Reinforcement Learning Policy Optimization:
A Unified Optimization Perspective on NPG, TRPO and PPO

Noah Stefancik
University of Rochester

December 9, 2025

1 Introduction

Reinforcement learning policy gradient methods such as NPG, TRPO, and PPO can be viewed from an optimization perspective. We review theoretical guarantees and examine relations of these common algorithms. We then move on to empirical results of these algorithms and extensions.

2 Background and Notation

2.1 Markov Decision Process

We formulate an infinite horizon discounted Markov decision process (MDP) as $M = (S, A, P, C, \gamma)$ where S is our state space, A is our action space, P is our transition probability, $C : S \times A \to [0, C_{max}]$ is our cost function, and $\gamma \in (0, 1)$ is our discount factor. Denote a policy of our agent by $\pi : S \to \Delta_A$ where Δ_A is the probability simplex over A. Define our value function as

$$V^\pi(s) := \mathbb{E}_{s_0,\cdots}[\sum_{t=0}^{\infty} \gamma^t C(s_t, a_t) \mid \pi, s_0 = s]$$

Our Q function of the state action function is given by

$$Q^\pi(s, a) := \mathbb{E}_{s_0,a_0,\cdots}[\sum_{t=0}^{\infty} \gamma^t C(s_t, a_t) \mid \pi, s_0 = s, a_0 = a]$$

The advantage function is given as the difference between these

$$A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s)$$

Note we use a cost function here rather than a reward function to align with the optimization standard of minimization. Denote our initial state distribution as μ. Also denote the non normalized discounted visitation frequencies as

$$\rho^\pi(s) = \sum_{k=1}^{\infty} \gamma^t Pr(s_t = s \mid \pi, \mu)$$

3 Policy Gradients

We parameterize π by θ and we can reformulate the reinforcement learning problem as an optimization problem

$$\begin{aligned}\theta^* &= \text{argmin}_\theta V^{\pi_\theta}(\mu) \\ &= \text{argmin}_\theta f(\theta)\end{aligned}$$

For notational simplicity we use $f(\theta) = V^{\pi_\theta}(\mu)$.

We can take the optimization problem above and apply a standard gradient descent yielding

$$\theta_{k+1} = \theta_k + \eta \nabla_\theta f(\theta)$$

Note there is a closed form (1)

$$\nabla_\theta f(\theta) = \sum_s \rho^{\pi_{\theta_k}}(s) \sum_a \nabla_\theta \pi_{\theta_k}(a \mid s) Q^{\pi_{\theta_k}}(s, a)$$

A natural extension to this method is from optimization is to take the first order Taylor approximation and a divergence term.

$$\theta_{k+1} = \text{argmin}_\theta \{f(\theta_k) + (\theta - \theta_k)^\top \nabla_\theta f(\theta_k) + \eta D_w(\theta, \theta_k)\}$$

Note we can drop the first term inside the argument since it does not depend on θ so we are left with the update rule

$$\theta_{k+1} = \text{argmin}_\theta \{(\theta - \theta_k)^\top \nabla_\theta f(\theta_k) + \eta D_w(\theta, \theta_k)\}$$

4 Natural Policy Gradient (NPG) (2)

If we take our divergence function w to be the negative entropy then we result in the KL divergence. We do this for NPG and it results in the update scheme

$$\theta_{k+1} = \text{argmin}_\theta \{(\theta - \theta_k)^\top \nabla_\theta f(\theta_k) + \eta KL(\pi_{\theta_k}(\cdot \mid s) \| \pi_\theta(\cdot \mid s)\}$$

where

$$\begin{aligned}KL(\pi_{\theta_k}(\cdot \mid s) \| \pi_\theta(\cdot \mid s) &= \sum_a \pi_{\theta_k}(a \mid s) \log(\frac{\pi_{\theta_k}(a \mid s)}{\pi_\theta(a \mid s)}) \\ &= \mathbb{E}_{a \sim \pi_{\theta_k}(\cdot|s)}[\log \frac{\pi_{\theta_k}(\cdot \mid s)}{\pi_\theta(\cdot \mid s)}]\end{aligned}$$

Unfortunately this minimization problem has no closed form and so we take the second order Taylor approximation KL divergence. Locally the zeroeth and first order terms are 0 and so we have

$$KL(\pi_{\theta_k}(\cdot \mid s) \| \pi_\theta(\cdot \mid s) \approx \frac{1}{2}(\theta - \theta_k)^\top F(\theta_k)(\theta - \theta_k) + O(\|\theta - \theta_k\|^3)$$

Now we transform the problem into a constraint optimization problem yielding

$$\theta_{k+1} = \arg\min_\theta \{(\theta - \theta_k)^\top \nabla_\theta f(\theta_k)\} \quad \text{s.t.} \quad \frac{1}{2}(\theta - \theta_k)^\top F(\theta_k)(\theta - \theta_k) \leq \delta$$

Computing the solution of the resulting problem yields

$$\theta_{k+1} = \theta_k - \sqrt{\frac{2\delta}{\nabla_\theta f(\theta_k)^\top F^{-1}(\theta_k)\nabla_\theta f(\theta_k)}}F^{-1}(\theta_k)\nabla_\theta f(\theta_k)$$

Alternatively we can change this to just a parameter step size which yields

$$\theta_{k+1} = \theta_k - \frac{1}{\lambda}F^{-1}(\theta_k)\nabla_\theta f(\theta_k)$$

5 Trust Region Policy Optimization (TRPO) (3)

5.1 Trust Region methods

Trust region methods are a class of methods that at each step approximate our function f over a small region. We "trust" this approximation to be close to the actual value of the function over this region. The subproblem that we reduce to could be over a ball, ellipsoid, or any valid region and the approximation could be any valid approximation but for the sake of this paper we take the most standard case of a Euclidean ball of radius r_k and the second order Taylor approximation of f

$$x_{k+1} = \min_{B(x_k,r_k)}\{f(x_k) + \langle\nabla f(x_k), x - x_k\rangle + \frac{1}{2}(x - x_k)^\top\nabla^2 f(x_k)(x - x_k)\}$$

TRPO enforces a trust region over the distibution space and emposes a KL ball constraint.

5.2 TRPO

Now we describe TRPO which takes inspiration and can be viewed as a trust region method. Note that we can express the change in our objective in terms of the advantage function

$$\begin{aligned}
f(\pi_\theta) &= f(\pi_{\theta_k}) + \mathbb{E}_{s_0,a_0,\cdots\sim\pi_\theta}[\sum_{t=0}^{\infty}\gamma^t A^{\pi_{\theta_k}}(s_t,a_t)] \\
&= f(\pi_{\theta_k}) + \sum_{t=0}^{\infty}\gamma^t\sum_s Pr(s_t = s \mid \pi_\theta)\sum_a \pi_\theta(a \mid s)A^{\pi_{\theta_k}}(s_t,a_t) \\
&= f(\pi_{\theta_k}) + \sum_s \rho^{\pi_\theta}(s)\sum_a \pi_\theta(a \mid s)A^{\pi_{\theta_k}}(s_t,a_t)
\end{aligned}$$

We introduce a new approximation since the dependency of $\rho^{\theta_\pi}(s)$ on θ_π is complex

$$L_{\pi_{\theta_k}}(\pi_\theta) = f(\pi_{\theta_k}) + \sum_s \rho^{\pi_{\theta_k}}(s)\sum_a \pi_\theta(a \mid s)A^{\pi_{\theta_k}}(s,a)$$

We define the following

$$KL^{\max}(\pi_{\theta_k},\pi_\theta) = \max_s KL(\pi_{\theta_k}(\cdot \mid s)\|\pi_\theta(\cdot \mid s))$$

$$\pi' = \arg\min_\pi L_{\pi_{\theta_k}}(\pi) \qquad \epsilon = \min_s \mathbb{E}_{a\sim\pi'(a|s)}[A^{\pi_{\theta_k}}(s,a)]$$

It can be shows that the following algorithm is non-increasing. This aligns with trust region where there is an optimal point within the trust region that is guaranteed to be as good or better than the current point, but outside of this region there is no such guarantee.

$$\pi_{k+1} = \arg\min_{\pi} \left[L_{\pi_k}(\pi) - C\, KL^{\max}(\pi_k, \pi) \right],$$
$$\text{where} \quad C = \frac{4\epsilon\gamma}{(1-\gamma)^2}.$$

To allow robustly large updates we use a constraint form of this and it follows that TPRO approximates this trust region method. We also take a heuristic of the KL terms since it is impractical due to the large number of constraints

$$KL^{\rho}(\pi_{\theta_k}, \pi_\theta) = \mathbb{E}_{s\sim\rho}[KL(\pi_{\theta_k}(\cdot \mid s), \pi_\theta(\cdot \mid s)]$$

It follows that the TRPO algorithm is given by

$$\min_{\theta} L_{\pi_{\theta_k}}(\pi_\theta) \quad \text{s.t.} \quad KL^{\rho^{\pi_{\theta_k}}}(\pi_{\theta_k}, \pi_\theta) \le \delta$$

5.3 Interpretations of TRPO

Note that the NPG method can be derived from this formulation. If we take the linear approximation of L and a quadratic approximation of the KL we yield exactly NPG. Note that this differs from TRPO which enforces the constraint at each update. One way this algorithm is implemented is to take a step in the natural gradient given this approximation. We then perform a backtracking line search to make sure the KL constraints is satisfied. Below is the algorithm for the backtracking line search.

```
**Algorithm:** TRPO Line Search
Compute   $D = \sqrt{\frac{2\delta}{\nabla_\theta f(\theta_k) F^{-1}(\theta_k) \nabla_\theta f(\theta_k)}}$
**for** $i = 0$ to $L$ **do**
    Compute $\theta = \theta_k - \alpha^i D$
    IF $KL(\pi_{\theta_{k+1}}, \pi_\theta) \le \delta$
        $\theta_{k+1} = \theta$
        **end for**
return $\theta_{k+1}$
```

Note this is just one way to implement the algorithm in a practical sense.

6 Proximal Policy Optimization (PPO) (4)

6.1 PPO

We introduce some notation and equations before we continue. Mainly we have the ratio

$$r_\theta(s,a) = \frac{\pi_\theta(a \mid s)}{\pi_{\theta_k}(a \mid s)}$$

It follows that we can write dropping the constant $f(\pi_{\theta_k})$. I drop the π in the notation so that this is clear as well

$$\begin{aligned} L_{\pi_{\theta_k}}(\theta) &= \sum_s \rho^{\pi_{\theta_k}}(s) \sum_a \pi_\theta(a \mid s) A^{\pi_{\theta_k}}(s,a) \\ &= \mathbb{E}_{s\sim\rho^{\pi_{\theta_k}}, a\sim\pi_{\theta_k}}[r_\theta(s,a) A^{\pi_{\theta_k}}(s,a)] \end{aligned}$$

PPO was heavily inspired by TRPO. In fact the motivation comes from the section of the TRPO paper which presented the KL penalty and not the constrained optimization problem. The KL penalty version of TRPO takes form

$$\theta_{k+1} = \arg\min_{\theta}\{L_{\pi_{\theta_k}}(\theta) - \beta \mathbb{E}_{s\sim\rho^{\pi_{\theta_k}}}[KL(\pi_{\theta_k}(\cdot \mid s), \pi_\theta(\cdot \mid s))]\}$$

It was realized that tuning β was impossible for multiple different tasks, or even over a singular task but in different stages of learning so PPO was created to alter the surrogate objective slightly. The clipped surrogate version of PPO uses a the clipped surrogate

$$L^{\text{clip}}_{\pi_{\theta_k}}(\theta) = \mathbb{E}_{s\sim\rho^{\pi_{\theta_k}}, a\sim\pi_{\theta_k}}[\max(r_\theta(s,a)A^{\pi_{\theta_k}}(s,a), \text{clip}(r_\theta(s,a), 1-\epsilon, 1+\epsilon)A^{\pi_{\theta_k}}(s,a))]$$

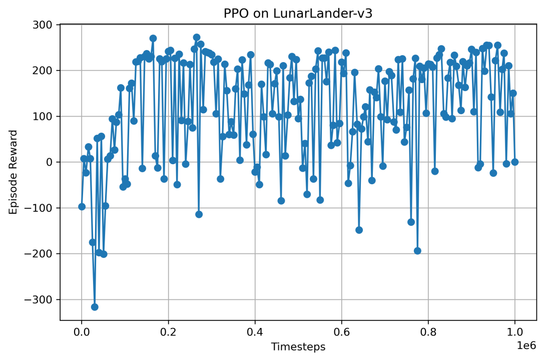
Where $\epsilon \in (0,1)$ is a tunable parameter. Our update scheme is simply

$$\theta_{k+1} = \arg\min_{\theta} L^{\text{clip}}_{\pi_{\theta_k}}(\theta)$$

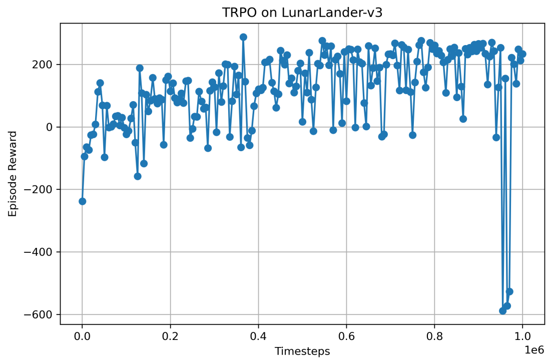
Another algorithm which was proposed in the paper which originally introduced multiple "proximal policy optimization algorithms" is given in terms of the KL penalty. That being said when people refer to PPO as a singular algorithm they typically mean the clipped version. This algorithm much closer resembles a proximal point algorithm, while PPO although inspired from proximal point like algorithms is not itslef a proximal point method.

7 Experiments

I implemented NPG and used stable-baselines3 as a library for implementations of PPO and TRPO. Gymnasium environments were used for the testing. I tested PPO and TRPO in the LunarLander-v3 task. Below I show graphs of the reward versus the timesteps. Note that we used cost instead of reward in this paper to align with the minimization framework in optimization.



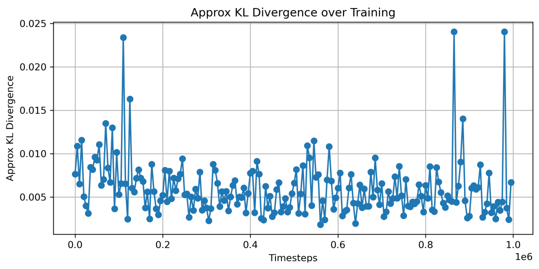
(a) PPO on the task



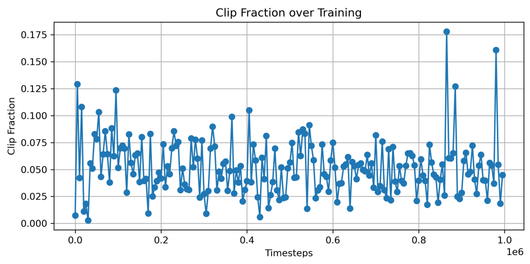
(b) TRPO on the task

Figure 1: Comparison of PPO and TRPO performance on LunarLander-v3.

Both TRPO and PPO attain similar rewards on the LunarLander-v3 task. Note that TRPO seems slightly more stable than PPO on this task. It also seems approach slower than PPO on this task especially originally, but hits the negative values much less frequently.



(a) PPO KL



(b) Clip ratio on PPO

Figure 2: Comparison of PPO and TRPO performance on LunarLander-v3.

Note how the KL is not a very constant value. This is what make choosing a specific β in the motivation for PPO quite difficult. Even over the same task the KL value is quite different. Also notice the clip fraction does change quite a lot over this task as well.

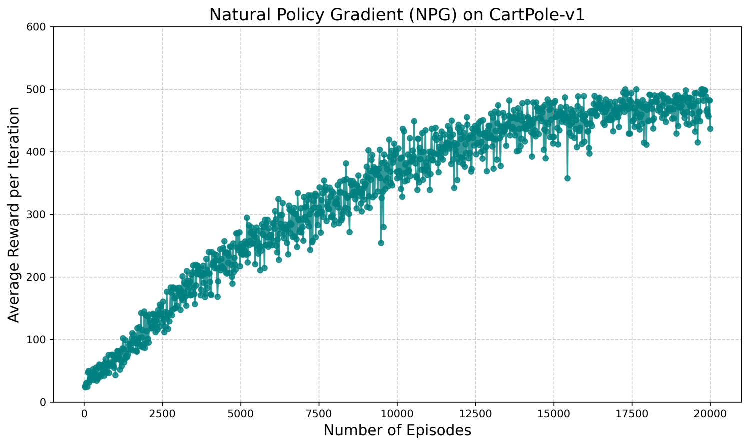


Figure 3: NPG on CartPole-v1

Cartpole is a classic and fairly simple reinforcement learning task. My implementation of NPG approaches the best solution where the reward is 500. It is fairly stable and does not jump too much at any timestep.

8 Related Extensions

We detail some extensions to each of these methods and framework which are of particular note.

8.1 Revisiting Natural Policy Gradients (5)

Natural gradients show up throughout many different optimization and reinforcement learning methods. This paper examines some methods proposed after the natural gradient was found.

Some later methods that they examined were Hessian Free-Optimization, Krylov Subspace Descent, and Topmoumoute online natural gradient algorithm (TONGA).

They found that both Hessian-Free Optimization and Krylov Subspace Descent were actually implementations of natural gradient descent.

8.2 Adaptive Trust Region and Mirror Descent Policy Optimization (6) (7)

The first paper argues that the view of TRPO a heuristic algorithm inspired by Conservative Policy Iteration (CPI) is naive. They showed that the TRPO's adaptive scaling mechanism is a natural RL version of of trust region methods.

The second paper examines TPRO from a mirror descnet perspective. The on policy MDPO shows and can be seen as a mirror descent perspective of TRPO.

9 Conclusion

In this report we examined NPG, TRPO, and PPO through a unified optimization perspective. We can put them into the language of constrained optimization, proximal methods, and mirror descent which reveal structural connections and similarities.

NPG emerges naturally by using a KL divergence and results in quite an elegant formulation. TRPO further extends this idea by enforcing an explicit trust region over the policy space, which under some assumptions can guarantee monotonic improvement of the policy. PPO, while inspired by TRPO, replaces the hard KL constraint with a clipping mechanism that achieves similar stability without requiring second order approximations or a line search.

Finally, we briefly discussed several extensions and reinterpretations. Together, these results illustrate how many RL algorithms can be understood as variations on a small number of optimization themes.

References

[1] Sutton, R. S., McAllester, D., Singh, S., Mansour, Y. (1999). Policy Gradient Methods for Reinforcement Learning with Function Approximation. *Advances in Neural Information Processing Systems, 12*, 1057–1063. https://papers.nips.cc/paper/1713-policy-gradient-methods-for-reinforcement-learning-with-function-approximation.pdf

[2] Kakade, S. (2001). *A natural policy gradient* (Doctoral dissertation, University of London). `https://proceedings.neurips.cc/paper_files/paper/2001/file/4b86abe48d358ecf194c56c69108433e-Paper.pdf`

[3] Schulman, J., Levine, S., Abbeel, P., Jordan, M., Moritz, P. (2015). Trust Region Policy optimization. arXiv (Cornell University), 1889–1897. https://doi.org/10.48550/arxiv.1502.05477

[4] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O. (2017). Proximal policy optimization algorithms. arXiv. https://arXiv.org/abs/1707.06347

[5] Pascanu, R., Bengio, Y. (2014). Revisiting Natural Gradient for Deep Networks. Axiv. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.767.8023

[6] Shani, L., Efroni, Y., Mannor, S. (2019). Adaptive Trust Region Policy Optimization: Global Convergence and Faster Rates for Regularized MDPs. Axiv. https://arxiv.org/pdf/1909.02769page14

[7] Efroni, Y., Ghavamzadeh, M., Tomas, M. (2021). Mirror Descent Policy Optimization [Journal-article]. https://arxiv.org/pdf/2005.09814