Proximal Policy Optimization with Dynamic Clipping

Student: Rishikesh Vaishnav
University of California, San Diego

Mentor: Sicun Gao, Ph.D.
University of California, San Diego

Abstract

Proximal Policy Optimization (PPO), a policy gradient algorithm drawing closely from the theory supporting Trust Region Policy Optimization (TRPO), has emerged as one of the most effective tools in reinforcement learning (RL) problems. PPO makes use of a loss function with a clipped importance sampling ratio using a single parameter ϵ. Although PPO shows promising empirical performance, it is vulnerable to problem-specific imbalances in its handling of positive and negative advantages. We investigate one such imbalance, addressing a discrepancy in expected penalty contributions of positive and negative estimators. By precisely calculating this discrepancy and minimizing it before each model update, we empirically demonstrate that eliminating this discrepancy can improve the overall performance of PPO.

1 Background

Note: This paper assumes knowledge of the common terms and concepts in reinforcement learning (see [3]).

Among current reinforcement learning (RL) algorithms, Policy Gradient methods have seen significant success at a wide range of tasks. These methods seek to learn performant policy distributions directly, rather than learning a value function to indirectly guide the policy. Using a policy $\pi_\theta(a)$ parameterized by θ, these algorithms have the general form:

Algorithm 1 Generic Policy Gradient

```
Initialize θ arbitrarily
while True do                                  ▷ loop forever
    θ_old ← θ
    rollout ← (s, a, r) from multiple
        episodes following π_θ
    Set θ to maximize the loss function
        L(rollout, θ, θ_old)
end while
```

Within this framework, the choice of a loss function is the key to an effective algorithm, and current research in reinforcement learning focuses particularly on finding new ways to represent this loss. A recent innovation in policy gradient methods is Trust Region Policy Optimization (TRPO) [1], which approximates the updates perfomed by an agent that uses the following loss function to guarantee monotonic improvement in the policy's performance:

$$L_{\theta_{old}}(\theta) - CD_{KL}^{max}(\theta, \theta_{old}), \tag{1}$$

where C is a constant,

$$L_{\theta_{old}}(\theta) = \mathbb{E}_t \left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} A_t \right], \tag{2}$$

and A_t is the advantage at time t. In TRPO implementations, this loss function results in relatively small step sizes, so in practice the penalty term $CD_{KL}^{max}(\theta, \theta_{old})$ is removed from equation (1), and we instead maximize (2) with a constraint on the KL divergence.

Implementations of TRPO generally have significant computational complexity relative to other RL algorithms, largely due to the need to calculate KL divergences and perform constraint optimization. To address this, a new algorithm based on the theory behind TRPO, Proximal Policy Optimization (PPO) [2], was developed.

Rather than maximizing (2) subject to a constraint, PPO maximizes the following "clipped" loss function:

$$L^{CLIP}(\theta) = \mathbb{E}_t \left[\min \left(r_t A_t, \text{clip}(r_t, 1-\epsilon, 1+\epsilon) A_t \right) \right], \tag{3}$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$.

This loss function can be maximized using standard optimization techniques and does not require constraint optimization, significantly simplifying implementation relative to TRPO. This loss function can be thought of as a heuristic approximation to the loss function from (1), where the minimization substitutes the penalty term.

PPO performs well in practice, in many cases outperforming TRPO. However, its simple equation does not allow for precise control over what penalties are actually introduced, and, as we will show, can exhibit problem-specific imbalances in the way positive and negative advantages are handled.

2 Expected Penalty Contributions

(3) can be split into positive and negative advantage cases as follows:

$$L^{CLIP}(\theta) = \mathbb{E}_t \left[\begin{cases} \min\left(r_t, \text{clip}(r_t, 1+\epsilon)\right) A_t & A_t > 0 \\ \max\left(r_t, \text{clip}(r_t, 1-\epsilon)\right) A_t & A_t < 0 \end{cases} \right]. \tag{4}$$

Now, let

$$r^+_{t,CLIP} = \min\left(r_t, \text{clip}(r_t, 1+\epsilon)\right)$$
$$r^-_{t,CLIP} = \max\left(r_t, \text{clip}(r_t, 1-\epsilon)\right).$$

It follows mathematically that $\mathbb{E}_t\left[r_t\right] = 1$, because ratios are sampled according to the old policy distribution. Therefore, it must be the case that $\mathbb{E}_t\left[r^+_{t,CLIP}\right] < 1$ and $\mathbb{E}_t\left[r^-_{t,CLIP}\right] > 1$. As the new policy changes relative to the old policy, clipping becomes more frequent, generally causing these expectations to become smaller and larger, respectively. Assuming independence between ratios and advantages, the effect of this is to make positive advantages less positive and make negative advatages more negative. This is a reflection of how penalization occurs when a new policy is learned.

We can define the "expected penalty contributions": $1 - \mathbb{E}_t[r^+_{t,CLIP}]$ and $\mathbb{E}_t[r^-_{t,CLIP}] - 1$. We would expect both of these values to become more positive as learning progresses in a single iteration.

In practice, the assumption of independence between ratios and advantages is incorrect, because if these were independent, we would expect to see a monotonically decreasing loss at each model update. In reality, however, an optimizer will specifically work around this by assigning ratios to advantages such that the loss tends to increase at each model update. Therefore, these expected ratios do not suggest the actual proportional contributions of positive and negative advantages to the overall loss.

However, it seems likely that these expectations have a significant influence on how positive and negative advatages are considered in calculating the overall loss. Addressing these expectations and controlling their relative values will affect the loss that is calculated, and could have an influence on overall performance.

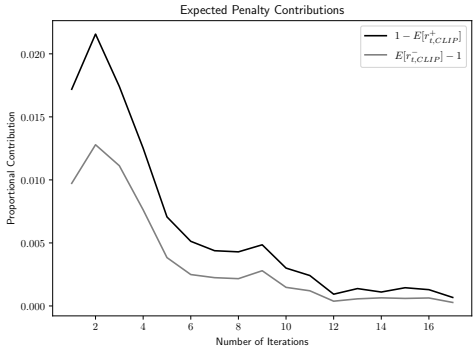

Figure 1: Empirical expected penalty contribution discrepancies. Expeceted values were calculated after the final model update in each iteration.

There is no reason to assume that the expected penalty contributions will agree with one another. We found that the both the rate at which these increased and the discrepancy between these two values was dependent on the choice of distribution to represent the policy.

3 Problem-Specific Expected Penalty Contribution Discrepancy

In order to investigate the effect of controlling the discrepancy between positive and negative expected penalty contributions, we focus our attention to a specific problem setting involving a continuous action space, with actions chosen according to a gaussian distribution with fixed standard deviation. From a simulated run, we observed that, for a single state, as the learned gaussian diverged from the original gaussian, the expected penalty contributions increased at different rates, resulting in a growing discrepancy between them (see simulation here: [link]). This discrepancy also appears empirically, as shown in figure 1. As suggested in the simulation, the empirical data shows that initial, high-learning phases are marked by a larger discrepancy and expected total penalty contribution than later, slower learning phases.

4 Addressing the Discrepancy

Using the aforementioned gaussian policy model, we can precisely calculate the discrepancy at a certain state using the following equation:

$$\begin{aligned}&(1 - E[r^+_{t,CLIP}]) - (E[r^-_{t,CLIP}] - 1)\\ &= \epsilon + (1-\epsilon)\int_{x^-}^{x^+} p(\mu_{old}, x)dx\\ &- \left(\int_{x^-}^{x^+} p(\mu, x)dx + 2\epsilon\int_{x^+}^{\infty} p(\mu_{old}, x)dx\right)\end{aligned}$$

where

$$x^+ = \frac{(\mu^2 - \mu^2_{old}) + 2\sigma^2 \ln(1+\epsilon)}{2(\mu - \mu_{old})},$$

$$x^- = \frac{(\mu^2 - \mu^2_{old}) + 2\sigma^2 \ln(1-\epsilon)}{2(\mu - \mu_{old})},$$

and μ and μ_{old} are the gaussian means predicted for this state for the new and old policies, respectively.

On its own, this equation does not allow for close control over the discrepancy. Consider allowing for distinct upper and lower ϵ, replacing the loss function with

$$L^{CLIP}(\theta) = \mathbb{E}_t\left[\min\left(r_t A_t, \text{clip}(r_t, 1-\epsilon^-, 1+\epsilon^+)A_t\right)\right]. \tag{5}$$

We can now write the discrepancy as:

$$\begin{aligned}&(1 - E[r^+_{t,CLIP}]) - (E[r^-_{t,CLIP}] - 1)\\ &= \epsilon^- + (1-\epsilon^-)\int_{x^-}^{x^+} p(\mu_{old}, x)dx\\ &- \left(\int_{x^-}^{x^+} p(\mu, x)dx + (\epsilon^+ + \epsilon^-)\int_{x^+}^{\infty} p(\mu_{old}, x)dx\right)\end{aligned} \tag{6}$$

Changing the subtraction between expected penalty contributions to an addition, we can also calculate the total expected penalty contribution:

$$\begin{aligned}&(1 - E[r^+_{t,CLIP}]) + (E[r^-_{t,CLIP}] - 1)\\ &= 2\int_{x^+}^{\infty} p(\mu, x)dx - (2 + \epsilon^+ - \epsilon^-)\int_{x^+}^{\infty} p(\mu_{old}, x)dx\\ &-\epsilon^- - (1-\epsilon^-)\int_{x^-}^{x^+} p(\mu_{old}, x)dx + \int_{x^-}^{x^+} p(\mu, x)dx\end{aligned} \tag{7}$$

From (6), we can see that increasing ϵ^+ and ϵ^- will generally have the effect of decreasing and increasing the discrepancy, respectively. We can define an algorithm that makes use of these equations in optimizing the clip parameters at every iteration. Given set initializations of ϵ^+ and ϵ^-, before calculating the loss function at each batch in each iteration, we can optimize these parameters to minimize this discrepancy while maintaining the total expected penalty contributions.

Algorithm 2 PPO with Dynamic Clipping

```
Initialize θ arbitrarily
Initialize ε⁺ and ε⁻
μ_old ← 0
while True do                                   ▷ loop forever
    θ_old ← θ
    rollout ← (s, a, r) from multiple episodes following π_θ
    for batch from rollout do
        if not first batch in rollout then
            μ ← the average absolute distance of the new policy's actions from the old policy's actions
            Minimize (6) while keeping (7) constant
        end if
        Perform an optimization step on (5)
    end for
end while
```

5 Results

Our tests were run on standard MuJoCo environments from OpenAI Gym. We ran tests with the original PPO environment using ϵ values of $0.1, 0.2, 0.3, 0.4$. We then ran tests with the new algorithm using these same ϵ values as initializers for both ϵ^+ and ϵ^-.

Overall, looking at individual environments and comparing the best runs (as defined by end-of-training performance) between the control (original PPO) and experimental algorithms, we saw that, compared to the control algorithm, the experimental algorithm either performed approximately the same as the control or slightly better. This suggests an improvement in performance. In all cases, the algorithm successfully reduced the empirical expected discrepancies, however this did not have a consistent effect on the actual discrepancies when calculating the loss.

Figures 2, 3, and 4 show the results for some environments. In these runs, the experimental and control algorithms were initialized with the same value of ϵ.

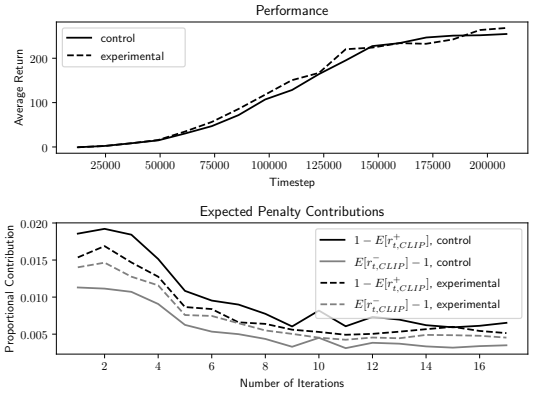

Figure 2: Results: Walker2d-v2 environment

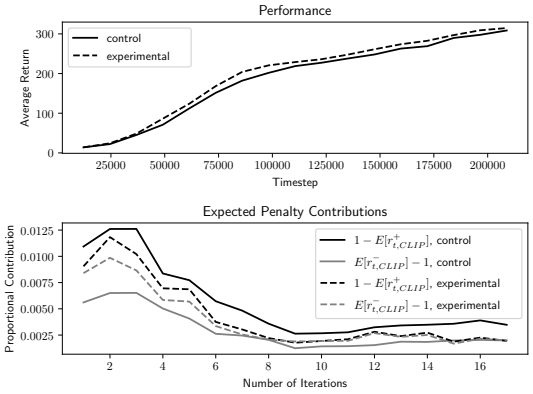

Figure 3: Results: Hopper-v2 environment

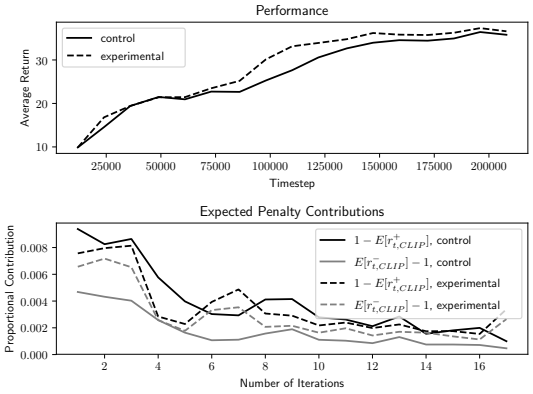

Figure 4: Results: Swimmer-v2 environment

6 Conclusions

The results of this research so far suggest that, within the framework of a PPO algorithm, using two dynamic clipping parameters that are optimized to address problem-specific imbalances is a promising technique to improve the performance of PPO.

There are many different directions in which we can continue to develop this research. In terms of runtime and computational complexity, it would likely be useful to use a problem-independent heuristic method to minimize the discrepancy. Additionally, the discrepancy can likely be more effectively reduced by applying ϵ-optimization on a per-state, rather than per-batch, basis. Another important consideration is how the expected discrepancy relates to the actual penalty contribution discrepancy.

7 Acknowledgements

I would like to thank my mentor, Dr. Sicun Gao, for being very attentive and supportive of my work, and for directing me to relevant research at the cutting-edge of reinforcement learning. I would also like to thank the Academic Enrichment Program at UCSD, especially Dr. Kirsten Kung, for hosting me during my work this summer as a member of the UC Scholars Program.

References

[1] John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. Trust region policy optimization. *CoRR*, abs/1502.05477, 2015.

[2] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.

[3] Richard S. Sutton and Andrew G. Barto. *Reinforcement learning - an introduction*. Adaptive computation and machine learning. MIT Press, 1998.