Proximal Policy Optimization (PPO)

policy gradient進階版

default reinforcement learning algorithm at OpenAI

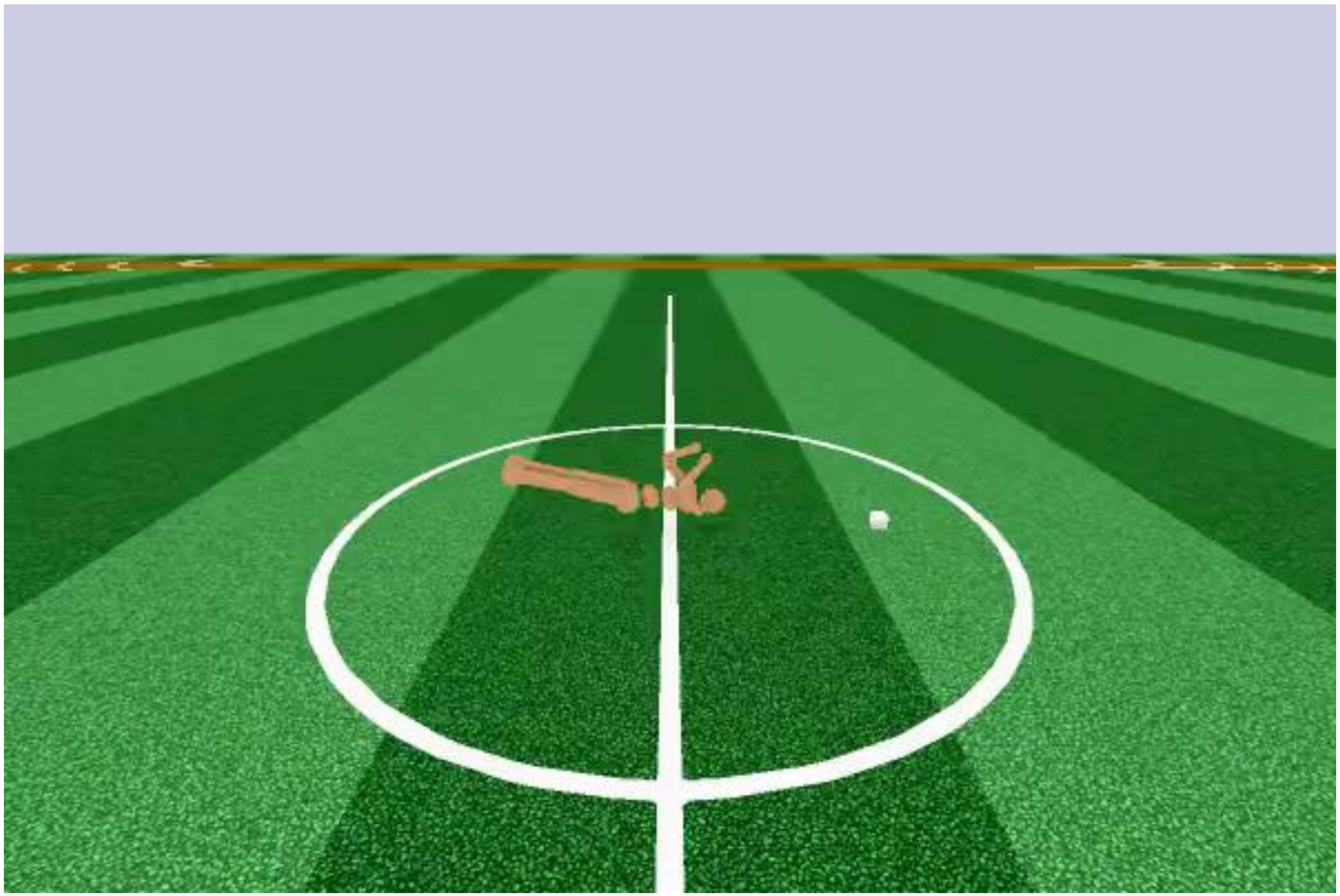
Policy Gradient → On-policy → Off-policy → Add constraint

DeepMind

https://youtu.be/gn4nRCC9TwQ

OpenAI

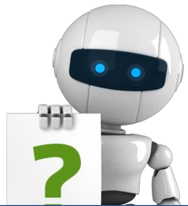
https://blog.openai.com/openai-baselines-ppo/



Policy Gradient (Review)

Basic Components

訓練過程可以調動的



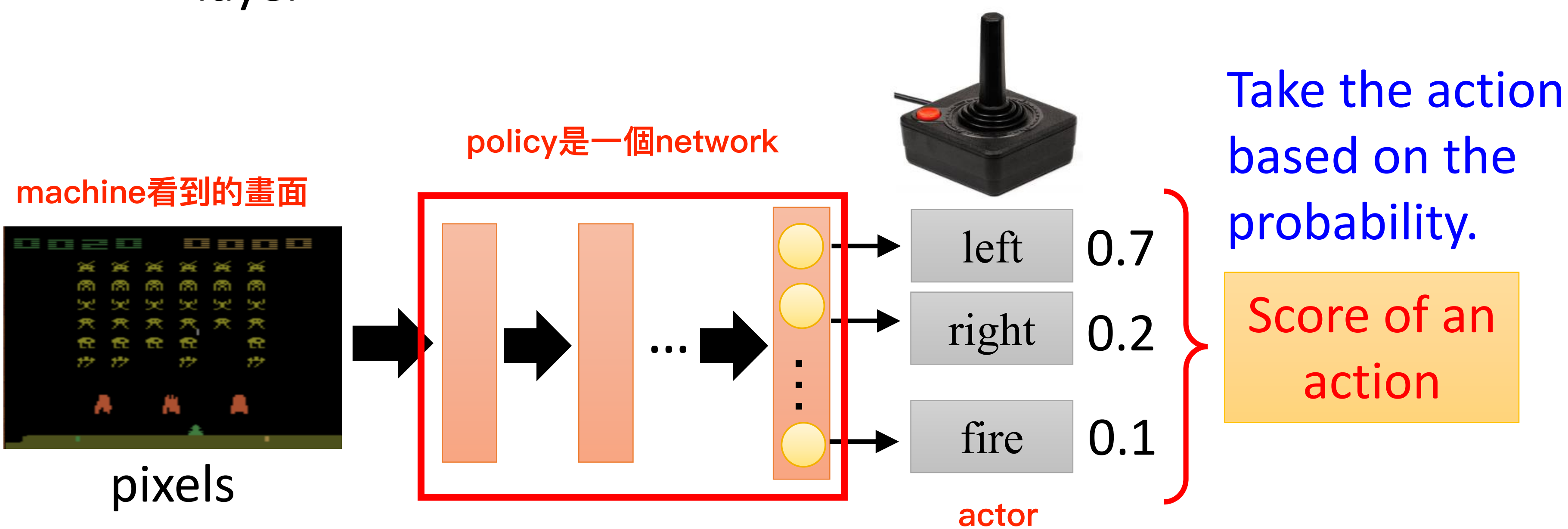
事先給定好的

You cannot control

	Actor	Env	Reward Function
Video Game			Get 20 scores when killing a monster
Go	AlphaGo Google DeepMind		The rule of GO

Policy of Actor

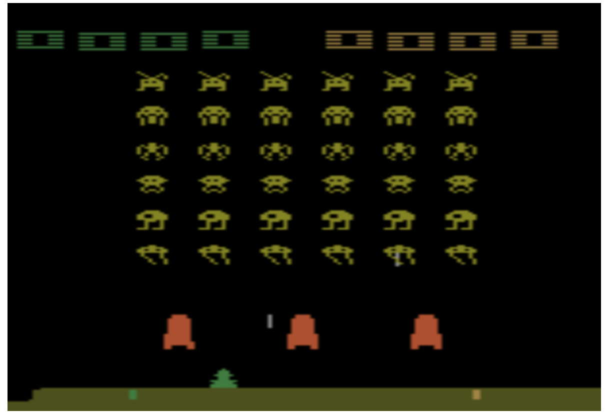
- Policy π is a network with parameter θ
  - Input: the observation of machine represented as a vector or a matrix
  - Output: each action corresponds to a neuron in output layer



Example: Playing Video Game

Start with observation s_1

Observation s_2

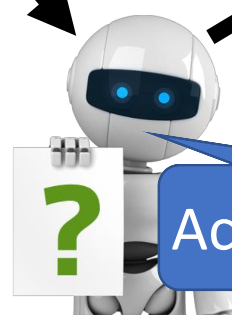


Observation s_3

遊戲初始的畫面，根據 NN policy決定actor

Action a_1: “right”

Obtain reward $r_1 = 0$



Action a_2 : “fire”

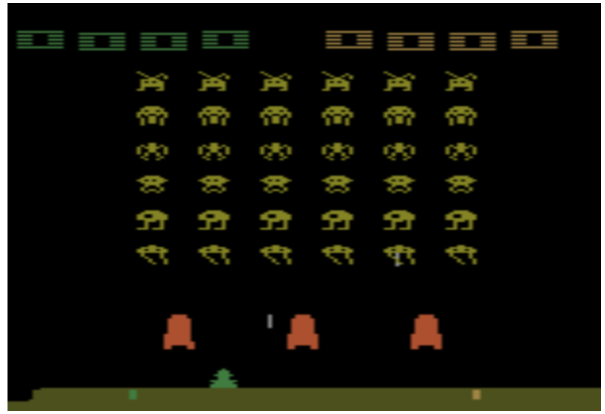
(kill an alien)

Obtain reward $r_2 = 5$

Example: Playing Video Game

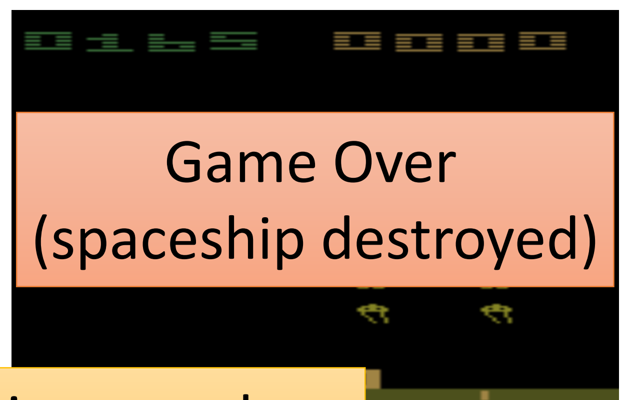
Start with observation s_1

Observation s_2



Observation s_3

After many turns



Obtain reward r_T

Action a_T

一場遊戲

This is an ***episode***.

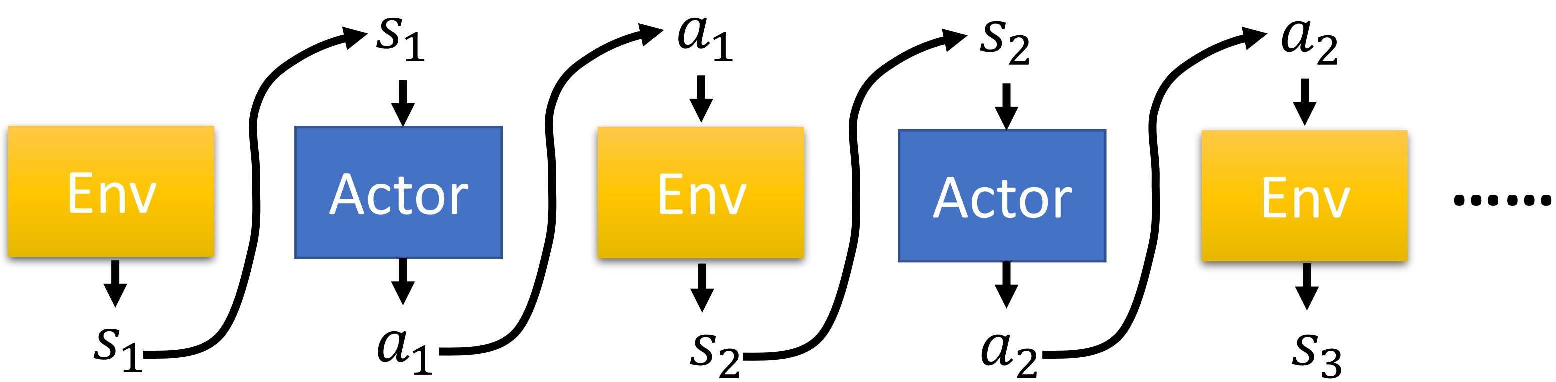
Total reward:

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

We want the total reward be maximized.

actor存在的目的就是想辦法提高R

Actor, Environment, Reward



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

trajectory發生的機率

$$p_\theta(\tau)$$

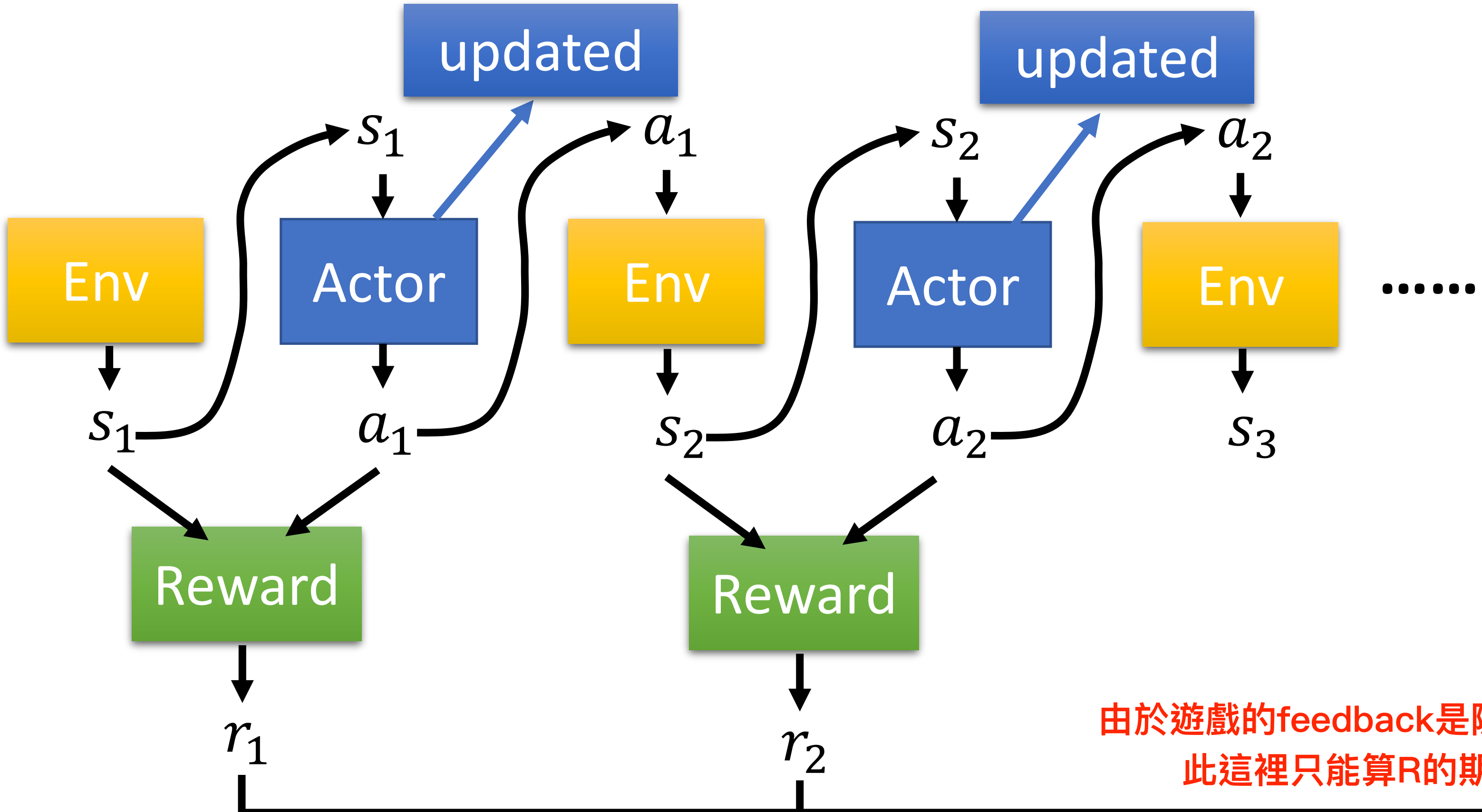
$$= p(s_1)p_\theta(a_1|s_1)p(s_2|s_1,a_1)p_\theta(a_2|s_2)p(s_3|s_2,a_2)\cdots$$

取決於自己學出來的N N

$$= p(s_1)\prod_{t=1}^{T} p_\theta(a_t|s_t)p(s_{t+1}|s_t,a_t)$$

取決於遊戲環境

Actor, Environment, Reward



由於遊戲的feedback是隨機的，因此這裡只能算R的期望值

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

Expected Reward

看到期望值就想到窮舉～

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

我們要做的事情就是maximum expected reward

Policy Gradient

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

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

舉例來說在GAN來說，這裡的R就是D

$R(\tau)$ do not have to be differentiable

It can even be a black box.

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

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

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

唯一需要算gradient的部分

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

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

Policy Gradient

用train好的參數跟遊戲玩一場，並且記錄下trajectory

Given policy π_θ

τ^1: (s_1^1, a_1^1) $R(\tau^1)$

(s_2^1, a_2^1) $R(\tau^1)$

⋮ ⋮

τ^2: (s_1^2, a_1^2) $R(\tau^2)$

(s_2^2, a_2^2) $R(\tau^2)$

⋮ ⋮

only used once

Update Model

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

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

Data Collection

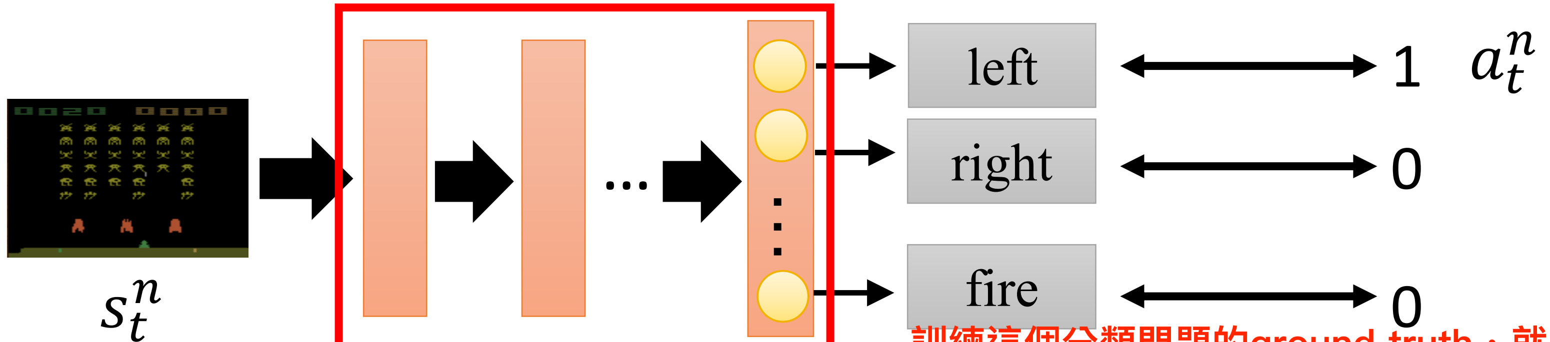
$$\theta \leftarrow \theta + \eta \nabla \bar{R}_\theta$$

Implementation

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

$s_t^n \quad a_t^n \quad R(\tau^n)$

Consider as classification problem



訓練這個分類問題的ground truth，就採用sampling，sample到哪個就當他GT label

classification network!!

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

TF, pyTorch …

一整場遊戲只有一個大R

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

在ＲＬ中只是多一個weight

Tip 1: Add a Baseline

這樣會造成大家都提升，只是有大有小

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

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

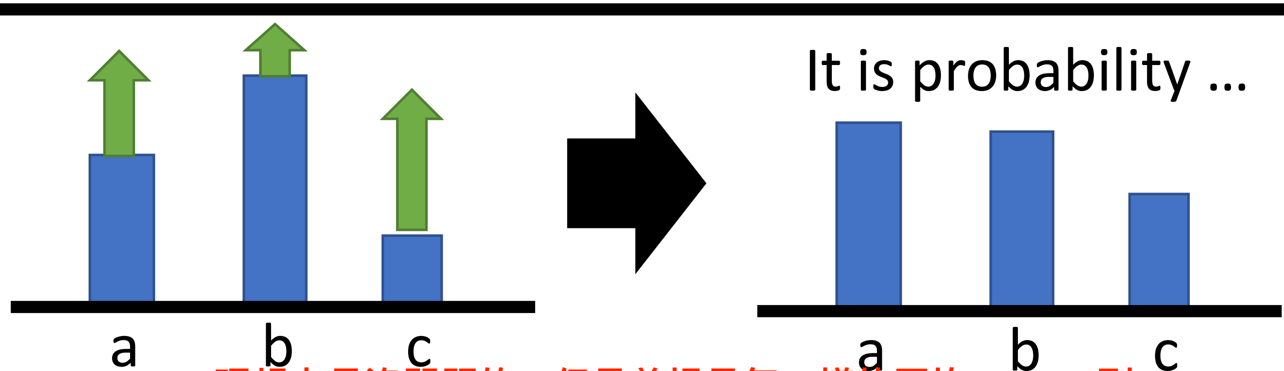
因此這邊做了一點感變：不要讓reward總是正的

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

大家同減去baseline b

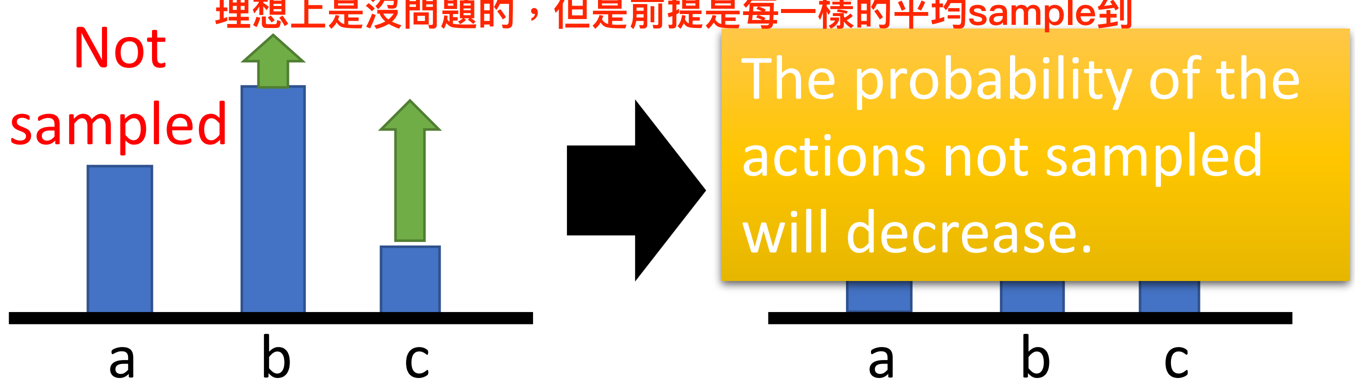
mean:每次episode計算一次

Ideal case



理想上是沒問題的，但是前提是每一樣的平均sample到

Sampling ……



譬如上面的例子，沒被sample到反而下降

Tip 2: Assign Suitable Credit

理想上這個問題只要sample夠多就可以被解決
但實作上無法sample所有情況，因此要涉及合理的credit

$\times 3$	$\times -2$	$\times -2$	$\times -7$	$\times -2$	$\times -2$
(s_a, a_1)	(s_b, a_2)	(s_c, a_3)	(s_a, a_2)	(s_b, a_2)	(s_c, a_3)
+5	+0	-2	-5	+0	-2

$R = +3$ $\qquad\qquad$ $R = -7$

同一場episode都是weighted
by same weight

但這是不公平的，因為說不定
只有某個action需要做調整

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

$$\boxed{\sum_{t'=t}^{T_n} r_{t'}^n}$$

只看這個時間點以後所得到的reward

Tip 2: Assign Suitable Credit

實際上在算R的時候需要跟環境

interaction，因此based on theta

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

相對其他action的好壞

How good it is if we take a_t other than other actions at s_t.

Estimated by "*critic*" (later)

Can be state-dependent

b 可以是state dependent，

一個NN的output

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

decay factor

$$\sum_{t'=t}^{T_n} r_{t'}^n \rightarrow \sum_{t'=t}^{T_n} \boxed{\gamma^{t'-t}} r_{t'}^n$$

Add discount factor $\gamma < 1$

代表說距離越遠影響越小

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

自己下自己學：on policy

佐為下棋、阿光在旁邊看

在旁邊看著學：off policy

每次更新完參數後又要跟環
境大量互動 collect data
這樣太花時間了！

On-policy → Off-policy

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

- 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

從p這個distribution sample x

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

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

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

但我們無法從p sample data，只能從另一個distribution sample (q)

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

Importance weight

Issue of Importance Sampling

mean一樣

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

但variance不一定一樣

$$Var_{x\sim p}[f(x)] \qquad Var_{x\sim q}[f(x)\frac{p(x)}{q(x)}]$$

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

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

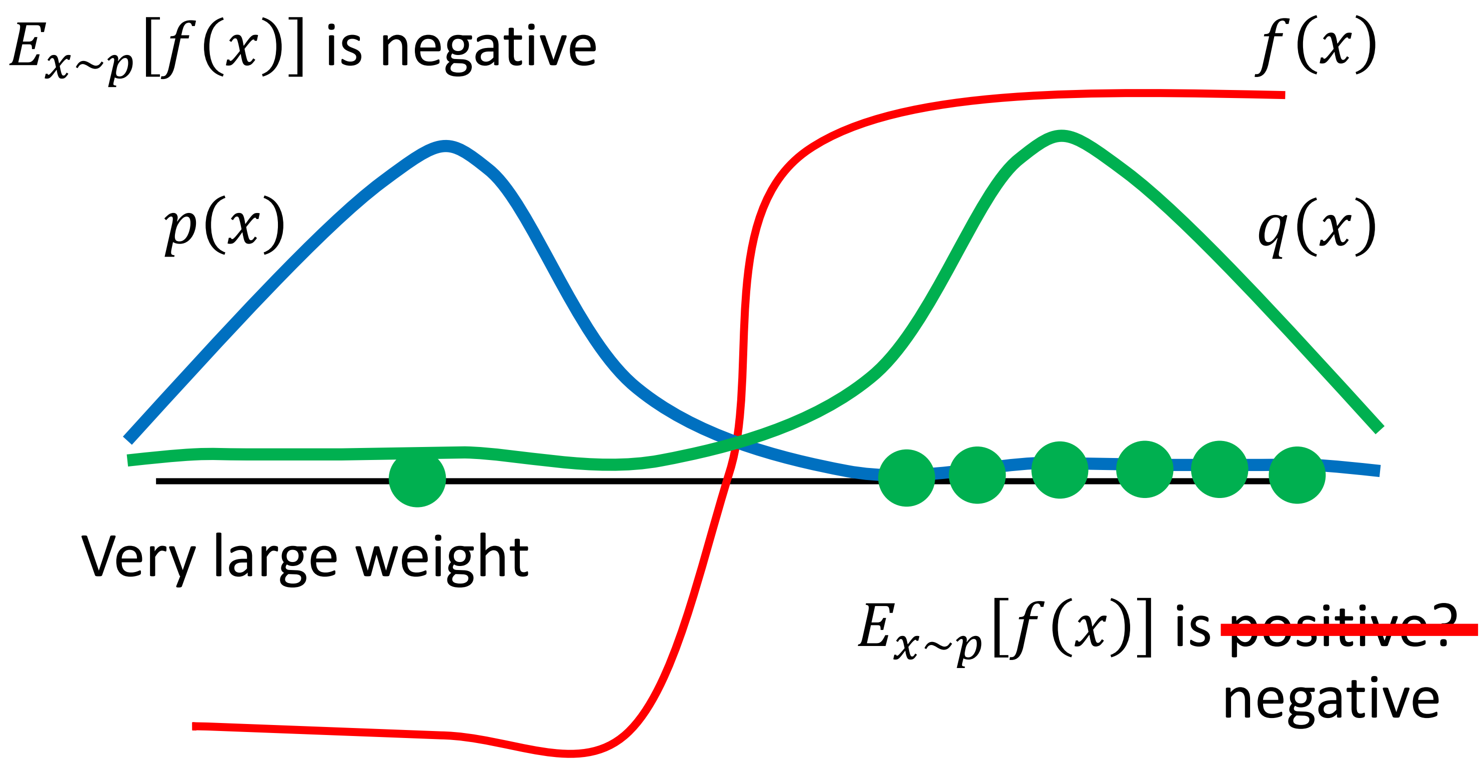
$$Var_{x\sim q}[f(x)\frac{p(x)}{q(x)}] = E_{x\sim q}\left[\left(f(x)\frac{p(x)}{q(x)}\right)^2\right] - \left(E_{x\sim q}\left[f(x)\frac{p(x)}{q(x)}\right]\right)^2$$

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

Issue of Importance Sampling

sample不夠多次，等式無法成立

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



On-policy → Off-policy

theta' 只負責做demo，負責互動後取出資料

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

- 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.

important weight

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

- Sample the data from 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

A: 這個state st，採取action at，互動完後得到的cumulative reward減去baseline

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

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

This term is from sampled data.

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

遊戲出現st的機率，太難計算了乾脆忽略他，說服自己不重要

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

network output probability

demo theta’

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

objective function

When to stop?

但這裡有一個限制，theta’與theta不能差太多，不然結果會太過不一致

Add Constraint

穩紮穩打，步步為營

PPO / TRPO

θ cannot be very different from θ'

Constraint on behavior not parameters

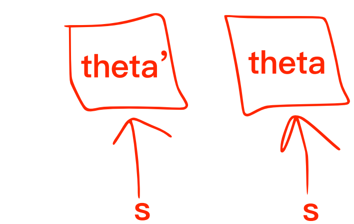
Proximal Policy Optimization (PPO)

對兩個action (network output) distribution算KL divergence

regularization，讓theta’ 與 theta不要差太多

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

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



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

TRPO (Trust Region Policy Optimization)

TRPO是將KL divergence變成一個constraint

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

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

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 using for demo to interact with the environment to collect $\{s_t, a_t\}$ and compute advantage $A^{\theta^k}(s_t,a_t)$
  - Find θ optimizing $J_{PPO}(\theta)$

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

Update parameters several times

adaptive KL divergence

當KL太大，表示最後一項沒有發揮作用，調大weight

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

Adaptive KL Penalty

PPO algorithm

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

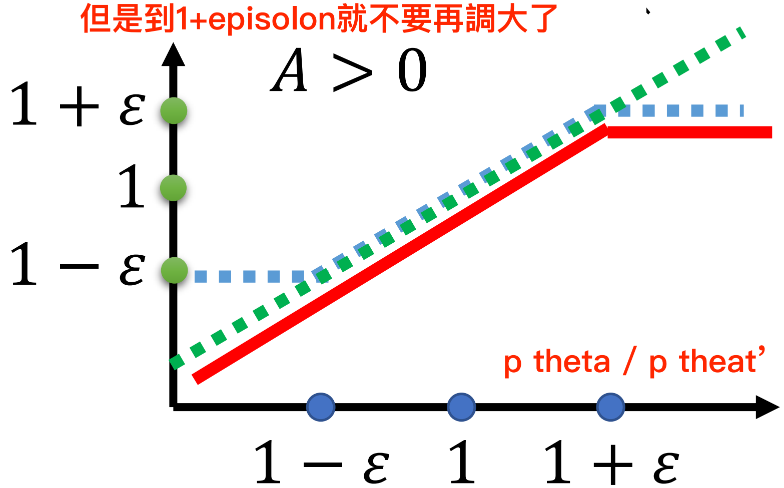
$$J^{\theta^k}(\theta) \approx \sum_{(s_t,a_t)} \frac{p_\theta(a_t|s_t)}{p_{\theta^k}(a_t|s_t)} A^{\theta^k}(s_t, a_t)$$

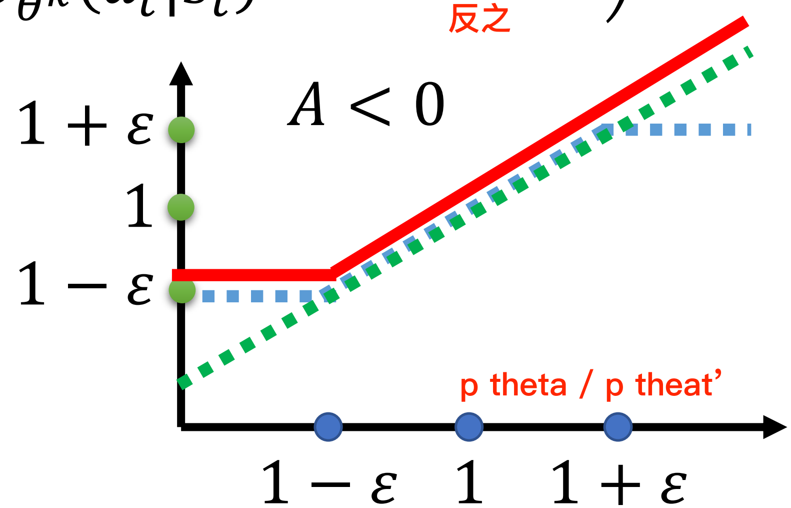
PPO2 algorithm

$$J_{PPO2}^{\theta^k}(\theta) \approx \sum_{(s_t,a_t)} \min\left(\frac{p_\theta(a_t|s_t)}{p_{\theta^k}(a_t|s_t)} A^{\theta^k}(s_t, a_t), \; clip\left(\frac{p_\theta(a_t|s_t)}{p_{\theta^k}(a_t|s_t)}, \overset{\text{floor}}{1-\varepsilon}, \overset{\text{ceil}}{1+\varepsilon}\right) A^{\theta^k}(s_t, a_t)\right)$$

當今天actor是正的，我們就盡量調大p theta，
但是到1+episolon就不要再調大了

反之





https://arxiv.org/abs/1707.06347

Experimental Results

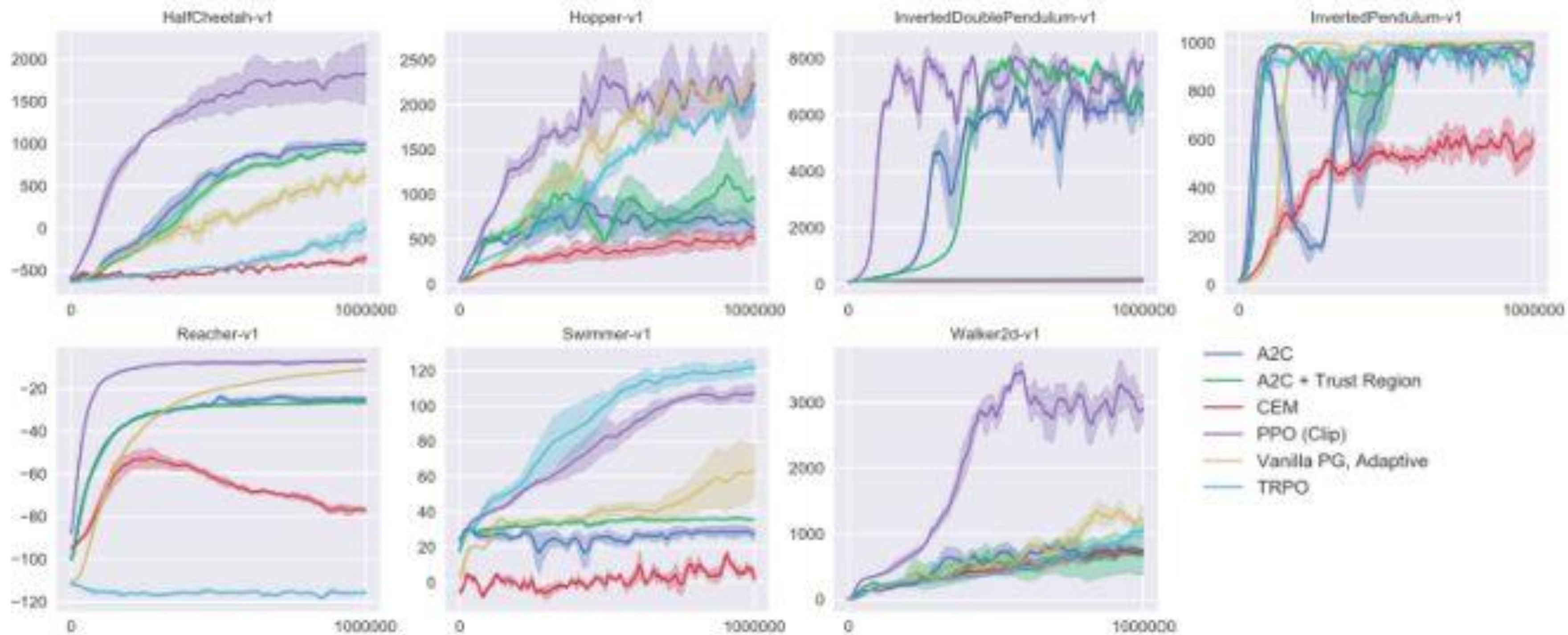


Figure 3: Comparison of several algorithms on several MuJoCo environments, training for one million timesteps.