REWARD ESTIMATION FOR VARIANCE REDUCTION IN DEEP REINFORCEMENT LEARNING

Joshua Romoff[1,2], **Alexandre Piché**[3], **Peter Henderson**[1], **Vincent Francois-Lavet**[1], **& Joelle Pineau**[1,2]
[1] McGill University, Montréal, Québec, Canada
[2] Facebook, Montréal, Québec, Canada
[3] MILA, Université de Montréal, Québec, Canada

ABSTRACT

In reinforcement learning (RL), stochastic environments can make learning a policy difficult due to high degrees of variance. As such, variance reduction methods have been investigated in other works, such as advantage estimation and control-variates estimation. Here, we propose a simple method for reducing variance, for general RL methods, by learning a reward estimator to train the value function. This results in theoretical reductions in variance in the tabular case, as well as empirical improvements in both the function approximation and tabular settings in environments where rewards are stochastic. To do so, we use a modified version of Advantage Actor Critic (A2C) on variations of Atari games.

1 INTRODUCTION

While reinforcement learning (RL) has had great successes in solving sequential decision-making tasks, high variance of some methods can make learning difficult when environments or rewards are strongly stochastic (Verbeeck et al., 2007; Henderson et al., 2018b). Several methods have been used to reduce variance, sometimes at the cost of bias. This includes generalized advantage estimation (Schulman et al., 2016), control-variates optimization (Grathwohl et al., 2017), updating the target policy via the expectation of its actions (Ciosek & Whiteson, 2018; Asadi et al., 2017), and updating the value function via the posterior mean of an estimated uncertain distribution (Henderson et al., 2017). Here, we propose a method for variance reduction by using a direct estimate of rewards $\hat{R}(s_t)$ to update the discounted value function $V_\gamma^\pi(s_t)$, rather than the sampled rewards. In the tabular case, this corresponds to using the sampled mean, while in the function approximation case this corresponds to learning a one-step value function $V_{\gamma=0}^\pi(s_t)$. We prove that this method results in theoretical variance reductions in the tabular case. We also show that it corresponds to performance gains in the tabular and function approximation settings in situations where rewards are highly stochastic.

1.1 BACKGROUND

We formulate our method with the fully observable Markov Decision Process (MDP). In an MDP, an agent can take an action a_t based on its current state s_t and receive a reward r_t, before transitioning to the next state of the MDP s_{t+1}. We focus on the discounted MDP case, where an agent tries to maximize the cumulative discounted reward $V_\gamma^\pi(s) = \left[\sum_{t=0}^{T} \gamma^t r_t | s_0 = s, \pi\right]$, also known as the discounted value of a policy. It is common to learn a value estimate of the current policy via temporal difference (TD) learning, where the current estimate of the value function is used to bootstrap the next estimate according to the Bellman equation $V_\gamma^\pi(s_t) = r_t + \gamma V_\gamma^\pi(s_{t+1})$, via the loss: $\left(r_t + \gamma V_\gamma^\pi(s_{t+1}) - V_\gamma^\pi(s_t)\right)^2$. In the case of Advantage Actor Critic (A2C), the synchronous version of Asynchronous Advantage Actor Critic (A3C) (Mnih et al., 2016), a stochastic parameterized policy (actor, $\pi_\theta(a|s)$) is learned from this value estimator via the TD error. That is, the actor loss becomes: $-\log \pi(a|s) \left(r_t + \gamma V_\gamma^\pi(s_{t+1}) - V_\gamma^\pi(s_t)\right)$.

2 REWARD ESTIMATION

To reduce variance in the value function updates, we introduce an estimator for the reward at a given state $\hat{R}(s_t)$. In the function approximation case, learning this reward estimator becomes a simple regression problem: $\left(r_t - \hat{R}(s_t)\right)^2$. We then use this reward estimator in the TD (Sutton, 1988) update of the value function, rather than the sample reward: $\left(\hat{R}(s_t) + \gamma V_\gamma^\pi(s_{t+1}) - V_\gamma^\pi(s_t)\right)^2$. Intuitively, where rewards are highly stochastic due to environment stochasticity or due to some stochastic process in reward assignment, this estimation will reduce the variance propagated to the value function. We use the A2C algorithm for our function approximation experiments with this modified update. Diagrams of this process can be seen in Figure 1. For function approximation experiments, we build on the PyTorch implementation of Kostrikov (2018).

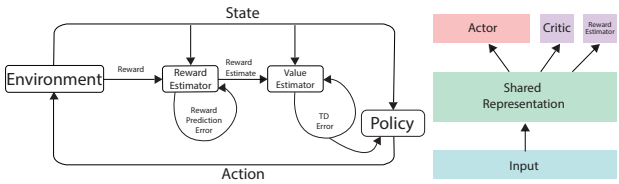

Figure 1: Left: The actor-critic update process with the reward estimator. Right: The network architecture used for Atari experiments, where there are several shared representation layers between the reward predictor, value function, and policy.

Variance Reduction in Tabular Settings: To determine whether our method for using a reward approximator reduces variance theoretically, we examine the tabular case. In this setting, we use the sample mean for the reward estimator: $\hat{R}_t(s) = \left[\frac{1}{N}\sum_i r_i \middle| s_i = s\right]$. That is, given N observed reward samples at a given state s, we determine the mean of those rewards. In this scenario, the sample mean is an unbiased estimator.

First, we determine the variance of the standard discounted Bellman equation (Bellman, 1957): $G_t^\gamma = R_t(s_t) + \gamma V_\pi^\gamma(s_{t+1})$. The variance of this Bellman estimate is: $\text{Var}\left[G_t^\gamma\right] = \text{Var}\left[R_t(s_t)\right] + \text{Var}\left[\gamma V_\pi^\gamma(s_{t+1})\right] + 2\text{Cov}\left[R_t, \gamma V_\pi^\gamma(s_{t+1})\right]$. We note that the right most term is equal to 0, due to the properties of the expectation in the Markovian setting. If we instead use an approximator for the reward, the Bellman equation becomes: $\hat{G}_t^\gamma = \hat{R}_t(s_t) + \gamma V_\pi^\gamma(s_{t+1})$. Similarly, the variance becomes: $\text{Var}\left[\hat{G}_t^\gamma\right] = \text{Var}\left[\hat{R}_t(s_t)\right] + \text{Var}\left[\gamma V_\pi^\gamma(s_{t+1})\right] + 2\text{Cov}\left[\hat{R}_t, \gamma V_\pi^\gamma(s_{t+1})\right]$. We note that the right most term in this case is also 0. Moreover, since we assume our approximation in the tabular case is simply the sample mean, we assume that $\text{Var}\left[\hat{R}_t(s_t)\right] = \frac{1}{N}\text{Var}\left[R_t(s_t)\right]$. Thus, we have that: $\text{Var}\left[\hat{G}_t^\gamma\right] = \frac{1}{N}\text{Var}\left[R_t(s_t)\right] + \text{Var}\left[\gamma V_\pi^\gamma(s_{t+1})\right]$. By re-arranging terms between the two variance estimates, we arrive at the inequality: $\frac{1}{N}\text{Var}\left[R_t(s_t)\right] \leq \text{Var}\left[R_t(s_t)\right]$. And thus the inequality: $\text{Var}\left[\hat{G}_t^\gamma\right] \leq \text{Var}\left[G_t^\gamma\right], \forall N \geq 1$. Therefore, by using the empirical mean of the rewards in a tabular setting, we can reduce variance. The intuitive benefit of this becomes clear in settings with highly stochastic rewards. That is, in a given state a reward may be provided with some probability $P(r)$ and otherwise is 0. In such a case, the error will propagate through longer MDP chains, whereas using the empirical mean will provide a more stable estimate, as will be demonstrated in subsequent experimental sections.

3 EXPERIMENTS

Tabular Experiments: We first investigate the tabular case. We construct a 5 state MDP as seen for value learning (an extended 10 state MDP can be seen in Appendix A). The MDPs contain deterministic transitions from left to right in the states, and the agent follows a fixed policy moving to the right and terminates on reaching the farthest state to the right. At each state it receives a stochastic reward of 1, 2, or 5 with a fixed probability of 0.5. The value function is updated

via the temporal difference (TD) error for 100 episodes. We measure the robustness to variance by evaluating the root mean squared error (RMSE) of the value function across the 100 episodes. As is seen in Figure 2, when using the reward estimator, the agent is able to learn more accurate representations of the value function even at high learning rates. This indicates a parity with the aforementioned theoretical variance reduction.

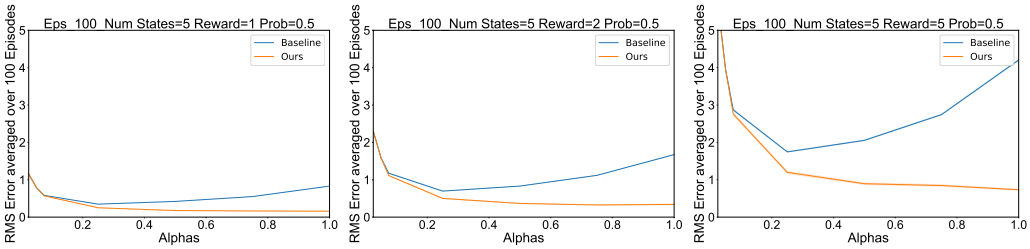

Figure 2: Tabular experiments with a 5-state MDP. In all cases, rewards are assigned with probability 0.5 and, set to 0 otherwise (rewards of $+1, +2, +5$, from left to right). The x-axis demonstrates various learning rates for the TD-update. We report the average RMSE over the first 100 episodes of learning - lower is better.

Atari Experiments: For evaluating our method with function approximation, we use 5 Atari games from the Arcade Learning Environment (ALE) (Bellemare et al., 2013). We experiment with adding a small amount of Gaussian noise at each time step to the reward signal (after clipping it to $[-1, 1]$). We use the exact same hyper parameters used in OpenAI's Baselines implementation (Dhariwal et al., 2017), but append an extra reward predictor head and use it to train our critic as described in section 2. We compare our approach to the standard A2C algorithm, as well as A2C with our architecture but without using the estimated reward in place of the true reward for the critic. The latter can be seen as the reward prediction auxiliary task described in Jaderberg et al. (2017). We report results, averaged over 3 random seeds, for rewards with and without added Gaussian noise (with mean 0 and variance 0.3) in Table 3. For the full training curves for different amounts of noise, see Appendix B. Across all games we see that our proposed method performs relatively better once noise has been introduced. This is especially true in Breakout, where we obtain 4x the performance of the baselines in the noisy environment. In Qbert and Seaquest, however, we see that our agent performs considerably worse than the baselines before noise is introduced. We posit that the cause of this to be poor reward prediction in these tasks.

Environment	No Reward Noise (% of Baseline)	With Reward Noise (% of Baseline)
SpaceInvaders	82.06	140.00
Breakout	100.00	400.00
BeamRider	101.63	124.87
Qbert	53.52	72.61
Seaquest	36.06	85.86
Average	74.65	164.67

Table 1: Comparison of the average episode reward over 10M steps of training between our approach to the best of both baselines (A2C and A2C with the reward prediction auxiliary task). The results are the average over 3 runs

4 Conclusion

Overall, we have demonstrated that by replacing the sampled reward with reward estimator in the value function update, we can reduce variance and improve learning overall. We expect that this methodology can be particularly useful in settings with high degrees of uncertainty. To address the short-comings in some Atari games, future work may involve learning a distributional reward estimator as in (Bellemare et al., 2017), off-policy experience replay for the reward estimator, or learn options for reward estimators as in (Henderson et al., 2018a).

REFERENCES

Kavosh Asadi, Cameron Allen, Melrose Roderick, Abdel-rahman Mohamed, George Konidaris, and Michael Littman. Mean actor critic. *arXiv preprint arXiv:1709.00503*, 2017.

M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

Marc G. Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *ICML*, volume 70 of *Proceedings of Machine Learning Research*, pp. 449–458. PMLR, 2017.

Richard Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, USA, 1 edition, 1957. URL http://books.google.com/books?id=fyVtp3EMxasC&pg=PR5&dq=dynamic+programming+richard+e+bellman&client=firefox-a#v=onepage&q=dynamic%20programming%20richard%20e%20bellman&f=false.

Kamil Ciosek and Shimon Whiteson. Expected policy gradients. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI)*, 2018.

Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, and Yuhuai Wu. Openai baselines. https://github.com/openai/baselines, 2017.

Will Grathwohl, Dami Choi, Yuhuai Wu, Geoff Roeder, and David Duvenaud. Backpropagation through the void: Optimizing control variates for black-box gradient estimation. *arXiv preprint arXiv:1711.00123*, 2017.

Peter Henderson, Thang Doan, Riashat Islam, and David Meger. Bayesian policy gradients via alpha divergence dropout inference. *Bayesian Deep Learning Workshop at NIPS*, 2017.

Peter Henderson, Wei-Di Chang, Pierre-Luc Bacon, David Meger, Joelle Pineau, and Doina Precup. OptionGAN: Learning Joint Reward-Policy Options using Generative Adversarial Inverse Reinforcement Learning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI)*, 2018a.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep Reinforcement Learning that Matters. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI)*, 2018b.

Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z. Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. *International Conference on Learning Representations*, 2017.

Ilya Kostrikov. Pytorch implementations of reinforcement learning algorithms. https://github.com/ikostrikov/pytorch-a2c-ppo-acktr, 2018.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, pp. 1928–1937, 2016.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016.

Richard S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3 (1):9–44, Aug 1988. ISSN 1573-0565. doi: 10.1023/A:1022633531479. URL https://doi.org/10.1023/A:1022633531479.

Harm Van Seijen, Hado Van Hasselt, Shimon Whiteson, and Marco Wiering. A theoretical and empirical analysis of expected sarsa. In *Adaptive Dynamic Programming and Reinforcement Learning, 2009. ADPRL'09. IEEE Symposium on*, pp. 177–184. IEEE, 2009.

Katja Verbeeck, Ann Nowé, Johan Parent, and Karl Tuyls. Exploring selfish reinforcement learning in repeated games with stochastic rewards. *Autonomous Agents and Multi-Agent Systems*, 14(3): 239–269, 2007.

A TABULAR MDP EXPERIMENT

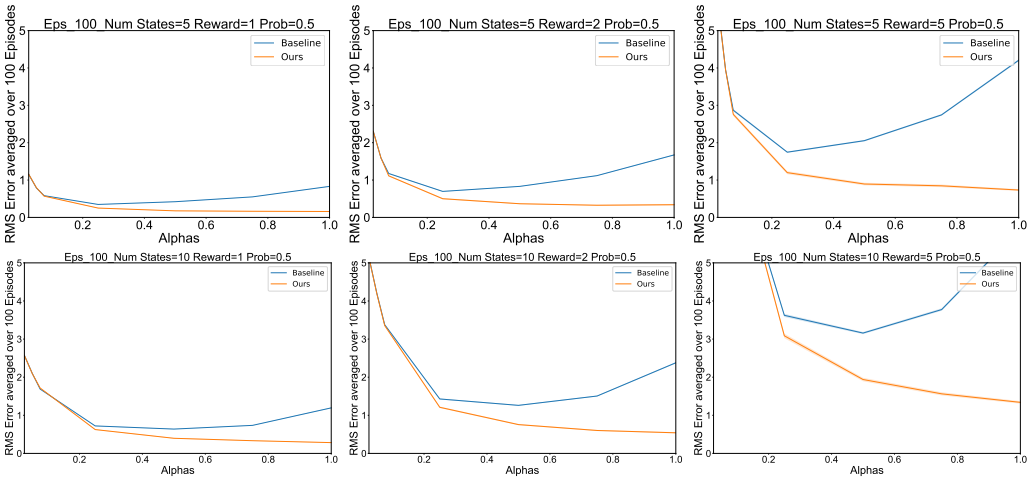

Figure 3: Tabular experiments with a 5-state MDP (top row) and a 10-state MDP (bottom row), with varying reward assignments at each states. In all cases, rewards are assigned with probability 0.5 and, set to 0 otherwise (rewards of $+1, +2, +5$, from left to right). The x-axis demonstrates various learning rates for the TD-update, as seen in similar variance analysis experiments (Van Seijen et al., 2009). As can be seen, using the sample mean in MDPs with stochastic processes greatly reduces the variance, allowing for higher learning rates to be used.

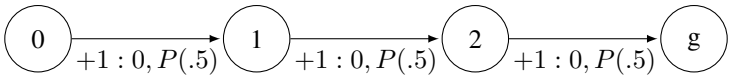

Figure 4: Depiction of the sample Markov Decision Process (MDP) used for the tabular case experiments.

B ATARI EXPERIMENTS

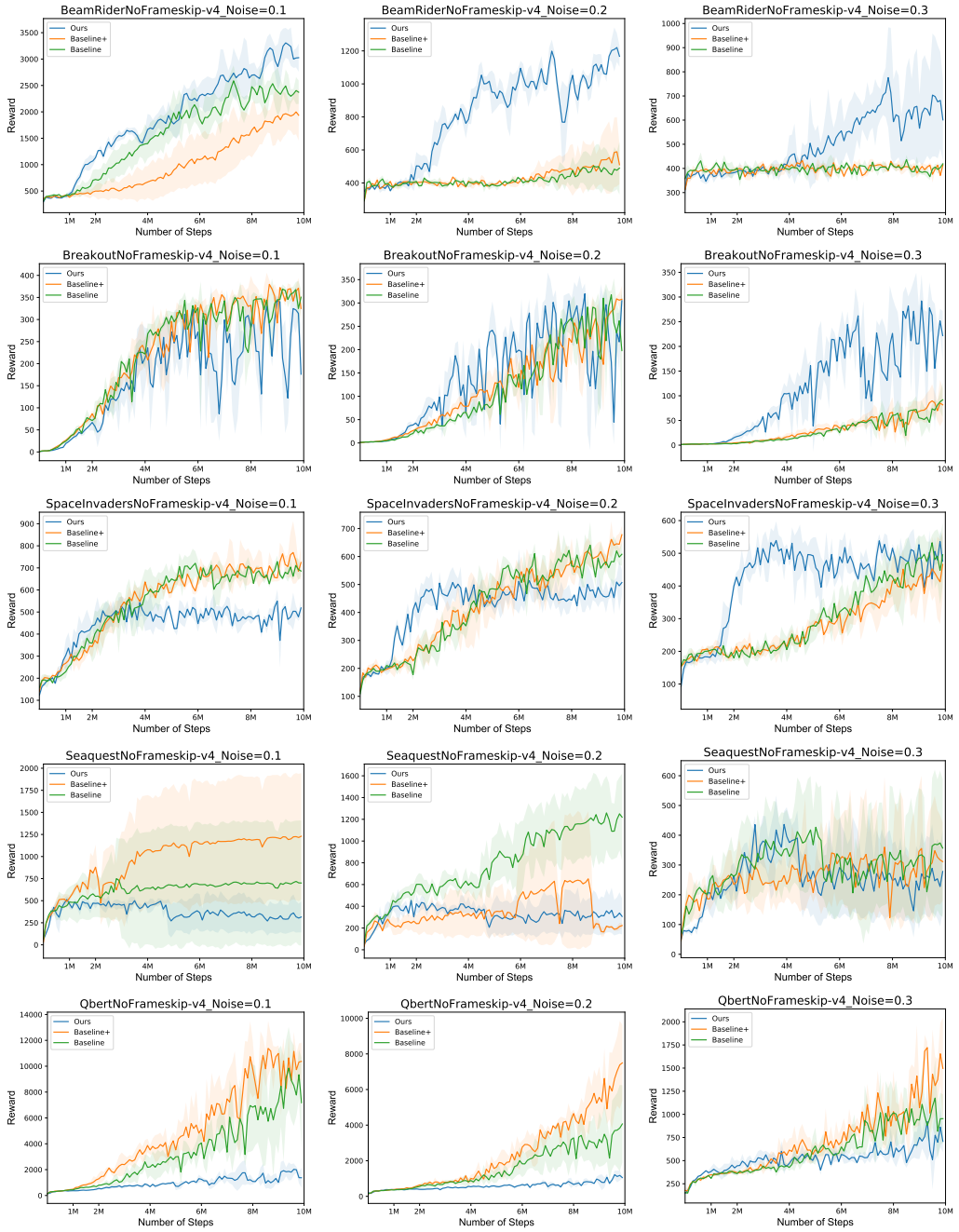

Figure 5: Full learning curves for 3 runs for five Atari Games over 10M training steps (corresponding with 40M raw frames). From top to bottom: Beam Rider, Breakout, Space Invaders, Seaquest, Qbert. We compare 4 different variance levels that each correspond to adding Gaussian noise centered at zero with the labelled variance. From left to right we have: 0.1, 0.2, 0.3. Our proposed method is labelled Ours, while the two baselines, A2C and A2C with the reward prediction auxiliary task, are labelled Baseline and Baseline+ respectively.