Adaptive Off-Policy Policy Gradient Methods

Xiang Gu, Josiah Hanna

Last Updated: August 16, 2019

1 INTRODUCTION

Reinforcement learning (RL) is learning how to map states to actions so as to maximize a numerical reward. The mapping from states to actions can be seen as a sequence of decisions making an agent has to make when interacting with the environment. A general and flexible model for the environment in reinforcement learning is a Markov Decision Processes (MDPs). When applying RL to a real-world problem modeled as a MDP, perhaps the most important task is *control*, where we seek an optimal policy so that the agent obtains maximal expected cumulative reward when following this policy.

Policy gradient methods are one of many RL algorithms used for control. They're conceptually simple yet proved to be powerful in RL problems such as robotics and games. Conventional policy gradient methods learn a policy (for the control task) while interacting with the environment using the same policy. An alternative way is to use *off-policy* policy gradient methods where we separate the *target policy* that is being learned for the control task from the *behavior policy* that is deployed to generate data. *Importance sampling* is a common technique that enables learning from data generated by following a different behavior policy from the target policy. However, off-policy methods usually have higher variance but it is not always the case. In certain problems, such as the presence of significant rare event, off-policy methods tend to have better performance.

An essential but natural question for off-policy policy gradient methods is which behavior policy one should use. Using a better behavior policy is a potential way to effectively reduce the policy gradient variance and thus improve learning. We proposed a general framework for searching and using an optimal behavior policy for off-policy policy gradient methods. We then give a simplified, more data-efficient and practical algorithm which is an instantiation of the general framework. Finally we will introduce three potential objectives to use on how to updating the behavior policy, and talk abou the pros and cons of each objective.

Our work also shows some of the difficulties of optimizing two policies at the same time.

2 PRELIMINARIES

2.1 Markov Decision Process and Reinforcement Learning Task

Reinforcement learning problems are formalized as Markov decision processes (MDPs), defined by the tuple $(\mathcal{S}, \mathcal{A}, p, r, \rho_0, \gamma)$, where $\mathcal{S}$ is a finite set of states, $\mathcal{A}$ is a finite set of actions, $p : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the environment transition probability distribution, $r : \mathcal{S} \to \mathbb{R}$ is the reward function, ρ_0 is the distribution of the starting state, and $\gamma \in [0, 1]$ is the discount factor. For simplicity, we will consider episodic environment throughout the whole paper.

Let π_θ denote a parameterized policy $\pi_\theta : \mathcal{S} \times \mathcal{A} \to [0, 1]$. At each episode, initial state S_0 is selected according to initial state distribution $S_0 \sim \rho_0(S)$. A trajectory $H := (S_0, A_0, R_0, ..., S_{T-1}, A_{T-1}, R_{T-1})$ is then generated by sampling actions according to the policy $A_t \sim \pi_\theta(A_t|S_t)$ and sampling states according to environment transition dynamics $S_t \sim p(S_t|S_{t-1}, A_{t-1})$. At each timestep, a numerical reward $R_t = r(S_t)$ is received. Let $G_t(H) := \Sigma_{l=t}^{T-1} \gamma^{l-t} R_l$ denote the discounted cumulative reward after (S_t, A_t) in trajectory H. If one trajectory terminates with less than T timesteps, we treat the terminal state as an *absorbing state* that transitions only to itself and receives only rewards of zero. So we will pad that trajectory with terminal states, actions that will lead back the terminal state itself, and zero rewards.

Our goal is to find the parameter $\theta^\star$ such that the *expected* cumulative reward of a trajectory, generated by simulating π_θ, is maximal. Formally,

$$\text{find } \theta^\star = \underset{\theta}{\operatorname{argmax}} \, \mathbb{E}_{H \sim \pi_\theta} \left[G_0(H) \right], \tag{1}$$

where $\mathbb{E}_{H \sim \pi_\theta}[\cdot]$ denotes the expectation of a random variable over all possible trajectories generated by following π_θ.

2.2 Policy Gradient Reinforcement Learning

Policy gradient methods are one method for solving RL problems. Instead of assigning a value function to each state (or state-action pair) and using it as an intermediate step to find the optimal policy for the task, policy gradient methods directly manipulate the policy towards a better policy without consulting a value function.

Policy gradient methods solve the optimization problem in 1 by using *gradient ascent*. They repeatedly updates parameter θ in the direction of gradient of expectation of cumulative reward

$$\theta \leftarrow \theta + \alpha \cdot \nabla_\theta \mathbb{E}_{H \sim \pi_\theta}[G_0(H)], \tag{2}$$

where the gradient term is [1]

$$g = \nabla_\theta \mathbb{E}_{H \sim \pi_\theta}[G_0(H)] = \mathbb{E}_{H \sim \pi_\theta} \left[\sum_{t=0}^{T-1} G_t(H) \nabla_\theta \log \pi_\theta(A_t|S_t) \right]. \tag{3}$$

Therefore, the policy gradient estimator is given by

$$g \approx \hat{g} = \left\langle \sum_{t=0}^{T-1} G_t(H) \nabla_\theta \log \pi_\theta(A_t|S_t) \right\rangle, \tag{4}$$

where $< \cdot >$ denotes an average over multiple trajectories. Typically the trajectories are sampled from the current policy, which is optimized through an iterative algorithm.

3 ADAPTIVE OFF-POLICY POLICY GRADIENT Methods

3.1 Converting Policy Gradient Method to an Off-Policy Version

Although policy gradient methods typically sample trajectories from current *target policy*, the choice of data collection policy – also known as *behavior policy* – can be made arbitrary through

[1] This gradient is just one potential answer among many others, which is also referred to as 'vanilla' REINFORCE [Williams, 1992][Sutton et al., 2000]. In this report, we use it as our policy gradient method for simplicity and demonstration.

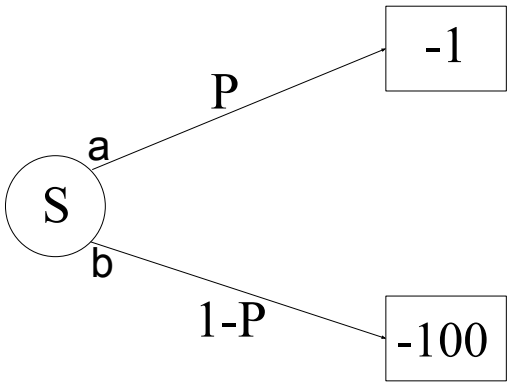


Figure 1: A simple Bandit problem. Current policy selects action a with probability P and it gives -1 reward, whereas it selects action b with probability $1 - P$ and it gives -100 reward.

importance sampling. Methods that deploy a different behavior policy from the target policy are referred to as *off-policy* method.
Equation 3 can be written in a way that allows off-policy learning

$$g = \mathbb{E}_{H\sim\pi_{\theta_b}}\left[\sum_{t=0}^{T-1} w(H)G_t(H)\nabla_\theta \log \pi_\theta(A_t|S_t)\right], \tag{5}$$

where $w(H) := \prod_{t=0}^{T-1} \frac{\pi_\theta(A_t|S_t)}{\pi_{\theta_b}(A_t|S_t)}$ is the *importance sampling ratio*. Equation 5 gives us another unbiased estimator of policy gradient. It allows off-policy learning from trajectories generated from a different policy π_{θ_b}. All we've changed is now we need to weight each trajectory by its importance sampling ratio, which accounts for the shift of state and action space visitation distribution between target and behavior policy.
Therefore, the off-policy policy gradient estimator is given by

$$g \approx \hat{g} = \left\langle \sum_{t=0}^{T-1} w(H)G_t(H)\nabla_\theta \log \pi_\theta(A_t|S_t) \right\rangle. \tag{6}$$

Now, we can update policy parameter θ using exactly the same formula in equation 2. This is the off-policy version of policy gradient and we call it *off-policy policy gradient method*.
Off-policy policy gradient is a generalization of on-policy policy gradient method. If we choose the behavior policy π_{θ_b} to be the same as target policy π_θ, then the off-policy policy gradient method becomes conventional on-policy policy gradient method.

3.2 A Simple Motivation Problem

In general, off-policy methods have higher variance than on-policy methods, but it is not always the case. Before we proceed, let's see an a simple example that serves as a motivation problem as to why we need off-policy version of policy gradient method and how it outperforms the conventional on-policy policy gradient method. Consider a simple Bandit problem (Figure 1) with one state and two actions which terminates after one step.

Suppose our initial policy is to choose actions uniformly random. Our goal is try to find the optimal policy for this MDP using policy gradient method. Obviously, one can immediately tell that the optimal policy is to always choose action a. But let's see what conventional policy gradient methods will do.
In the beginning of the learning process, we will collect a roughly even number of -1 rewards and -100 rewards because $P \approx 0.5$. The policy gradient estimate will point to the direction that pushes the policy away from action b, which is good. However, in the middle of the learning process, as target policy keeps moving towards action a, P will be significantly larger than $1 - P$. Thus, it's highly likely that all samples we collected at this point are all reward -1. The policy gradient estimate will then point to the opposite direction of action a because they're *negative* rewards. So our policy will be pushed back towards action b, until it finally collects -100 rewards again and realize that the other action is actually worse. The target policy is then pushed towards action a again. This bouncing around of target policy is bad for trying to learn an optimal policy. Actually, the target policy will never truly converge to the optimal policy, because with the limited capacity of samples collected at each iterations it is inevitable for the target policy to collect samples that are all of reward -1 at a future iteration. And thus it will pull the target policy back towards action b.

Let's try to apply off-policy policy gradient method with a uniformly random behavior policy. At each iteration we will choose actions by following this random behavior policy that is capable of collecting both -1 rewards and -100 rewards for the policy gradient estimate no matter what the target policy currently is. Consequently, we can keep seeing the -100 rewards throughout all iterations even if the target policy is very close to the optimal policy. All those -100 rewards keeps 'reminding' the agent of this bad action so it won't forget and fall back. In this way, our policy gradient estimate can always point to the correct direction and push the target policy towards the optimal policy.

3.3 Adaptive Off-Policy Policy Gradient Methods

The last section shows that off-policy can be preferable to on-policy methods. A natural next question is: which behavior policy should we use in off-policy policy gradient method? Conventional policy gradient method uses current target policy as behavior policy, but this is not necessarily the optimal choice. The main contribution of our work is to provide algorithms that search and use the optimal behavior policy for off-policy policy gradient to use.
At each iteration, we first run a subroutine to update the behavior policy to be a better behavior policy by interacting with the environment and collecting samples. [2]. Next we use this updated behavior policy to collect more samples and update the target policy using the gradient given by off-policy policy gradient methods (equation 6). In principle, we expect that this new framework learns the optimal target policy faster because at each iteration we have a better behavior policy that results in a policy gradient with lower variance. This framework is complementary to any policy gradient method. And we call this framework *adaptive off-policy policy gradient method*, with its pseudocode shown in Algorithm 1.

In practice, provided that the change of target policy between two consecutive iterations is small, we can expect that the optimal behavior policies between two consecutive iterations are very similar. So it will be very date-inefficient to run a full algorithm from scratch to find the optimal behavior policy at each iteration. Alternatively, we can update behavior policy only *one step* in the direction given by BPO at each iteration on the basis of the previous behavior policy in last

[2]We delay the discussion of how to find this optimal behavior policy in section 3.4 The idea is to search and use a behavior policy that reduces the variance of off-policy policy gradient estimator.

Algorithm 1 Framework of Adaptive Off-Policy Policy Gradient Method

Require: Initial target and behavior policy parameters θ and θ_b respectively; A subroutine behavior policy optimizer $BPO(H, \theta)$ that returns policy gradient estimate for updating behavior policy toward a better one w.r.t. current target policy π_θ using samples H generated from π_θ; A subroutine target policy optimizer $TPO(H, \theta_b)$ that returns policy gradient estimate to update target policy using samples H generated from π_{θ_b}; Step size α and β.

Ensure: A learned target policy parameter θ such that $\pi_\theta \approx \pi_\theta^\star$

```
initialize θ_b = θ
loop
  repeat
    sample trajectories from behavior policy: H ~ π_{θ_b}
    update behavior policy θ_b ← θ_b + α * BPO(H, θ)
  until behavior policy parameter θ_b converges
  sample trajectories from updated behavior policy: H ~ π_{θ_b}
  update target policy θ ← θ + β * TPO(H, θ_b)
end loop
```

iteration. All we have to modify on the framework is now we only update behavior policy for one step, and then we directly collect samples and update target policy. And we call it *simplified* adaptive off-policy policy gradient method.

However, there is one potential issue when applying this desired algorithm to large problems – instability. As shown in the framework, there are two step sizes α and β in the algorithm. It might be difficult to tune these parameters because these parameters might be problem specific and one parameter might be affected by the other.

3.4 Choices of the Behavior Policy

We now proceed to discuss how to search for a better behavior policy. Recall that the policy gradient estimator is:

$$g \approx \hat{g} = \left\langle \sum_{t=0}^{T-1} w(H) G_t(H) \nabla_\theta \log \pi_\theta(A_t|S_t) \right\rangle.$$

We want a measure of policy gradient variance. We can then use this measure as an objective for behavior policy search. We want an objective that we can optimize w.r.t. the behavior policy parameters such that the new behavior policy gives us lower variance gradient estimates.

3.4.1 Gradient Objective: A Full Solution

When it comes to measure the quality of an estimator, the most natural objective is to use the mean-squared error (MSE) of the estimator. Our task is thus to search for the behavior policy that minimize the MSE of the policy gradient estimator. Formally,

$$\text{find } \theta_b^\star = \operatorname*{argmin}_{\theta_b} MSE \left[\sum_{t=0}^{T-1} w(H) G_t(H) \nabla_\theta \log \pi_\theta(A_t|S_t) \right]. \tag{7}$$

Define $\Gamma = \sum_{t=0}^{T-1} w(H) G_t(H) \nabla_\theta \log \pi_\theta(A_t|S_t)$. The gradient of this objective w.r.t. θ_b is then given by the following theorem:

Theorem 1. $\nabla_{\theta_b} MSE[\Gamma] = \mathbb{E}_{H\sim\pi_{\theta_b}}\left[-\Gamma^2 \sum_{t=0}^{T-1} \nabla_{\theta_b} \log \pi_{\theta_b}(A_t|S_t)\right]$

Proof. Proofs for all theoretical results are included in Appendix A. □

Note that Γ here is, in most cases, a vector-valued random variables instead of a scalar-valued random variable. As appendix shows, however, we can proceed the derivation as if Γ is a scalar-valued random variable. Although this objective does exactly what we want by targeting searching for the behavior policy that gives an estimator with minimal MSE, some possible setbacks of it are 1). computation is more expensive because we need to compute the term $\nabla_\theta \log \pi_\theta(A_t|S_t)$ in Γ, which might be a large vector with millions of components (e.g. when we use large neural network to implement policies); 2). it depends on the parameterization of target policy. Choice of parameterization can affect the variance measure. It is known to affect learning in vanilla policy gradient methods [Kakade, 2002].

3.4.2 G_0 Objective: A Straightforward Choice from Prior Work

Prior work in policy evaluation literature has proposed an algorithm – Behavior Policy Gradient (BPG) – to search for the optimal behavior policy when using off-policy method to evaluation a given target policy [Hanna et al., 2017]. A straightforward approach to our problem is to directly apply the same approach. Formally,

$$\text{find } \theta_b^\star = \operatorname*{argmin}_{\theta_b} MSE[w(H)G_0(H)]. \tag{8}$$

As shown in the paper, the gradient of the objective w.r.t. θ_b is

$$\nabla_{\theta_b} \text{MSE}[w(H)G_0(H)] = \mathbb{E}_{H\sim\pi_{\theta_b}}\left[-(w(H)G_0(H))^2 \sum_{t=0}^{T-1} \nabla_{\theta_b} \log \pi_{\theta_b}(A_t|S_t)\right]. \tag{9}$$

The obvious advantage of using this objective is that this is a straightforward application of previous work. The downside of it is that it was original developed for policy evaluation, not policy gradient. So it might not be able to indicate what we want to accomplish. Thus it does not directly minimize the variance of the policy gradient estimate.

3.4.3 Advantage Objective: A Compromise

Alternatively, there is an objective we can use that avoids the parameterization dependency problem in gradient objective yet remains relevant to the policy gradient estimator – MSE of the sum of re-weighted IS returns $MSE\left[\sum_{t=0}^{T-1} w(H)G_t(H)\right]$, which is the first term in the policy gradient estimator ignoring the gradient-log term. Formally, our task is

$$\text{find } \theta_b^\star = \operatorname*{argmin}_{\theta_b} MSE\left[\sum_{t=0}^{T-1} w(H)G_t(H)\right]. \tag{10}$$

The gradient of this objective w.r.t. θ_b is given by the following theorem:

Theorem 2. $\nabla_{\theta_b} MSE\left[\sum_{t=0}^{T-1} w(H)G_t(H)\right] = \mathbb{E}_{H\sim\pi_{\theta_b}}\left[-\left(\sum_{t=0}^{T-1} w(H)G_t(H)\right)^2 \sum_{t=0}^{T-1} \nabla_{\theta_b} \log \pi_{\theta_b}(A_t|S_t)\right]$

Proof. Proofs for all theoretical results are included in Appendix B. □

This objective is a compromise of the first two objectives. It is more related to gradient objective than G_0 objective, and it also alleviate the potential issue of the dependency of the parameterization. If we compare all three objectives with each other, one might find that actually we're finding a balance between complexity and relevance. And there are actually more potential objectives one can think of. There three objectives, however, lie on both ends and the middle point of the 'spectrum' comprised of all possible objectives. On one end, the G_0 objective gives us the simplest objective yet it might suffer from irrelevance to the task. On the other end, the full objective accurately depicts the task we're trying to address but it might not be the best choice in practice because of the dependency of parameterization and costly computation. The advantage objective is a compromise between these two which has advantages of the two objectives without severe setbacks.

4 EMPIRICAL RESULTS

We devised and performed several experiments to study the performance of the adaptive off-policy policy gradient framework and try to understand its behavior. Our experiments are designed to answer the following questions:

1. How good is the performance of adaptive off-policy policy gradient methods, compared with its on-policy and off-policy (with a fixed behavior policy) counterpart?
2. What is the cause of the performance improvement in the previous question, if any?
3. What are the performance differences using different objectives listed above?

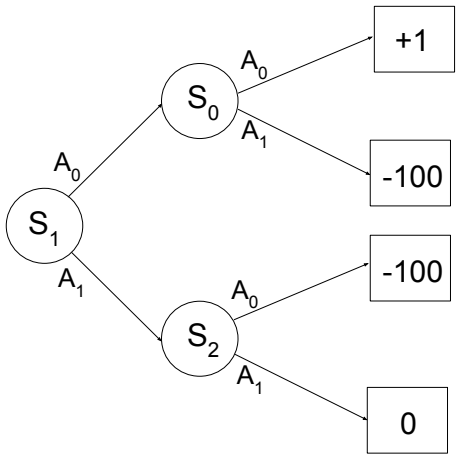


Figure 2: A simple two-step Bandit problem for experiment 2. The problem has three states S_0, S_1, and S_2. At each state, there are two actions A_0 and A_1 the agent can take. The agent always starts in S_1. It takes the agent to S_0 with zero reward if A_0 is taken, and S_1 with zero reward if A_1 is taken. In state S_0, if the agent takes action A_0 it receives reward +1 and then the episode terminates. If the agent takes action A_1 it receives reward -100 and the episode terminates. Similarly, in state S_1, the episodes terminates with -100 reward if action A_0 is taken, and reward 0 if action A_1 is taken.

4.1 Empirical Set-up

We designed three experiments to try to answer these three questions. Here we will briefly describe the set-up of the experiment.

1. We chose REINFORCE algorithm as the behavior policy baseline, and tested three methods – on-policy (naive) REINFORCE, off-policy REINFORCE with fixed behavior policy [3], and adaptive off-policy REINFORCE – on optimizing a 4-armed Bandit problem. One of the four actions result in a high magnitude reward. Results are averaged over 10 runs with different random number generator seeds in 95% confidence interval.

2. We devised a two-step MDP as shown in Figure 2. The problem is simple enough that we can exactly compute the variance ($Var[\hat{g}] = E[\hat{g}^2] - E[\hat{g}]^2$) for both current behavior policy and target policy. In other words, we're trying to answer the question "do we find a better behavior policy in terms of variance of policy gradient estimator". We will ignore the second term $E[\hat{g}]^2 = g^2$ when plotting the results because it's a constant when comparing two behavior policies. Results are averaged over 25 runs with different random number generator seeds in 95% confidence interval.

3. We tested our framework on a cartpole problem modified based on OpenAI Gym [Brockman et al., 2016]. A pole is attached by an un-actuated joint to a cart, which moves along a frictionless track. The system is controlled by applying a force of +1 or -1 to the cart. The pendulum starts upright, and the goal is to prevent it from falling over. A reward of +1 is provided for every timestep that the pole remains upright. The episode ends either when 25 time steps are reached, or when the pole is more than 15 degrees from vertical or the cart moves more than 2.4 units from the center. Results are averaged over 25 runs with different random number generator seeds in 95% confidence interval.

4.2 Empirical Results

1. Figure 3 is the result of experiment 1. The adaptive off-policy method (IS-REINFORCE-BPG) learns the optimal policy faster than the other two methods. At the same time, we can see that the static off-policy version (IS-REINFORCE) doesn't perform well, compared with the conventional on-policy version (REINFORCE).

2. Figure 4 is the result of experiment 2. In on-policy method (Figure 4a), since the target policy is identical to behavior policy, the resulting curves overlap on each other. We can see that the variance increase first and then decrease. However, if we use adaptive off-policy method, the variance of policy gradient estimate will only decrease as the learning process goes. And it's strictly better than using the target policy to generate data from. All three objectives mentioned above show a similar behavior.

3. Figure 5 is the result of experiment 3. From Figure 5a, we can see that the on-policy REINFORCE has a stable average performance to optimize the target policy but individual runs might bounce around, and the off-policy version has a similar performance as the on-policy version. For the adaptive off-policy version (Figure 5b, 5c, and 5d), however, some runs improved and stayed in the optimal target policy without any bouncing around, whereas some runs degraded. The average performance is nevertheless still much better than the on-policy version. Another interesting thing to notice is that the entropy of the target policy in adaptive off-policy methods decreases much faster than the on-policy version to zero.

[3] It keeps using the initial behavior policy throughout all iterations. The initial behavior policy is searched and found by a BPG algorithm. We also refer this version as static off-policy methods, in contrast with the adaptive off-policy method.

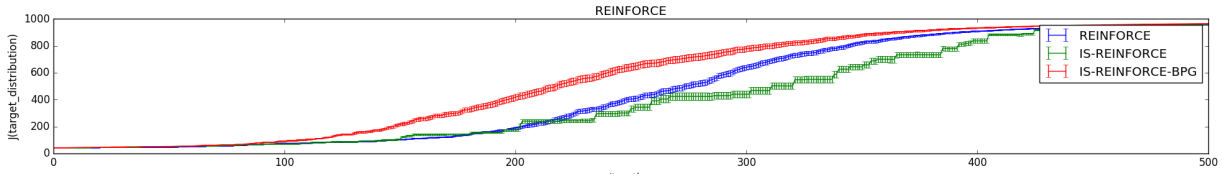
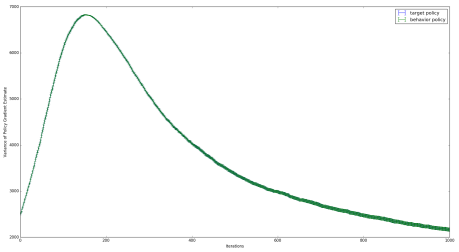
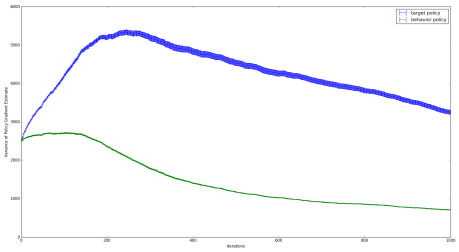


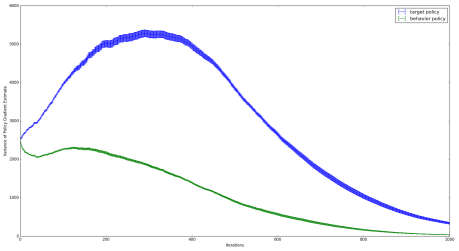
Figure 3: Results of experiment 1. The x-axis is the number of iterations of learning process. The y-axis is the expected per-action reward under current target policy. We desire a control algorithm that can improve the expected per-action reward most with less number of iterations. REINFORCE refers to the on-policy REINFORCE algorithm. IS-REINFORCE refers to the static off-policy REINFORCE. IS-REINFORCE-BPG refers to adaptive off-policy REINFORCE with G_0 objective.



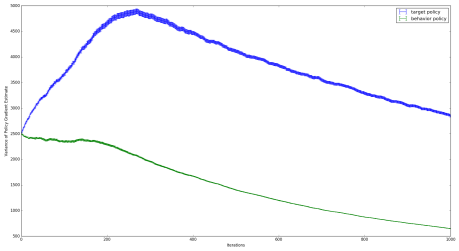
(a) On-Policy



(b) Adaptive Off-Policy with Gradient Objective



(c) Adaptive Off-Policy with Advantage Objective

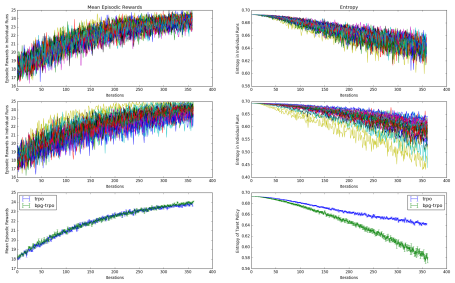


(d) Adaptive Off-Policy with G_0 Objective

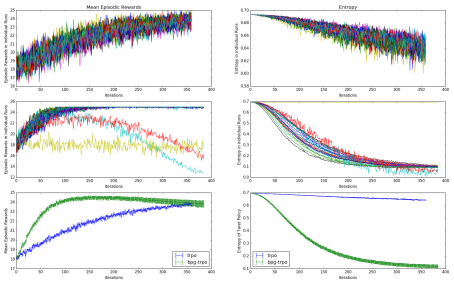
Figure 4: Results of experiment 2. The x-axis is the number of iterations of learning process. The y-axis is the variance of policy gradient estimate (shifted by a constant) computed analytically. We desire a control algorithm that gives smaller estimate variance provided it's an unbiased estimate. Subplot (a) shows the result of using on-policy REINFORCE on this two-step Bandit problem. Subplot (b) (c) (d) show the result of using adaptive off-policy REINFORCE with different objectives.

5 DISCUSSION AND FUTURE WORK

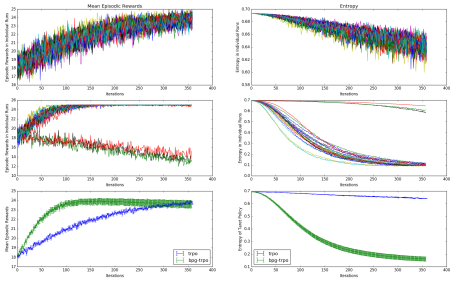
Experiment shows one limitation in adaptive off-policy policy gradient – instability. It did improve the learning process but at the same time several individual runs might not be able to perform as well as expected. This might be correlated to the potential problem of synchronizing the update of target policy and behavior policy, which then reflects on the tuning of update step sizes. In our opinion, this is the major issue that makes off-policy methods more challenging. However, we



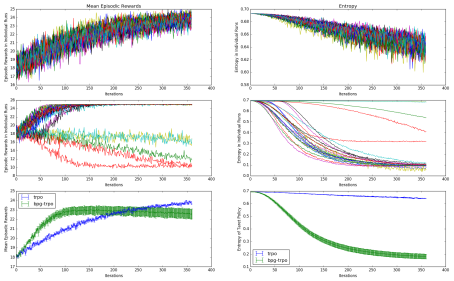
(a) Off-Policy with fixed behavior policy baseline



(b) Adaptive Off-Policy with Gradient Objective



(c) Adaptive Off-Policy with Advantage Objective



(d) Adaptive Off-Policy with G_0 Objective

Figure 5: Results of experiment 3. Within each figure, left column represents the mean episodic rewards of current target policy, whereas the right column represents the entropy of current target policy. The first row is the performance of on-policy REINFORCE with individual runs. The second row is the performance of adaptive off-policy REINFORCE with individual runs with different objectives. The third row is the average of 25 runs with two methods plotted on the same subplot for comparison.

also saw the advantage of using adaptive off-policy policy gradient methods – faster and steady learning. The reason is, in a high level, off-policy uses a behavior policy that keeps sampling actions which will be otherwise neglected under target policy, so the agent won't forget those action and thus be able to push the policy always towards the right direction without bouncing. Prior work also showed that off-policy improves the learning process even more with the presence of significant rare events [Ciosek and Whiteson, 2017]. In addition, the entropy of behavior policy will collapse pretty fast for adaptive off-policy policy gradient methods, meaning the behavior policy becomes deterministic very quickly, if the step size of behavior policy is too large. And it will be very hard for behavior policy to swing back to non-deterministic after it becomes deterministic. In the case, the behavior policy might violate the *coverage* requirement of importance sampling [4], and therefore the results with off-policy methods will be biased.

Several potential future work include 1). add an entropy bonus to the behavior policy update to alleviate the fast entropy collapse problem; 2). intermix on-policy and off-policy methods for a stable and fast learning. It is conceivable that off-policy and on-policy methods are suitable for different phases of the learning process. Then it's an interesting question on how to automatically detect the setting when it's better to stick to on-policy methods, and when the variance of policy

[4]Coverage means every action taken under target policy π_θ is also taken, at least occasionally, under behavior policy π_{θ_b}. That is, $\pi_\theta(a|s) > 0$ implies $\pi_{\theta_b}(a|s) > 0$.

gradient estimator of on-policy method starts to increase and it's thus more beneficial to switch to off-policy method to decrease the variance.

6 RELATED WORK

Previous work in general stochastic gradient descent (SGD) context proposed an idea that SGD can be accelerated via adaptive Importance Sampling [Bouchard et al., 2015]. However, their work focus on general SGD, and had merely tested their algorithm in a simple reinforcement learning (RL) environment with simplest policy gradient method. Our work instead focuses on RL setting, and we tested it on more complicated RL environments with more advanced policy gradient algorithm.

Adaptive off-policy policy gradient methods are closely related to existing work on policy gradient method and off-policy evaluation. Policy gradient methods have been extensively used to approach complex RL problems (e.g. high-dimensional continuous state and action space control) and currently it seems to be the only feasible framework for those hard RL problems. There are lots of ways people tried to improve policy gradient methods [Kober and Peters, 2012] [Schulman et al., 2015a] [Schulman et al., 2015b]. Most of the policy gradient methods use an on-policy version where our work focus on off-policy version. The adaptive off-policy policy gradient framework is complementary to any policy gradient methods.

In off-policy evaluation literature, existing work discussed one essential question with off-policy evaluation: which behavior policy is optimal to deploy to generate data for evaluating the target policy [Hanna et al., 2017]. It proposed an algorithm to search for this optimal behavior policy. Then the same authors followed up this idea and extended it to do control with policy gradient methods [Hanna and Stone, 2018] . However, in that paper, they just talked about the possibility of using an optimal behavior policy computed at the beginning of the learning process, and kept it fixed throughout all subsequent iterations. Another limitation is that they directly copied and used the same objective as in the off-policy evaluation paper to update the behavior policy, which doesn't necessarily capture the task here in control. Our work is a natural follow-up on this paper where we adapt the behavior policy at all iterations and we investigated several possible objectives.

Finally, some prior work discussed a similar idea as our work but from a totally different perspective [Ciosek and Whiteson, 2017]. Like us, they discussed the idea of adapting sampling procedure throughout all iterations to speed up policy gradient methods. They, however, chose to adapt the environment variables at each iteration, yet sticking to an on-policy method. It's possible and reasonable to adapt the environment variables if we're learning with simulation. But this is also one big limitation from our point of view because simulations are not always available (or not even possible) and we cannot thus access the environment variables. Instead, we think that adapting behavior policy is a more general and feasible way to approach this problem because either in simulation or in real life, one can always choose and change the behavior policy to deploy without any limitations.

7 CONCLUSION

In this work, we investigated the idea of adapting behavior policy in off-policy policy gradient methods. We proposed three possible objectives for optimizing behavior policy and tested them on three simple RL problems. Empirical results show adapting behavior policy did help learning process to be faster and prevents the bouncing problem occurred in on-policy method. One major issue, however, is the instability in individual runs. The reason for it is the problem of synchronizing

the update of both target policy and behavior policy, which might solved by careful tuning of step size parameters. Finally we discussed two potential ideas for future work.

References

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

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*, pages 1057–1063, 2000.

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

Josiah P Hanna, Philip S Thomas, Peter Stone, and Scott Niekum. Data-efficient policy evaluation through behavior policy search. *arXiv preprint arXiv:1706.03469*, 2017.

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

Kamil Andrzej Ciosek and Shimon Whiteson. Offer: Off-environment reinforcement learning. In *AAAI*, pages 1819–1825, 2017.

Guillaume Bouchard, Théo Trouillon, Julien Perez, and Adrien Gaidon. Online learning to sample. *arXiv preprint arXiv:1506.09016*, 2015.

Jens Kober and Jan Peters. Reinforcement learning in robotics: A survey. In *Reinforcement Learning*, pages 579–610. Springer, 2012.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International Conference on Machine Learning*, pages 1889–1897, 2015a.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015b.

Josiah P Hanna and Peter Stone. Towards a data efficient off-policy policy gradient. 2018.

A Derivation of Derivative of Gradient Objective

Since $\Gamma = \sum_{t=0}^{T-1} w(H) G_t(H) \nabla_\theta \log \pi_\theta(A_t|S_t)$ is an unbiased estimator of g, its mean-squared error $MSE[\Gamma]$ is equal to its variance $Var[\Gamma]$. So minimizing $MSE[\Gamma]$ is equal to minimizing $Var[\Gamma]$. However, Γ is not a scalar-valued random variable in most cases when the parameter of target policy is a vector with more than one component.

$$\Gamma = \begin{bmatrix} \Gamma_0 \\ \Gamma_1 \\ \vdots \\ \Gamma_{N-1} \end{bmatrix} \tag{11}$$

Therefore, the variance of Γ is also a vector which is equal to applying the variance operator to each of its component.

$$Var[\Gamma] = \begin{bmatrix} Var[\Gamma_0] \\ Var[\Gamma_1] \\ \vdots \\ Var[\Gamma_{N-1}] \end{bmatrix} \tag{12}$$

So it is actually an ill-defined way to say "our task is to 'minimize' $Var[\Gamma]$" because $Var[\Gamma]$ is a vector. It doesn't make sense to "minimize" a vector. It makes sense to minimize a function of the vector that outputs a scalar. And this function should reflect the goal of the optimization. That is, optimizing the function also gives us a reasonable optimized vector to some extent.
Here, we define the function to be the summation over all components $\sum_{n=0}^{N-1} Var[\Gamma_n]$.
Expand this objective and we have

$$\begin{aligned} \sum_{n=0}^{N-1} Var[\Gamma_n] &= \sum_{n=0}^{N-1} \left[\mathbb{E}_{H\sim\pi_{\theta_b}}[\Gamma_n^2] - \mathbb{E}_{H\sim\pi_{\theta_b}}[\Gamma_n]^2 \right] \\ &= \sum_{n=0}^{N-1} \left[\mathbb{E}_{H\sim\pi_{\theta_b}}[\Gamma_n^2] - g_n^2 \right] \\ &= \mathbb{E}_{H\sim\pi_{\theta_b}} \left[\sum_{n=0}^{N-1} [\Gamma_n^2] \right] - \sum_{n=0}^{N-1} g_n^2 \\ &= \mathbb{E}_{H\sim\pi_{\theta_b}} \left[\Gamma^2 \right] - g^2 \end{aligned} \tag{13}$$

Equation 13 tells us that we can just treat Γ in $Var[\Gamma]$ as a normal single random variable and proceed the derivation although it is actually a vector of random variables and it involves a self-defined objective upon the variance.
we differentiate $MSE[\Gamma]$ w.r.t. θ_b:

$$\begin{aligned} \nabla_{\theta_b} \text{MSE}[\Gamma] = \nabla_{\theta_b} Var[\Gamma] &= \nabla_{\theta_b} \left(\mathbb{E}_{H\sim\pi_{\theta_b}}[\Gamma^2] - g^2 \right) \\ &= \nabla_{\theta_b} \mathbb{E}_{H\sim\pi_{\theta_b}}[\Gamma^2] = \nabla_{\theta_b} \left(\sum_H \Pr(H|\pi_{\theta_b}) \Gamma^2 \right) \\ &= \sum_H \left[\Pr(H|\pi_{\theta_b}) \nabla_{\theta_b} \Gamma^2 + \Gamma^2 \nabla_{\theta_b} \Pr(H|\pi_{\theta_b}) \right] \end{aligned} \tag{14}$$

Consider the last term $\nabla_{\theta_b}\Pr(H|\pi_{\theta_b})$ first in more detail:

$$\begin{aligned}
\nabla_{\theta_b}\Pr(H|\pi_{\theta_b}) &= \nabla_{\theta_b} \underbrace{\left(\rho_0(S_0)\prod_{t=0}^{T-2} p(S_{t+1}|S_t, A_t)\right)}_{\text{environment transitions}} \underbrace{\left(\prod_{t=0}^{T-1}\pi_{\theta_b}(A_t|S_t)\right)}_{\text{policy selections}} \\
&= \Pr(H)\nabla_{\theta_b}\prod_{t=0}^{T-1}\pi_{\theta_b}(A_t|S_t) \\
&= \Pr(H)\left(\prod_{t=0}^{T-1}\pi_{\theta_b}(A_t|S_t)\right)\left(\sum_{t=0}^{T-1}\frac{\nabla_{\theta_b}\pi_{\theta_b}(A_t|S_t)}{\pi_{\theta_b}(A_t|S_t)}\right) \\
&= \Pr(H|\pi_{\theta_b})\sum_{t=0}^{T-1}\frac{\nabla_{\theta_b}\pi_{\theta_b}(A_t|S_t)}{\pi_{\theta_b}(A_t|S_t)} \\
&= \Pr(H|\pi_{\theta_b})\sum_{t=0}^{T-1}\nabla_{\theta_b}\log\pi_{\theta_b}(A_t|S_t),
\end{aligned} \tag{15}$$

where $\Pr(H)$ denotes the environment transition part (independt of θ_b).
Continuing on equation 14, we have

$$\begin{aligned}
\nabla_{\theta_b}\text{MSE}[\Gamma] &= \sum_H\left[\Pr(H|\pi_{\theta_b})\nabla_{\theta_b}\Gamma^2 + \Gamma^2\Pr(H|\pi_{\theta_b})\sum_{t=0}^{T-1}\nabla_{\theta_b}\log\pi_{\theta_b}(A_t|S_t)\right] \\
&= \sum_H\Pr(H|\pi_{\theta_b})\left[\nabla_{\theta_b}\Gamma^2 + \Gamma^2\sum_{t=0}^{T-1}\nabla_{\theta_b}\log\pi_{\theta_b}(A_t|S_t)\right] \\
&= \mathbb{E}_{H\sim\pi_{\theta_b}}\left[\Gamma^2\sum_{t=0}^{T-1}\nabla_{\theta_b}\log\pi_{\theta_b}(A_t|S_t) + \nabla_{\theta_b}\Gamma^2\right].
\end{aligned} \tag{16}$$

consider the last term $\nabla_{\theta_b}\Gamma^2$ first:

$$\begin{aligned}
\nabla_{\theta_b}\Gamma^2 &= 2\Gamma\nabla_{\theta_b}\Gamma = 2\Gamma\nabla_{\theta_b}\left(\sum_{t=0}^{T-1} w(H)G_t(H)\nabla_\theta\log\pi_\theta(A_t|S_t)\right) \\
&= 2\Gamma\sum_{t=0}^{T-1}\nabla_{\theta_b}\left(\frac{\Pr(H|\pi_\theta)}{\Pr(H|\pi_{\theta_b})}G_t(H)\nabla_\theta\log\pi_\theta(A_t|S_t)\right) \\
&= -2\Gamma\sum_{t=0}^{T-1}\left[\Pr(H|\pi_\theta)G_t(H)\nabla_\theta\log\pi_\theta(A_t|S_t)\frac{\nabla_{\theta_b}\Pr(H|\pi_{\theta_b})}{\Pr(H|\pi_{\theta_b})^2}\right] \\
&= -2\Gamma\sum_{t=0}^{T-1}\left[w(H)G_t(H)\nabla_\theta\log\pi_\theta(A_t|S_t)\left(\sum_{t=0}^{T-1}\nabla_{\theta_b}\log\pi_{\theta_b}(A_t|S_t)\right)\right] \\
&= -2\Gamma\left(\sum_{t=0}^{T-1}\nabla_{\theta_b}\log\pi_{\theta_b}(A_t|S_t)\right)\sum_{t=0}^{T-1} w(H)G_t(H)\nabla_\theta\log\pi_\theta(A_t|S_t) \\
&= -2\Gamma^2\sum_{t=0}^{T-1}\nabla_{\theta_b}\log\pi_{\theta_b}(A_t|S_t)
\end{aligned} \tag{17}$$

Continuing on equation 16, we have

$$\nabla_{\theta_b}\text{MSE}[\Gamma] = \mathbb{E}_{H\sim\pi_{\theta_b}}\left[-\Gamma^2\sum_{t=0}^{T-1}\nabla_{\theta_b}\log\pi_{\theta_b}(A_t|S_t)\right]. \tag{18}$$

B Derivation of Derivative of Advantage Objective

In policy evaluation, a reasonable way to define the 'goodness' of a policy is to use the expected return of the starting state $\mathbb{E}_{H\sim\theta}[G_0(H)]$. And an off-policy version to approach this evaluation problem is to use the expected re-weighted importance sampling return $\mathbb{E}_{H\sim\theta_b}[w(H)G_0(H)]$ as our measurement. [Hanna et al., 2017] proposed an novel way to search for this approach to find the "optimal" behavior policy for policy evaluation. The optimal behavior policy is defined to be the one with minimal mean squared error of re-weighted importance sampling return estimator. Formally,

$$\text{find } \theta_b^\star = \underset{\theta_b}{\text{argmin}}\, MSE\left[w(H)G_0(H)\right].$$

In the paper, they gave a theorem that states the derivative of $MSE[w(H)G_0(H)]$ is

$$\nabla_{\theta_b}\text{MSE}[w(H)G_0(H)] = \mathbb{E}_{H\sim\pi_{\theta_b}}\left[-(w(H)G_0(H))^2\sum_{t=0}^{T-1}\nabla_{\theta_b}log\pi_{\theta_b}(A_t|S_t)\right], \tag{19}$$

which we also directly used in our G_0 objective (equation 9). Recall that, for advantage objective, our task is to optimize $MSE[\sum_{t=0}^{T-1}w(H)G_t(H)]$. If we substitute our objective $\sum_{t=0}^{T-1}w(H)G_t(H)$ for $w(H)G_0(H)$ in equation 19, then the results immediately follows

$$\nabla_{\theta_b}\text{MSE}\left[\sum_{t=0}^{T-1}w(H)G_t(H)\right] = \mathbb{E}_{H\sim\pi_{\theta_b}}\left[-\left(\sum_{t=0}^{T-1}w(H)G_t(H)\right)^2\sum_{t=0}^{T-1}\nabla_{\theta_b}\log\pi_{\theta_b}(A_t|S_t)\right]. \tag{20}$$