ML Lecture 23: Deep Reinforcement Learning - II

> 臺灣大學人工智慧中心 科技部人工智慧技術暨全幅健康照護聯合研究中心 http://ai.ntu.edu.tw/

前情提要

- Reinforcement Learning 的方法主要分為
  - Policy-based
    - 會 learn 一個負責做事的 actor
  - Value-based
    - 會 learn 一個不做事的 critic，專門批評

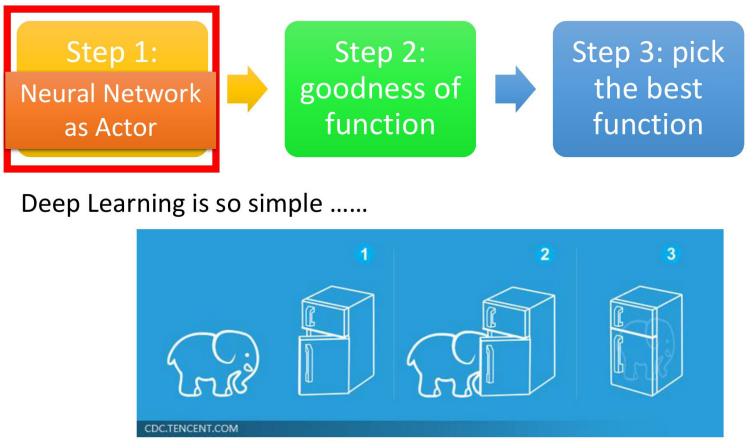
Policy-based Approach

- Actor (or Policy)
  - Actor 就是一個 function，通常寫成 pi
    - input 就是 machine 看到的 observation
    - output 就是 machine 要採取的 action
    - 透過 reward 幫助我們找出這個 Actor (function)

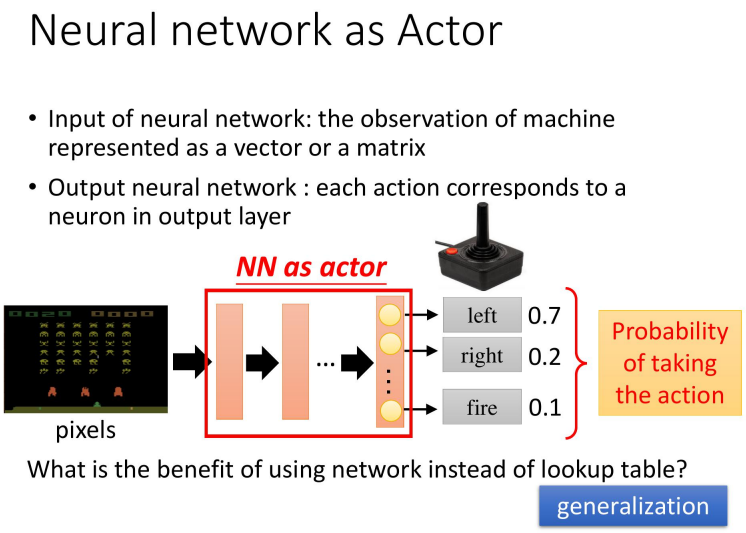
找 Actor 三步驟

一、決定 function 長甚麼樣子 (例如：NN)

-



- Neural Network 決定了一個 function space，所以 Actor 可以是一個 NN



  - Input：用一個 vector 來描述一堆 pixel 的 observation
  - output：現在可以採取的 action
- Neural Network 的好處：
  - Policy 是 stochastic，意思是 Policy 的 output 其實是個機率，會根據機率在同一個畫面有不同action。傳統的作法是存一個 table，看到什麼 observation 就採取什麼 action
  - 即使是沒有看過的東西，也有可能得到一個合理的結果

二、決定一個 function (Actor) 的好壞

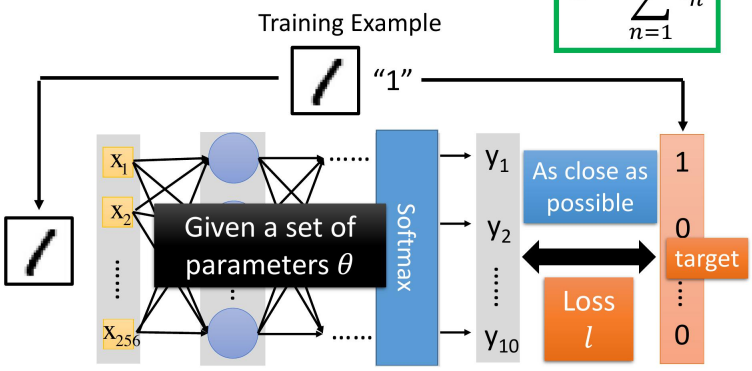
- **<u>先回顧：Supervised learning 怎麼評估 Actor 好壞</u>**
  -

Goodness of Actor

Total Loss:

$$L = \sum_{n=1}^{N} l_n$$

- Review: Supervised learning



- 假設給一個 Neural Network 其參數已經知道是 theta
  - 有一堆 training example 在做 image classification，就把 image 丟進去看 output 跟 target 像不像，如果越像代表 function 越好
- 會定義一個東西叫做 Loss，算每一個 example 的 Loss 合起來就是 Total Loss
  - 找一個參數去 minimize 這個 Total Loss
- **<u>再來看 Reinforcement Learning 怎麼評估 Actor 好壞</u>**
  -

Goodness of Actor

- Given an actor $\pi_\theta(s)$ with network parameter θ
- Use the actor $\pi_\theta(s)$ to play the video game
  - Start with observation s_1
  - Machine decides to take a_1
  - Machine obtains reward r_1
  - Machine sees observation s_2
  - Machine decides to take a_2
  - Machine obtains reward r_2
  - Machine sees observation s_3
  - ……
  - Machine decides to take a_T
  - Machine obtains reward r_T

END

Total reward: $R_\theta = \sum_{t=1}^{T} r_t$

Even with the same actor, R_θ is different each time

Randomness in the actor and the game

We define $\bar{R}_\theta$ as the *<u>expected value</u>* of R_θ

$\bar{R}_\theta$ evaluates the goodness of an actor $\pi_\theta(s)$

- 假設有一個 Actor (Neural Network)，他的參數是 theta，用 pi 下標 theta
  - Actor 是一個 function，input 就是一個 s（s 就是 Actor 看到的 observation）
- 要知道一個 Actor 表現好還是不好，就讓 Actor 實際的去玩一下這個遊戲
  - 玩完遊戲以後得到的 Total Reward 可以寫成 Rθ，Rθ 就是 r1 + r2 一直加到 rT
  - 把所有在每一個 step 得到的 reward合起來，就是在這一個 episode 裡面得到的 Total Reward
  - 由於 Actor 是 stochastic，遊戲本身也有隨機性，所以 Rθ 是一個 random variable
  - 目標：maximize Rθ 的期望值
- Rθ 期望值實際上要怎麼計算出來

-

Goodness of Actor

- An episode is considered as a trajectory τ
  - $\tau = \{s_1, a_1, r_1, s_2, a_2, r_2, \cdots, s_T, a_T, r_T\}$
  - $R(\tau) = \sum_{t=1}^{T} r_t$
  - If you use an actor to play the game, each τ has a probability to be sampled
    - The probability depends on actor parameter θ: $P(\tau|\theta)$

$$\bar{R}_\theta = \sum_\tau R(\tau) P(\tau|\theta) \approx \frac{1}{N} \sum_{n=1}^{N} R(\tau^n)$$

Sum over all possible trajectory

Use π_θ to play the game N times, obtain $\{\tau^1, \tau^2, \cdots, \tau^N\}$

Sampling τ from $P(\tau|\theta)$ N times

- 假設一場遊戲就是一個 trajectory τ，τ 是一個 sequence 裡面包含了 state、observation，看到這個 observation 以後 take 的 action，還有得到的 reward，還有新的 observation、take 的 action、得到的 reward 等等的一個 sequence
- R(τ) 代表這個 trajectory 在這場遊戲最後得到的 Total Reward，所有的小 r summation 起來就是 total 的 reward
- 每一個 τ 都會有一個出現的機率， τ 代表某一種可能的從遊戲開始到結束的過程，可以用機率來描述他，寫作 P(τ | θ)
- Rθ bar 就寫成 summation over 所有可能的 τ (所有可能的遊戲進行的過程) 的期望 reward
  - 每一個 τ，都有一個機率 P(τ | θ)，和一個 reward R(τ)，乘起來就是期望 reward
- 實際上要窮舉所有的 τ 是不可能的，所以讓 Actor 去玩這個遊戲玩 N 場，有點 sample 的感覺

三、選一個最好的 Actor

- 用 Gradient Descent
  -

Gradient Ascent

• Problem statement

$$\theta^* = arg \max_{\theta} \bar{R}_\theta$$

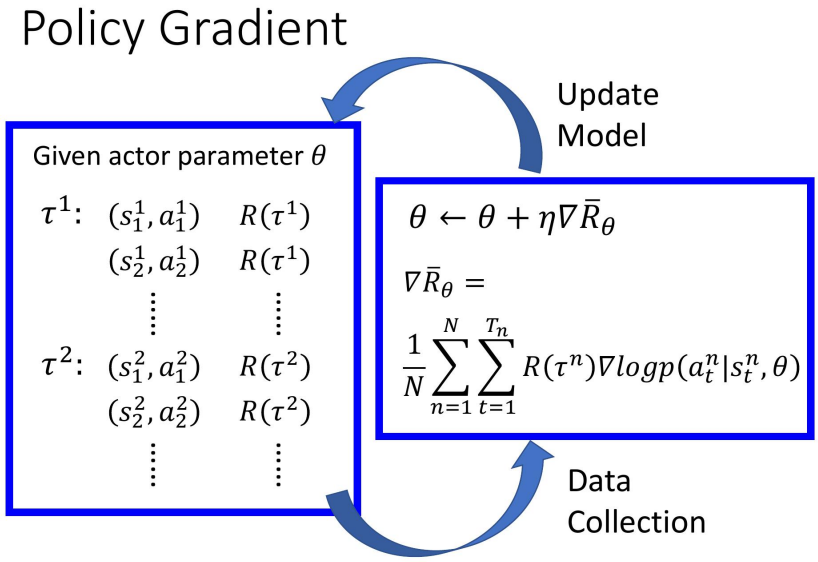
• Gradient ascent
  • Start with θ^0
  • $\theta^1 \leftarrow \theta^0 + \eta \nabla \bar{R}_{\theta^0}$
  • $\theta^2 \leftarrow \theta^1 + \eta \nabla \bar{R}_{\theta^1}$
  • ……

$$\theta = \{w_1, w_2, \cdots, b_1, \cdots\}$$

$$\nabla \bar{R}_\theta = \begin{bmatrix} \partial \bar{R}_\theta / \partial w_1 \\ \partial \bar{R}_\theta / \partial w_2 \\ \vdots \\ \partial \bar{R}_\theta / \partial b_1 \\ \vdots \end{bmatrix}$$

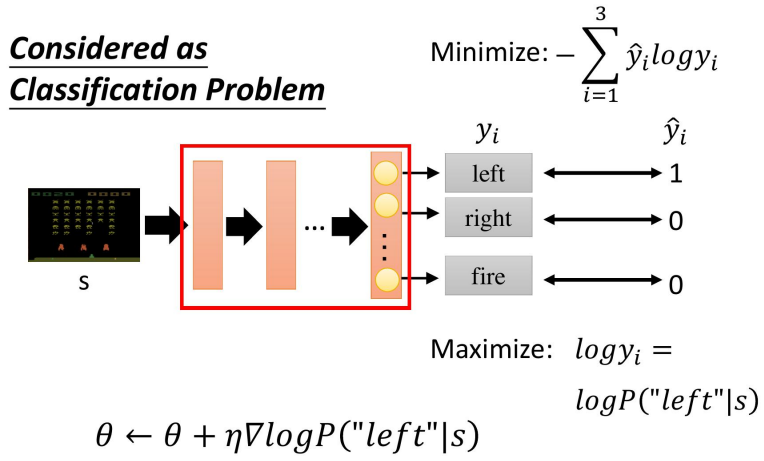
- 目標就是要最大化這個 Rθ bar
  - 先隨機的找一個初始的 θ0，隨機找一個初始的 Actor
  - 然後計算在使用初始的 Actor 的情況下，你的參數對 Rθ bar 的微分，再去 update 你的參數得到 θ1
  - 依此類推……
- 實際運算
  - Rθ bar = summation over 所有的 τ，R(τ) * P(τ | θ)
  - R(τ) 跟 θ 是沒任何關係的，不需要微分，所以可以是個不可微的黑盒子
- 一些問題排解：如果 R(τ) 永遠是正的，會發生什麼事
  - 像玩 Space Invader，得到的 reward 都是正的，殺了外星人就得到正的分數，最糟就是殺不到外星人得到分數是 0
  - 因為實作的時候，我們做的是 sampling，有可能只 sample 到 b、c 這個 action (純移動)，而沒 sample 到 a (射擊)
  - 可能 a 這個 action machine 從來沒試過它，不知道它的 R(τ) 到底有多大，又因為 b 跟 c 機率都會增加，a 沒 sample 到，機率就自動減少
- 解法：reward 要減掉一個 bias
  - 這個 bias 叫做 baseline，好過 baseline 才把那個 action 的機率增加，小於 baseline 把它 action 的機率減小
  - 這樣子就不會造成某一個 action 沒被 sample 到它的機率就會變小

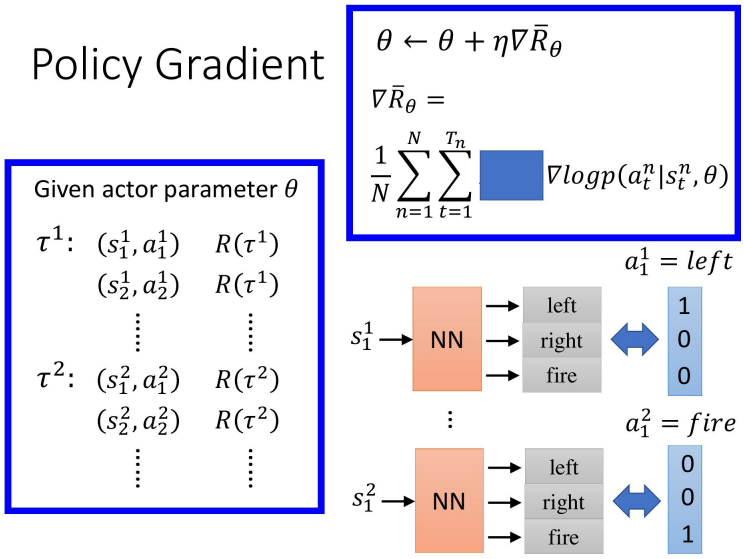
Policy Gradient



- 首先先有一個 actor，參數是 θ (random initialize)
- 先收集資料：
  - 拿這個初始的 actor, θ 去玩 N 次遊戲，收集到 N 個 trajectory (τ)
    - 假設收集到一個 τ^1 (trajectory 1)
      - τ^1 裡面有 state 1，state 1 採取了 action a1
      - 以此類推
    - 玩完這個遊戲以後，可以算出一個 total reward，R(τ)
  - 用上圖式子去 update 參數 θ，有了一個新的 actor
- 再收集資料：（因為 actor 是新的可能，會得到不太一樣的分布，會得到一個不太一樣的結果）
  - 再 update 參數 θ
- 以此類推

問題：上圖公式是什麼意思：

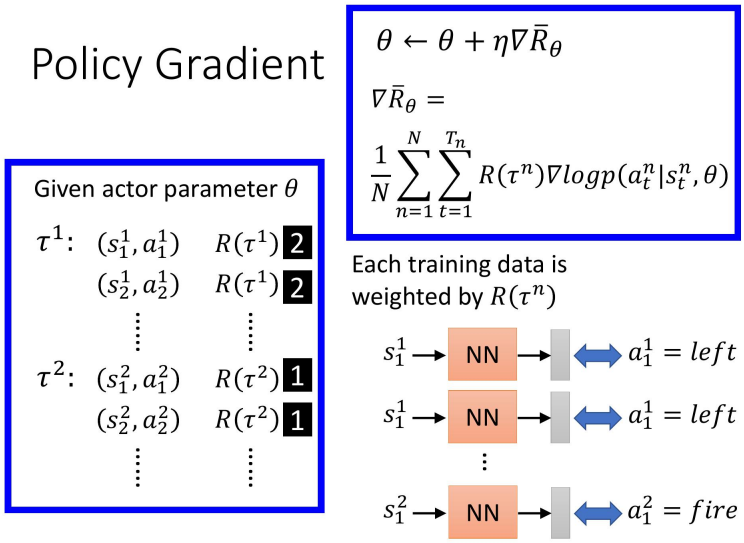


- ∑: summation over 某一個 trajectory 所有的 time step
- ∇ log p(a(上標 n, 下標 t) | s(上標n, 下標t))
  - 

  - 假設現在要做的是一個分類的問題
    - 有一個 network，有一個 actor，這個 actor 當作是一個 classifier
    - 這個 classifier 做的事情是 given 一個畫面 S，它分類說，我們現在應該要採取哪一個 action（有 3 個可以採取的 action）
  - 在做分類的時候，要 train 一個 classifier，要有 labeled data， 要給 network 一個 target
    - 假設現在的目標是 1、0、0（left 是正確的類別）
    - 把 network 的 output 叫做 yi，target 叫做 yi\head
    - 分類問題是在 minimize **cross entropy**
      - cross entropy 就是：summation over 每一個 dimension、summation over 所有的 class
      - 把 yi\head * log(yi)，前面取負號

- 實際上我們在做的事情，本來是一個負號加 minimize
  - 也是 maximize log(yi)
  - 所謂的 yi ，其實就是 P("left"|s)
  - 我們希望我們 network 的 output，跟訂下來的 target (left) 越接近越好
- 所以這是所要的 objective function，要去 maximize 它，對它算一個 gradient

- R(τ)： 常數項， total reward
  -



  - 如果把 R(τ^n) 拿掉，當作那一項等於 1
    - 意思就是：training data裡面有一個 s1、a1(left)
    - 把 s1 丟到 network 裡面，他會給我們 left、right 跟 fire 的機率，希望這個機率跟 1、0、0 越接近越好
    - 就變成了一個分類的問題
  - 實際上，我們在 update 這個式子的時候，真正在做的事情是
    - 現在有一筆 training data，input 和 target 就是這個樣子，請把這個分類問題做對
    - 有一個不一樣的地方就是，加了 reward，就是給一個權重，讓這筆example 被複製 R(τ) 次

Value-based Approach

- Critic 概念

Critic

- A critic does not determine the action.
- Given an actor π, it evaluates the how good the actor is

An actor can be found from a critic.

e.g. Q-learning

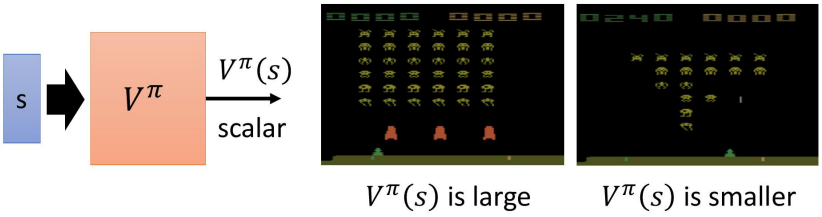
http://combiboilersleeds.com/picaso/critics/critics-4.html

- learn 一個 network 它不做事（不會決定 Action）
- learn 一個 function，這個 function 可以知道現在看到的 observation 有多好
- （其實也可以從 Critic 得到一個 Actor，這樣就是 Q Learning）

Critic

Critic

- State value function $V^{\pi}(s)$
  - When using actor π, the *cumulated* reward expects to be obtained after seeing observation (state) s



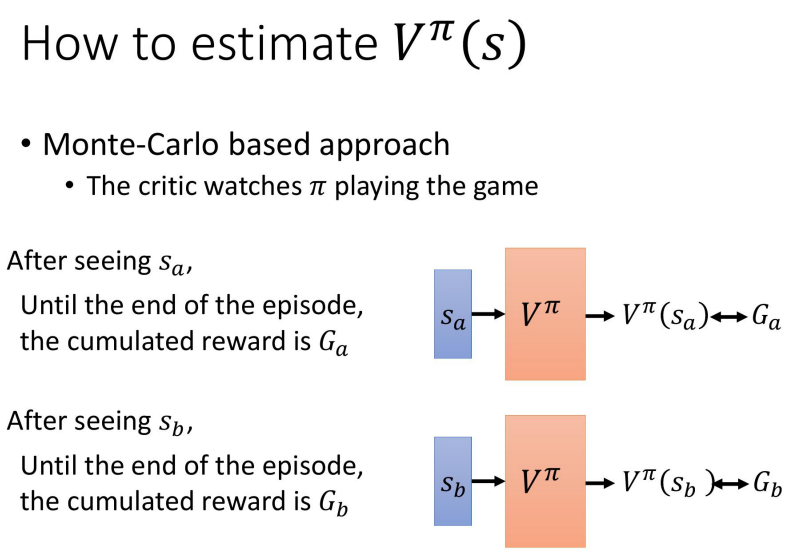
- Critic 並沒有辦法決定要採取哪一個 action
  - 給一個 actor pi，Critic 可以告訴你說這個 actor pi 有多好
- state value function，寫成 V(pi) of x
  - 在給定一個 actor pi 的前提下，假設看到一個 observation or state s，會告訴你接下來一直到遊戲結束的時候，得到 reward 的總和期望值有多大
  - 得到的是看到這個 state 之後，所有 accumulated 的 reward 的期望值
- 以下圍棋為例：
  - 假設你已經有一個下圍棋的 agent，叫做 pi
  - 給它一個 observation，就是棋盤的盤勢，比如說，出手天元

    - V(pi) of x 就是：假設出手下在天元，接下來獲勝的機率有多大
- Critic 的工作，就是衡量一個 actor 好不好
    - 以上左圖的 observation，丟到 Critic 裡可能會 output 一個很大的正值
        - 因為還有很多 alien 可以殺，所以會得到很高的分數
    - 以上右圖的 observation，丟到 Critic 裡可能會會得到相對比較少的值
        - 因為 alien 變得比較少
        - 而且屏障消失了，所以可能很快就會死，分數就比較少

怎麼評估 Critic

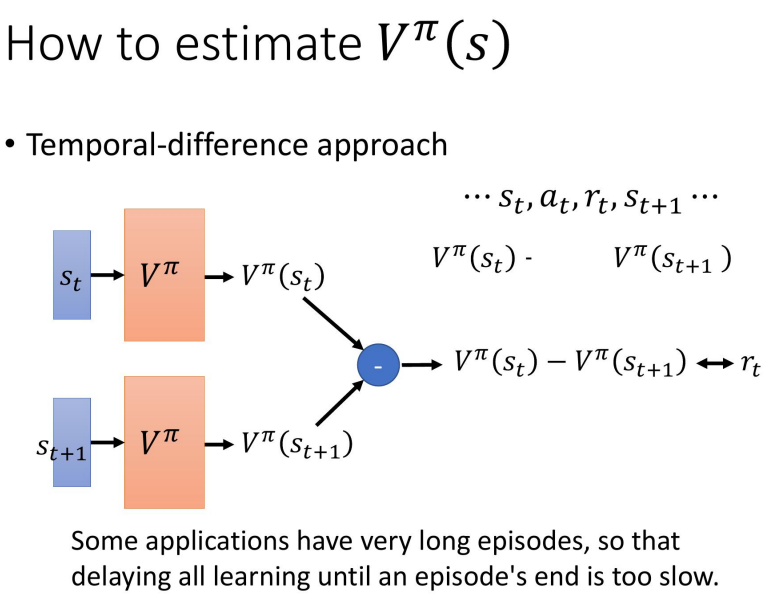
- Critic 其實會隨著 actor 的不同，而得到不同的分數
- 兩個方法，一個是 Monte-Carlo，一個是 Temporal-Difference

Monte-Carlo 的方法 (MD)



- 較直觀
    - 直接去看那個 actor 玩遊戲，假設現在 Critic 觀察到，actor pi 在經過這個 state Sa 以後，它得到的 accumulated 的 reward 是 Ga
    - Critic 就要學說如果 input state Sa，那我的 output 要跟 Ga 越接近越好
- regression 問題
- actor 要調它的參數，那它的 output 跟 Ga 越接近越好

Temporal-Difference 的方法 (TD)

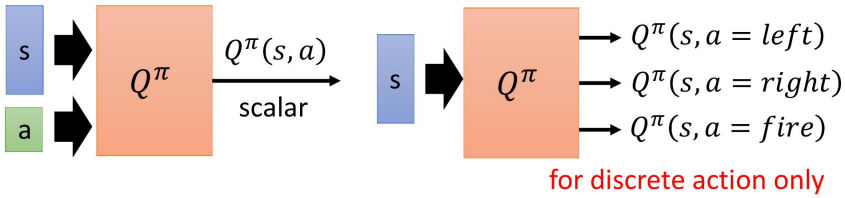


- 較不直觀
  - 一樣讓 Critic 去看 actor 玩遊戲，當看到 actor 在 state st 採取 action at，得到 reward rt，然後跳到 state s(t+1)
  - 一次做一個data，不用等到遊戲結束
  - actor 只要在某一個 state 採取某一個行為，Critic 就可以學了
- 為什麼Critic 這樣就可以學：
  - based on 上圖式子，在 s(t+1) 和 s(t) 中間它們差了 reward 就是 r(t)
  - 即使不知道 accumulated reward 是多少，但我知道 s(t) 輸出的值跟 s(t+1) 輸出的值，中間差了 r(t)，就可以 learn 下去
- 好處：
  - 有些遊戲非常的長，這樣遊戲玩到一半的時候，就可以開始 update 你的 network，不會拖太久

Q function

Another Critic

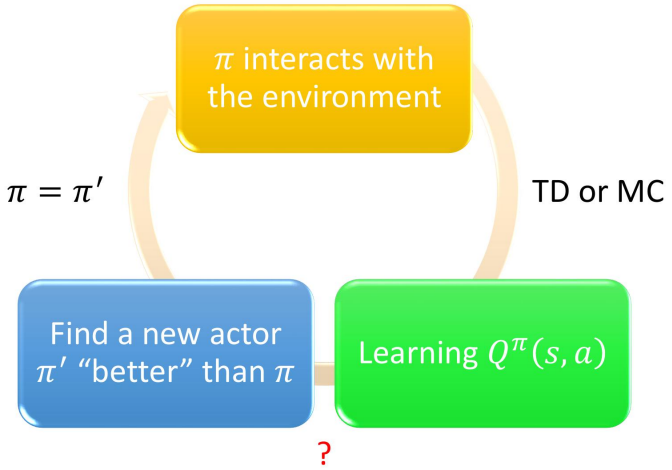
- State-action value function $Q^{\pi}(s,a)$
  - When using actor π, the *cumulated* reward expects to be obtained after seeing observation s and taking a



- 上面提到的那種 critic 沒有辦法拿來決定 action
- 有另外一種 critic 可以拿來決定 action， 這種 critic 叫做 Q function
  - input 是一個 state，**一個 action**
  - 在給定一個 actor pi 的前提之下，在 observation s，**採取了 action a**，到遊戲結束的時候，會得到多少 accumulated reward
  - 會窮舉所有的 a，再搭配 s 代入 Q function

Q-Learning

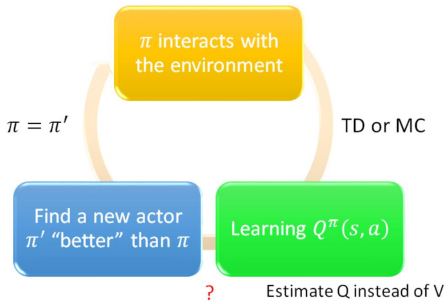
Q-Learning



- 用 Q function 找出一個比較好的 actor，這一招就叫做 Q learning
- 整個 process：
  - 先有一個已經初始的 actor pi，然後這個 actor pi，去跟這個環境互動

- 然後 critic 去觀察 (TD or MC) 這個 actor pi 它跟環境的互動
- 估測說，給定這個 actor 的前提之下，在某一個 state 採取某一個 action，得到的 Q value 是多少
- 可以保證：我們一定能夠找到一個新的、比原來的 pi 更好的 actor pi
- 重點在紅色問號那步：
  - 只要量得出 Q function，就一定可以找到一個更好的 actor pi prime

Q-Learning



- Given $Q^{\pi}(s,a)$, find a new actor π' "better" than π
  - "Better": $V^{\pi'}(s) \geq V^{\pi}(s)$, for all state s

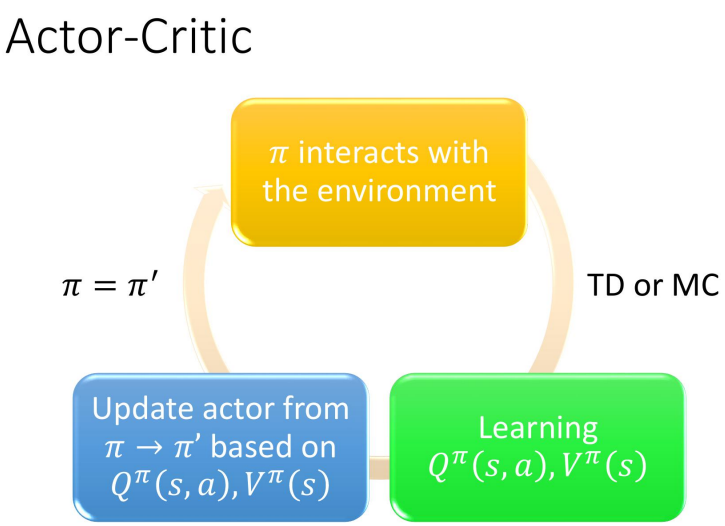
$$\pi'(s) = arg\max_{a} Q^{\pi}(s,a)$$

➢ π' does not have extra parameters. It depends on Q

➢ Not suitable for continuous action a

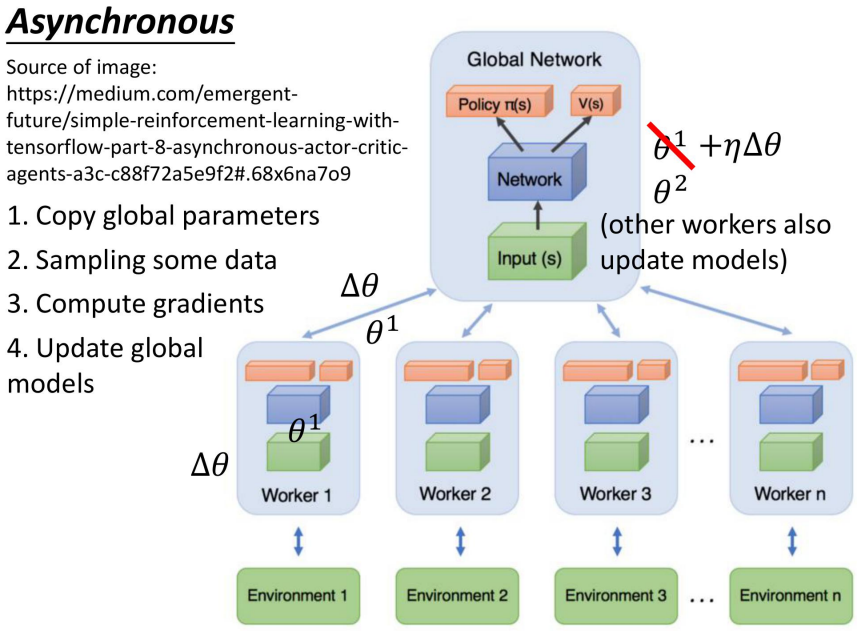
- 什麼叫做 pi prime 一定比 pi 好：
  - pi prime 比 pi 好的定義是：
    - 給 **<u>所有可能的 state s</u>**，如果用 pi 去玩這個遊戲，得到的 reward，一定會小於用 pi prime 得到的 accumulated reward
- 怎麼找到一個比較好的 actor pi prime：
  - 給定一個 Q function，某一個 state 的時候，窮舉所有可能的 action，看哪一個 action 的 Q value 最大
  - 然後這個 a，pi prime 就說這就是它的輸出了
- 但是如果今天 action 無法窮舉，就無法使用
- Q learning 的 trick => *rainbow* 的 paper
  - 有 7 種不同的 DQN 的 tip，對應到彩虹的 7 個顏色

Actor-Critic



- Actor+Critic 的精神
  - actor 不要看環境的 reward，而是看 critic
  - （因為環境有隨機性，reward變化太大）
- 知名方法
  - A2C
    - Advantage Actor-Critic
  - A3C
    - Asynchronous Advantage Actor-Critic

A3C (Asynchronous Advantage Actor-Critic)



- Asynchronous 的意思：
  - 有一個 global 的 network、global 的 actor 跟 global 的 critic
  - 每到學習的時候呢，就去跟 global 的 actor 和 critic copy 一組參數過來，讓這個 actor 實際去跟環境互動（類似開分身的概念）
  - 那互動完以後，Critic 就會告訴 actor 說要怎麼樣 update 參數。把這個 updated 參數，傳回去 global 的 network
  - 每一個分身，都會傳一個 update 的方向，合起來可以一起做 update，等於就是做平行的運算
- 實作上，要做 asynchronous 這一招，前提是要有很多很多的 machine 這樣子
  - 如果只有一台 machine，就只能用 A2C

小結論

- Actor 跟 Critic 可以合在一起 train
  - 好處：簡單講就是比較強

Inverse reinforcement learning

- Imitation learning 的一種
- 在 inverse reinforcement learning 裡面
  - 只有 environment 跟 actor，沒有 reward function
  - 還有這個 expert demo trajectory
  - 意思是：有高手去把這個遊戲，玩了 N 遍給 machine 看
- 沒有 reward function 很正常?
  - 多數生活中的 case，都是沒有 reward function 的 (不像圍棋有明確輸贏，電玩有明確得分)

    - 比如：自駕車、chat bot (用一些自己訂出來的 reward，有時候會很奇怪)
- 雖然不知道最好的 actor 是什麼，但是我們有專家 (expert)
    - 專家去玩了 N 場遊戲，告訴我們說厲害的人玩這個遊戲，看起來是怎麼樣的
    - 根據專家的 demo，還有 environment，加上 inverse reinforcement learning，可以推出 reward function 應該長什麼樣子
- 用 inverse reinforcement learning 的方法去 **推出 reward function**，最後再用 reinforcement learning 的方法去找出最好的 actor
    - 概念：那些 experts 永遠是對的
    - 訂一個 reward function，一定要讓 expert 得到的分數，比 actor 得到的分數高 (先射箭，再畫靶)
    - 根據新的 reward，actor 去 maximize 新的 reward function 以後，再去跟環境互動，他就會得到新的 trajectory
    - 當他變得跟老師一樣厲害以後，再改一下規格，讓老師算出來的分數，還是比較高
- 整體概念跟 GAN 很像
    - generator 換個名字叫做 actor
    - discriminator 換個名字叫做 reward function