Comparing Trust Region Policy Optimization and Natural Policy Gradient in Reinforcement Learning

Jerke van den Berg (11034106)

Ruihan Sun (12516414)

Tom Lieberum (13253042)

Erik Jenner (13237896)

1 Introduction

Policy optimization algorithms have become very popular in reinforcement learning, as they are applicable in continuous action spaces. They are also efficient, as they directly optimize the solution (policy) we are interested in when solving a problem.

The use of gradient methods for policy optimization has received increasing attention over the last few years, as they have been proven to be quite useful for challenging sequential decision-making tasks (Lovatto et al., 2019). For example, playing Atari games from pixels and solving simulated robotics tasks are progressing faster than before with the use of gradient methods for policy optimization.

Two popular gradient-based policy optimization algorithms are **Trust Region Policy Optimization** (TRPO) and **Natural Policy Gradient** (NPG). Both update their parameters in the same direction, but their step sizes are chosen in different ways – specifically, TRPO can be interpreted as adapting its learning rate at each step. In this report, we determine how large these learning rate adaptations are and whether there is any interpretable trend in them. We also analyze the effect of different environments and parameters on these questions. **TODO:** Mention KL update sizes or other things we end up including **TODO:** Give some motivation for why we investigate step size specifically

1.1 Natural Policy Gradient

With NPG, we calculate the Fisher Information matrix of our policy formula. This matrix consists of the second order derivative of the log of the policy formula with respect to all possible combinations of parameters in our formula. By multiplying the gradients of the current parameters (put into a vector) with the inverse of this matrix, we take into account parameterization for less noise in our updates compared to vanilla gradient ascent. By making use of the Fisher Information matrix, we also ensure that the KL divergence between our updates is constant.

1.2 Trust Region Policy Optimization

NPG is motivated by achieving updates with a fixed KL divergence. But the size of its step does not explicitly depend on a maximum KL divergence but instead on a less interpretable learning rate. Furthermore, the KL divergence between policies is not actually constant since the underlying derivation assumes infinitesimal updates.

TRPO improves upon both of these issues: it updates the parameters in the same direction as NPG but with a step size that depends only on an explicitly chosen maximum KL divergence. Instead of always taking the full step it also backtracks to ensure that the actual KL divergence is no larger than that maximum (it can however be smaller).

2 Related work

The first distinction that can be made between TRPO and NPG is by their use of hyperparamters. In TRPO, a maximum KL divergence between the old policy and the new is used as a constraint directly. NPG uses a (fixed) learning rate to control its update stepsize alongside a KL divergence constraint met in the second order derivative approximation. The effect of both these approaches has been researched by (Paul et al., 2019), which argue that though TRPO has an advantage by enforcing KL divergence as a constraint directly, the difference is much less noticeable when applying efficient hyperparameter tuning for NPG.

(Shani et al., 2019) showed that using TRPO efficiently results in an iteration complexity of $O(1/\epsilon)$. This result is compared to the entropy-regularized approach for NPG by (Cen et al., 2020), which outperforms this measure.

3 Methods

To compare the performance of TRPO and NPG, we will run experiments on MountainCar-V0, MountainCarContinuous-V0, Pendulum-V0 and CartPole-V1. These environments are easy and well implemented in OpenAI baseline (Dhariwal et al., 2017), which allows us to study our research question more easily. Since they are also computationally inexpensive, we are able to run a relatively large number of experiments. They are also diverse: MountainCar-V0 and CartPole-V1 are discrete control environments but MountainCar-V0 is more stable. MountainCarContinuous and Pendulum on the other hand have continuous action spaces, which makes them a natural fit for our study of policy optimization methods.**TODO**

We train an Actor-Critic model with TRPO and with NPG on all of these environments. For discrete action spaces, we use a categorical distribution to model the policy, for continuous spaces we use a normal distribution with input-dependent variance and mean. In addition to different environments, we use different values for the maximum KL divergence (TRPO) or learning rate (NPG). We chose to focus on varying these hyperparameters because they are specific to the question we're investigating. While things like the number of layers of the Critic or the activation functions we use might have an effect on performance, they are likely not as important for the general trends of step sizes as the parameters that directly influence the step size calculations. **TODO:** Also vary network architecture, policy distributions, GAE params, other things?

TODO: Describe network architecture, maybe in appendix?

All experiments are run with at least five different seeds to determine the range of fluctuations (Henderson et al. (2018) demonstrates that outcomes can vary strongly across different random seeds). The seeds are used for both the OpenAI gym environment, as well as PyTorch, ensuring completely deterministic runs. The final results are created by a run with previously unused seeds, to avoid accidentally fitting to specific seeds.

Our implementation of NPG and TRPO is based on Lee (2019), with some modifications to allow us to run all our experiments.

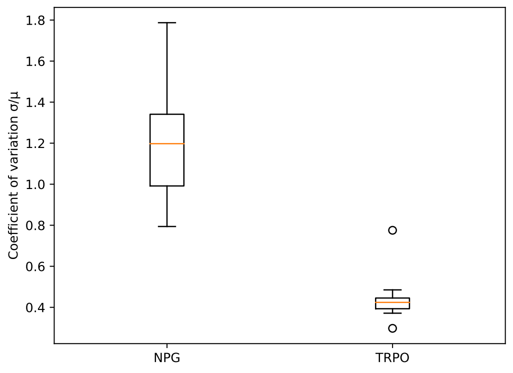


Figure 1: Coefficient of variation (CV) of KL updates for NPG and TRPO. The CV was calculated across all the updates in one episode. The box plot then shows the spread of the CV over 10 different seeds. The environment was CartPole-v1.

4 Experiments and Results

The adaptive step size of TRPO is designed to lead to a KL divergence between old and new policy that is roughly the same for each update. Because of the finite step sizes, this will not hold exactly, but we expect the KL divergences to vary less than for NPG. Figure 1 shows that this is indeed the case: the coefficient of variation of KL divergences between policies, i.e. the normalized standard deviation, is significantly larger for NPG than for TRPO. The underlying KL divergences are all shown in fig. 2. We can also observe that NPG sometimes takes very large steps (in terms of KL divergence) and much smaller steps most of the time. For TRPO this is more uniform. So NPG has a constant step size but this does not translate to constant KL divergences. In contrast, TRPO varies its step size to achieve KL divergences that are closer to constant. **TODO:** Test this on other environments as well.

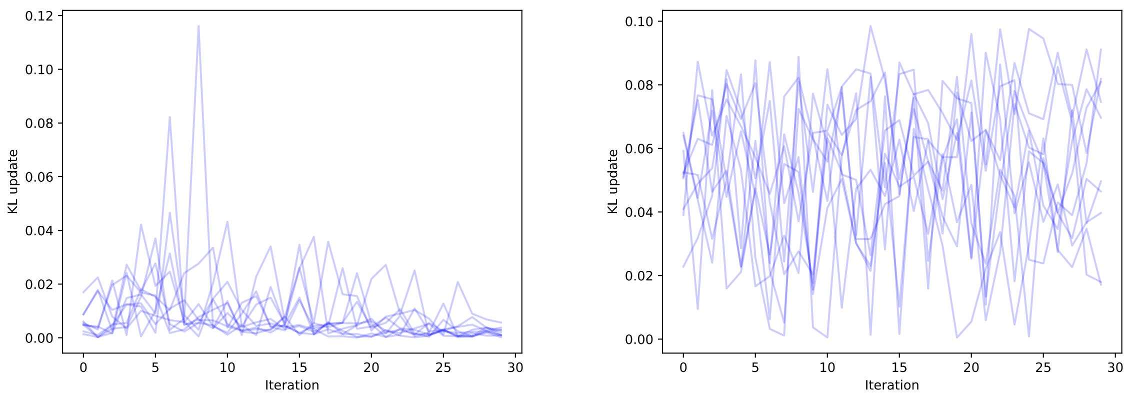
We can also see this from a different perspective, by displaying the step-size. Here we would expect a constant line for NPG and a variable step-size for TRPO. The step-sizes for the Pendulum-v0 environment are shown in fig. 3

5 Discussion

References

Shicong Cen, Chen Cheng, Yuxin Chen, Yuting Wei, and Yuejie Chi. 2020. Fast global convergence of natural policy gradient methods with entropy regularization. *arXiv preprint arXiv:2007.06558*.

Prafulla Dhariwal, Christopher Hesse, Oleg Klimov,



(a) NPG (b) TRPO

Figure 2: KL divergences for 10 runs with different random seeds on the CartPole-v1 environment

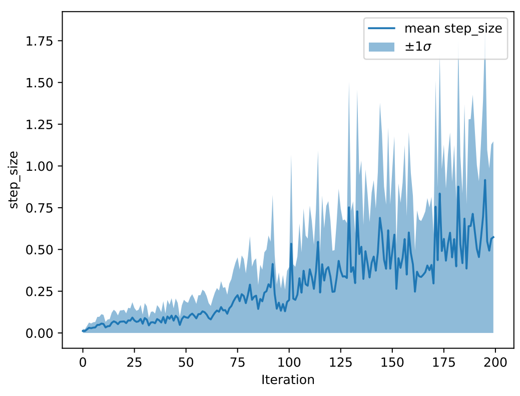


Figure 3: Step-size for TRPO in the Pendulum-v0 environment, averaged over 4 seeds.

Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, Yuhuai Wu, and Peter Zhokhov. 2017. Openai baselines. `https://github.com/openai/baselines`.

Peter Henderson, R. Islam, Philip Bachman, Joelle Pineau, D. Precup, and D. Meger. 2018. Deep reinforcement learning that matters. *ArXiv*, abs/1709.06560.

Woongwon Lee. 2019. Policy gradient algorithms. `https://github.com/reinforcement-learning-kr/pg_travel`.

Ângelo Gregório Lovatto, Thiago Pereira Bueno, and Leliane Nunes Barros. 2019. Analyzing the effect of stochastic transitions in policy gradients in deep reinforcement learning. In *2019 8th Brazilian Conference on Intelligent Systems (BRACIS)*, pages 413–418. IEEE.

Supratik Paul, Vitaly Kurin, and Shimon Whiteson. 2019. Fast efficient hyperparameter tuning for policy gradient methods. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 4616–4626. Curran Associates, Inc.

Lior Shani, Yonathan Efroni, and Shie Mannor. 2019. Adaptive trust region policy optimization: Global convergence and faster rates for regularized mdps. *arXiv preprint arXiv:1909.02769*.

A