Batch and Sequential Policy Optimization with Doubly Robust Objectives

Alex Lewandowski
Department of Computing Science
University of Alberta
Edmonton, Alberta
alex3@ualberta.ca

Dale Schuurmans
Department of Computing Science
University of Alberta
Edmonton, Alberta
daes@ualberta.ca

Abstract

In this paper, we investigate a policy's ability to exploit batches of experience when trained on different objectives. We introduce objectives based on doubly robust estimation that are orders of magnitude better in their ability to exploit batches of experience. In addition, we compare these objectives in the both batch and sequential settings, with one and multiple steps of policy iteration respectively. We find that batch policy iteration introduces difficulties distinct from the on-policy setting where a policy only updates with a single trajectory before generating a new trajectory. Despite exploration receiving considerable attention, exploitation in the context of optimization can also be quite challenging for current objectives.

1 Introduction

A reinforcement learning agent must simultaneously choose its actions to generate data and use that data to optimize an objective towards actions that provide high return. This coupling between optimization and action selection is alluring, but even the simpler problem of action selection, via exploration and exploitation, is not well understood. The focus of this paper is exploitation, which we investigate by looking at different objectives in batch reinforcement learning.

Suppose that, in a contextual bandit problem, we are given complete feedback as if it were a supervised learning problem. In this case, we would observe the rewards for actions not taken. Using this information, we could form a target for our policy using all action-values – not just for the action taken by the behavior policy. This would then bypass the need for a REINFORCE estimator [1] since we would be able to optimize through the expectation. The key problem is that we do not have an oracle to provide us with information on actions not taken. In reinforcement learning, a separate function approximator is often used to estimate the action-values [2]. However, alternating between updates of two separate function approximators is a saddle-point problem that can cause instabilities. While some work has investigated novel objectives in the contextual bandit setting [3, 4], we consider new objectives for reinforcement learning.

In this paper, we study how doubly robust estimation with different choices of objective function affect an agent's ability to learn. A key aspect of this approach is the unification of actor and critic in function approximation. While actor-critic models have separate networks for the policy and for the action-value function, we use only a single shared network. In doing so, we leverage both generalization and auxiliary tasks by having one network predict both action-value functions and the policy distribution. To benchmark our objectives we look at off-policy problems in the cartpole environment [5, 6]. In particular, we investigate learning from batches of experience, with and without sequential policy improvement, as well as learning from pretrained policies.

Optimization Foundations for Reinforcement Learning Workshop at NeurIPS 2019, Vancouver, Canada.

2 Off-policy policy optimization

Due to the policy gradient theorem [7], expected return is often the quantity that policy gradient methods optimize. However, recent work suggests that the optimization landscape induced by the expected return is not well suited to gradient based optimization [8, 4]. This stands in contrast to supervised learning, which can afford to use cross-entropy as a surrogate for misclassification cost to allow for robust convergence of neural networks.

We first motivate new objectives from the perspective of batch reinforcement learning, where policy improvement (and hence the policy gradient theorem) is not strictly needed. Current work in deep reinforcement learning constitutes a blend of batch and online reinforcement learning. While agents act online, their experience is stored in an experience replay buffer to be used at a later time for updating. As a result, there is a lack of clear separation between success in the batch off-policy setting and the online on-policy setting.

To begin, we consider the softmax policy over preferences for each discrete action,

$$\pi_\theta(A = a|s_t) = \frac{e^{Q_\theta(s_t,a)}}{\sum_a e^{Q_\theta(s_t,a)}}.$$

Where Q_θ scores the relative preference of certain actions and is not necessarily equal to the action-value function $Q_\pi(s_t, a)$.

While we are optimizing a soft-max policy, the experience (or data) that we are using comes from another behavior policy $\beta(a|s)$. Using experience from another policy is referred to as off-policy learning, and policy-gradient methods leverage importance sampling for an unbiased estimate of some quantity X_t,

$$\mathbb{E}_\pi[X_t] = \mathbb{E}_\beta[\frac{\pi(a_t|s_t)}{\beta(a_t|s_t)}X_t].$$

Implicitly, importance sampling ignores potential generalization across actions not taken. In what follows, we address this problem by using doubly robust estimates of the return.

2.1 Estimating the return

To properly measure the difference between the policy's distribution and the estimates of the return, we must maintain estimates of the action-value. The first issue in batch reinforcement learning is that the rewards (and hence returns) are unobserved for actions not taken. At time t, we have an estimate of $Q(s_t, a_t)$ by the Monte Carlo return $G_t = \sum_{i=t}^{T} r_i$. For other actions $a \neq a_t$, we must estimate $Q(s_t, a)$ and we do so using the preferences of our policy. In particular, we use an idea similar to the doubly robust estimator from statistics and policy evaluation [9, 10].

$$\hat{Q}_{DR}(s_t, a) = Q_\theta(s_t, a) + \mathbf{1}_{a=a^t}\frac{\rho_{1:(t-1)}G_t - Q_\theta(s_t, a)}{\beta(a_t|s_t)}.$$

where $\rho_{1:(t-1)} = \prod_{t'=1}^{t-1}\frac{\pi(a_{t'}|s_{t'})}{\beta(a_{t'}|s_{t'})}$ is the importance correction. That is, we use the preferences that output from our policy as estimates of the action-value for all actions not taken. However, we reweigh the action-value for actions taken proportional to the proposal probability under the behavior policy $\beta(a_t|a_t)$. Furthermore, we can write a distribution over Q using the softmax function,

$$p_{\hat{Q}}(A = a|s_t) = \frac{e^{\hat{Q}_{DR}(s_t,a)}}{\sum_a e^{\hat{Q}_{DR}(s_t,a)}}.$$

2.2 Imputed policy objectives

Now that we have estimates of the return using doubly robust imputation, we can use this information in our objectives. Whenever we would normally compare the policy with the action-value of the action take, we will now compare the entire distribution. By requiring the function approximator to satisfy the imputation and requirement, we are requiring the function approximator to generalize. Then, what objective function should we use? We first look to the reinforcement learning literature. First, we observe that the recursive definition of doubly robust imputation [10],

$$V_{DR}^t = V_\theta(s_t) + \rho_t(r_t + V_{DR}^{t+1} - Q_\theta(s_t, a_t)).$$

where $V_\theta(s_t) = \sum_a \pi_\theta(a|s_t) Q_\theta(s_t, a)$. We can unroll this objective to obtain the Monte-Carlo objective,

$$J(\theta) = \sum_{t=1}^{T} \sum_{t'=t}^{T} \rho_{1:t}(r_t + V_\theta(s_t) - Q_\theta(s_t, a_t)).$$

Next, we look at the supervised learning literature. One common method is ***KL Minimization*** $KL(P||\,Q)$ which minimizes the distance between two distributions P and Q. If we are learning a policy π_θ, then the corresponding 'label" distribution is the softmax over estimated returns $p_{\hat{Q}}(a|s_t) =$ softmax$(\hat{Q}_{DR})$. In supervised learning, the labels are provided and fixed. This is not the case for doubly robust imputation, since the function approximator is used to estimate the return for actions not taken. If the labels were indeed fixed, we would similarly prefer the forward KL divergence in supervised learning since it would simplify to the cross entropy $\sum_a p_{\hat{Q}}(a) \log \pi(a)$. Note that this is a hybrid between the REINFORCE estimator [11] and all-action policy gradient [12].

As a result, we instead use the backward KL divergence $KL(\pi_\theta||\,p_{\hat{Q}})$. In the actor-critic literature, where $\hat{Q}$ is estimated using a critic with separate weights, this is sometimes referred to as entropy regularized expected return. To see why, we momentarily ignore the outer summation over time and the conditional on state s_t,

$$KL(\pi_\theta||\,p_{\hat{Q}}) = \sum_a [\pi_\theta(a) \log \pi_\theta(a) - \pi_\theta(a) \log p_{\hat{Q}}(a)].$$

Where $H = \sum_a \pi_\theta(a) \log \pi_\theta(a)$ is the entropy of the distribution π_θ. When this term is added to an objective, it biases the policy towards a uniform distribution.

To use these objectives for reinforcement learning, we can consider the sum of KL divergences at each time step over the horizon T. This is justified by the fact that the policy is conditionally independent given the state, and we have that

$$KL\left[\pi_\theta(\tau)||\,p_{\hat{Q}}(\tau))\right] = \sum_{t=1}^{T} KL\left[\pi_\theta(a|s_t)||\,p_{\hat{Q}}(a|s_t)\right].$$

Simplifying this further, we fix and omit s_t for brevity. The doubly robust backward KL divergence can be decomposed to find that,

$$KL\left[\pi_\theta(a))||\,p_{\hat{Q}}(a))\right] = -\pi_\theta(a_t)\frac{\rho_{1:(t-1)}G_t - q_\theta(a_t)}{\beta(a_t)} + \sum_a [\pi_\theta(a) \log \pi_\theta(a) + e^{\hat{Q}_{DR}(a)} - \pi_\theta(a) q_\theta(a)].$$

Hence, the doubly robust backward KL objective decomposes to expected return with state-action baseline with an additional entropy term, normalizing constant and value term. Interestingly, the normalizing term is dependent on θ and will contribute to the gradient of the objective.

It is unclear whether the entropy regularization component is important since $\hat{Q}_{DR}$ depends on the policy parameters. We now define a new objective without this term,

$$J(\theta) = -\sum_a \pi_\theta(a) \log p_{\hat{Q}}(a) = -\pi_\theta(a_t)\frac{\rho_{1:(t-1)}G_t - q_\theta(a_t)}{\beta(a_t)} + \sum_a [e^{\hat{Q}_{DR}(a)} - \pi_\theta(a) q_\theta(a)]$$

which is just the cross entropy between the policy and the doubly robust imputation of return.

3 Experiments

We compare our imputed objectives (Expected Return - Doubly Robust, Backward KL, Backward KL no entropy) against classical objectives such as Expected return with and without importance correction on the cartpole environment [5, 6]. The start state is randomized and the maximum possible return is 200, where the episode terminates. While a simple domain, the series of off-policy and batch settings that we propose remains a challenging problem in the literature [13].

For all of our experiments, we average over 30 different seeds and, for each seed, the policy is evaluated multiple times on the environment. The shaded region corresponds to one standard error

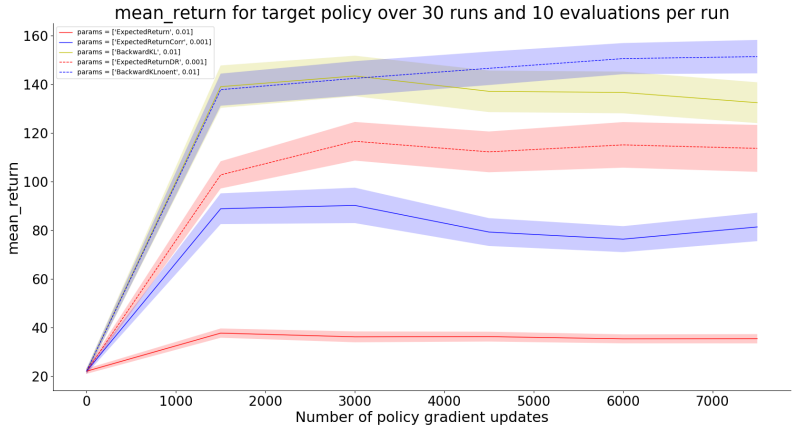

Figure 1: Batch policy optimization with 200 trajectories sampled from a uniformly random behavior policy. KL-based objectives perform orders of magnitude better than conventional algorithms. Doubly robust return also performs better than conventional importance sampling

around the mean over the 30 different runs. The function approximator for the softmax policy is a neural network with a single hidden layer of 32 neurons and relu activation functions. We use the Adam optimizer [14] and all learning rates are individually tuned for each algorithm, between $\alpha = 10^{-i}$ for $i \in [1, 2, 3, 4, 5, 6]$. Lastly, action selection is stochastically sampled from the soft-max distribution.

3.1 Batch and sequential reinforcement learning

For the batch reinforcement learning problem, the agent is provided 200 trajectories from a uniformly random policy (mean episode length of 22). Each objective is then optimized by taking a minibatch of 4 trajectories. For each seed, the agent is evaluated on the environment 10 for every 15 epochs (for a total of 150 epochs of training). Referring to Figure 1, we see that both KL objectives are able to produce excellent policies without ever directly interacting with the environment. Unsurprisingly, uncorrected expected return is unable to learn anything from the batch of trajectories. However, even importance sampling is unable to match the performance of the doubly robust objectives.

As hypothesized in Section 2.2, we see very little difference between the backward KL divergence with and without entropy. This seems to suggest that the entropy term is not the main contributor to the success of the doubly robust backward KL objective.

3.2 Batch policy iteration and on-policy learning

In batch policy iteration, agents are initially trained on 50 trajectories sampled from a uniformly random policy. Afterwards, an optimized policy is extracted and used to generate a new batch of 50 trajectories. Referring to Figure 2, we find that this degrades performance of the KL-based agents relative to doubly robust expected return. Still, the imputed objectives are still significantly better than the classical expected return objectives.

Contrasting this with the on-policy performance in Figure 3 (left), we see that all agents perform roughly equally by the end of the last episode. Early on, doubly robust backward KL performs significantly better however this performance gap degrades. More interestingly however, the imputed objectives are unable to take advantage of more epochs for a single trajectory as seen in Figure 3 (right).

3.3 Batch learning with pretrained behavior policy

The difference between the on-policy and batch policy iteration setting is striking, but confounded by factors involving the sudden change in data distribution and its effect on the optimizer's learning rate.

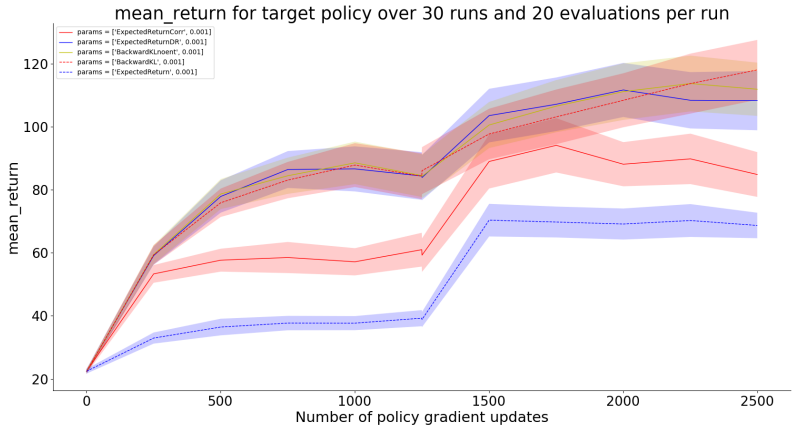

Figure 2: Batch off-policy optimization with one step of policy iteration. Initially, 50 trajectories are sampled from a uniformly random behavior policy. After 100 epochs, 50 new trajectories are sampled from the optimized policy.

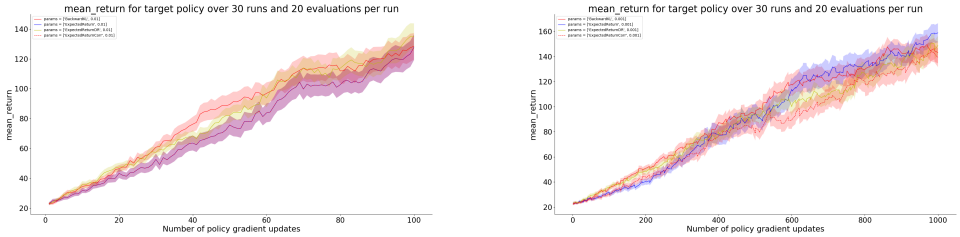

Figure 3: Policy optimization with a single trajectory per update. Doubly robust Backward KL performs statistically significantly better early on, but eventually all objectives yield similar policies. Left: 1 epoch. Left: 10 epochs.

To control for this, we pretrain a policy with an on-policy expected return objective to two levels of performance (50 episodes: mean return of 70, 100 episodes: mean return of 140). All objectives are provided this policy and a batch of 50 trajectories to optimize it further.

In Figure 4 (top), the policy and its trajectories are initially from an expected return objective trained to achieve a mean return of 70. The initial performance is already quite good, however the imputed objectives are able to optimize it further. As expected, on-policy expected return struggles to improve performance past the initial value. The same is true in Figure 4 (bottom), except that the degree of improvement is less. Interestingly, backward KL is able to improve the policy more than imputed expected return.

4 Related Work

Besides pioneering work on doubly robust policy evaluation [10, 15], there has been some work using doubly robust policy optimization to contextual bandits [9, 4]. Since we consider the sequential setting of reinforcement learning, the problem is much harder.

The objective function for the policy network in soft actor critic [16] is similar to the backward KL divergence objective that we investigate in this paper. However, we use a single network to parameterize the action-value function and the policy. Hence, the learning dynamics are quite different in that targets are constantly moving. Moreover, our objective function is not quite an actor critic even if we used separate networks for each function.

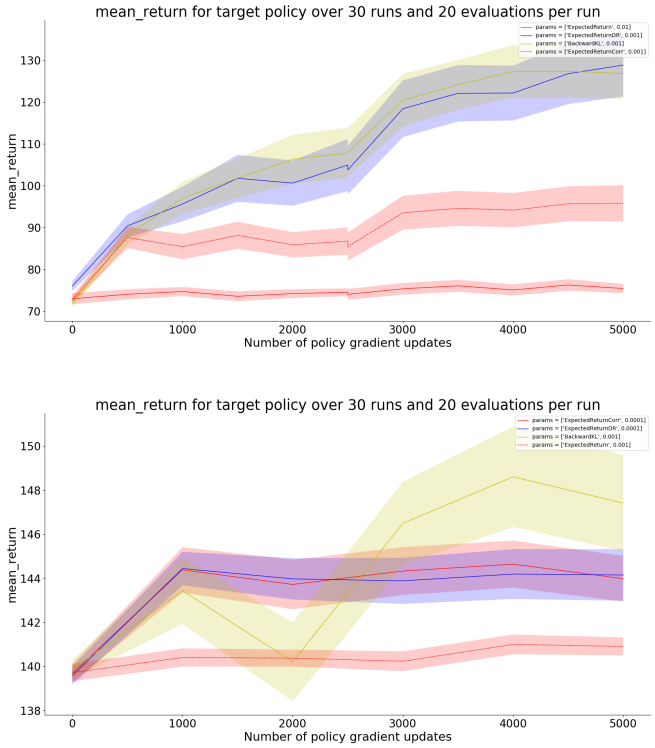

Figure 4: Batch off-policy optimization with a pretrained policy. Top: Two steps of policy iteration with an initial policy that achieves a mean return of 70. Bottom: One step of policy iteration with an initial policy that achieves a mean return of 140.

Lastly, our treatment of the return as distribution is distinct from distributional RL [17]. While their work considers the distribution of return of the state-action pair $Z(S, A)$, while we consider the conditional distribution $Z(A|s)$. Put another way, distributional RL considers returns random even for a fixed action while this works considers randomness in returns only through the randomness in the policy.

5 Discussion

In this paper we explored KL minimization objectives for reinforcement learning using doubly robust imputation. We showed that a single network can learn to act and predict return from uniformly random behavior on the cartpole environment. What is surprising, is that these off-policy methods are equally effective in the on-policy setting. This novel formulation does not aim to decrease variance. Instead, this objective leverages generalization in the optimization process by forcing the policy to learn how to impute action-values.

Three issues remain and are outside the scope of this paper. First, how can we estimate action-values for actions not taken when actions are continuous? Second, what inductive bias does the imputation scheme produce? And last, what is the interaction between objective function and imputation scheme? Indeed, doubly robust imputation will not work with continuous actions because the event that we observe a particular action has measure zero. We will then have to change the imputation in such a way that still produces the same inductive bias. But it is unclear if the same objective functions will be the best performers.

This new perspective on learning objectives sheds some light on issues in optimization for RL. Namely, the problem of behaving optimal on multiple streams of data can conflict with learning on newer streams of data. As a result, our work suggests a trade-off between how much you can learn through temporal interactions and through a supervised learning approach.

References

[1] Ronald J Williams and Jing Peng. Function optimization using connectionist reinforcement learning algorithms. *Connection Science*, 3(3):241–268, 1991.

[2] Richard S. Sutton and Andrew G. Barto. *Reinforcement learning - an introduction*. Adaptive computation and machine learning. MIT Press, 2018.

[3] Thorsten Joachims, Adith Swaminathan, and Maarten de Rijke. Deep learning with logged bandit feedback. 2018.

[4] Minmin Chen, Ramki Gummadi, Chris Harris, and Dale Schuurmans. Surrogate losses for batch policy optimization in one-step decision making, 2019.

[5] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

[6] Andrew G Barto, Richard S Sutton, and Charles W Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE transactions on systems, man, and cybernetics*, (5):834–846, 1983.

[7] 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.

[8] Zafarali Ahmed, Nicolas Le Roux, Mohammad Norouzi, and Dale Schuurmans. Understanding the impact of entropy on policy optimization. *CoRR*, 2018.

[9] Miroslav Dudik, John Langford, and Lihong Li. Doubly robust policy evaluation and learning. *CoRR*, 2011.

[10] Nan Jiang and Lihong Li. Doubly robust off-policy value evaluation for reinforcement learning. *CoRR*, 2015.

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

[12] Kamil Ciosek and Shimon Whiteson. Expected policy gradients. *CoRR*, 2017.

[13] Yao Liu, Adith Swaminathan, Alekh Agarwal, and Emma Brunskill. Off-policy policy gradient with state distribution correction. *CoRR*, 2019.

[14] Diederik P Kingma and Jimmy Ba. Adam: a method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[15] Miroslav Dudik, Dumitru Erhan, John Langford, and Lihong Li. Doubly robust policy evaluation and optimization. *CoRR*, 2015.

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

[17] Marc G. Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. *CoRR*, 2017.