Proximal Policy Optimization (PPO)

default reinforcement learning algorithm at OpenAI

From on-policy to off-policy

Using the experience more than once

On-policy v.s. Off-policy

- On-policy: The agent learned and the agent interacting with the environment is the same.
- Off-policy: The agent learned and the agent interacting with the environment is different.

阿光下棋

佐為下棋、阿光在旁邊看

On-policy → Off-policy

$$\boxed{\nabla \bar{R}_\theta = E_{\underline{\tau \sim p_\theta(\tau)}}[R(\tau)\nabla log p_\theta(\tau)]}$$

- Use π_θ to collect data. When θ is updated, we have to sample training data again.

Goal: Using the sample from $\pi_{\theta'}$ to train θ. θ' is fixed, so we can re-use the sample data.

Importance Sampling

x^i is sampled from $p(x)$

We only have x^i sampled from $q(x)$

$$\boxed{E_{x\sim p}[f(x)]} \approx \cancel{\frac{1}{N}\sum_{i=1}^{N} f(x^i)}$$

$$= \boxed{\int f(x)p(x)dx} = \boxed{\int f(x)\frac{p(x)}{q(x)}q(x)dx} = \boxed{E_{x\sim q}[f(x)\boxed{\frac{p(x)}{q(x)}}]}$$

Importance weight

Issue of Importance Sampling

$$\boxed{E_{x\sim p}[f(x)] = E_{x\sim q}[f(x)\frac{p(x)}{q(x)}]}$$

$$\mathrm{V}ar_{x\sim p}[f(x)] \qquad \mathrm{V}ar_{x\sim q}[f(x)\frac{p(x)}{q(x)}]$$

$$VAR[X]$$
$$= E[X^2] - (E[X])^2$$

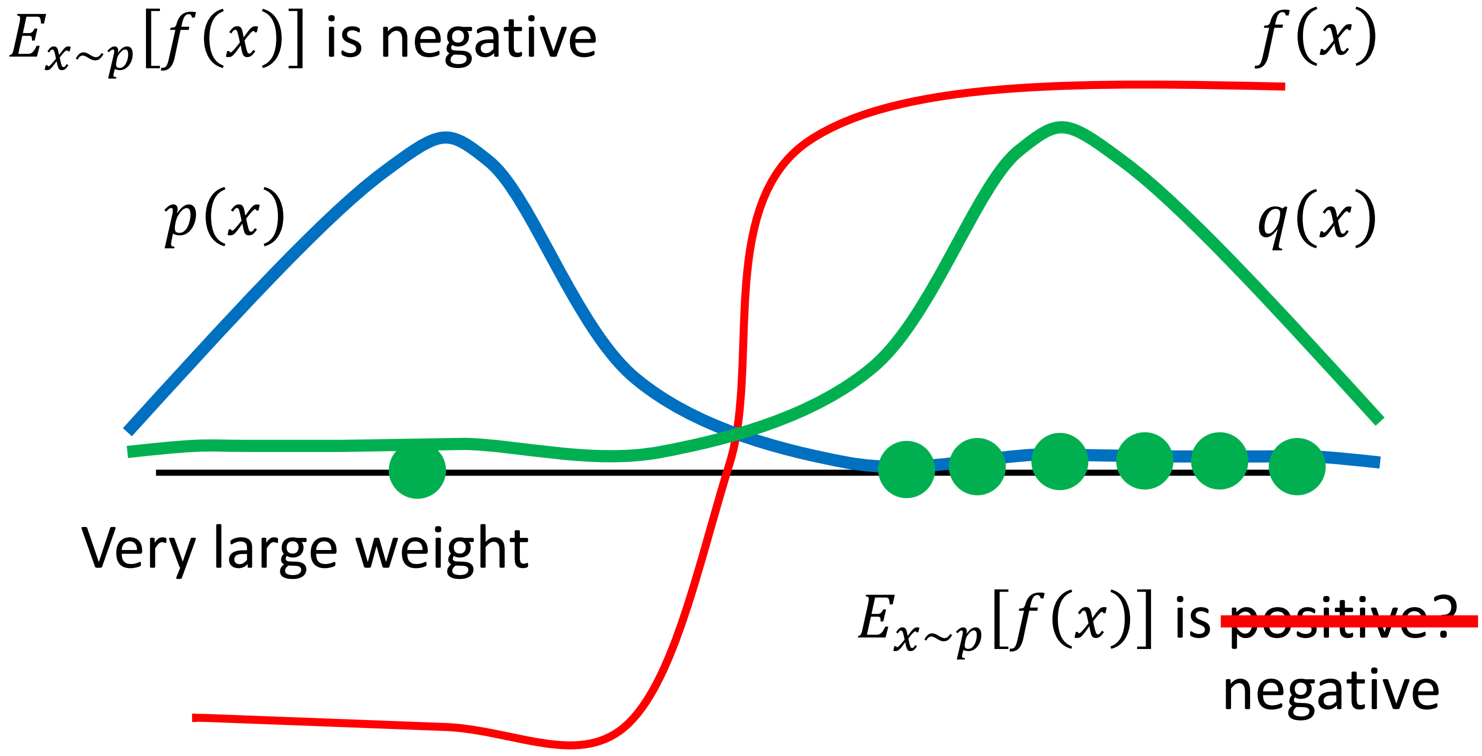
$$\mathrm{V}ar_{x\sim p}[f(x)] = E_{x\sim p}[f(x)^2] - \left(E_{x\sim p}[f(x)]\right)^2$$

$$\mathrm{V}ar_{x\sim q}[f(x)\frac{p(x)}{q(x)}] = E_{x\sim q}\left[\left(f(x)\frac{p(x)}{q(x)}\right)^2\right] - \left(E_{x\sim q}\left[f(x)\frac{p(x)}{q(x)}\right]\right)^2$$

$$= E_{x\sim p}\left[f(x)^2\boxed{\frac{p(x)}{q(x)}}\right] - \left(E_{x\sim p}[f(x)]\right)^2$$

Issue of Importance Sampling

$$E_{x\sim p}[f(x)] = E_{x\sim q}[f(x)\frac{p(x)}{q(x)}]$$



On-policy → Off-policy

$$\nabla \bar{R}_\theta = E_{\tau \sim p_\theta(\tau)}[R(\tau) \nabla log p_\theta(\tau)]$$

- Use π_θ to collect data. When θ is updated, we have to sample training data again.
- Goal: Using the sample from $\pi_{\theta'}$ to train θ. θ' is fixed, so we can re-use the sample data.

$$\nabla \bar{R}_\theta = E_{\tau \sim p_{\theta'}(\tau)}\left[\boxed{\frac{p_\theta(\tau)}{p_{\theta'}(\tau)}} R(\tau) \nabla log p_\theta(\tau)\right]$$

- Sample the data from θ'.
- Use the data to train θ many times.

Importance Sampling

$$E_{x \sim p}[f(x)] = E_{x \sim q}[f(x)\frac{p(x)}{q(x)}]$$

On-policy → Off-policy

$$\nabla f(x) = f(x) \nabla log f(x)$$

Gradient for update

$$= \boxed{E_{(s_t,a_t)\sim\pi_\theta}[A^{\theta}(s_t,a_t)\nabla log p_\theta(a_t^n|s_t^n)]}$$

$$A^{\theta'}(s_t,a_t)$$

This term is from sampled data.

$$= E_{(s_t,a_t)\sim\boxed{\pi_{\theta'}}}[\boxed{\frac{P_\theta(s_t,a_t)}{P_{\theta'}(s_t,a_t)}}\cancel{A^{\boxed{\theta}}(s_t,a_t)}\nabla log p_\theta(a_t^n|s_t^n)]$$

$$= E_{(s_t,a_t)\sim\pi_{\theta'}}[\frac{p_\theta(a_t|s_t)}{p_{\theta'}(a_t|s_t)}\cancel{\frac{p_\theta(s_t)}{p_{\theta'}(s_t)}}A^{\theta}(s_t,a_t)\nabla log p_\theta(a_t^n|s_t^n)]$$

$$\boxed{J^{\theta'}(\theta) = E_{(s_t,a_t)\sim\pi_{\theta'}}\left[\frac{p_\theta(a_t|s_t)}{p_{\theta'}(a_t|s_t)}A^{\theta'}(s_t,a_t)\right]}$$

When to stop?

Add Constraint

穩紮穩打，步步為營

PPO / TRPO

θ cannot be very different from θ'

Constraint on behavior not parameters

Proximal Policy Optimization (PPO)

$$\nabla f(x) = f(x)\nabla log f(x)$$

$$\boxed{J_{PPO}^{\theta'}(\theta) = J^{\theta'}(\theta) - \beta KL(\theta, \theta')}$$

$$J^{\theta'}(\theta) = E_{(s_t,a_t)\sim\pi_{\theta'}}\left[\frac{p_\theta(a_t|s_t)}{p_{\theta'}(a_t|s_t)}A^{\theta'}(s_t,a_t)\right]$$

TRPO (Trust Region Policy Optimization)

$$J_{TRPO}^{\theta'}(\theta) = E_{(s_t,a_t)\sim\pi_{\theta'}}\left[\frac{p_\theta(a_t|s_t)}{p_{\theta'}(a_t|s_t)}A^{\theta'}(s_t,a_t)\right]$$

$$KL(\theta, \theta') < \delta$$

PPO algorithm

$$J^{\theta^k}(\theta) \approx \sum_{(s_t,a_t)} \frac{p_\theta(a_t|s_t)}{p_{\theta^k}(a_t|s_t)} A^{\theta^k}(s_t,a_t)$$

- Initial policy parameters θ^0
- In each iteration
  - Using θ^k to interact with the environment to collect $\{s_t, a_t\}$ and compute advantage $A^{\theta^k}(s_t, a_t)$
  - Find θ optimizing $J_{PPO}(\theta)$

$$J_{PPO}^{\theta^k}(\theta) = J^{\theta^k}(\theta) - \beta KL\left(\theta, \theta^k\right)$$

Update parameters several times

- If $KL\left(\theta, \theta^k\right) > KL_{max}$, increase β
- If $KL\left(\theta, \theta^k\right) < KL_{min}$, decrease β

Adaptive KL Penalty

PPO algorithm

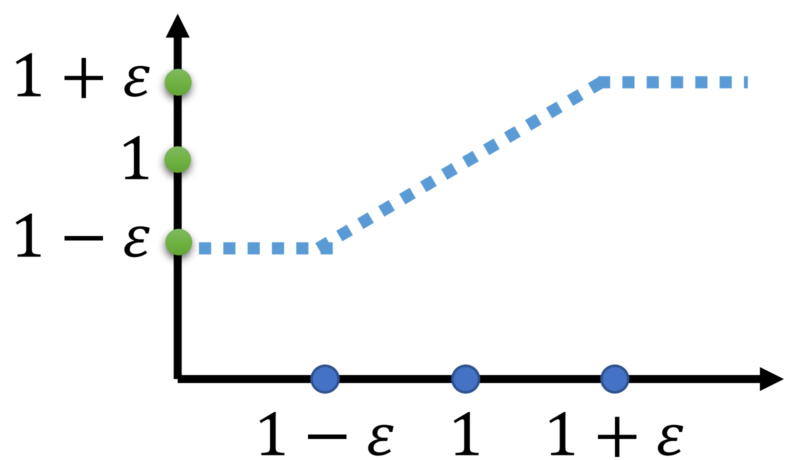
$$J_{PPO}^{\theta^k}(\theta) = J^{\theta^k}(\theta) - \beta KL(\theta, \theta^k)$$

$$J^{\theta^k}(\theta) \approx \sum_{(s_t,a_t)} \frac{p_\theta(a_t|s_t)}{p_{\theta^k}(a_t|s_t)} A^{\theta^k}(s_t, a_t)$$

PPO2 algorithm

$$J_{PPO2}^{\theta^k}(\theta) \approx \sum_{(s_t,a_t)}$$

$$\boxed{clip\left(\frac{p_\theta(a_t|s_t)}{p_{\theta^k}(a_t|s_t)}, 1-\varepsilon, 1+\varepsilon\right) A^{\theta^k}(s_t, a_t)}$$



$$\frac{p_\theta(a_t|s_t)}{p_{\theta^k}(a_t|s_t)}$$

PPO algorithm

$$J_{PPO}^{\theta^k}(\theta) = J^{\theta^k}(\theta) - \beta KL(\theta, \theta^k)$$

$$J^{\theta^k}(\theta) \approx \sum_{(s_t,a_t)} \frac{p_\theta(a_t|s_t)}{p_{\theta^k}(a_t|s_t)} A^{\theta^k}(s_t, a_t)$$

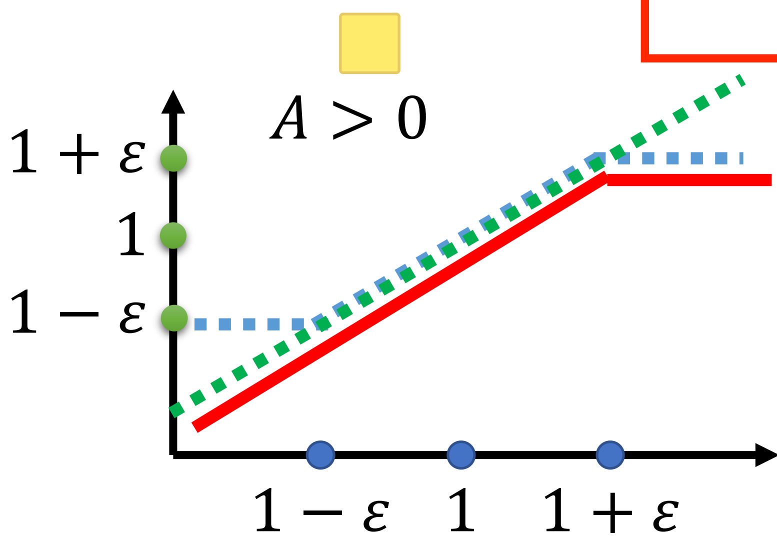
PPO2 algorithm

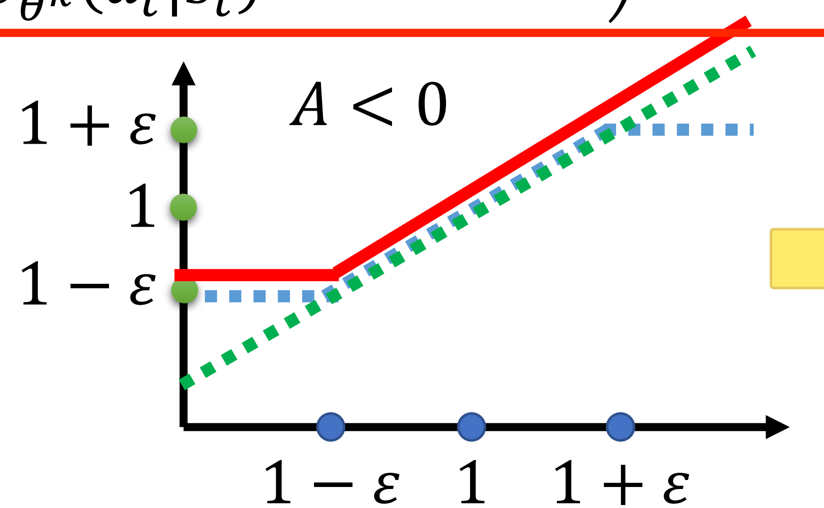
綠線

$$J_{PPO2}^{\theta^k}(\theta) \approx \sum_{(s_t,a_t)} min\left(\boxed{\frac{p_\theta(a_t|s_t)}{p_{\theta^k}(a_t|s_t)} A^{\theta^k}(s_t, a_t),}\right.$$

藍線

$$\left.\boxed{clip\left(\frac{p_\theta(a_t|s_t)}{p_{\theta^k}(a_t|s_t)}, 1-\varepsilon, 1+\varepsilon\right) A^{\theta^k}(s_t, a_t)}\right)$$





https://arxiv.org/abs/1707.06347

Experimental Results

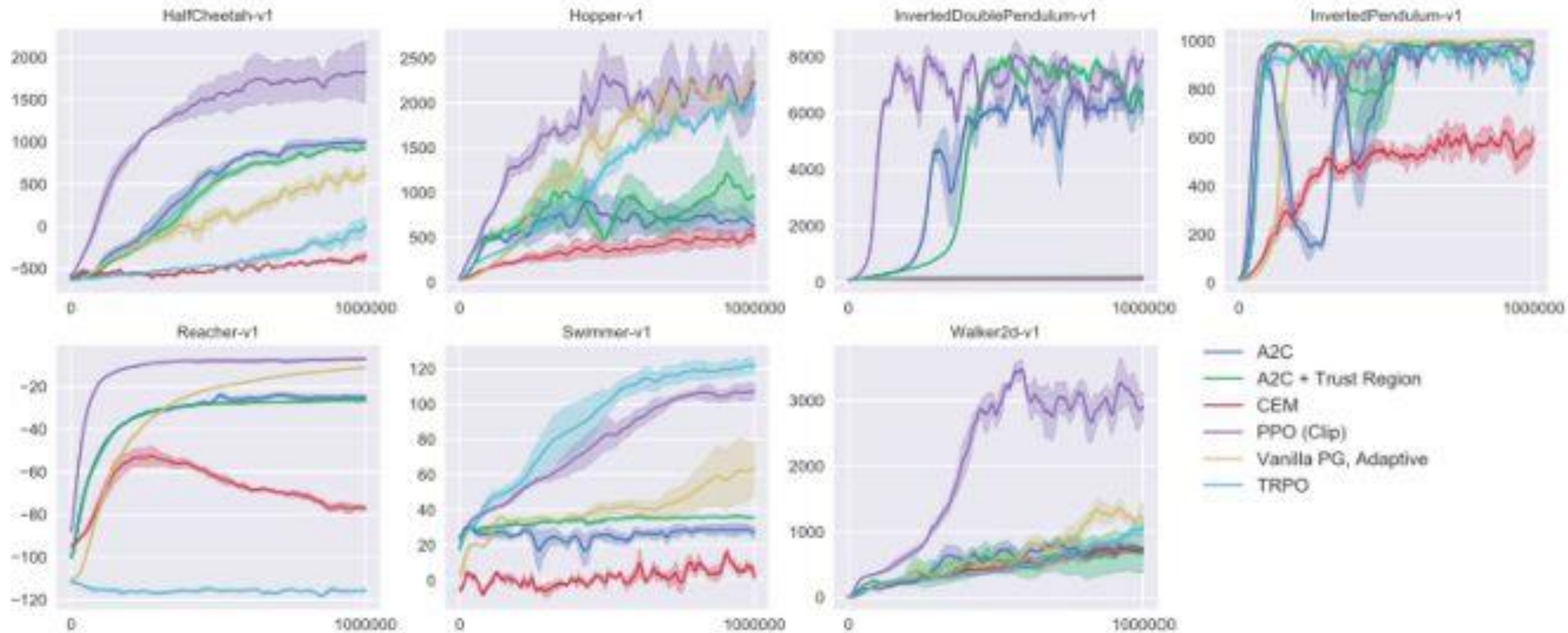


Figure 3: Comparison of several algorithms on several MuJoCo environments, training for one million timesteps.