The Sufficiency of Off-policyness: PPO is insufficient according to an Off-policy Measure

Xing Chen¹, Dongcui Diao², Hechang Chen¹, Hengshuai Yao³, Jielong Yang¹, Haiyin Piao⁵, Zhixiao Sun⁵, Bei Jiang², and Yi Chang¹

¹School of Artificial Intelligence, Jilin University
²Department of Mathematical and Statistical Sciences, University of Alberta
³Department of Computing Science, University of Alberta
⁴Huawei Technologies Canada
⁵School of Electronics and Information, Northwestern Polytechnical University

Abstract

One of the major difficulties of reinforcement learning is learning from off-policy samples, which are collected by a different policy (behavior policy) from what the algorithm evaluates (the target policy). Off-policy learning needs to correct the distribution of the samples from the behavior policy towards that of the target policy. Unfortunately, important sampling has an inherent high variance issue which leads to poor gradient estimation in policy gradient methods. We focus on an off-policy Actor-Critic architecture, and propose a novel method, called Preconditioned Proximal Policy Optimization (P3O), which can control the high variance of importance sampling by applying a preconditioner to the Conservative Policy Iteration (CPI) objective. This preconditioning uses the sigmoid function in a special way that when there is no policy change, the gradient is maximal and hence policy gradient will drive a big parameter update for efficient exploration of the parameter space. This is a novel exploration method that has not been studied before given that existing exploration methods are based on the novelty of states and actions. We compare with several best-performing algorithms on both discrete and continuous tasks and the results showed that PPO is insufficient in off-policyness, and our P3O is more off-policy than PPO according to the “off-policyness” measured by the DEON metric, and P3O explores in a larger policy space than PPO. Results also show that our P3O maximizes the CPI objective better than PPO during the training process.

1 Introduction

Real-world problems like medication dosing and autonomous driving pose a great challenge for learning algorithms that require lots of interaction with the environment in order for them to do well. Humans have the ability to learn from others, drawing lessons from others’ experience and quickly learning a skill without first doing it when perceiving it for the first time. Later, when there is a chance to practice, the skills obtained from off-line lessons are quickly adapted and improved. It appears we are still far from obtaining this remarkable learning ability in Artificial Intelligence.

Off-policy learning is one promising paradigm to achieve this goal, which evaluates a target policy using a behavior policy that generates experience [Precup et al., 2001] [Sutton et al., 2009]. The paradigm seems characteristic and general enough for obtaining skills from other sources. With off-policy learning, one agent can reuse experience from itself or even the other agents, where the
samples are collected in a different way from what it is interested to learn now. Off-policy learning seems powerful but it is tricky in nature. The mismatch between the distribution of the behavior policy and that of the target policy poses a big challenge for the learning process to be stable. Even for policy evaluation, following the gradient of the temporal difference error easily diverges and some special treatment to the gradient needs to be done to make the underlying ordinary differential equation stable, e.g., see [Sutton et al., 2009]. For policy gradient methods, this is no difference. This means the importance sampling ratio, widely adopted in recent popular algorithms, can be problematic. The problem nature can be seen in a simple example. How to drive a Tesla where there are lots of cows walking randomly on the street? In the gigantic training data set, there may be just a few moments this scenario happens. For an agent to learn this skill well, the objective function must re-weigh the samples in the training data, elevating these few samples much higher than they appear in the training data. As one would imagine, this will lead to high variances because we only have a few samples for this particular scenario. This is the key problem we address in this paper for the context of reinforcement learning.

We consider the environment is not reset-free which is true with real-world applications. In this case, we can only sample a limited number of trajectories, and the issue becomes more severe and results in high regret during learning. In this context, the Conservative Policy Iteration CPI objective [Kakade and Langford, 2002] is a key element of the spectrum of recent popular algorithms including Trust Region Policy Optimization (TRPO) [Schulman et al., 2015], Proximal Policy Optimization (PPO) [Schulman et al., 2017] and many others. It is based on an importance sampling ratio between the new and the old policy, which can cause high variances and leads to a poor gradient estimation and unstable performances for policy gradient methods. TRPO avoids this problem by using a hard threshold for the policy change. PPO uses a clipping for the importance sampling ratio to make sure it is not too far from one and then the final objective is the minimum of the clipped and the un-clipped objective.

The more general topic than off-policy learning is sample-efficient learning. In deep reinforcement learning, sample efficiency can be drawn from the following characteristics. 1) Efficient policy representation. Ensuring that the updated policy is close to the original policy is a good practice although there is some approximation error [Tomar et al., 2020]. The CPI with clipping used by PPO [Schulman et al., 2017], TPPO [Wang et al., 2020] and TR-PPO [Wang et al., 2019b] ensures we only search new policies that are not too far from the existing one. These methods are all first-order optimization algorithms with a low computational cost. 2) Convex optimization. [Tomar et al., 2020] simplified the problem of maximizing the trajectory’s return and used convex optimization solvers by minimizing the Bregman-divergence. 3) Second-order methods that take advantage of Hessian matrix. For example, trust-region computes an approximation to the second-order gradient, such as TRPO [Schulman et al., 2015] and ACKTR [Wu et al., 2017]. These methods usually have a high computation cost. 4) Off-policy learning. Besides the CPI objective based methods, [Wang et al., 2016] truncated the importance sampling ratio with bias correction and use stochastic dueling network for stable learning. [Haarnoja et al., 2018] proposed an off-policy formulation that enables reuse of previously collected data for efficiency. A more detailed review is contained in the Appendix.

In this paper, our methodology is to improve the policy representation in the clipping methods, which results in a better control of the variances caused by importance sampling. This is achieved by applying the preconditioning technique to the CPI objective, and regularizing the policy change by the KL-divergence. Usually preconditioning is applied to an iterative method such as linear system solvers [Saad, 2003; Yao and Liu, 2008]. Recently, there are a few works that apply preconditioning in deep learning to accelerate the learning process, e.g., see [Li et al., 2016; Sappol et al., 2019]. Our work is a novel application of preconditioning to control variances in policy gradient estimation. Our preconditioning technique has an interesting property in that it encourages exploration when the policy change is little and switches to exploitation when policy change is big.

2 Background

This section reviews MDPs and recent popular algorithms TRPO and PPO. The key to TRPO and PPO are their objective functions, both of which are based on an approximation to the value function of a new policy if we know its advantages over the original policy.
Markov Decision Process. An MDP is defined by \((\mathcal{S}, \mathcal{A}, \mathcal{P}, R, \lambda)\), where \(\mathcal{S}\) is the state space, \(\mathcal{A}\) is the action space, for each \(a \in \mathcal{A}\), \(\mathcal{P}\) is a probability measure assigned to a state \(s \in \mathcal{S}\), which we denote as \(\mathcal{P}(\cdot | s, a)\). Define \(R : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}\) as the reward function, where \(\mathbb{R}\) is the real space. \(\lambda \in (0, 1)\) is the discount factor. We consider stochastic policies in this paper. Denote a stochastic policy by a probability measure \(\pi\) applied to a state \(s\): \(\pi(\cdot | s) \rightarrow [0, 1]\). At a time step \(t\), the agent observes the current state \(s_t\) and takes an action \(a_t\). The environment sends the next state \(s_{t+1}\) and a scalar reward \(R_{t+1} = R(s_t, a_t)\) to the agent. The main task of the agent is to find an optimal policy that maximizes the expected sum of discounted future rewards:

\[
V_\pi(s) = \mathbb{E}_\pi \left[ \sum_{t=0}^{\infty} \lambda^t R_t \right],
\]

where \(a_t \sim \pi(\cdot | s_t)\) and \(s_{t+1} \sim \mathcal{P}(\cdot | s_t, a_t)\) for all \(t \geq 0\).

The state-action value function for the policy is defined similarly except the first action taken is not necessarily according to the policy:

\[
Q_\pi(s, a) = \mathbb{E}_\pi \left[ \sum_{t=0}^{\infty} \lambda^t R_t \right],
\]

where \(a_t \sim \pi(\cdot | s_t)\) for \(t \geq 1\).

We will use \(A_\pi\) to denote the advantage function, which characterizes the advantage of an action \(a\) at a state \(s\) by \(A_\pi(s, a) = Q_\pi(s, a) - V_\pi(s)\). Note that if \(a \sim \pi(\cdot | s)\), then the advantage is zero. So the advantage tells us how advantageous an action is, comparing to the action that would be otherwise taken according to policy \(\pi\). In the remainder of the paper, \(\tilde{A}\) is an approximation to the advantage, which is simply the difference between the state-action function and the value function both estimated by an individual algorithm.

Let \(\rho_0\) be the initial state distribution. Let \(\eta(\pi)\) be the expected discounted reward:

\[
\eta(\pi) = \mathbb{E}_{s \sim \rho_0} \left[ V_\pi(s) \right].
\]

Now suppose we are interested in another policy \(\tilde{\pi}\). Let \(d_{\tilde{\pi}}\) be the stationary distribution of the policy. According to Kakade and Langford [2002], the expected return of \(\tilde{\pi}\) can be calculated in terms of \(\eta(\pi)\) and its advantage over \(\pi\) in a straight-forward way:

\[
\eta(\tilde{\pi}) = \eta(\pi) + \sum_s \rho_\pi(s) \sum_a \tilde{\pi}(a | s) A_\pi(s, a).
\]

Here \(\rho_\pi(s) = \sum_{t=0}^{\infty} \lambda^t d_\pi(s_t), s_0 \sim \rho_0\) and the actions are chosen according to \(\tilde{\pi}\), which is just the sum of discounted visitation probabilities [Schulman et al. 2015] approximated \(\eta(\tilde{\pi})\) by replacing \(\rho_\pi(s)\) with \(\rho_\pi(s)\) in the right-hand side:

\[
\hat{\eta}(\tilde{\pi}) = \eta(\pi) + \sum_s \rho_\pi(s) \sum_a \tilde{\pi}(a | s) A_\pi(s, a).
\]

Note in our algorithm, the new policy will be \(\tilde{\pi}\). The benefit of this approximation is that the expected return of the new policy \(\tilde{\pi}\) can be calculated based on the past samples which are from the old policy \(\pi\). Note for any parameter value \(\theta_0\), because of the way that \(\hat{\eta}\) is defined, we have

\[
\hat{\eta}_{\pi_{\theta_0}}(\pi_{\theta_0}) = \eta(\pi_{\theta_0}), \quad \nabla_\theta \hat{\eta}_{\pi_{\theta_0}}(\pi_{\theta_0})|_{\theta = \theta_0} = \nabla_\theta \eta(\pi_{\theta_0})|_{\theta = \theta_0}. \tag{1}
\]

This means that a small gradient descent update of \(\theta_0\) to improve \(\hat{\eta}_{\pi_{\theta_0}}(\pi_{\theta_0})\) also improves \(\eta(\pi_{\theta_0})\).

The TRPO objective. TRPO, PPO and our algorithm all aim to improve the policy incrementally by maximizing the advantage of the new policy over the old one, considering the influence from importance sampling. Sample-based TRPO maximizes the following Conservative Policy Iteration (CPI) objective

\[
L^{\text{CPI}}(\theta) = \tilde{\mathbb{E}}_{\pi_\theta}[r_t(\theta) \tilde{A}_{\pi_{\theta_{t-1}}}(s_t, a_t)],
\]

where \(r_t(\theta) = \pi_\theta(a_t | s_t) / \pi_{\theta_{t-1}}(a_t | s_t)\), constrained by that the difference between the new policy and the old policy is smaller than a threshold. Here the operator \(\tilde{\mathbb{E}}\) means an empirical average over a finite number of samples. Notably, the importance sampling ratio \(r_t(\theta)\) can be very large. To avoid this problem, TRPO uses a hard threshold for the policy change instead of a regularization because “it is difficult to choose a single regularization factor that would work for different problems” according to Schulman et al. [2017]. TRPO then uses the trust region method which is a second-order method that maximizes the objective function with a quadratic approximation. The advantage \(\tilde{A}_{\pi_{t-1}}\) is replaced by the \(Q_{\pi_{t-1}}\) in TRPO.
The PPO objective. This policy objective function is used in deriving the PPO algorithm [Schulman et al., 2017], which was motivated to invent a first order method of TRPO. The PPO algorithm first samples a number of trajectories using policy \( \pi_{\text{old}} \), each trajectory with \( T \) time steps. For each trajectory, the advantage is computed according to

\[
\hat{A}_t = \lambda^{T-t} V(s_T) + \sum_{k=t}^{T-1} \lambda^{k-t} r_k - V(s_t).
\]

Given the advantages and the importance sampling ratios (to re-weigh the advantages), PPO maximizes the following objective:

\[
L^{\text{PPO}}(\theta) = \mathbb{E}_t \min \left\{ r_t(\theta) \hat{A}_t, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon) \hat{A}_t \right\},
\]

\[
= \mathbb{E}_t \min \left\{ r_t(\theta) \hat{A}_t, \min \left\{ \max\{r_t(\theta) \hat{A}_t, (1 - \epsilon) \hat{A}_t\}, (1 + \epsilon) \hat{A}_t\right\} \right\},
\]

where the last equation assumes \( \hat{A}_t \geq 0 \). If \( \hat{A}_t < 0 \), the objective is similarly complicated. This shows that although PPO performs really well in practice, its objective is rather hard to comprehend due to the use of min and max and their special way of composition. Presumably the role of the two composed operators is to battle instability caused by importance sampling and maintain performance across different tasks. It is not the first time that importance sampling causes trouble for reinforcement learning. For the discrete-action problems, both off-policy evaluation and off-policy learning are known to suffer from high variances due to the product of a series of such ratios each of which can be bigger than one especially when the behavior policy and the target policy are dissimilar, e.g., see [Precup et al., 2001, Sutton et al., 2009]. Importance sampling is unfortunately necessary because the nature of reinforcement learning is off-policy. That is, the agent needs to constantly improve its behavior using experience that is imperfect in the sense that it does not behave optimally at learning. Importance sampling ratios arise because one needs to correct the weighting of the objectives such as advantages, from the behavior policy towards the weighting that would be otherwise under a target and improved policy, e.g., see [Precup et al., 2001, Schulman et al., 2015].

3 The Preconditioned Off-policy Learning Method

In this section, we propose a new objective function by preconditioning. We also use the KL divergence to ensure small and smooth policy changes between the update.

3.1 The Scopic Objective

Inspired by the CPI objective and the PPO objective, we propose the following objective:

\[
L^{sc}(\theta) = \mathbb{E}_t \left[ \sigma \left( \tau (r_t(\theta) - 1) \right) \frac{\hat{A}_t}{\tau} \right]
\]

where \( \sigma \) is the sigmoid function and \( \tau \) is the temperature. The advantage \( \hat{A}_t \) is computed in the same way as in PPO. We term this new objective function the Scopic objective, short for sigmoidal conservative policy iteration objective without clipping. Intuitively, according to the Scopic objective, the agent learns to maximize the scaled advantages of the new policy over the old one whilst maintaining stability by feeding the importance sampling ratio to the sigmoid function. By using the sigmoid, theoretically the importance sampling ratio is allowed from zero to infinity while the output is still in a small range, \([\sigma(-\tau), 1]\). Thus the new policy is allowed to be optimized in a policy space that is much larger than the clipped surrogate objective. Because the PPO objective is clipped, PPO would not have any information such as the gradient for new policies whose importance sampling ratios over the current policy are beyond the range of \([1 - \epsilon, 1 + \epsilon]\).

Besides the controlled importance sampling ratio range in a “soft” way, there is an interesting property of the Scopic objective that is very beneficial for reinforcement learning here. The input of the sigmoid is zero if there is no change in the policy at a state. Note that the gradient of the sigmoid achieves the maximum in this case. This means when policy change is zero or little, the sigmoid drives for a big parameter update and hence explore in the policy space. The effect of a big change in \( \theta \) leads to a big change in \( \pi_{\theta} \) as well, and thus the action selection has a big change, meaning that our method
effectively adapts the parameter update magnitude to explore the action space. On the other hand, when the new policy changes a great deal from the old policy, the gradient of the sigmoid grows small which gives the parameter little update. The effect is that the agent will fix to a close neighborhood of the policy and utilize the knowledge built in the policy, which leads to exploitation. So by using the Scopic objective, the agent learns to balance between exploration and exploitation automatically via the gradient magnitude that is adapted by the sigmoid function. This method of exploration is novel for reinforcement learning and it has not been explored in previous research to the best of our knowledge. Existing methods of exploration are mostly based on the novelty of states and actions, boiling down to count based methods such as UCT [Kocsis and Szepesvári, 2006] and UCB-1 [Auer, 2002]. The count based methods have a wide applications such as computer games, e.g., the use of UCT in AlphaGo [Silver et al., 2016], and the contextual bandits [Bubeck and Cesa-Bianchi, 2012] in recommender systems [Li et al., 2010], etc. The count based methods, by definition, only apply to discrete spaces. Though it is possible to extend to some smoothed version in the continuous case such as kernel regression UCT [Yee et al., 2016], such methods depend on a choice of kernels and a regression procedure that is performed on a data set of samples. Our approach via the sigmoidally preconditioned objective balances exploration and exploitation in a natural way given an online sample and it does not involve other samples, which is very computationally efficient.

In the Scopic objective, the term $4/\tau$ may appear odd at first sights. Let us explain this choice here. First note the important case when there is no change in the new policy from the old one. This momentary on-policy learning can be recovered by $\tau = 2$. This can be seen from when the input of the sigmoid function is zero, meaning that the learning reduces to on-policy at least for the first mini-batch update. Refer to the definition of $\hat{\eta}$ in Eq.1 for any parameter value $\theta_0$. For the choice of term of $4/\tau$, we can derive

$$L_{sc}(\theta_0) = \hat{\eta}(\theta_0), \quad \nabla_\theta L_{sc}(\theta)|_{\theta=\theta_0} = \nabla_\theta \hat{\eta}(\theta)|_{\theta=\theta_0}.$$  

Thus this ensures that the gradient descent update of the Scopic objective will improve $\hat{\eta}$ and hence $\eta$ for the case of on-policy learning.

The Scopic loss can be viewed as a preconditioning technique. Let $p(\theta) = \sigma(\tau(r_t(\theta) - 1))$, the gradients of the Scopic objective and the CPI objective are:

$$\nabla_\theta L_{sc} = \bar{E}_t[4p(\theta)(1 - p(\theta))\nabla_\theta r_t(\theta)\hat{A}_t], \quad \nabla_\theta L_{cpi} = \bar{E}_t[\nabla_\theta r_t(\theta)\hat{A}_t]$$

Thus stochastic gradient ascent update for the Scopic objective is a modification from that of the CPI objective. This is similar to preconditioning in iterative methods [Saad, 2003], but note here the preconditioner is stochastic and applies to the stochastic gradient. Figure 1 shows the objective function and the gradient for CPI, the PPO objective and our Scopic objective.

### 3.2 KL Divergence

To handle the problem more effectively, we also consider the KL-divergence between the new policy and the old policy to make sure learning is close to on-policy learning and the importance sampling
The objective function is 

\[ L_{vf}(w) = \mathbb{E}_t \left[ r_t + \lambda V(s_{t+1}) - V(s_t) \right] \nabla_w V(w). \]

because the objective function is 

\[ L_{vf}(w) = \mathbb{E}_t \left[ r_t + \lambda V(s_{t+1}) - V(s_t) \right] \nabla_w V(w). \]

Our Preconditioned Proximal Policy Optimization (P3O) algorithm boils down to the gradient ascent maximizing \( L^{P3O} \) and the gradient descent minimizing \( L_{vf} \). The P3O algorithm is shown in Appendix 1.

4 Experiments

We tested the performance of our P3O algorithm versus baselines in both continuous- and discrete tasks in OpenAI Gym [Brockman et al., 2016] and the Arcade Learning Environment [Bellemare et al., 2013]. The task include Ant-v2, HalfCheetah-v2, and Walker2d-v2 for continuous tasks, for which the policy is parameterized using Gaussian distribution. Discrete tasks include Enduro-v4, Breakout-v4, and BeamRider-v4. The observation of discrete environments is four stacking frames RGB image of the screen, and the policy is parameterized using Softmax distribution. In addition to TRPO and PPO, we also include A2C [Mnih et al., 2016], ACKTR [Wu et al., 2017], and DualClip-PPO [Ye et al., 2020] as baselines. We evaluate the episodic accumulated reward during the training process of each algorithm. We run each algorithm in the six environments with four random seeds and set the training time steps to be ten million for the discrete tasks and three million for continuous tasks. Code is available at https://github.com/raincchio/P3O.

4.1 Performance Comparison

In Figure 2 the learning curves of TRPO and A2C are very flat: showing ineffectiveness for these discrete environments. The TRPO’s performance was similar to the empirical results in [Wu et al., 2017]. The reason why it did not perform well may be due to that it is hard to set a proper parameter for the KL-divergence constraint since it varies in different environments. ACKTR is a second-order, natural gradient algorithm with a much higher computational cost per time step. However, it still did not outperform PPO which is a first-order method. Dualclip-PPO inherits the clipping operation from PPO objective and adds another max operator with an additional parameter [Ye et al., 2020].
The algorithm was applied to the game of Honor of Kings and achieved competitive plays against human professionals. However, there was no baseline compared. In our experiments, the algorithm performed close to or worse than PPO. That PPO is better than all the other four baselines shows that it is indeed important to control the high variance issue of importance sampling. Our P3O outperformed all the baselines including the best performing PPO for the tasks. In the next subsection, we investigate into why this happened.

4.2 The DEON Off-policy Measure and Policy Space Comparison

In order to understand the performance comparison between PPO and our P3O, we compared the maximum deviation from on-policy learning (DEON) measure, defined by \( y = \max(|r - 1|) \), where the maximum was taken over all the samples in the collected trajectories. This measure computed during the training process of PPO and P3O are compared in Figure 3. It shows that the deviation of our P3O from on-policy learning is much bigger than PPO during training: P3O is more off-policy than PPO. This means P3O explores in a much bigger policy space than PPO. The clipping operation in the PPO objective prevents the algorithm from exploring beyond the policy space whose importance sampling ratio over the old policy is within the range, \([1 - \epsilon, 1 + \epsilon]\). Together with the performance comparison in Figure 2 this shows that the new Scopic objective via sigmoidally preconditioning that is used by our P3O is a very effective way of conducting exploration in the parameter space which results in an efficient exploration of the action space.

![Figure 3: The DEON off-policy measure of our P3O vs. PPO during training. P3O’s deviation are much higher than PPO, showing P3O explores in a policy space that is much bigger than PPO due to that the clipping in the PPO’s objective makes it fail to discover better policies beyond the clip range.](image)

Note for discrete tasks, all the importance sampling ratios finally converged close to one due to the use of the decaying learning rate. This is interesting because it shows the learning rate decay can give us an on-policy learning algorithm in the long run. For the continuous tasks where the fixed learning rate is used, the DEON measure of P3O is still increasing in the end, while for PPO it drops close to zero. In particular, the average DEON measure is up to as big as 20.0 for P3O which performs better than PPO. This shows clipping the importance sampling ratio in the range \([1 - \epsilon, 1 + \epsilon]\) by the PPO objective is far from enough to cover good policies. In the experiments, we used the \( \epsilon = 0.2 \), same as used in the PPO paper [Schulman et al., 2017]. The DEON metric being still large in the end of learning means P3O is still exploring. The larger policy search space and the consistent exploration leads to a bigger improvement in performance for continuous tasks than for discrete tasks (See Figure 2 and note the continuous tasks have a much coarser scale in the y-axis).
Figure 4: Plotting the CPI objective during training of P3O and PPO. P3O maximizes the CPI objective much better than PPO except for BeamRider.

4.3 CPI Objective Comparison

This experiment was initiated due to a note that both the PPO objective and the Scopic objective originates from the CPI objective. As shown by the TRPO and PPO algorithms, directly maximizing the CPI objective is problematic due to high variances with importance sampling. PPO and our P3O can be viewed as special methods for maximizing the CPI objective. Thus, one important question is, how much is the CPI objective maximized in either of the two algorithms? We thus calculated the CPI objective (without any clipping or sigmoid preconditioning) in the training process of the algorithms. The result is shown in Figure 4. For both the discrete and continuous tasks except BeamRider, P3O consistently maximizes the CPI objective better than PPO. This means the Advantage of the new policy is consistently larger than the old policy with sigmoid preconditioning than with clipping. For the discrete tasks, the CPI objective of P3O finally converged close to that of PPO. This is also because the decay learning rate was used for discrete tasks which became really small in the end leading to the importance sampling being one and thus the Scopic objective reducing to the CPI objective. The CPI of continuous tasks is even much bigger. Because of the use of the fixed learning rate, P3O still actively explores even in the end of learning and keeps discovering new policies whose Advantage is much bigger than the old one.

4.4 Sensitivity to Hyper-parameters

The hyper-parameter studies were performed over three dimensions (number of updates, batch size, learning rate). The numbers of epoch updates were either 5 or 10. The batch size were either 32 or 64. The learning rate were either constant $10^{-4}$ or the decay scheduling. The decay schedule started with a learning rate of $3 \times 10^{-4}$ and decayed linearly, which was used in OpenAI’s PPO implementation. This leads to eight hyper-parameter combinations whose results are shown in Figure 5. Group (c) performed the best for PPO (consistent with the best result in the PPO paper) and Group (h) was the best for P3O. This shows PPO prefers the decay schedule while our method prefers the fixed learning rate for continuous tasks. For the best hyper-parameter group for both algorithms, P3O’s variance is much lower than PPO.

4.5 Ablation Studies

As the P3O objective has two parts, the sigmoidal preconditioner and the KL-divergence regularizer, we conducted an ablation study. The following three variants were studied: P3O-S removes the sigmoidal preconditioner and replaces it with the identity mapping; P3O-K removes the KL-divergence regularizer; P3O-SK removes both the preconditioner and KL-divergence, reducing to
Figure 5: Hyper-parameter sensitivity studies of PPO (green) and our P3O (red) on HalfCheetah, with two schedulings of each of the three dimensions: the number of epochs on the samples, batch size, and learning rate. In terms of the best case, P3O (group h) performed better than PPO (group c; consistent with the original PPO paper) with higher mean reward and lower variances.

Figure 6: Ablation studies. P3O-S removes the sigmoid preconditioner and P3O-K removes KL-divergence regularizer; while P3O-SK removes both, reducing to the TRPO objective. This shows both factors are important to the performance of P3O; in addition, preconditioning contributes more than the regularization.

the TRPO objective. Figure 6 shows the performance of all these variants decreased sharply. Notably, P3O-K (with only the sigmoid preconditioning) outperforms P3O-S (with only the KL-divergence regularizer), showing that the sigmoid preconditioning is the primary driver of improvement in P3O.

5 Conclusion

In this paper, we proposed a new method of off-policy learning that deals with the high variance issue of importance sampling with a novel preconditioning technique via the sigmoid function. The new method exhibits exploratory behavior when the new policy is close to the old policy and exploits if the new policy is already a big change from the old one. Our objective function employs the KL divergence regularizer between the new policy and the old policy. The two techniques enable us not to rely on the clipping technique used by PPO. There is almost no increase in the computational cost of our method.
We compared our P3O algorithm with five recent deep reinforcement learning baselines in both discrete and continuous environments. Results show that our method achieves better performance than the baselines. We found that PPO is insufficient in off-policyness, and our P3O deviates more from on-policy learning than PPO, according to a measure of off-policyness during training which is called DEON. This suggests that our method is “more off-policy” and its policy search space is larger than PPO. In addition, the CPI objective of our P3O algorithm is larger than PPO except for one task being close, showing that our algorithm finds better policies that give larger Advantages.

References

Peter Auer. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3(Nov):397–422, 2002.

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

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

Sébastien Bubeck and Nicolo Cesa-Bianchi. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *arXiv preprint arXiv:1204.5721*, 2012.

Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Vlad Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Ian Dunning, et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *International Conference on Machine Learning*, pages 1407–1416. PMLR, 2018.

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, pages 1587–1596. PMLR, 2018.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

Chloe Ching-Yun Hsu, Celestine Mendler-Dünner, and Moritz Hardt. Revisiting design choices in proximal policy optimization. *arXiv preprint arXiv:2009.10897*, 2020.

Sham Kakade and John Langford. Approximately optimal approximate reinforcement learning. In *In Proc. 19th International Conference on Machine Learning*. Citeseer, 2002.

Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *Proceedings of the 17th European Conference on Machine Learning*, ECML’06, page 282–293, Berlin, Heidelberg, 2006. Springer-Verlag. ISBN 354045375X. doi: 10.1007/11871842_29. URL https://doi.org/10.1007/11871842_29

Chunyuan Li, Changyou Chen, David Carlson, and Lawrence Carin. Preconditioned stochastic gradient langevin dynamics for deep neural networks. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web - WWW ’10*. ACM Press, 2010. doi: 10.1145/1772690.1772758. URL https://doi.org/10.1145/1772690.1772758

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, 2019.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.
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, pages 1928–1937. PMLR, 2016.

Doina Precup, Richard S. Sutton, and Sanjoy Dasgupta. Off-policy temporal difference learning with function approximation. In Proceedings of the Eighteenth International Conference on Machine Learning, ICML ’01, page 417–424, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc. ISBN 1558607781.

Yousef Saad. Iterative Methods for Sparse Linear Systems. Other Titles in Applied Mathematics. SIAM, second edition, 2003. ISBN 978-0-89871-534-7. doi: 10.1137/1.9780898718003. URL http://www-users.cs.umn.edu/~saad/IterMethBook_2ndEd.pdf.

Johannes Sappl, Laurent Seiler, Matthias Harders, and Wolfgang Rauch. Deep learning of preconditioners for conjugate gradient solvers in urban water related problems. arXiv preprint arXiv:1906.06925, 2019.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In International conference on machine learning, pages 1889–1897. PMLR, 2015.

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

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. nature, 529(7587):484–489, 2016.

Richard S Sutton and Andrew G Barto. Reinforcement learning: An introduction. MIT press, 2018.

Richard S Sutton, Hamid Maei, and Csaba Szepesvári. A convergent o(n) temporal-difference algorithm for off-policy learning with linear function approximation. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, Advances in Neural Information Processing Systems, volume 21. Curran Associates, Inc., 2009. URL https://proceedings.neurips.cc/paper/2008/file/e0c641195b27425bb056ac56f8953d24-Paper.pdf.

Manan Tomar, Lior Shani, Yonathan Efroni, and Mohammad Ghavamzadeh. Mirror descent policy optimization, 2020. URL https://arxiv.org/abs/2005.09814.

Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In Proceedings of the AAAI conference on artificial intelligence, volume 30, 2016.

Qing Wang, Yingru Li, Jiechao Xiong, and Tong Zhang. Divergence-augmented policy optimization. In Proceedings of the 33rd International Conference on Neural Information Processing Systems, pages 6099–6110, 2019a.

Yuhui Wang, Hao He, Xiaoyang Tan, and Yaozhong Gan. Trust region-guided proximal policy optimization, 2019b.

Yuhuai Wu, Elman Mansimov, Roger B Grosse, Shun Liao, and Jimmy Ba. Scalable trust-region method for deep reinforcement learning using kronecker-factorized approximation. Advances in neural information processing systems, 30:5279–5288, 2017.

Hengshuai Yao and Zhi-Qiang Liu. Preconditioned temporal difference learning. In Proceedings of the 25th international conference on Machine learning, pages 1208–1215, 2008.
Deheng Ye, Zhao Liu, Mingfei Sun, Bei Shi, Peilin Zhao, Hao Wu, Hongsheng Yu, Shaojie Yang, Xipeng Wu, Qingwei Guo, et al. Mastering complex control in moba games with deep reinforcement learning. In Proceedings of the AAAI Conference on Artificial Intelligence, pages 6672–6679, 2020.

Timothy Yee, Viliam Lisy, and Michael Bowling. Monte carlo tree search in continuous action spaces with execution uncertainty. In Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI’16, page 690–696. AAAI Press, 2016. ISBN 9781577357704.
A Algorithm

The pseudo-code of our P3O algorithm is shown in Algorithm 1.

Algorithm 1 Preconditioner Proximal Policy Optimization

Input: a simulation environment
Output: policy $\pi_\theta$ for the environment
Initialize policy parameters $\theta_\pi, \theta_{\pi_{old}}$ and value network parameters $\theta_w$, learning rate for each parameter $\lambda_\pi, \lambda_v$, update number $T$, number of samples $N$, data buffer $D$
Simulate as many as possible agents (depending on hardware) with policy $\pi_\theta$ to interact with the environment

for iteration $= 1$ to $m$ do
    repeat
        $a_t \sim \pi_\theta(a_t|s_t)$
        $s_{t+1} \sim P(s_{t+1}|s_t, a_t)$
        $D \leftarrow D \cup (s_t, a_t, r_t, s_{t+1})$
    until $t \geq NT$
    Take all the buffer data to compute advantage $\hat{A}$

for epoch $= 1$ to $T$ do
    Sample $N$ mini-batch samples from Buffer $D$
    $\theta_\pi \leftarrow \theta_\pi - \lambda_\pi \nabla_\theta L_{sc}^{\pi}$
    $\theta_w \leftarrow \theta_w - \lambda_v \nabla_w L_{vf}^{\pi}$
end for
Clear data buffer $D$
end for

B Related Works

In this section, we briefly review the related works from four aspects.

Convex optimization methods. Some work has developed approaches that simplify the optimization problem of reinforcement learning into a convex optimization problem. In the paper [Yao and Liu, 2008], a framework of policy evaluation algorithms called the preconditioned temporal difference (PTD) learning is introduced. And in the paper [Li et al., 2016, Sappl et al., 2019], they use the preconditioning for deep neural network effective training. DAPO [Wang et al., 2019a] proved that the convex optimization method could be used in the reinforcement learning problem and include the Bregman divergence to ensure minor and safe policy updates with off-policy data. Euclidean distance and Kullback Leibler (KL) divergence are instances of Bregman divergence. MDPO [Tomar et al., 2020] adds a proximity term that restricts two consecutive policies to be close to each other to the common expected reward objective, iteratively updates the policy by approximately solving a trust-region problem.

Compute Hessian Matrix. Recently, there are many methods to improve sample efficiency based on the Trust-region second-order gradient optimization algorithms [Schulman et al., 2015, Wu et al., 2017]. These methods calculate the approximate Hessian matrix to find an ideal gradient. However, the computational cost is enormous, and the Hessian matrix is not tackled for complex neural networks. Although [Wu et al., 2017] designed a particular neural network struct for simplifying computation, the network structure is not generalizable for complex application situations.

Policy Approximation. Several studies go deeper into the policy approximation in RL, one of the famous is the clipping mechanism of the PPO algorithm, which relies upon optimizing parametrized policies by heuristic method. In [Wang et al., 2020], they adopt a new clipping function to support a rollback behavior to restrict the ratio between the new policy and the old one. In the paper [Ye et al., 2020], when the estimate of the advantage is negative, their dual-clip operator will result in a better performance of the policy. In [Hsu et al., 2020], they review the multiple failure modes of the PPO algorithm and use beta distribution instead of the Gaussian distribution to get better performance.
However, this is not to improve the algorithm but to replace the structure more suitable for specific environments. These studies only analyzed the effects of the PPO’s clip operation and did not put forward the theory to explain why the clipping is needed. For the estimation of $\eta(\pi)$ under a behavior policy $\hat{\pi}$, possible methods include Retrace [Wang et al., 2016] providing an estimator of state-action value, and V-trace [Espeholt et al., 2018] providing an estimator of state value. But that two methods also suffer from the estimation variance.

Off-Policy Method. For value-based algorithms, sample efficiency has also been considered from the aspect of data usage. In the DQN [Mnih et al., 2015] algorithm, a replay buffer for learning previous experiences is used to improve sample efficiency. In the double-DQN [Van Hasselt et al., 2016], the overestimation problem is eliminated by decoupling the two steps of target Q action selection and target Q calculation. The DDPG [Lillicrap et al., 2019] is a deterministic policy gradient algorithm based on actor-critic, solving continuous action space tasks that DQN cannot solve. TD3 [Fujimoto et al., 2018] follows the idea of double-DQN, using two independent critics to prevent overestimation, but can only be used for environments with continuous action spaces. In the SAC [Haarnoja et al., 2018], they introduce entropy into a deep reinforcement learning algorithm, which provides sample-efficient learning while retaining the benefits of entropy maximization.

Compared with the previous method, our method uses preconditioner to better deal with the infinite variance problem. Low-variance gradients make learning more efficient, allowing fixed step-sizes to be used, potentially reducing the overall number of steps needed to reach convergence and resulting in faster training. And we can guarantee monotonic policy improvement by cooperating with the KL divergence constraint.