Implementation Matters in Deep RL: A Case Study on PPO and TRPO[1][2]

[1] "Implementation Matters in Deep RL: A Case Study on PPO and TRPO" by Engstrom et al., ICLR 2020

[2] Summary: http://tiny.cc/zh0lnz

Overview

Overview

- Performance of many RL algorithms depends a lot on tricks used in implementation

Overview

- Performance of many RL algorithms depends a lot on tricks used in implementation
- Authors dived into PPO and found out that it is not an exception

Overview

- Performance of many RL algorithms depends a lot on tricks used in implementation
- Authors dived into PPO and found out that it is not an exception
- Key observation: tricks used in PPO give more boost than PPO itself

Overview

- Performance of many RL algorithms depends a lot on tricks used in implementation
- Authors dived into PPO and found out that it is not an exception
- Key observation: tricks used in PPO give more boost than PPO itself
- PPO+tricks works a bit better than TRPO+tricks

Off-policy PG vs TRPO vs PPO

Off-policy PG vs TRPO vs PPO

A common part between TRPO and PPO is the off-policy policy gradient (with importance sampling):

$$J_{\mathrm{PG}}(\theta) = \mathop{\mathbb{E}}_{(s_t, a_t) \sim \pi} \left[\frac{\pi_\theta \left(a_t | s_t\right)}{\pi \left(a_t | s_t\right)} \hat{A}_\pi \left(s_t, a_t\right) \right] \tag{1}$$

Off-policy PG vs TRPO vs PPO

A common part between TRPO and PPO is the off-policy policy gradient (with importance sampling):

$$J_{\mathrm{PG}}(\theta) = \mathop{\mathbb{E}}_{(s_t, a_t) \sim \pi} \left[\frac{\pi_\theta (a_t | s_t)}{\pi (a_t | s_t)} \hat{A}_\pi (s_t, a_t) \right] \tag{1}$$

TRPO differs from PG by constraining the optimization step:

$$\begin{array}{ll} \max_\theta & J_{\mathrm{PG}}(\theta) \\ \text{s.t.} & D_{KL}\left(\pi_\theta(\cdot|s) \| \pi(\cdot|s)\right) \leq \delta \end{array} \tag{2}$$

Off-policy PG vs TRPO vs PPO

A common part between TRPO and PPO is the off-policy policy gradient (with importance sampling):

$$J_{\mathrm{PG}}(\theta) = \mathop{\mathbb{E}}_{(s_t, a_t) \sim \pi} \left[\frac{\pi_\theta \left(a_t | s_t \right)}{\pi \left(a_t | s_t \right)} \hat{A}_\pi \left(s_t, a_t \right) \right] \tag{1}$$

TRPO differs from PG by constraining the optimization step:

$$\begin{array}{ll} \max_\theta & J_{\mathrm{PG}}(\theta) \\ \text{s.t.} & D_{KL} \left(\pi_\theta(\cdot|s) \| \pi(\cdot|s) \right) \leq \delta \end{array} \tag{2}$$

PPO differs from PG by clipping the ratio inside the objective:

$$J_{\mathrm{PPO}} = \mathop{\mathbb{E}}_{(s_t, a_t) \sim \pi} \left[\min \left(\mathrm{clip} \left(\rho_t, 1 - \varepsilon, 1 + \varepsilon \right) \hat{A}_\pi \left(s_t, a_t \right), \rho_t \hat{A}_\pi \left(s_t, a_t \right) \right) \right] \tag{3}$$

where

$$\rho_t = \frac{\pi_\theta \left(a_t | s_t \right)}{\pi \left(a_t | s_t \right)}. \tag{4}$$

A ton of trick in PPO paper

Authors looked at the PPO implementation by OpenAI and found a lot of tricks in it.

A ton of trick in PPO paper

Authors looked at the PPO implementation by OpenAI and found a lot of tricks in it. The main ones were:

A ton of trick in PPO paper

Authors looked at the PPO implementation by OpenAI and found a lot of tricks in it. The main ones were:

1. Value function clipping (during value function fitting)

A ton of trick in PPO paper

Authors looked at the PPO implementation by OpenAI and found a lot of tricks in it. The main ones were:

1. Value function clipping (during value function fitting)
2. Reward scaling

A ton of trick in PPO paper

Authors looked at the PPO implementation by OpenAI and found a lot of tricks in it. The main ones were:

1. Value function clipping (during value function fitting)
2. Reward scaling
3. Orthogonal init + layer scaling

A ton of trick in PPO paper

Authors looked at the PPO implementation by OpenAI and found a lot of tricks in it. The main ones were:

1. Value function clipping (during value function fitting)
2. Reward scaling
3. Orthogonal init + layer scaling
4. Learning rate annealing

What is the main source of a good PPO performance?

What is the main source of a good PPO performance?

Authors ran several experiments for PPO and TRPO with and without tricks and obtained the following results

	WALKER2D-V2	HOPPER-V2	HUMANOID-V2
PG +tricks	2867	2371	831
PG +PPO	2735	2142	674
PG +TRPO	2791	2043	586
PG +PPO +tricks	3292	2513	806
PG +TRPO+tricks	3050	2466	1030

What is the main source of a good PPO performance?

Authors ran several experiments for PPO and TRPO with and without tricks and obtained the following results

	WALKER2D-V2	HOPPER-V2	HUMANOID-V2
PG +tricks	2867	2371	831
PG +PPO	2735	2142	674
PG +TRPO	2791	2043	586
PG +PPO +tricks	3292	2513	806
PG +TRPO+tricks	3050	2466	1030

So,

- tricks improve the performance better than PPO or TRPO
- original paper should have used tricks for TRPO as well