南京航空航天大學

Nanjing University of Aeronautics and Astronautics

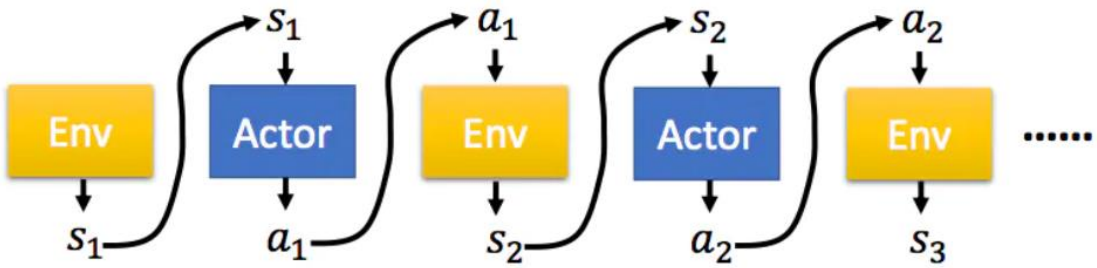
IMPLEMENTATION MATTERS IN DEEP POLICY GRADIENTS: A CASE STUDY ON PPO AND TRPO

Logan Engstrom*, Andrew Ilyas*, Shibani Santurkar[1], Dimitris Tsipras[1],
Firdaus Janoos[2], Larry Rudolph[1,2], and Aleksander Mądry[1]

[1]MIT [2]Two Sigma
{engstrom,ailyas,shibani,tsipras,madry}@mit.edu
rudolph@csail.mit.edu, firdaus.janoos@twosigma.com

ICLR 2019

Background-RL



Trajectory $\tau = \{s_1, a_1, s_2, a_2, \cdots, s_T, a_T\}$

$$\pi_\theta(\tau) = \pi(s_1)\pi_\theta(a_1 \mid s_1)\pi_\theta(s_2 \mid s_1, a_1)\pi_\theta(a_2 \mid s_2)\cdots$$

$$R(\tau) = \sum_i r_i \qquad \pi^* = \arg\max_\pi E_{\tau\sim\pi}[R(\tau)] \quad \tau = s_1, a_1, s_2, a_2 \cdots$$

Background-Policy gradient

$$J(\theta) = \sum_{\tau} \pi_\theta(\tau) R(\tau)$$

$$J(\theta) = E_t[\log \pi_\theta(a_t \mid s_t)(R(\tau) - \text{base})]$$

$$\begin{aligned}
\nabla_\theta J(\theta) &= \sum_{\tau} \nabla_\theta \pi_\theta(\tau) R(\tau) \\
&= \sum_{\tau} \nabla_\theta \pi_\theta(\tau) \frac{\pi_\theta(\tau)}{\pi_\theta(\tau)} R(\tau) \\
&= \sum_{\tau} \pi_\theta(\tau) \nabla_\theta \log \pi_\theta(\tau) R(\tau) \\
&= E[\nabla_\theta \log \pi_\theta(\tau) R(\tau)] \\
&= E_t[\nabla_\theta \log \pi_\theta(a_t \mid s_t) R(\tau)]
\end{aligned}$$

Policy Gradient Methods for Reinforcement Learning with Function Approximation

Background-Actor-Critic

Advantage function: $A_t = Q(s_t, a_t) - V(s_t)$

$$\Longrightarrow \quad J(\theta) = E_t[\pi_\theta(a_t \mid s_t) A_t]$$

Importance sampling: $E_{x\sim p}[f(x)] = \int f(x)p(x)dx = \int f(x)\frac{p(x)}{q(x)}q(x)dx = E_{x\sim q}\left[f(x)\frac{p(x)}{q(x)}\right]$

$$\Longrightarrow \quad J(\theta) = E_t\left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{old}}(a_t \mid s_t)} A_t\right]$$

Trpo: $\max_\theta \quad \mathbb{E}_{(s_t,a_t)\sim\pi}\left[\frac{\pi_\theta(a_t \mid s_t)}{\pi(a_t \mid s_t)}\widehat{A}_\pi(s_t, a_t)\right]$

Ppo: $\max_\theta \mathbb{E}_{(s_t,a_t)\sim\pi}\left[\min\left(\text{clip}(\rho_t, 1-\varepsilon, 1+\varepsilon)\widehat{A}_\pi(s_t, a_t), \rho_t\widehat{A}_\pi(s_t, a_t)\right)\right]$

Motivation

Existing deep RL methods to be brittle, hard to reproduce, unreliable across runs, and sometimes outperformed by simple baseline.

⇩

how do the multitude of mechanisms used in deep RL training algorithms impact agent behavior?

⇩

Maybe code-level optimizations fundamentally change algorithms' operation.

Analysis code-level optimizations

- Value function clipping.

Originally suggest: $L^V = (V_{\theta_t} - V_{targ})^2$

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

Analysis code-level optimizations

- Reward scaling.

Algorithm 1 PPO scaling optimization.

procedure INITIALIZE-SCALING()
    $R_0 \leftarrow 0$
    $RS =$ RUNNINGSTATISTICS() ▷ New running stats class that tracks mean, standard deviation
procedure SCALE-OBSERVATION(r_t) ▷ Input: a reward r_t
    $R_t \leftarrow \gamma R_{t-1} + r_t$ ▷ γ is the reward discount
    ADD(RS, R_t)
    return r_t/STANDARD-DEVIATION(RS) ▷ Returns scaled reward

Analysis code-level optimizations

- Orthogonal initialization and layer scaling:

  Instead of using the default weight initialization scheme for networks, the implementation uses an orthogonal initialization scheme with scaling that varies from layer to layer.

- Adam learning rate annealing:

  the implementation sometimes anneals the learning rate of for optimization.

Experiment

Table 1:List of algorithms studied in this work, with their crucial properties.

Algorithm	Section	Step method	Uses PPO clipping?	Uses PPO optimizations?
PPO	—	PPO	✓	As in (Dhariwal et al., 2017)
PPO-M	Sec. 3	PPO	✓	✗
PPO-NoClip	Sec. 4	PPO	✗	Found via grid search
TRPO	—	TRPO	—	✗
TRPO+	Sec. 5	TRPO	—	Found via grid search

Experiment

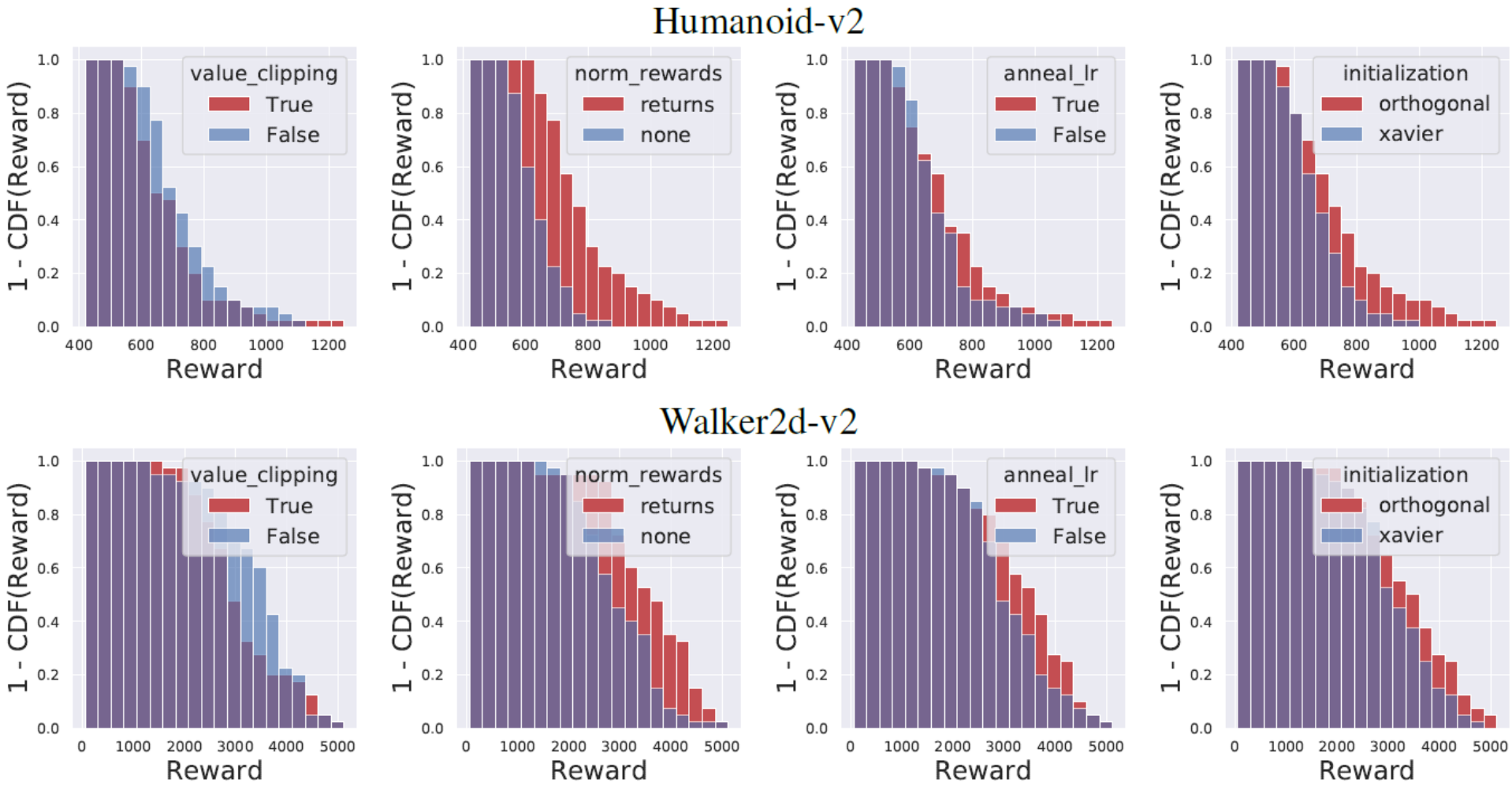
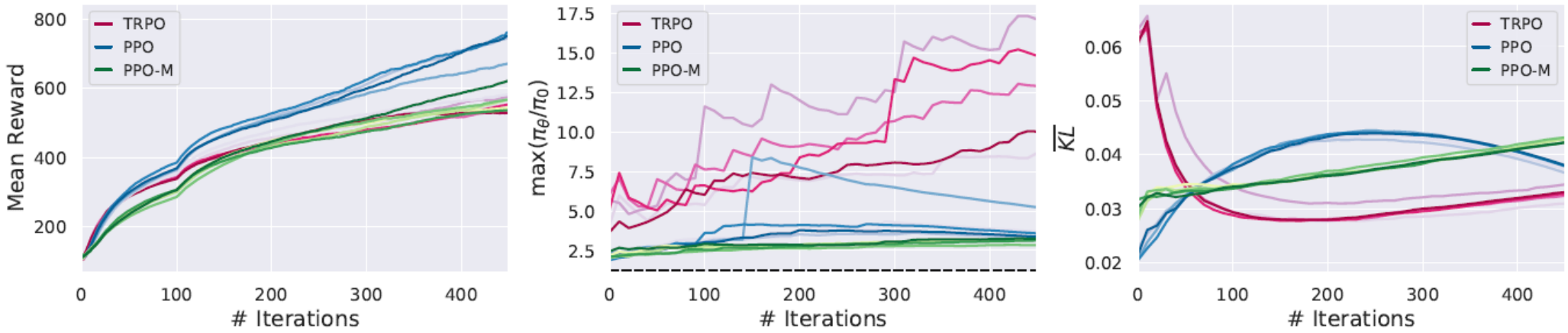


Figure 1:An ablation study on the first four optimizations (value clipping, reward scaling, network initialization, and learning rate annealing).

Experiment



Humanoid-v2(train)

Figure 2:Per step mean reward, maximum ratio (c.f. (2)), mean KL, and mean KL for agents trained to solve the MuJoCo Humanoid-v2 task.

Experiment

Table 2:Full ablation of step choices (PPO or TRPO) and presence of code-level optimizations measuring agent performance on benchmark tasks.

Step	MuJoCo Task Walker2d-v2	Hopper-v2	Humanoid-v2
PPO	3292 [3157, 3426]	2513 [2391, 2632]	806 [785, 827]
PPO-M	2735 [2602, 2866]	2142 [2008, 2279]	674 [656, 695]
TRPO	2791 [2709, 2873]	2043 [1948, 2136]	586 [576, 596]
TRPO+	3050 [2976, 3126]	2466 [2381, 2549]	1030 [979, 1083]
AAI	242	99	224
ACLI	557	421	444

$$\mathrm{AAI} = \max\{|\mathrm{PPO} - \mathrm{TRPO+}|, |\mathrm{PPO\text{-}M} - \mathrm{TRPO}|\},$$

$$\mathrm{ACLI} = \max\{|\mathrm{PPO} - \mathrm{PPO\text{-}M}|, |\mathrm{TRPO+} - \mathrm{TRPO}|\}.$$

Experiment

Table 3:Comparison of PPO performance to PPO without clipping.

	WALKER2D-V2	HOPPER-V2	HUMANOID-V2
PPO	3292 [3157, 3426]	2513 [2391, 2632]	806 [785, 827]
PPO (BASELINES)	3424	2316	—
PPO-M	2735 [2602, 2866]	2142 [2008, 2279]	674 [656, 695]
PPO-NOCLIP	2867 [2701, 3024]	2371 [2316, 2424]	831 [798, 869]

THANKS