Proximal Policy Optimization (PPO)

default reinforcement learning algorithm at OpenAI
It might look goofy ...
Policy Gradient (Review)
## Basic Components

<table>
<thead>
<tr>
<th>Video Game</th>
<th>Env</th>
<th>Reward Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actor</td>
<td></td>
<td>You cannot control</td>
</tr>
<tr>
<td>Get 20 scores when killing a monster</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Go</th>
<th></th>
<th>The rule of GO</th>
</tr>
</thead>
<tbody>
<tr>
<td>AlphaGo Google DeepMind</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**The rule of GO**

You cannot control the Actor.
Policy of Actor

- Policy $\pi$ is a network with parameter $\theta$
  - Input: the observation of machine represented as a vector or a matrix
  - Output: each action corresponds to a neuron in output layer

Take the action based on the probability.

Score of an action

left: 0.7
right: 0.2
fire: 0.1

pixels
Example: Playing Video Game

Start with observation $s_1$

Obtain reward $r_1 = 0$
Action $a_1$: “right”

Observation $s_2$

Obtain reward $r_2 = 5$
Action $a_2$: “fire” (kill an alien)

Observation $s_3$
Example: Playing Video Game

Start with observation $s_1$

Observation $s_2$

Observation $s_3$

After many turns

Game Over (spaceship destroyed)

Obtain reward $r_T$

Action $a_T$

This is an **episode**.

Total reward:

$$ R = \sum_{t=1}^{T} r_t $$

We want the total reward be maximized.
Actor, Environment, Reward

Trajectory \[ \tau = \{s_1, a_1, s_2, a_2, \ldots, s_T, a_T\} \]

\[ p_\theta(\tau) = p(s_1)p_\theta(a_1|s_1)p(s_2|s_1, a_1)p_\theta(a_2|s_2)p(s_3|s_2, a_2)\ldots \]

\[ = p(s_1) \prod_{t=1}^{T} p_\theta(a_t|s_t) p(s_{t+1}|s_t, a_t) \]
Actor, Environment, Reward

\[ \begin{align*}
    \text{Env} & \xrightarrow{s_1} \text{Actor} \xrightarrow{a_1} \text{Env} \\
    & \downarrow s_1 \quad \downarrow a_1 \quad \downarrow s_2 \quad \downarrow a_2 \quad \downarrow s_3
\end{align*} \]

\[ R(\tau) = \sum_{t=1}^{T} r_t \]

\[ \bar{R}_\theta = \sum_{\tau} R(\tau)p_\theta(\tau) = E_{\tau \sim p_\theta(\tau)}[R(\tau)] \]
Policy Gradient

$$\bar{R}_{\theta} = \sum_{\tau} R(\tau)p_{\theta}(\tau) \quad \nabla \bar{R}_{\theta} = ?$$

$$\nabla \bar{R}_{\theta} = \sum_{\tau} R(\tau)\nabla p_{\theta}(\tau) = \sum_{\tau} R(\tau)p_{\theta}(\tau) \frac{\nabla p_{\theta}(\tau)}{p_{\theta}(\tau)}$$

$R(\tau)$ do not have to be differentiable
It can even be a black box.

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

$$= \sum_{\tau} R(\tau)p_{\theta}(\tau) \nabla \log p_{\theta}(\tau)$$

$$= E_{\tau \sim p_{\theta}(\tau)}[R(\tau)\nabla \log p_{\theta}(\tau)] \approx \frac{1}{N} \sum_{n=1}^{N} R(\tau^n)\nabla \log p_{\theta}(\tau^n)$$

$$= \frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T_n} R(\tau^n) \nabla \log p_{\theta}(a_t^n | s_t^n)$$
Policy Gradient

Given policy $\pi_\theta$

$\tau^1: \begin{cases} (s^1_1, a^1_1) & R(\tau^1) \\ (s^1_2, a^1_2) & R(\tau^1) \end{cases}$

$\tau^2: \begin{cases} (s^2_1, a^2_1) & R(\tau^2) \\ (s^2_2, a^2_2) & R(\tau^2) \end{cases}$

only used once

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

$\theta \leftarrow \theta + \eta \nabla \bar{R}_\theta$

$\nabla \bar{R}_\theta = \frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T_n} R(\tau^n) \nabla \log p_\theta(a^n_t | s^n_t)$
\[ \theta \leftarrow \theta + \eta \nabla \bar{R}_\theta \]
\[ \nabla \bar{R}_\theta = \frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T_n} R(\tau^n) \nabla \log p_\theta(a_t^n | s_t^n) \]

Consider as classification problem

\[
\frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T_n} \log p_\theta(a_t^n | s_t^n) \]

TF, pyTorch ...

\[
\frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T_n} R(\tau^n) \log p_\theta(a_t^n | s_t^n) \]
Tip 1: Add a Baseline

\[
\theta \leftarrow \theta + \eta \nabla R_{\theta}
\]

It is possible that \( R(\tau^n) \) is always positive.

\[
\nabla \bar{R}_\theta \approx \frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T_n} (R(\tau^n) - b) \nabla \log p_\theta(a_t^n | s_t^n) \quad b \approx E[R(\tau)]
\]

Ideal case

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Sampling

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Not sampled

It is probability ...

The probability of the actions not sampled will decrease.
Tip 2: Assign Suitable Credit

\[
\begin{align*}
\times 3 & \quad \times -2 & \quad \times -2 & \quad \times -7 & \quad \times -2 & \quad \times -2 \\
(s_a, a_1) & \quad (s_b, a_2) & \quad (s_c, a_3) & \quad (s_a, a_2) & \quad (s_b, a_2) & \quad (s_c, a_3) \\
\text{+5} & \quad \text{+0} & \quad \text{-2} & \quad \text{-5} & \quad \text{+0} & \quad \text{-2}
\end{align*}
\]

\[R = +3\quad R = -7\]

\[
\nabla \bar{R}_\theta \approx \frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T_n} (R(t^n) - b) \nabla \log p_\theta(a_t^n | s_t^n) \sum_{t'=t}^{T_n} r_{t'}^n
\]
Tip 2: Assign Suitable Credit

How good it is if we take $a_t$ other than other actions at $s_t$.
Estimated by “critic” (later)

Can be state-dependent

$$\nabla \tilde{R}_\theta \approx \frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T_n} (R(t^n) - b) \nabla \log p_{\theta}(a_t^n | s_t^n)$$

$$\sum_{t'=t}^{T_n} r_{t'} \rightarrow \sum_{t'=t}^{T_n} \gamma^{t'-t} r_{t'}$$
Add discount factor $\gamma < 1$
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.
On-policy → Off-policy

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

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

**Importance Sampling**

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

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

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

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

Importance weight
Issue of Importance Sampling

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

\[
\text{VAR}_x[f(x)] = \text{VAR}_q[f(x) \frac{p(x)}{q(x)}]
\]

\[
\text{VAR}_x[f(x)] = E_{x \sim p}[f(x)^2] - (E_{x \sim p}[f(x)])^2
\]

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

\[
= E_{x \sim p}\left[f(x)^2 \frac{p(x)}{q(x)}\right] - (E_{x \sim p}[f(x)])^2
\]
Issue of Importance Sampling

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

- \( E_{x \sim p}[f(x)] \) is negative
  - Very large weight

- \( E_{x \sim p}[f(x)] \) is positive?
  - negative
On-policy → Off-policy

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

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

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

- Sample the data from \( \theta' \).
- Use the data to train \( \theta \) many times.

\underline{Importance Sampling}

\[ E_{x \sim p}[f(x)] = E_{x \sim q}[f(x) \frac{p(x)}{q(x)}] \]
On-policy $\rightarrow$ Off-policy

Gradient for update

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

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

This term is from sampled data.

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

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

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

When to stop?
Add Constraint

穩紮穩打，步步為營
PPO / TRPO

$$\theta$$ cannot be very different from $$\theta'$$
Constraint on behavior not parameters

**Proximal Policy Optimization (PPO)**

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

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

**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

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

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

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

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

Adaptive KL Penalty

Update parameters several times
**PPO algorithm**

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

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

**PPO2 algorithm**

\[ J_{\text{PPO2}}^{\theta^k}(\theta) \approx \sum_{(s_t, a_t)} \text{clip} \left( \frac{p_\theta(a_t|s_t)}{p_{\theta^k}(a_t|s_t)}, 1 - \varepsilon, 1 + \varepsilon \right) A^{\theta^k}(s_t, a_t) \]
**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**

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

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

\[
A > 0
\]

\[
A < 0
\]
Experimental Results

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