

“Implementation Matters in Deep Policy Gradients: A Case Study on PPO and TRPO”

Alexey Larionov (MSc-2 IST)
Ilya Barskiy (MSc-2 IST)

RL 2021 Course

Skoltech



Project task

- Study implementation of Trust Region Policy Optimization (TRPO) and Proximal Policy Optimization (PPO) methods for OpenAI Gym problems.
- Apply “code-level optimizations” to the implementations as described in the paper ¹
- Verify some of the paper’s results about true nature of PPO’s gain in cumulative reward over TRPO

¹ Engstrom L, Ilyas A, Santurkar S, Tsipras D, Janoos F, Rudolph L, Madry A. Implementation matters in deep policy gradients: A case study on ppo and trpo. arXiv preprint arXiv:2005.129.272020 May 25.

<https://arxiv.org/abs/2005.12729>

Preliminary theory

TRPO:

$$\rho_t = \frac{\pi_{\theta}(a_t|s_t)}{\pi(a_t|s_t)}$$

PPO

Importance sampling with KL divergence constraint. Updated policy is improved, and kept in vicinity of the previous policy. Optimal step needs to compute Hessians.

Enforces the trust region by clipping policies ratios around 1. Practically it approximates KL divergence without directly computing it. More sample efficient than TRPO.

$$\begin{aligned} & \max_{\theta} \mathbb{E}_{(s_t, a_t) \sim \pi} [\rho_t \hat{A}_{\pi}(s_t, a_t)] \\ \text{s.t. } & D_{KL}(\pi_{\theta}(\cdot|s) \parallel \pi(\cdot|s)) \leq \delta, \quad \forall s \end{aligned}$$

$$\begin{aligned} & \max_{\theta} \mathbb{E}_{(s_t, a_t) \sim \pi} [\min(\rho_t \hat{A}_{\pi}(s_t, a_t), \\ & \quad \text{clip}(\rho_t; [1 - \varepsilon, 1 + \varepsilon]) \hat{A}_{\pi}(s_t, a_t))] \end{aligned}$$

Code-level optimizations of PPO

1. Value network's loss with a clipped value function

$$L^V = \max[(V_{\theta_t} - V_{targ})^2, (\text{clip}(V_{\theta_t}, V_{\theta_{t-1}} - \varepsilon, V_{\theta_{t-1}} + \varepsilon) - V_{targ})^2]$$

2. Reward Scaling

$$R_t \leftarrow \gamma R_{t-1} + r_t \quad r_t \leftarrow \frac{r_t}{\text{std}\{R_i\}_{i=0}^t}$$

3. Reward Clipping

$$r_t \in [-5, 5] \text{ or } r_t \in [-10, 10]$$

4. Observation Normalization (mean-zero, variance-one)

5. Observation Clipping

$$s_t \in [-10, 10]$$

6. Global Gradient Clipping

$$\|\nabla_{\vartheta}\|_{l_2} < 0.5$$

Code-level optimizations of PPO

6. Orthogonal initialization & scaling

weights matrices contain orthogonal columns/rows

different layers have different initial scales

8. Adam learning rate annealing

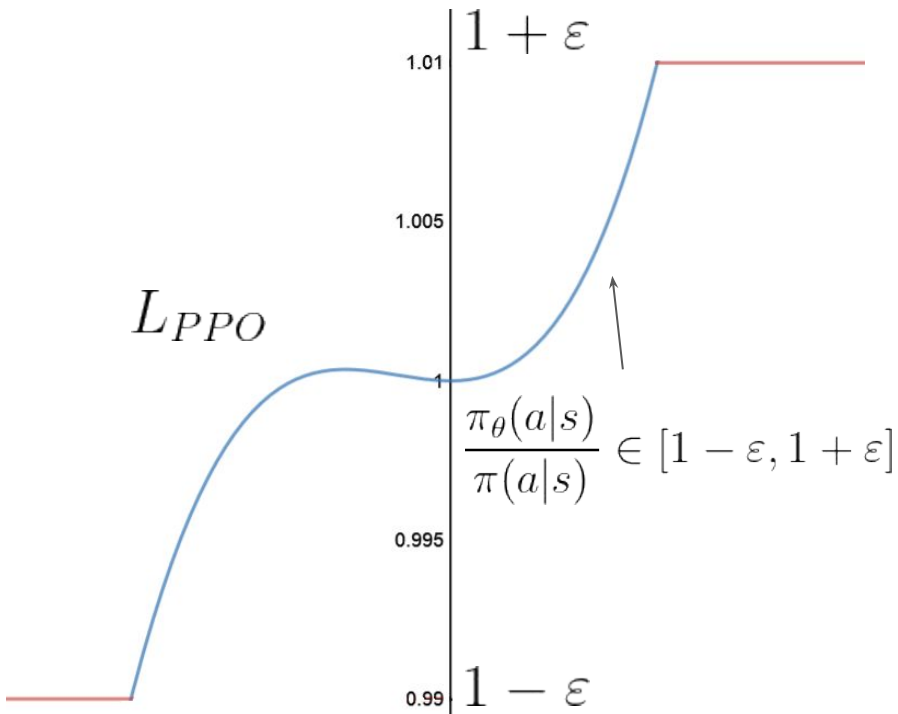
depending on the task, a plethora of annealing schemes is tested

9. Hyperbolic tangent activations

both for policy and value networks

Underlying reasons

PPO is optimized over clipped objective, but the gradient is unbounded



- If policies ratio is clipped, $\nabla_\theta L_{PPO} = 0$
- If the ratio is near 1, $\nabla_\theta L_{PPO} \neq 0$ and the gradient magnitude can be large enough to leave the trust region

Paper's hypothesis: size of the step is determined solely by the steepness of the surrogate landscape (which is improved via code-level optimizations)

Methods

Train configurations

- TRPO and PPO as implemented in [OpenAI baselines](#)
- TRPO_MAX - a TRPO extended with code-level optimizations from PPO
- PPO_MIN - a PPO with disabled code-level optimizations

Compare them by reward gain, and also by

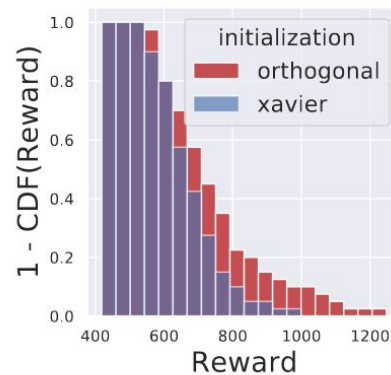
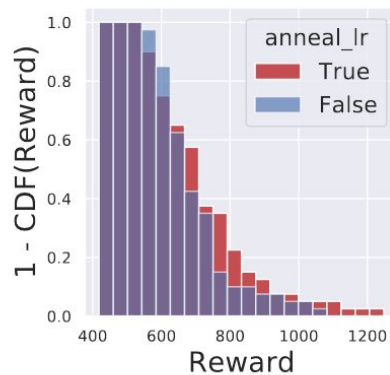
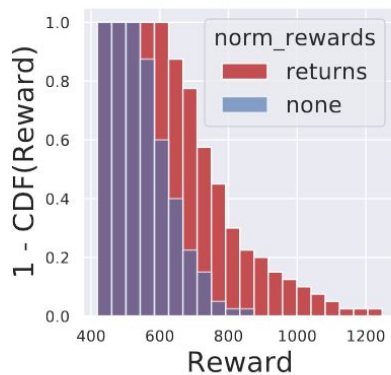
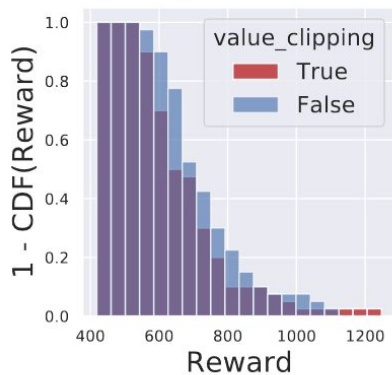
Average code-level improvement; Average algorithmic improvement

$$ACLI \triangleq \max\{|PPO - PPO_MIN|, |TRPO_MAX - TRPO|\}$$

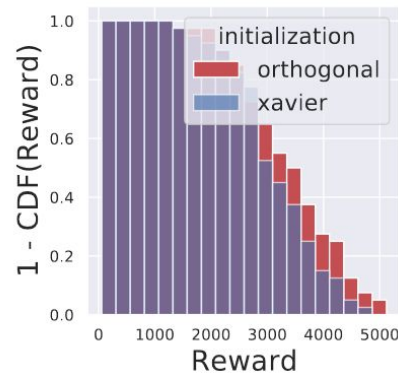
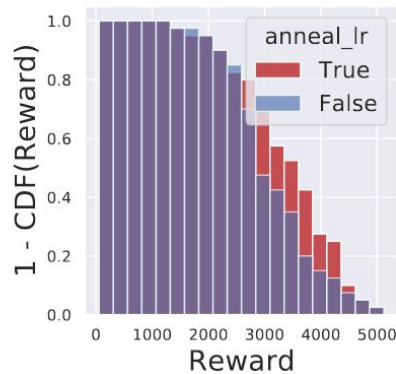
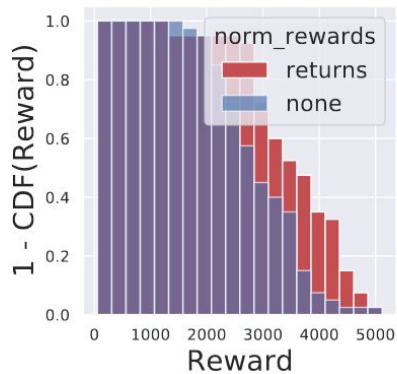
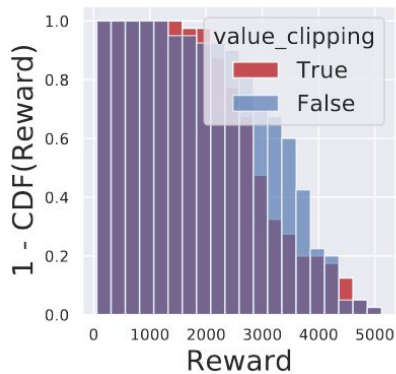
$$AAI \triangleq \max\{|PPO - TRPO_MAX|, |PPO_MIN - TRPO|\}$$

Methods, non-replicated

Humanoid-v2



Walker2d-v2





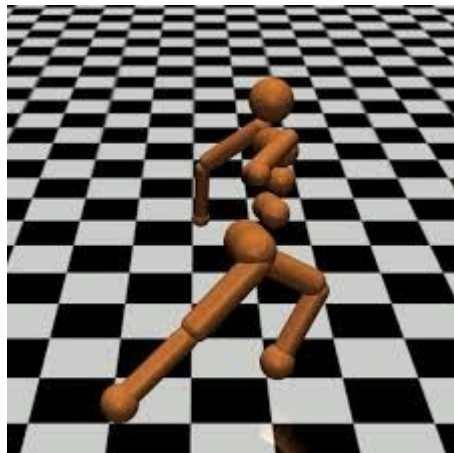
Repository

<https://github.com/laralex/Sk-reinforcement-learning>

Based on MuJoCo environments (Humanoid-v2, Hopper-v2)

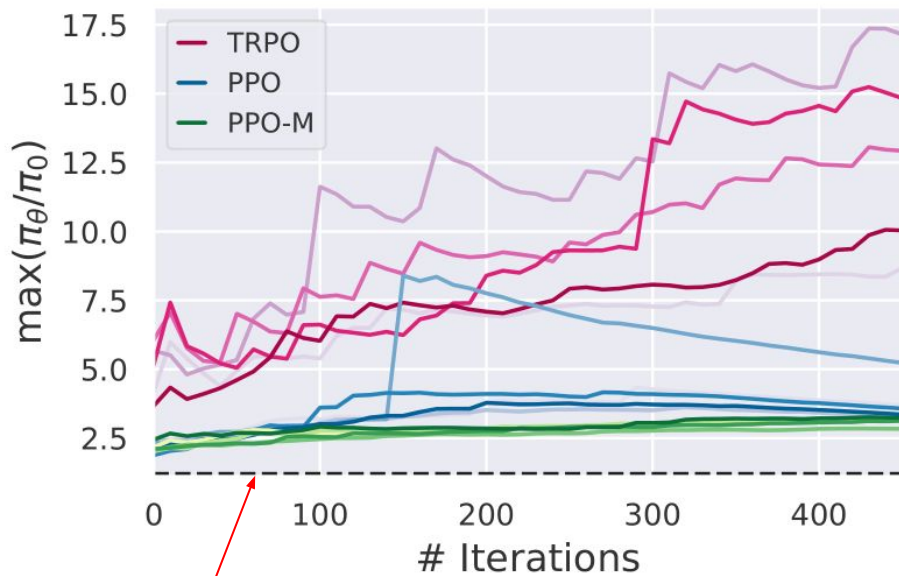
Run with ``python run.py --render configs/{config_name}.yaml``

-  [run.py](#) - the loop of training for the given config
- [configs/](#) - YAML files with parameters of all experiments
-  [src/code_level_optim.py](#) - all the code for code-level optimizations
- [src/actor.py](#) - learnable policy Neural Network (MLP)
- [src/critic.py](#) - learnable value function Neural Network (MLP)
- [src/utility.py](#) - command line arguments parsing, experiment's config parsing
- [materials/](#) - images, the presentation, the original paper file

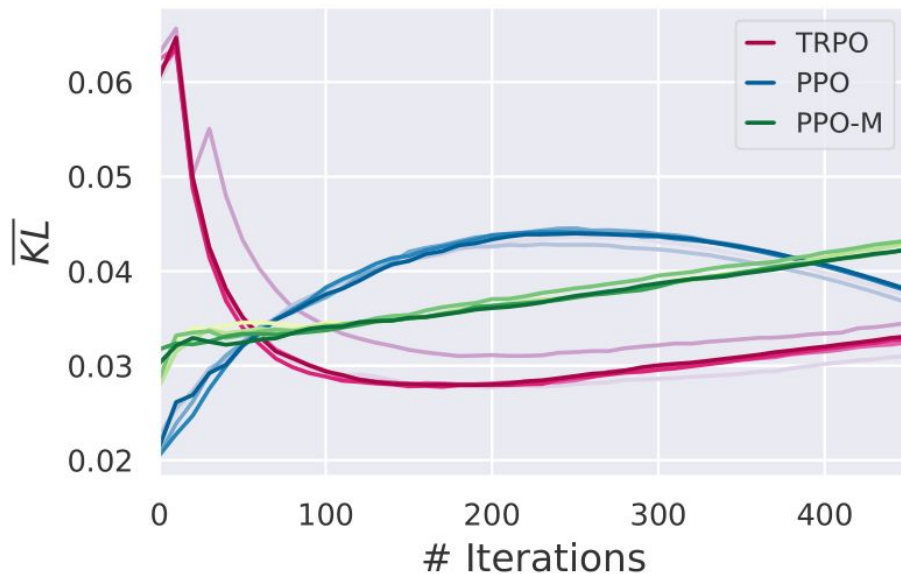


Results

We've implemented the code-level optimizations, but couldn't implement TRPO in time without copy-pasting it from the referenced paper



clipping epsilon = 0.2



Results

$$ACLI \triangleq \max\{|PPO - PPO_MIN|, |TRPO_MAX - TRPO|\}$$

$$AAI \triangleq \max\{|PPO - TRPO_MAX|, |PPO_MIN - TRPO|\}$$

STEP	MuJoCo TASK		
	WALKER2D-V2	HOPPER-V2	HUMANOID-V2
PPO	3292 [3157, 3426]	2513 [2391, 2632]	806 [785, 827]
PPO-MIN	2735 [2602, 2866]	2142 [2008, 2279]	674 [656, 695]
TRPO	2791 [2709, 2873]	2043 [1948, 2136]	586 [576, 596]
TRPO-MAX	3050 [2976, 3126]	2466 [2381, 2549]	1030 [979, 1083]
AAI	242	99	224
ACLI	557	421	444

Conclusion

- We investigated the intuition of TRPO and PPO in more details
- Implemented code-level optimizations
- We couldn't replicate results of full ablation study, as it requires extensive grid search over a lot of hyperparameters
- Mania, H., Guy, A., & Recht, B. (2018). Simple random search provides a competitive approach to reinforcement learning. *arXiv preprint arXiv:1803.07055*.

Implementation matters!

Thanks for the attention