mathbf{x}(s, a)\|$. Although one might expect bounded features in some scenarios (e.g., a robotic arm in a factory), it is plausible that in others the features are either unbounded, or the maximum norm is quite large, rendering the bound uninformative. Another limitation is its assumption of linear function approximation; however informative, there is dependence on the policy parameterization given that Proposition 2 involves the gradient of the log probabilities.

6 Related Work

The most relevant work to ours is the AC(λ) algorithm (Degris et al., 2012), an existing incremental policy gradient update with eligibility traces. In contrast with our work, AC(λ) uses on an online TD(λ) critic to both provide complete estimates of the return to bootstrap off of, and be used as a state-value baseline. The algorithm's derivation assumes the use of a state-value baseline, that even the Monte Carlo extreme ($\lambda = 1$) results in a fundamentally different incremental update from this work. We empirically explore AC(λ)'s inclusion of a critic in Section 7. Beyond AC(λ), much work on online, incremental learning algorithms with eligibility traces are in the value-based approaches for reinforcement learning (Sutton, 1988; Seijen & Sutton, 2014), while the aforementioned recent policy gradient methods generally process data from completed episodes in batches.

7 Empirical Evaluation

For our empirical evaluation, we focus first on precise experimental questions in small-scale domains, before understanding the properties of our method in large-scale control environments. We address only the on-policy setting to facilitate a deeper analysis of our algorithms.

7.1 Environments

AdaptChain is a 50-state 1D episodic grid world. An agent starts on the left, receives a reward of -1 for moving left, and 0 for moving right. Moving off the left-most state keeps the agent in place, while moving off the right-most state terminates with a reward of 10.

SwitchStay is a deterministic 2-state continuing MDP where in each state, an agent has the option to stay in its current state, or switch to the other. In one state, staying and switching receives rewards of 1 and -1 respectively, and in the other state, the same actions receive rewards of 2 and 0.

We further use OpenAI Gym's **CartPole-v0** and **LunarLander-v2** environments (Brockman et al., 2016). They are physics-based problems requiring the use of function approximation.

7.2 Trace Accuracy over Time

In this portion, we evaluate the extent to which our gradient expression in Equation 6 for the discounted setting is invalidated as π_θ changes. We are interested in the following questions.

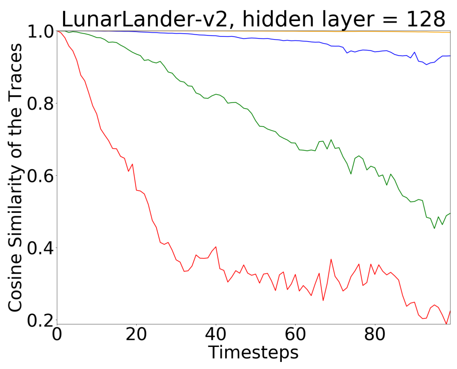
1) For a given policy π_θ at time t with trace $\mathbf{e}_t$, how close is $\mathbf{e}_t$ to what the trace would have been at time t, if our initial policy had been π_θ, if we had obtained the same trajectory, and if we had not updated π_θ at all?

We use the cosine similarity to measure closeness. If the cosine similarity between our current trace $\mathbf{e}_t$ and the counterfactual trace is close to 1, that result would indicate that the policy gradient expression in Equation 6 remained approximately valid, even for a changing π_θ. We note that although we could have added importance sampling to correct the trajectories, doing so would not have changed the values of the cosine similarities.

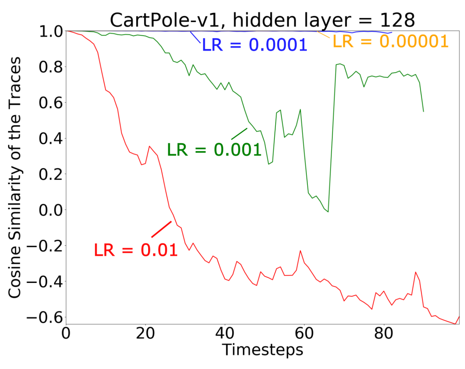
We initialize 100 random policies (i.e., 100 different agents) and apply updates according to Algorithm 1. At every time step t, for each agent, we compute the cosine similarity between the current trace $\mathbf{e}_t$ and $\gamma^t \sum_{j=0}^{t} \nabla_\theta \log \pi_{\theta_0}(a_j \mid s_j)$, where π_{θ_0} is the initial policy of the given agent and $(s_0, a_0, \cdots, s_t, a_t)$ is the current trajectory of that agent.

In Figure 4, we show a representative sample of our cosine similarity plots (additional figures for other hidden layer sizes may be found in Appendix A.3). Before proceeding with the analysis, let us first note that some of the lines cut off before 100 time steps; this cut-off happened because of the inherent episode termination in the environment itself. For example, environment interaction in CartPole ends when the pole falls over or the cart exceeds the horizontal boundaries. As different policies lead to a different number of time steps before termination, when calculating the mean cosine similarity we ignore the contribution of those agents that have terminated.

As expected, the cosine similarity tends to decrease over time as π_θ is updated. The rate of decrease is higher for larger learning rates, consistent with the intuition that large changes in π_θ should in-

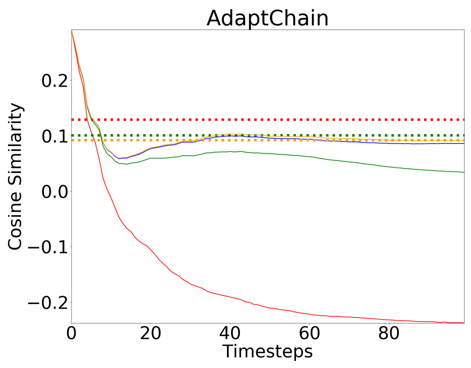


(a)

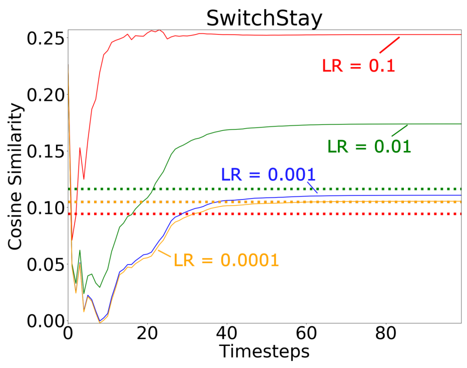


(b)

Figure 1: Each line is the mean over 100 random policies; the standard errors are smaller than the width of the lines. At every time step, the policy is updated as in Algorithm 1. Each value on the plot is the cosine similarity of the trace at that time, compared to what the trace would have been if the initial policy had been what the current policy is now, and if the exact same trajectory had been followed–the so-called counterfactual trace. The counterfactual trace is calculated with Equation 5 using the current policy and the stored trajectory up until that point.



(a)



(b)

Figure 2: Each line is the mean over 100 random policies; the standard errors are smaller than the width of the lines. At every time step, the policy is updated as in Algorithm 1. Each point on a solid line is the cosine similarity of $\sum_{j=0}^{t} R_{t+1}\mathbf{e}_t$ with $\nabla_\theta J_\gamma(\pi_{\theta_t})$. The dotted lines represent the cosine similarity of $\sum_{t=0}^{T} \gamma^t R_{t+1} \sum_{j=0}^{t} \nabla_\theta \pi_{\theta_0}(a_j \mid s_j)$ (REINFORCE) and $\nabla_\theta J_\gamma(\pi_{\theta_t})$.

validate the trace expression to a greater degree. Moreover, for the lower range of learning rates (≤ 0.0001), the cosine similarity seems to remain close to 1 for all 100 time steps.

For the settings where the cosine similarity did decrease, this decrease was not monotonic, and indeed the cosine similarity sometimes increased. This result is consistent with Proposition 2: what is important in assessing the estimation error is not necessarily the number of updates applied, but the actual policy probabilities. It is possible that applying more updates to a policy could move it closer in policy probabilities to the initial policy, although it may be further in terms of parameter space. Lastly, we note that the spike in Figure 1b at 60 steps for the learning rate of 0.001 is an artifact of the fact that at 60 time steps, all runs had already terminated except for one whose cosine similarity consistently remained close to 0.8.

2) For a given policy π_θ at time t with trace $\mathbf{e}_t$, how close is the IPG update to the exact policy gradient? As this question requires calculation of the exact policy gradient, we limit our consideration of this question to tabular environments.

As in the previous experiment, we initialize 100 random policies. This time, for each time step, we measure the cosine similarity between $\nabla_\theta J_\gamma(\pi_{\theta_t})$ and $\sum_{i=0}^{t} R_{i+1}\mathbf{e}_j$. At the end of 100 time steps, we additionally compute what REINFORCE would have used for its end-of-episode update on this trajectory (if the initial policy had not been modified), and compare it with the exact policy gradient.

In Figure 2, we show some representative plots for the respective environments, with additional plots in Appendix A.3. First, cosine similarities tended to be quite poor in general, with the mean cosine similarity being quite far from one. Yet, in some cases the cosine similarity was no worse, and sometimes better, than the cosine similarity of REINFORCE, suggesting that using Equation 6 with a changing policy is not unreasonable. In SwitchStay, the mean consine similarity actually tended to increase over time, which is consistent with our prior observation that the important factor is not necessarily the number of updates, but the difference in poicy probabilities. For REINFORCE, we emphasize that we are *not* computing Monte Carlo estimates of the policy gradient for a single policy; if we were, we would expect that the cosine similarities of REINFORCE would be close to 1. Our focus on the single-sample behaviour of REINFORCE is meant to capture how REINFORCE might perform in practicewhere it is common not to have many trajectories of the same policy available for policy gradient estimation.

Finally, there did not seem to be a consistent relationship between the learning rate and the cosine similarity. Indeed, a higher learning rate seemed detrimental to cosine similarity in AdaptChain, but was beneficial in SwitchStay. This result is strange since for the environment settings, a smaller learning rate implied greater fidelity of the trace to either a true, counterfactual trace or to the policy gradient.

7.3 Control

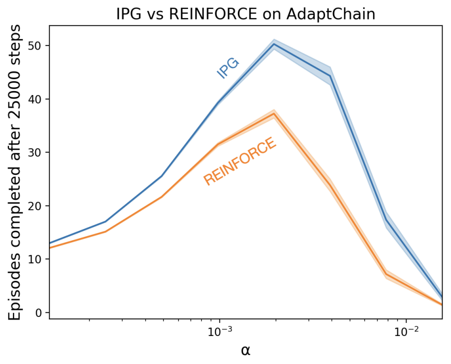
We conclude our experimental evaluation with a comparison between IPG and REINFORCE (Williams, 1992) in two control problems. As typically done in empirical work on policy gradient methods, we consider the semi-gradient variants of each algorithm (Nota & Thomas, 2019).

1) Our first comparison is in the tabular AdaptChain environment. In particular, this environment emphasizes a scenario where IPG will perform a different update from REINFORCE through excessive state revisitations. We expect that through the interim fixed-horizon policy gradient updates, IPG will be able to adapt mid-episode to the negative feedback for moving left, and tend to go right upon revisiting previous states. We swept over various step sizes, and Figure 3a shows the number of episodes completed over 25,000 steps, averaged across 1,000 independent runs.

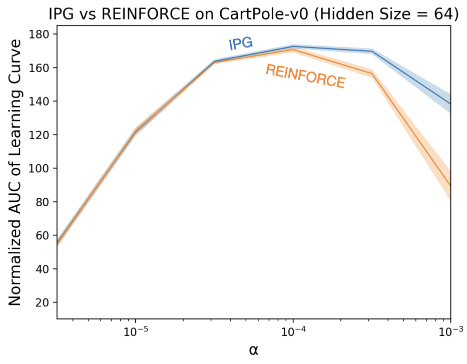
Consistent with our hypotheses, IPG performed very different updates from REINFORCE, and consistently outperformed REINFORCE because of being able to adapt online. IPG seems to support larger learning rates, and we see the curves converge as learning rates decrease, in line with IPG matching REINFORCE should the policy change slowly.

2) Next, we perform an ablation study in CartPole-v0, where the policy is parameterized by a fully-connected neural network with 2 hidden layers. Our emphasis is on whether our observations from the tabular setting remain when function approximation is used. The use of state features can be viewed as exacerbating the excessive state revisitation scenario, as relatively different states may still have features in common. We further compare IPG with the AC(λ) algorithm (Degris et al., 2012) to see the impact of an online-learned state-value baseline. We fix $\lambda = 1$ for AC(λ) so that each algorithm aims to use Monte Carlo estimates of the return. We performed 30 independent runs of 100,000 steps for each parameter setting, and at each step, we recorded the mean return over the last 10 episodes. Figure 3b shows a representative comparison between IPG and REINFORCE in terms of the area under each parameter setting's learning curve (AUC). Figure 3c shows a similar comparison between IPG and AC(1), and Figure 3d shows the learning curves under each algorithm's parameter setting with the largest AUC. Complete results can be found in Appendix A.3.

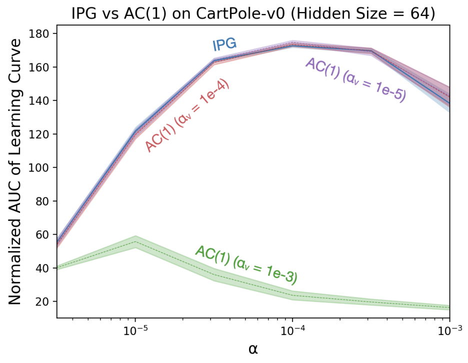
IPG and REINFORCE seemingly follow the same trends observed in AdaptChain. With AC(1), a large value learning rate can greatly hinder learning, highlighting potential issues with an inaccurate critic. With smaller value learning rates, AC(1) approaches but does not outperform IPG. We found that AC(1) with small learning rates still had rather inaccurate values. We suspect that by performing small updates, the value estimates remained small, resulting in an update similar to IPG.



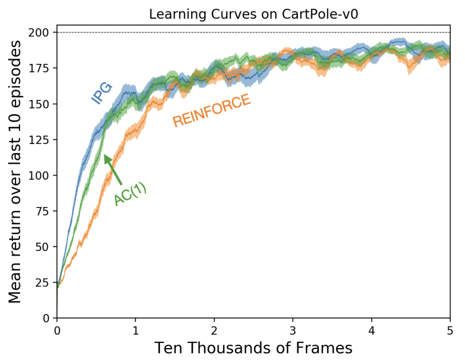
(a) Episodes completed after 25,000 steps in the AdaptChain environment. Results are averaged over 1,000 runs, and shaded regions represent one standard error. Note the log-scale of the x-axis.



(b) AUCs over 100,000 steps in the CartPole-v0 environment. Results are averaged over 30 runs, and shaded regions represent one standard error. Note the log-scale of the x-axis.



(c) AUCs over 100,000 steps in the CartPole-v0 environment. Results are averaged over 30 runs, and shaded regions represent one standard error. Note the log-scale of the x-axis.



(d) Learning curves under each algorithm's best parameters in terms of AUC. Results are averaged over 30 runs, and shaded regions represent one standard error.

8 CONCLUSION

We introduce an incremental policy gradient method that **1)** does not require a critic and **2)** can perform policy updates at each time step. This method is based upon an alternative formulation of the policy gradient, from which we extract eligibility traces that may be updated incrementally. Theoretically, we **1)** draw a connection between the gradient of the discounted future state visitation distribution and a particular sum of these traces and **2)** analyze the policy-gradient estimation error our method introduces through policy updates. Empirically, we **1)** characterize the extent to which this estimation error occurs and **2)** investigate the practical performance of IPG.

There is much promising future work: **1)** It would be interesting to adapt IPG to produce online analogues of state-of-the-art methods that use a critic, such as PPO, to understand further when/if a critic should be used, as our results provide an example where there was no clear benefit in using an online-learned state-value baseline. **2)** Our implementation of IPG through an eligibility trace seems analagous to an online learning procedure, while recent work (Shani et al., 2019) has connected policy gradient and online learning methods. Further work in this direction might unearth new algorithms that can take advantage of recent advances in online learning, such as parameter-free algorithms (Cutkosky & Orabona, 2018). **3)** Our empirical evaluation focused on the on-policy case to get clearer answers regarding properties of IPG. However, it would be fruitful to assess how it fares in the online, off-policy setting.

REFERENCES

Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Remi Munos, Nicolas Heess, and Martin Riedmiller. Maximum a posteriori policy optimisation. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=S1ANxQW0b`.

Zafarali Ahmed, Nicolas Le Roux, Mohammad Norouzi, and Dale Schuurmans. Understanding the impact of entropy on policy optimization. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 151–160, Long Beach, California, USA, 2019.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Ashok Cutkosky and Francesco Orabona. Black-box reductions for parameter-free online learning in banach spaces. *arXiv preprint arXiv:1802.06293*, 2018.

Thomas Degris, Martha White, and Richard S. Sutton. Off-policy actor-critic. *CoRR*, abs/1205.4839, 2012. URL `http://arxiv.org/abs/1205.4839`.

Matthew Fellows, Anuj Mahajan, Tim GJ Rudner, and Shimon Whiteson. Virel: A variational inference framework for reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 7120–7134, 2019.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

Andrew Ilyas, Logan Engstrom, Shibani Santurkar, Dimitris Tsipras, Firdaus Janoos, Larry Rudolph, and Aleksander Madry. A closer look at deep policy gradients. In *International Conference on Learning Representations*, 2020.

Sham Kakade and John Langford. Approximately optimal approximate reinforcement learning. In *ICML*, volume 2, pp. 267–274, 2002.

Sham M. Kakade. A natural policy gradient. In *Advances in neural information processing systems*, pp. 1531–1538, 2002.

Jens Kober and Jan R Peters. Policy search for motor primitives in robotics. In *Advances in neural information processing systems*, pp. 849–856, 2009.

Vijay R Konda and John N Tsitsiklis. Actor-critic algorithms. In *Advances in neural information processing systems*, pp. 1008–1014, 2000.

Timothy P Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning.

Ofir Nachum, Bo Dai, Ilya Kostrikov, Yinlam Chow, Lihong Li, and Dale Schuurmans. Algaedice: Policy gradient from arbitrary experience. *arXiv preprint arXiv:1912.02074*, 2019.

Gerhard Neumann et al. Variational inference for policy search in changing situations. In *Proceedings of the 28th International Conference on Machine Learning, ICML 2011*, pp. 817–824, 2011.

Chris Nota and Philip S. Thomas. Is the policy gradient a gradient? 2019.

Theodore J. Perkins and Mark D. Pendrith. On the existence of fixed points for q-learning and sarsa in partially observable domains. In *Proceedings of the Nineteenth International Conference on Machine Learning*, ICML '02, pp. 490–497, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc. ISBN 1558608737.

Theodore J Perkins and Doina Precup. A convergent form of approximate policy iteration. In *Advances in neural information processing systems*, pp. 1627–1634, 2003.

Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 2014.

Walter Rudin et al. *Principles of mathematical analysis*, volume 3. McGraw-hill New York, 1964.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In Francis Bach and David Blei (eds.), *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 1889–1897, Lille, France, 2015.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Harm Seijen and Rich Sutton. True online td(lambda). volume 32 of *Proceedings of Machine Learning Research*, pp. 692–700, Bejing, China, 22–24 Jun 2014. PMLR. URL http://proceedings.mlr.press/v32/seijen14.html.

Lior Shani, Yonathan Efroni, and Shie Mannor. Adaptive trust region policy optimization: Global convergence and faster rates for regularized mdps. *arXiv preprint arXiv:1909.02769*, 2019.

R. S. Sutton. Learning to predict by the methods of temporal differences. *Machine learning*, 3(1): 9–44, 1988.

Richard S. Sutton. *Temporal Credit Assignment in Reinforcement Learning*. PhD thesis, 1984.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT press, 2018.

Richard S. Sutton, David A. McAllester, Satinder P. Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, pp. 1057–1063, 2000.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

A Appendix

A.1 Alternative Formulation of the Discounted Policy Gradient

Our goal in this section is to show the following.

$$\begin{aligned}\sum_{s,a} d_\gamma^{\pi_\theta}(s)\pi_\theta(a \mid s) q_\gamma^{\pi_\theta}(s,a)\nabla_\theta \log \pi_\theta(a \mid s)& \\ = \mathbb{E}_{p_0,\pi_\theta}\left[\sum_{t=0}^{\infty}\gamma^t R_{t+1}\sum_{j=0}^{t}\nabla_\theta \log \pi_\theta(A_j \mid S_j)\right].& \qquad (11)\end{aligned}$$

First, since the partial sums of $d_\gamma^{\pi_\theta}$ actually converge absolutely to $d_\gamma^{\pi_\theta}$, we can use Mertens's theorem on Cauchy products to write

$$\begin{aligned}
d_\gamma^{\pi_\theta}(s) q_\gamma^{\pi_\theta}(s,a) &= \left(\sum_{t=0}^{\infty} \gamma^t \Pr(S_t = s \mid p_0, \pi_\theta)\right) \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{\infty} \gamma^t R_{t+1} \mid S_0 = s, A_0 = a\right] \\
&= \left(\sum_{t=0}^{\infty} \gamma^t \Pr(S_t = s \mid p_0, \pi_\theta)\right) \sum_{t=0}^{\infty} \mathbb{E}_{\pi_\theta}\left[\gamma^t R_{t+1} \mid S_0 = s, A_0 = a\right] \\
&\qquad \triangleright \text{ boundedness of the reward and Fubini's theorem} \\
&= \sum_{t=0}^{\infty} \sum_{j=0}^{t} \gamma^t \Pr(S_j = s \mid p_0, \pi_\theta) \mathbb{E}_{\pi_\theta}\left[R_{t-j+1} \mid S_0 = s, A_0 = a\right] \\
&\qquad \triangleright \text{ Mertens's theorem} \\
&= \sum_{t=0}^{\infty} \sum_{j=0}^{t} \gamma^t \Pr(S_j = s \mid p_0, \pi_\theta) \mathbb{E}_{\pi_\theta}\left[R_{t+1} \mid S_j = s, A_j = a\right] \\
&\qquad \triangleright \text{ using the Markov property to reindex the reward term}
\end{aligned}$$

Note that $\Pr(S_j = s \mid p_0, \pi_\theta)\pi_\theta(a \mid s) = \Pr(S_j = s, A_j = a \mid p_0, \pi_\theta)$. Hence, from substituting into the policy gradient term,

$$\begin{aligned}
\nabla_\theta J_\gamma(\pi_\theta) &= \sum_{s,a} \sum_{t=0}^{\infty} \sum_{j=0}^{t} \gamma^t \Pr(S_j = s \mid p_0, \pi_\theta) \pi_\theta(a \mid s) \mathbb{E}_{\pi_\theta}\left[R_{t+1} \mid S_j = s, A_j = a\right] \nabla_\theta \log \pi_\theta(a \mid s) \\
&= \sum_{s,a} \sum_{t=0}^{\infty} \sum_{j=0}^{t} \gamma^t \Pr(S_j = s, A_j = a \mid p_0, \pi_\theta) \mathbb{E}_{\pi_\theta}\left[R_{t+1} \mid S_j = s, A_j = a\right] \nabla_\theta \log \pi_\theta(a \mid s) \\
&= \sum_{t=0}^{\infty} \gamma^t \sum_{j=0}^{t} \sum_{s,a} \Pr(S_j = s, A_j = a \mid p_0, \pi_\theta) \mathbb{E}_{\pi_\theta}\left[R_{t+1} \mid S_j = s, A_j = a\right] \nabla_\theta \log \pi_\theta(a \mid s) \\
&= \sum_{t=0}^{\infty} \gamma^t \sum_{j=0}^{t} \mathbb{E}_{S_j, A_j}\left[\mathbb{E}_{\pi_\theta}\left[R_{t+1} \mid S_j, A_j\right] \nabla_\theta \log \pi_\theta(A_j \mid S_j)\right] \\
&= \sum_{t=0}^{\infty} \gamma^t \sum_{j=0}^{t} \mathbb{E}_{p_0, \pi_\theta}\left[\nabla_\theta \log \pi_\theta(A_j \mid S_j) R_{t+1}\right] \\
&= \mathbb{E}_{p_0, \pi_\theta}\left[\sum_{t=0}^{\infty} \gamma^t R_{t+1} \sum_{j=0}^{t} \nabla_\theta \log \pi_\theta(A_j \mid S_j)\right],
\end{aligned}$$

where the final expectation is over the trajectory distribution under the Markov chain induced by π_θ and p_0.

A.2 PROOFS

The following lemma will be useful.

Lemma 1. *Under Assumption 1, we have that the following quantities on both sides exist, and that equality holds.*

$$\sum_{t=0}^{\infty} \nabla_\theta \gamma^t \Pr(S_t = s \mid p_0, \pi_\theta) = \nabla_\theta \sum_{t=0}^{\infty} \gamma^t \Pr(S_t = s \mid p_0, \pi_\theta). \tag{12}$$

Proof. Our general approach is to use uniform convergence and the Weierstrass M-test to allow the exchange of the gradient and the series.

The partials sums $S_n \doteq \sum_{t=0}^{n} \gamma^t \Pr(S_t = s \mid p_0, \pi_\theta)$ are differentiable as π_θ is differentiable, and each $\Pr(S_t = s \mid p_0, \pi_\theta)$ is simply a product of terms that do not involve θ (i.e., the transition distribution) and a product of terms with π_θ. Moreover, S_n converges (component-wise) to the unnormalized future state visitation distribution. And therefore the r.h.s exists.

Now, the question is whether $\nabla_\theta S_n$ converges to anything (again, component-wise), that is, if the left hand side exists. In fact, we will show that the S_n converge uniformly by applying the Weierstrass M-test to the individual terms $f_t \doteq \nabla_\theta \gamma^t \Pr(S_t = s \mid p_0, \pi_\theta)$. Recall that for the Weierstrass M-test, we must find $M_t \in \mathbb{R}$ such that for any θ, $\|f_t\| \leq M_t$, where $\|\cdot\|$ is any Banach space norm (e.g., the Euclidean norm), and show that $\sum_n M_t \leq \infty$. This will be enough for uniform convergence. convergence of $\nabla_\theta S_n$.

Let $f_t \doteq \nabla_\theta \gamma^t \Pr(S_t = s \mid p_0, \pi_\theta)$. We have

$$\begin{aligned}
\|f_t\| &= \|\nabla_\theta \gamma^t \Pr(S_t = s | p_0, \pi_\theta)\|, \\
&= \left\| \gamma^t \sum_{s_i, a_i, 0 \leq i \leq t-1} \Pr(S_0 = s_0, A_1 = a_1, \cdots, S_t = s) \sum_{j=0}^{t-1} \nabla_\theta \log \pi_\theta(a_j \mid s_j) \right\|, \\
&\leq \gamma^t \sum_{s_i, a_i, 0 \leq i \leq t-1} \Pr(S_0 = s_0, A_1 = a_1, \cdots, S_t = s) \sum_{j=0}^{t-1} \|\nabla_\theta \log \pi_\theta(a_j \mid s_j)\|, \\
&\leq \gamma^t \sum_{s_i, a_i, 0 \leq i \leq t-1} \Pr(S_0 = s_0, A_1 = a_1, \cdots, S_t = s) Ct, \\
&\leq \gamma^t Ct.
\end{aligned}$$

Now, note that

$$C \sum_{t=0}^{\infty} \gamma^t t = \frac{C\gamma}{(1-\gamma)^2} < \infty. \tag{13}$$

Hence, the partial sums $S_n = \sum_{t=0}^{n} \nabla_\theta \gamma^t \Pr(S_t = s \mid p_0, \pi_\theta)$ converge uniformly. The uniform convergence of these partial sums implies that both quantities in Equation 12, and in particular that we may interchange the series and the gradient (see (Rudin et al., 1964) for example). □

Proposition 1 (Expected Sum of Traces, Discounted Setting)**.** *Under Assumption 1, the following holds for every state s.*

$$\nabla_\theta d_\gamma^{\pi_\theta}(s) = \gamma \sum_{t=0}^{\infty} \Pr(S_t = s \mid p_0, \pi_\theta) \mathbb{E}[\mathbf{e}_{t-1} \mid S_t = s] \tag{7}$$

Proof. For notational compactness, let us define $p(s_0 \mid s_{-1}, a_{-1}) \doteq p_0(s_0)$. We can write

$$\begin{aligned}
&\Pr(S_t = s \mid p_0, \pi_\theta) \\
&\quad = \sum_{s_i, a_i;\, 0 \leq i \leq t-1} p(s \mid s_{t-1}, a_{t-1}) \prod_{j=0}^{t-1} \pi_\theta(a_j \mid s_j) p(s_j \mid s_{j-1}, a_{j-1}).
\end{aligned}$$

Hence, using the log-likelihood trick,

$$\begin{aligned}
&\nabla_\theta \Pr(S_t = s \mid p_0, \pi_\theta) \\
&\quad = \sum_{s_i, a_i;\, 0 \leq i \leq t-1} p(s \mid s_{t-1}, a_{t-1}) \prod_{j=0}^{t-1} \pi_\theta(a_j \mid s_j) p(s_j \mid s_{j-1}, a_{j-1}) \sum_{j=0}^{t-1} \nabla_\theta \log \pi_\theta(a_j \mid s_j)
\end{aligned}$$

Note that for a trajectory $(s_0, a_0, \cdots s_j, a_j)$ we defined our traces in the discounted setting as

$$\mathbf{e}_j \doteq \gamma^j \sum_{i=0}^{j} \nabla_\theta \log \pi_\theta(a_j \mid s_j).$$

We define $\mathbf{e}_{-1} \doteq 1$ as an edge case.

Hence, we can write $\nabla_\theta \Pr(S_t = s \mid p_0, \pi_\theta)$ as a kind of marginalisation of $\mathbf{e}_{t-1}$. Strictly speaking, as $\Pr(S_t \mid p_0, \pi_0)$ is not equal to 1 in general, we cannot write $\nabla_\theta \Pr(S_t = s \mid p_0, \pi_\theta)$ as an expectation. However, we can introduce a normalization term to write the expression more compactly.

$$\begin{aligned}
&\nabla_\theta \Pr(S_t = s \mid p_0, \pi_\theta) \\
&\quad = \frac{\Pr(S_t = s \mid p_0, \pi_\theta)}{\Pr(S_t = s \mid p_0, \pi_\theta)} \sum_{s_i, a_i;\, 0 \le i \le t-1} p(s \mid s_{t-1}, a_{t-1}) \left(\prod_{j=0}^{t-1} \pi_\theta(a_j \mid s_j) p(s_j \mid s_{j-1}, a_{j-1}) \right) \gamma^{1-t} \mathbf{e}_{t-1} \\
&\quad = \Pr(S_t = s \mid p_0, \pi_\theta) \sum_{s_i, a_i;\, 0 \le i \le t-1} \underbrace{\frac{p(s \mid s_{t-1}, a_{t-1}) \left(\prod_{j=0}^{t-1} \pi_\theta(a_j \mid s_j) p(s_j \mid s_{j-1}, a_{j-1}) \right)}{\Pr(S_t = s \mid p_0, \pi_\theta)}}_{(a)} \gamma^{1-t} \mathbf{e}_{t-1}.
\end{aligned}$$

Let us consider what (a) represents. The numerator is simply the joint probability of a trajectory $(s_0, a_0, \cdots, s_{t-1}, a_{t-1}, s)$. The denominator is the probability that such a trajectory ends in s. Hence, we can interpret (a) as the conditional probability of a trajectory $(s_0, a_0, \cdots, s_{t-1}, a_{t-1})$ given that such a trajectory will end in s. Hence, we will write

$$\nabla_\theta \Pr(S_t = s \mid p_0, \pi_\theta) = \Pr(S_t = s \mid p_0, \pi_\theta) \gamma^{1-t} \mathbb{E}[\mathbf{e}_{t-1} \mid S_t = s]$$

Putting it all together,

$$\begin{aligned}
\nabla_\theta \sum_{t=0}^{\infty} \gamma^t \Pr(S_t = s \mid p_0, \pi) &= \sum_{t=0}^{\infty} \gamma^t \nabla_\theta \Pr(S_t = s \mid p_0, \pi_\theta), \\
&= \gamma \sum_{t=0}^{\infty} \Pr(S_t = s \mid p_0, \pi_\theta) \mathbb{E}[\mathbf{e}_{t-1} \mid S_t = s],
\end{aligned}$$

where Lemma 1 ensures that we can interchange the gradient and infinite sum in the first equality.

□

Proposition 2. *Assume we have a trajectory* $(s_0, a_0, r_1, \cdots, s_t, a_t, r_{t+1})$. *Let* $\theta, \theta_j \in \mathbb{R}^d$ *be arbitrary, for* $0 \le j \le t$. *Then,*

$$\left\| \gamma^t r_{t+1} \sum_{j=0}^{t} \nabla_\theta \log \pi_{\theta_j}(a_j \mid s_j) - \gamma^t r_{t+1} \sum_{j=0}^{t} \nabla_\theta \log \pi_\theta(a_j \mid s_j) \right\| \tag{8}$$

$$\le \gamma^t |r_{t+1}| \sum_{j=0}^{t} \|\pi(\cdot \mid s_j) - \pi_j(\cdot \mid s_j)\|_1 \max_a \|\mathbf{x}(s_j, a)\|, \tag{9}$$

$$\le \gamma^t |r_{t+1}| (t+1) \max_j \left(\|\pi(\cdot \mid s_j) - \pi_j(\cdot \mid s_j)\|_1 \max_a \|\mathbf{x}(s_j, a)\| \right) \tag{10}$$

Proof. In the following, we will write π_j in place of π_{θ_j} and π in place of π_θ to reduce mess.

For a linear Boltzmann policy, a straightforward calculation provides the following gradient.

$$\nabla_\theta \log \pi_\theta(a \mid s) = \mathbf{x}(s, a) - \sum_b \pi_\theta(b \mid s) \mathbf{x}(s, b).$$

Hence,

$$\begin{aligned}
\|\nabla_\theta \log \pi_j(a \mid s) - \nabla_\theta \log \pi(a \mid s)\| &= \left\| \sum_b \mathbf{x}(s, b)(\pi(b \mid s) - \pi_j(b \mid s)) \right\| \\
&\le \sum_b \|\mathbf{x}(s, b)\| \cdot |(\pi(b \mid s) - \pi_j(b \mid s))| \\
&\le \|\pi(\cdot \mid s) - \pi_j(\cdot \mid s)\|_1 \max_a \|\mathbf{x}(s, a)\|
\end{aligned}$$

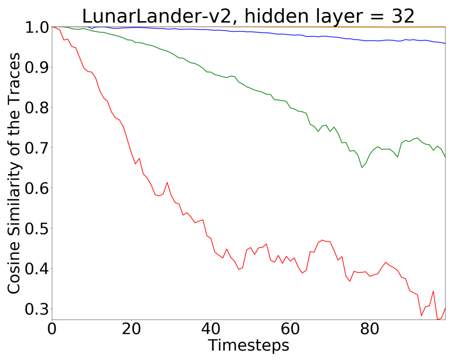
Substituting,

$$\begin{aligned}
&\left\|\gamma^t r_{t+1} \sum_{j=0}^{t} \nabla_\theta \log \pi_{\theta_j}(a_j \mid s_j) - \gamma^t r_{t+1} \sum_{j=0}^{t} \nabla_\theta \log \pi_\theta(a_j \mid s_j)\right\| \\
&\quad \leq \gamma^t |r_{t+1}| \sum_{j=0}^{t} \left\|\nabla_\theta \log \pi_{\theta_j}(a_j \mid s_j) - \nabla_\theta \log \pi_\theta(a_j \mid s_j)\right\|, \\
&\quad \leq \gamma^t |r_{t+1}| \sum_{j=0}^{t} \|\pi(\cdot \mid s_j) - \pi_j(\cdot \mid s_j)\|_1 \max_a \|\mathbf{x}(s_j, a)\|,
\end{aligned}$$

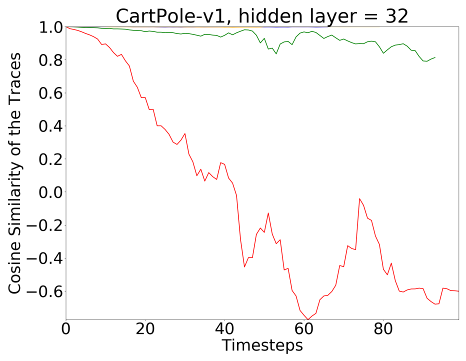
thus proving the first inequality. The second inequality follows immediately.

□

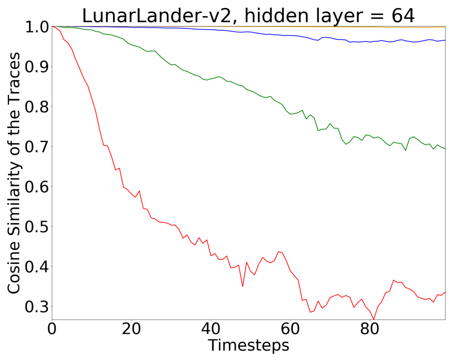
A.3 Additional Plots



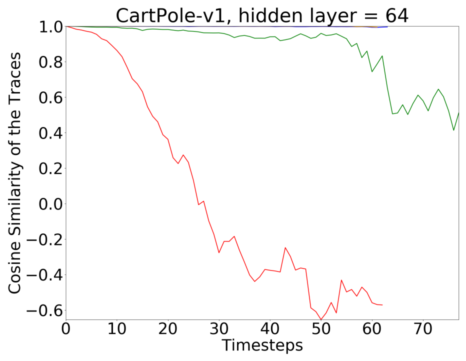
(a)



(b)



(c)



(d)

Figure 4: Each line is the mean over 100 random policies; the standard errors are smaller than the width of the lines. At every time step, the policy is updated as in Algorithm 1. Each value on the plot is the cosine similarity of the trace at that time, compared to what the trace would have been if the initial policy had been what the current policy is now, and if the exact same trajectory had been followed–the so-called counterfactual trace. The counterfactual trace is calculated with Equation 5 using the current policy and the stored trajectory up until that point.

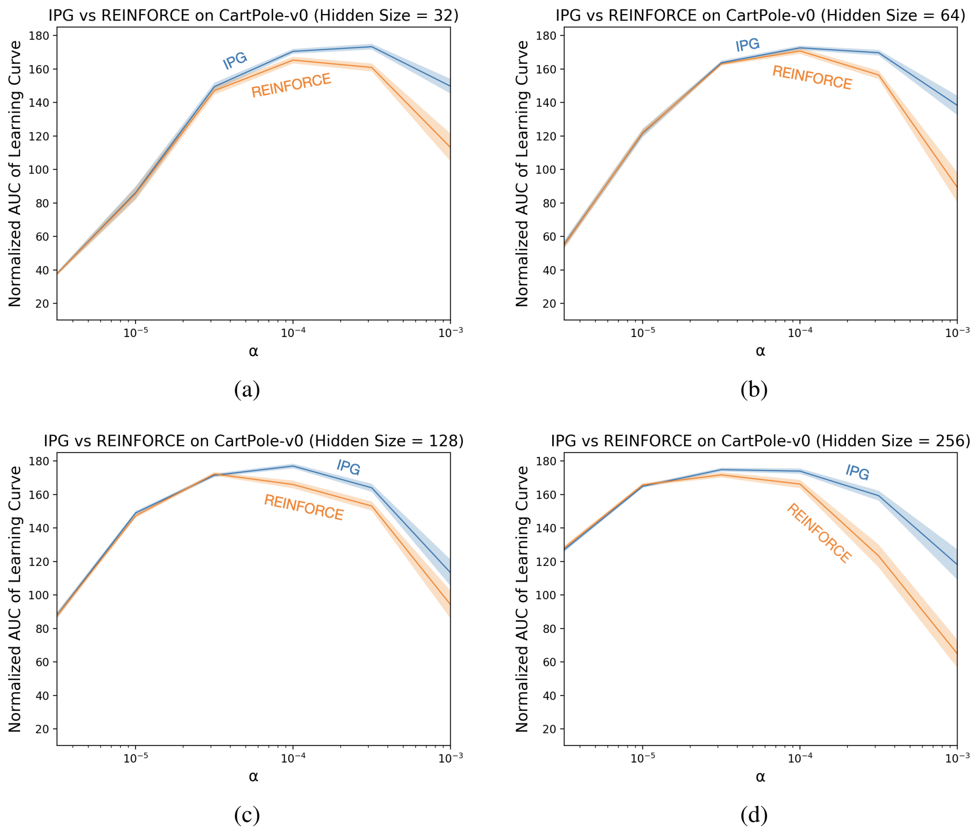
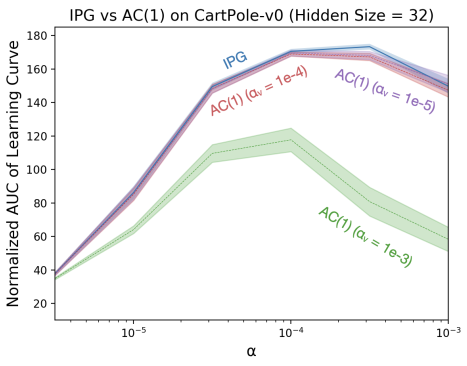
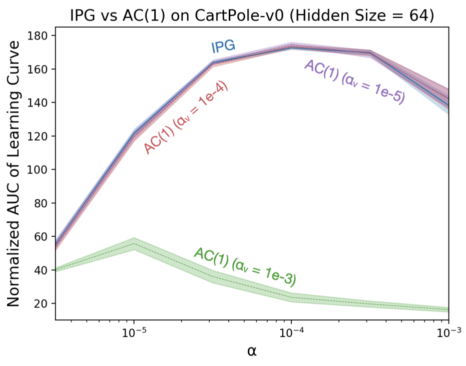


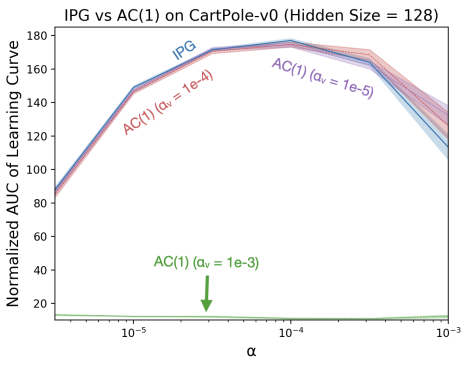
Figure 5: AUCs over 100,000 steps in the CartPole-v0 environment. Results are averaged over 30 runs, and shaded regions represent one standard error. Note the log-scale of the x-axis.



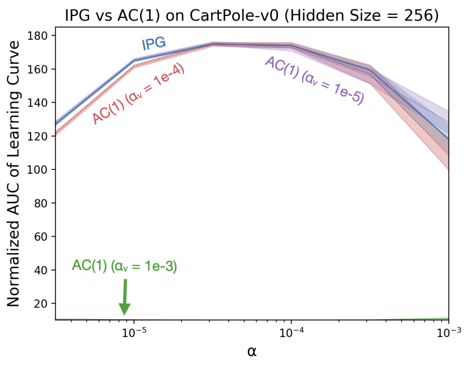
(a)



(b)



(c)



(d)

Figure 6: AUCs over 100,000 steps in the CartPole-v0 environment. Results are averaged over 30 runs, and shaded regions represent one standard error. Note the log-scale of the x-axis.

A.4 An Attempt at Deriving IPG for the Average Reward Setting

Unlike REINFORCE, IPG is suitable for both episodic and continuing tasks. In continuing tasks, it's typical to consider the average reward objective, and tempting to ask if there's a version of IPG which directly optimizes this objective. We were not, however, able to find the policy gradient of this objective in a form similar to Equation 3, from which the IPG update can be derived. Nevertheless, we provide our initial attempt in here and leave this as an avenue for future work.

In this section, we try to derive the policy gradient that takes a similar form as equation 3.

The average reward rate is defined as

$$r(\pi_\theta) \doteq \lim_{t\to\infty} \mathbb{E}\left[R_t | S_0, A_{0:t-1} \sim \pi_\theta\right]. \tag{14}$$

The limit is assumed to exist and to be independent of the initial state S_0 (an ergodicity assumption; see Puterman (2014)). The policy gradient theorem for average reward setting (Sutton et al., 2000) provides the gradient of Equation 14.

$$\begin{aligned}\nabla_\theta r(\pi_\theta) &= \sum_s d^{\pi_\theta}(s) \sum_a q^{\pi_\theta}(s,a)\pi_\theta(a|s)\nabla_\theta \log \pi_\theta(a|s) \\ &= \lim_{t\to\infty} \mathbb{E}\left[\left(\nabla_\theta \log \pi_\theta(A_t \mid S_t)\right)\sum_{j=t}^{\infty}(R_{j+1} - r(\pi_\theta))\right],\end{aligned} \tag{15}$$

where $q^{\pi_\theta}(s,a) \doteq \mathbb{E}_{\pi_\theta}[\sum_{t=0}^{\infty}(R_{t+1} - r(\pi_\theta))|S_0 = s, A_0 = a]$ is the *differential action-value* of $(s,a) \in \mathcal{S} \times \mathcal{A}$ (Sutton & Barto, 2018), and d^{π_θ} is the stationary distribution of π_θ.

For simplicity, we only consider the on-policy case. Using the policy gradient theorem for continuing tasks, we have

$$
\begin{aligned}
\nabla_\theta r(\pi_\theta) &= \lim_{t\to\infty} \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(A_t \mid S_t) \sum_{j=t}^{\infty}(R_{j+1} - r(\pi_\theta))\right]. \\
&= \lim_{t\to\infty} \frac{1}{t+1}\mathbb{E}_{\pi_\theta}\left[\sum_{i=0}^{t} \nabla_\theta \log \pi_\theta(A_i \mid S_i) \sum_{j=i}^{\infty}(R_{j+1} - r(\pi_\theta))\right] \qquad \text{(Cesaro convergence)} \\
&= \lim_{t\to\infty} \frac{1}{t+1}\mathbb{E}_{\pi_\theta}[\\
&\quad \nabla_\theta \log \pi_\theta(A_0|S_0)((R_1 - r(\pi_\theta)) + (R_2 - r(\pi_\theta)) + (R_3 - r(\pi_\theta)) + \dots) \\
&\quad + \nabla_\theta \log \pi_\theta(A_1|S_1)((R_2 - r(\pi_\theta)) + (R_3 - r(\pi_\theta)) + (R_4 - r(\pi_\theta)) + \dots) \\
&\quad + \dots \\
&\quad + \nabla_\theta \log \pi_\theta(A_t|S_t)((R_{t+1} - r(\pi_\theta)) + (R_{t+2} - r(\pi_\theta)) + (R_{t+3} - r(\pi_\theta)) + \dots)] \\
&\quad = \lim_{t\to\infty} \frac{1}{t+1}\Bigg(\mathbb{E}_{\pi_\theta}[(R_1 - r(\pi_\theta))\big(\nabla_\theta \log \pi_\theta(A_0|S_0)\big) \\
&\quad + (R_2 - r(\pi_\theta))\big(\nabla_\theta \log \pi_\theta(A_0|S_0) + \nabla_\theta \log \pi_\theta(A_1|S_1)\big) \\
&\quad + \dots \\
&\quad + (R_t - r(\pi_\theta))\big(\nabla_\theta \log \pi_\theta(A_0|S_0) + \nabla_\theta \log \pi_\theta(A_1|S_1) + \dots + \nabla_\theta \log \pi_\theta(A_{t-1}|S_{t-1})\big)] \\
&\quad + \mathbb{E}_\mu[\\
&\quad (R_{t+1} - r(\pi_\theta))\big(\nabla_\theta \log \pi_\theta(A_0|S_0) + \nabla_\theta \log \pi_\theta(A_1|S_1) + \dots + \nabla_\theta \log \pi_\theta(A_t|S_t)\big) \\
&\quad + (R_{t+2} - r(\pi_\theta))\big(\nabla_\theta \log \pi_\theta(A_0|S_0) + \nabla_\theta \log \pi_\theta(A_1|S_1) + \dots + \nabla_\theta \log \pi_\theta(A_t|S_t)\big) \\
&\quad + (R_{t+3} - r(\pi_\theta))\big(\nabla_\theta \log \pi_\theta(A_0|S_0) + \nabla_\theta \log \pi_\theta(A_1|S_1) + \dots + \nabla_\theta \log \pi_\theta(A_t|S_t)\big) \\
&\quad + \dots]\Bigg) \\
&\quad = \lim_{t\to\infty} \frac{1}{t+1}\Bigg(\mathbb{E}_{\pi_\theta}\left[\sum_{j=0}^{t-1}(R_{j+1} - r(\pi_\theta)) \sum_{k=0}^{j} \nabla_\theta \log \pi_\theta(A_k|S_k)\right] \\
&\quad + \mathbb{E}_{\pi_\theta}\left[\sum_{j=t}^{\infty}(R_{j+1} - r(\pi_\theta)) \sum_{k=0}^{t} \nabla_\theta \log \pi_\theta(A_k|S_k)\right]\Bigg)
\end{aligned}
$$

This involves a term that is similar to equation 3

$$
\lim_{t\to\infty} \frac{1}{t+1}\mathbb{E}_\mu\left[\sum_{j=0}^{t-1}(R_{j+1} - r(\pi_\theta)) \sum_{k=0}^{j} \nabla_\theta \log \pi_\theta(A_k|S_k)\right].
$$

which should be well-defined, and the other term

$$
\lim_{t\to\infty} \frac{1}{t+1}\mathbb{E}_\mu\left[\sum_{j=t}^{\infty}(R_{j+1} - r(\pi_\theta)) \sum_{k=0}^{t} \nabla_\theta \log \pi_\theta(A_k|S_k)\right]
$$

, which should also be well-defined and equal to 0.

However, it is not clear to us if both terms are well-defined and if the second term equals to 0.

A.5 Unifying IPG and REINFORCE

While AdaptChain highlights a scenario where online adaptation via interim fixed-horizon policy gradient updates is beneficial, there may exist scenarios where it can be detrimental. For example, with large learning rates, it may over-greedify toward relatively myopic outcomes. Waiting a bit before performing updates can also dampen the policy gradient discrepancies caused by a changing policy. This motivates a unifying algorithm with an adjustable update frequency.

The REINFORCE update can be recovered by accumulating (but not applying) each step's IPG update, and applying them all at once at the end of an episode. This suggests that the extremes can be interpolated by only applying some proportion of the stored update, saving the remainder for later time steps. In particular, we specify the update frequency through a *probability of updating*: with probability ω the algorithm applies (and clear) the accumulated IPG updates, and with probability $1-\omega$ it will store the current update to be applied later. At episode termination, all remaining accumulated updates will be performed deterministically.

This gives the following, *expected* incremental update, denoted IPG(ω):

$$\begin{aligned}
\mathbf{e}_t &\leftarrow \gamma \mathbf{e}_{t-1} + \gamma^t \nabla_\theta \log \pi_\theta(A_t|S_t) \\
\Delta\theta_t &\leftarrow (1-\omega)\Delta\theta_{t-1} + \alpha R_{t+1}\mathbf{e}_t \\
\theta_t &\leftarrow \theta_{t-1} + \omega\Delta\theta_t
\end{aligned}$$

with the following, additional update performed on episode termination:

$$\theta_t \leftarrow \theta_t + (1-\omega)\Delta\theta_t$$