Introduction of Reinforcement Learning

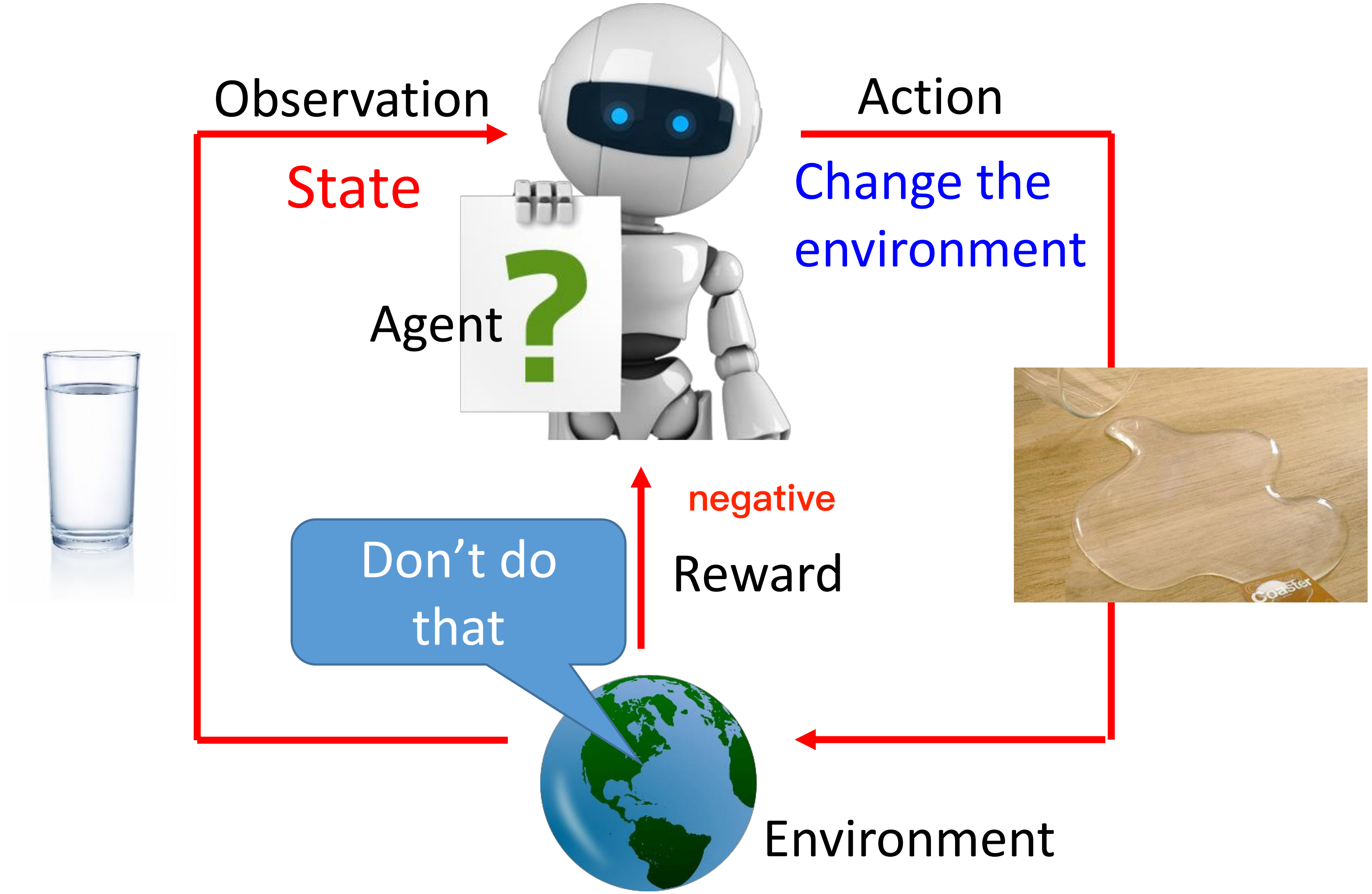
Deep Reinforcement Learning

Deep Reinforcement Learning: AI = RL + DL

Reference

- Textbook: Reinforcement Learning: An Introduction
  - http://incompleteideas.net/sutton/book/the-book.html
- Lectures of David Silver
  - http://www0.cs.ucl.ac.uk/staff/D.Silver/web/Teaching.html (10 lectures, around 1:30 each)
  - http://videolectures.net/rldm2015_silver_reinforcement_learning/ (Deep Reinforcement Learning)
- Lectures of John Schulman
  - https://youtu.be/aUrX-rP_ss4

Scenario of Reinforcement Learning



Machine會調整它採取的行為使得它得到的reward可以最大化

Scenario of Reinforcement Learning

Agent learns to take actions maximizing expected reward.

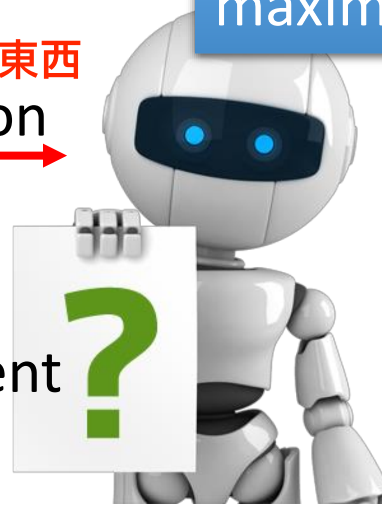
Machine可以感測到的東西

Observation

State

Action

Change the environment



Agent

positive

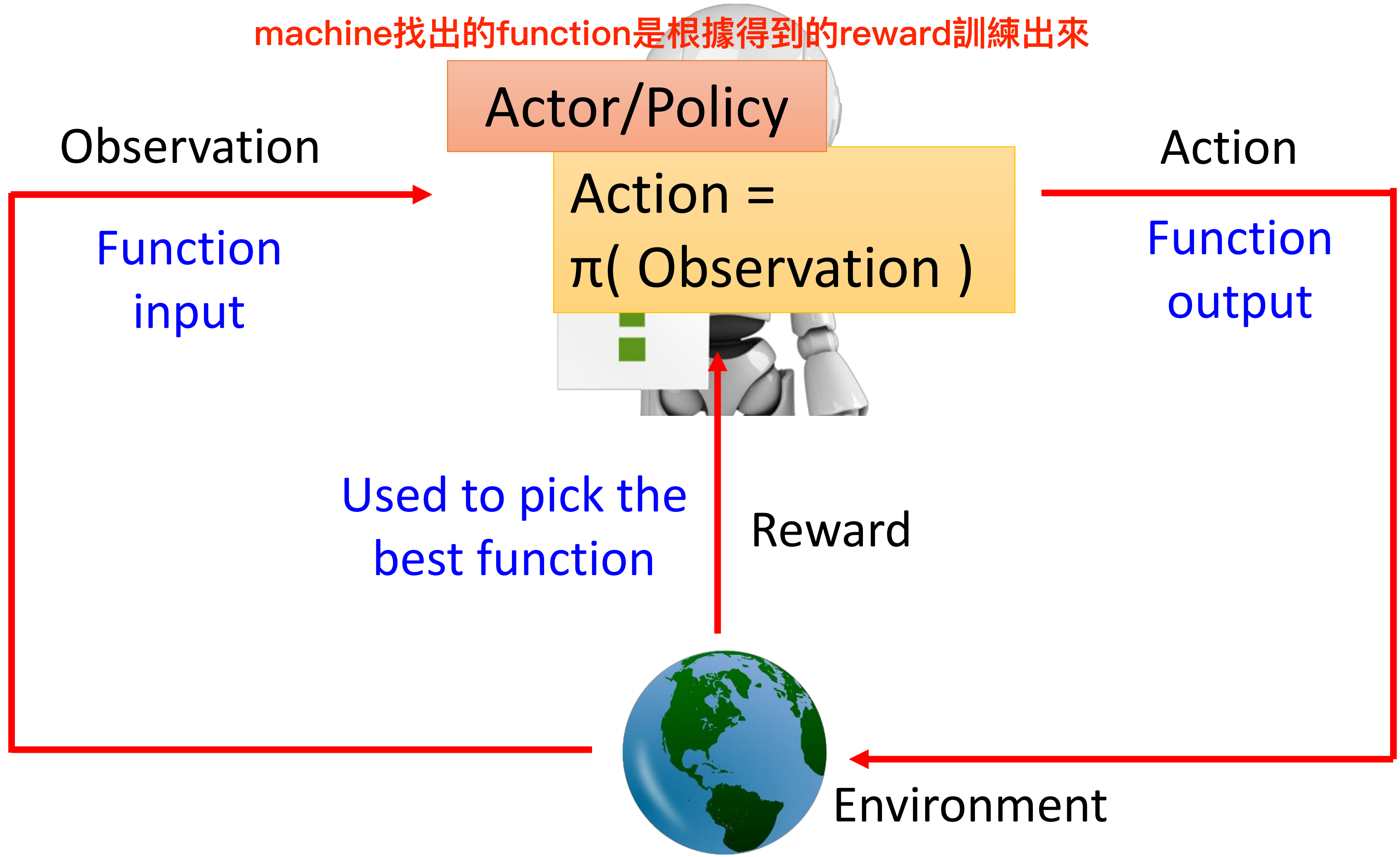
Reward

Thank you.

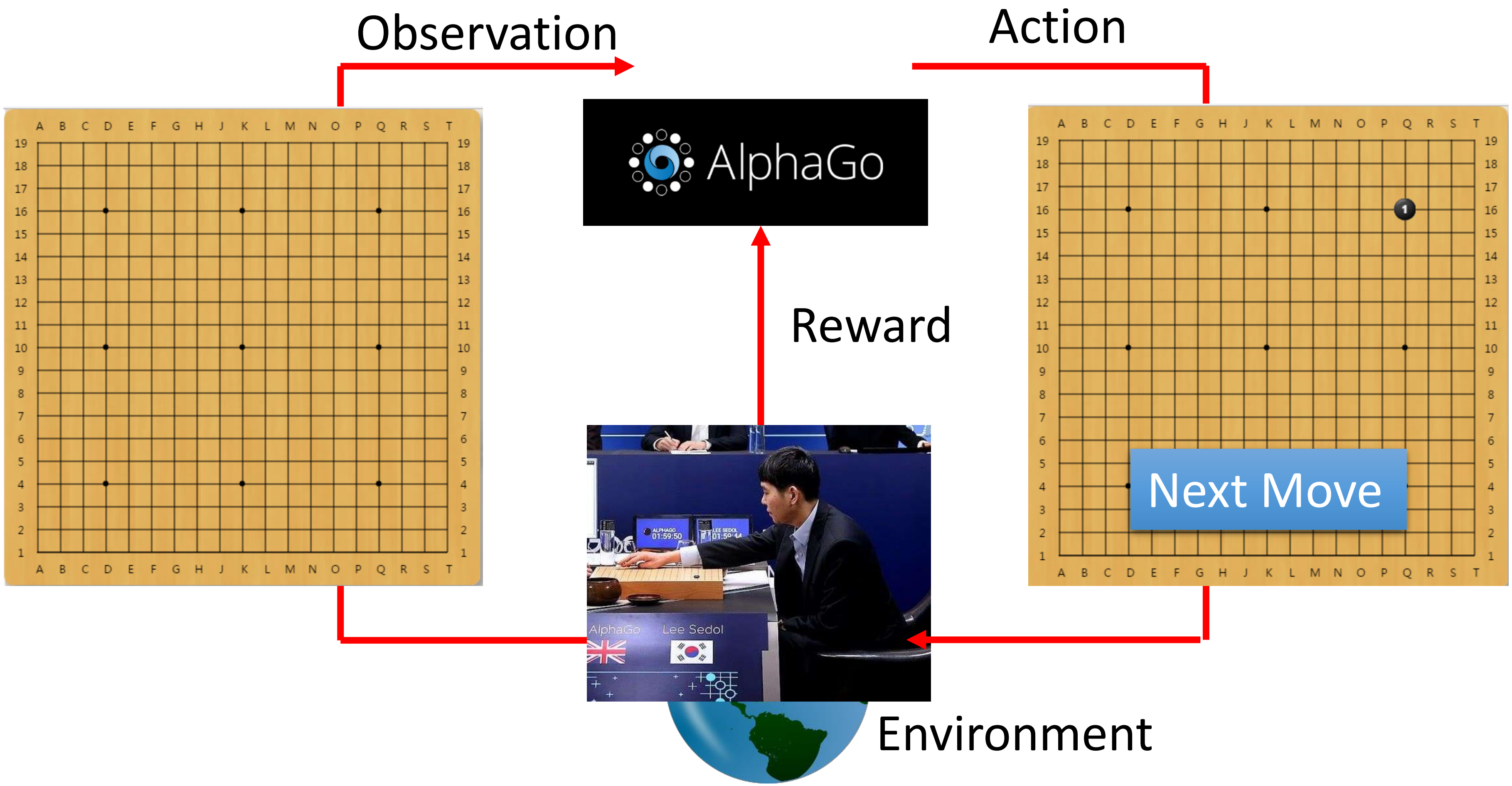
Environment

https://yoast.com/how-to-clean-site-structure/

Machine Learning
≈ Looking for a Function

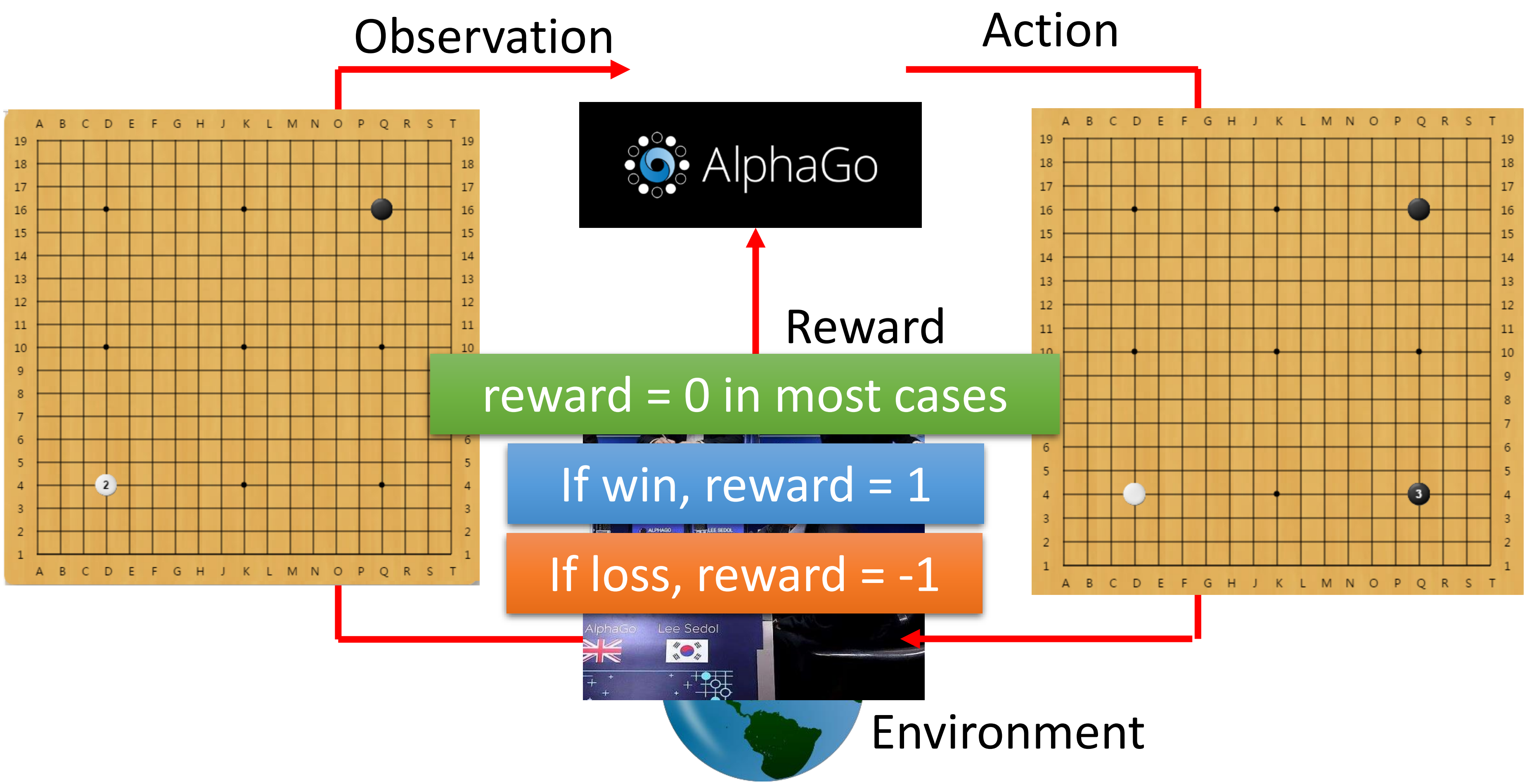


Learning to play Go



Learning to play Go

Agent learns to take actions maximizing expected reward.



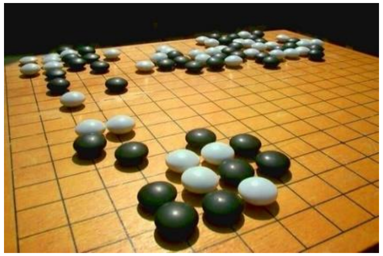
不好train，因為只有在最後一步下完（整盤棋結束得到輸贏結果）才能得到reward

Learning to play Go

照著棋譜學

- Supervised: Learning from teacher

Next move: "5-5"



Next move: "3-3"

- Reinforcement Learning Learning from experience

讓machine直接跟人pk，根據輸贏結果決定如何學習

First move → …… many moves …… → Win!

(Two agents play with each other.)

先學到一個程度在開始learn出兩個machine自己互相下

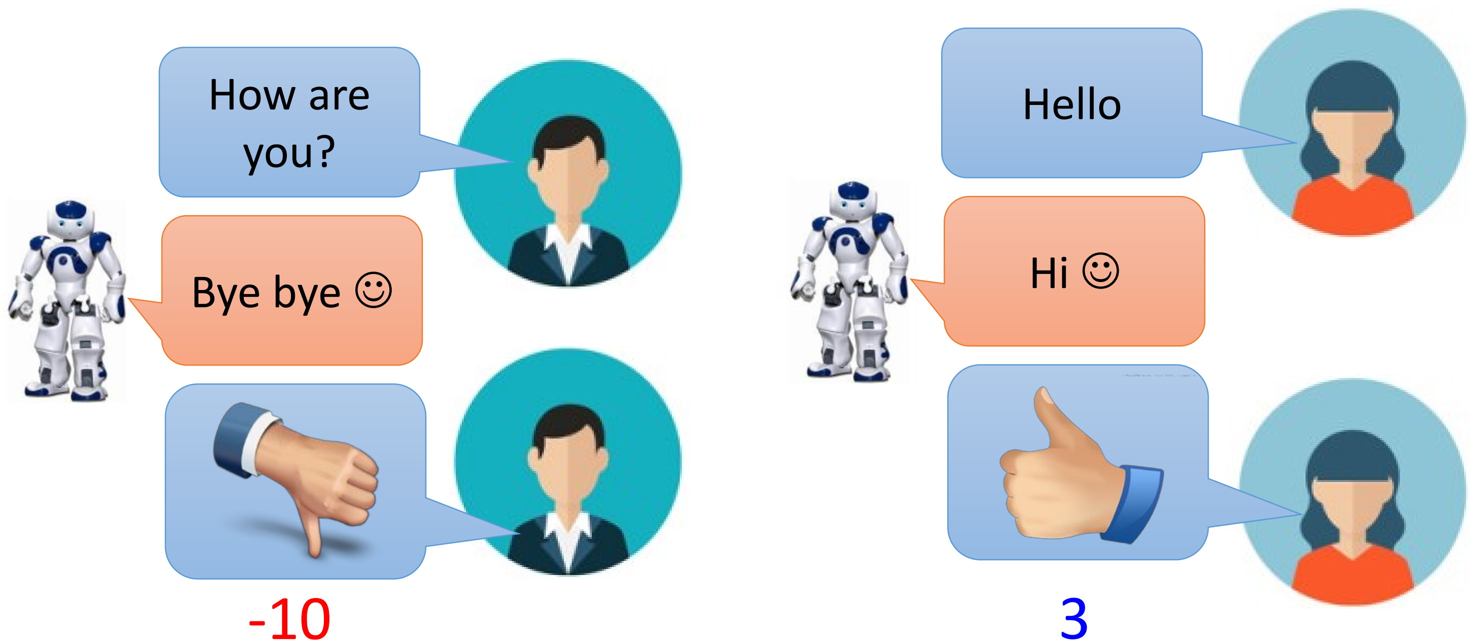
Alpha Go is supervised learning + reinforcement learning.

Learning a chat-bot

https://image.freepik.com/free-vector/variety-of-human-avatars_23-2147506285.jpg

http://www.freepik.com/free-vector/variety-of-human-avatars_766615.htm

- Machine obtains feedback from user



- Chat-bot learns to maximize the ***expected reward***

機器會想辦法maximize得到的reward

然而跟人對話到結束是很花時間的，因此可以learn兩個machine護療

Learning a chat-bot

- Let two agents talk to each other (sometimes generate good dialogue, sometimes bad)

How old are you?

See you.

See you.

See you.

How old are you?

I am 16.

I though you were 12.

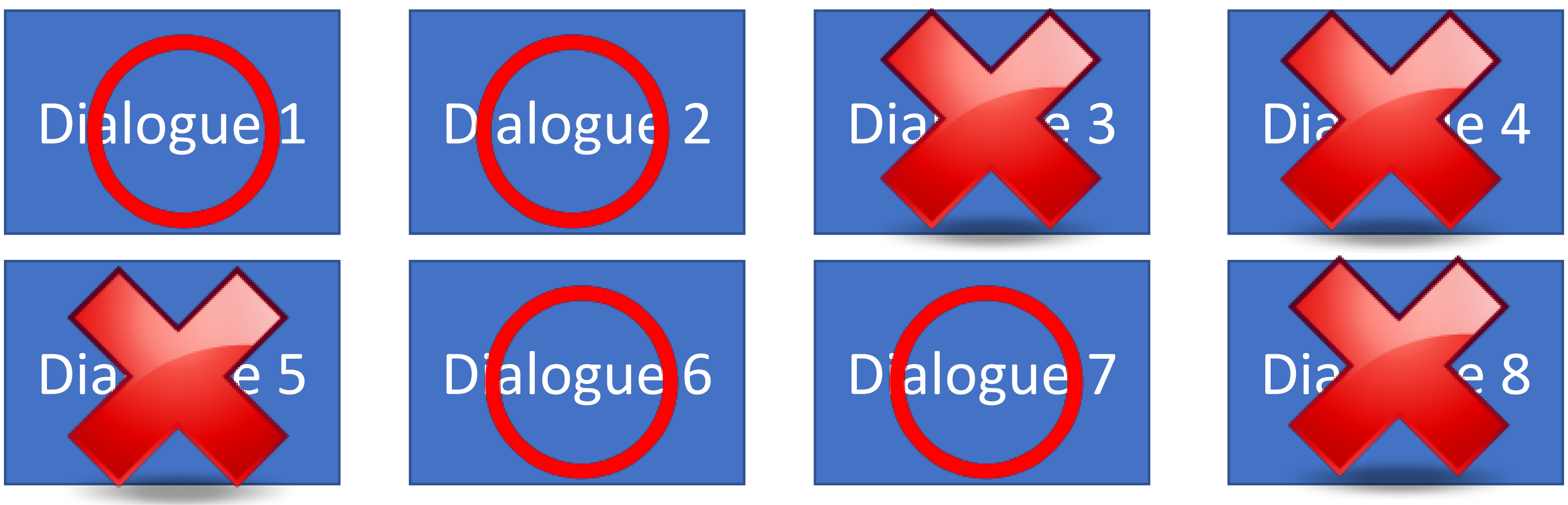
What make you think so?

定義一個predefined的rule，根據對話內容給許好或不好的評分

Learning a chat-bot

- By this approach, we can generate a lot of dialogues.
- Use some pre-defined rules to evaluate the goodness of a dialogue

互聊完後再根據定義好的function去label這些對話內容



Machine learns from the evaluation

Deep Reinforcement Learning for Dialogue Generation
https://arxiv.org/pdf/1606.01541v3.pdf

Learning a chat-bot

- Supervised

"Hello" — Say "Hi"

"Bye bye" — Say "Good bye"

- Reinforcement

……. ……. ……

Hello ☺ ……

Agent Agent

Bad

人跟機器亂聊，如果人離開對話則給予penalty

More applications

- Flying Helicopter
  - https://www.youtube.com/watch?v=0JL04JJjocc
- Driving
  - https://www.youtube.com/watch?v=0xo1Ldx3L5Q
- Robot
  - https://www.youtube.com/watch?v=370cT-OAzzM
- Google Cuts Its Giant Electricity Bill With DeepMind-Powered AI 利用RL使他們的server省下大量的電
  - http://www.bloomberg.com/news/articles/2016-07-19/google-cuts-its-giant-electricity-bill-with-deepmind-powered-ai
- Text generation
  - https://www.youtube.com/watch?v=pbQ4qe8EwLo

Example: Playing Video Game

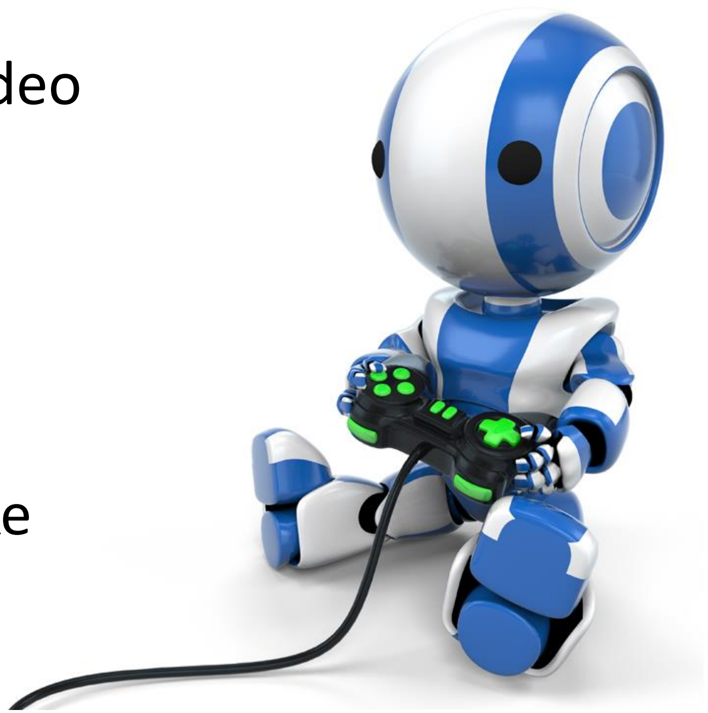
- Widely studies:
  - Gym: https://gym.openai.com/
  - Universe: https://openai.com/blog/universe/

雖然遊戲中也是有遊戲的AI，但是他們都是已經寫好的定義
而我們這邊是讓machine以人的觀點來玩遊戲

Machine learns to play video games as human players

➢ What machine observes is pixels

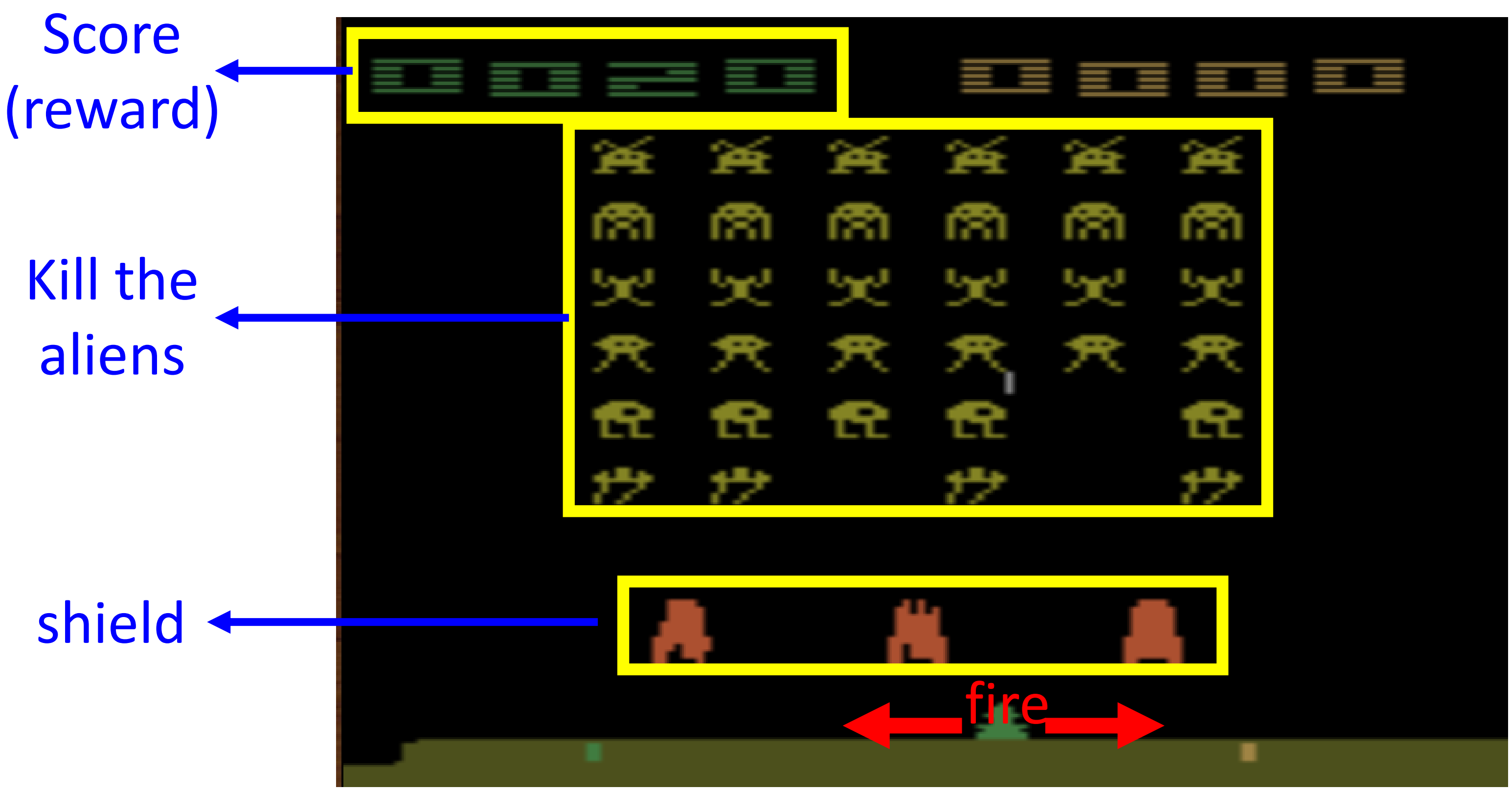
➢ Machine learns to take proper action itself



Example: Playing Video Game

- Space invader

Termination: all the aliens are killed, or your spaceship is destroyed.



Example: Playing Video Game

- Space invader
  - Play yourself: http://www.2600online.com/spaceinvaders.html
  - How about machine: https://gym.openai.com/evaluations/eval_Eduozx4HRyqgTCVk9ltw

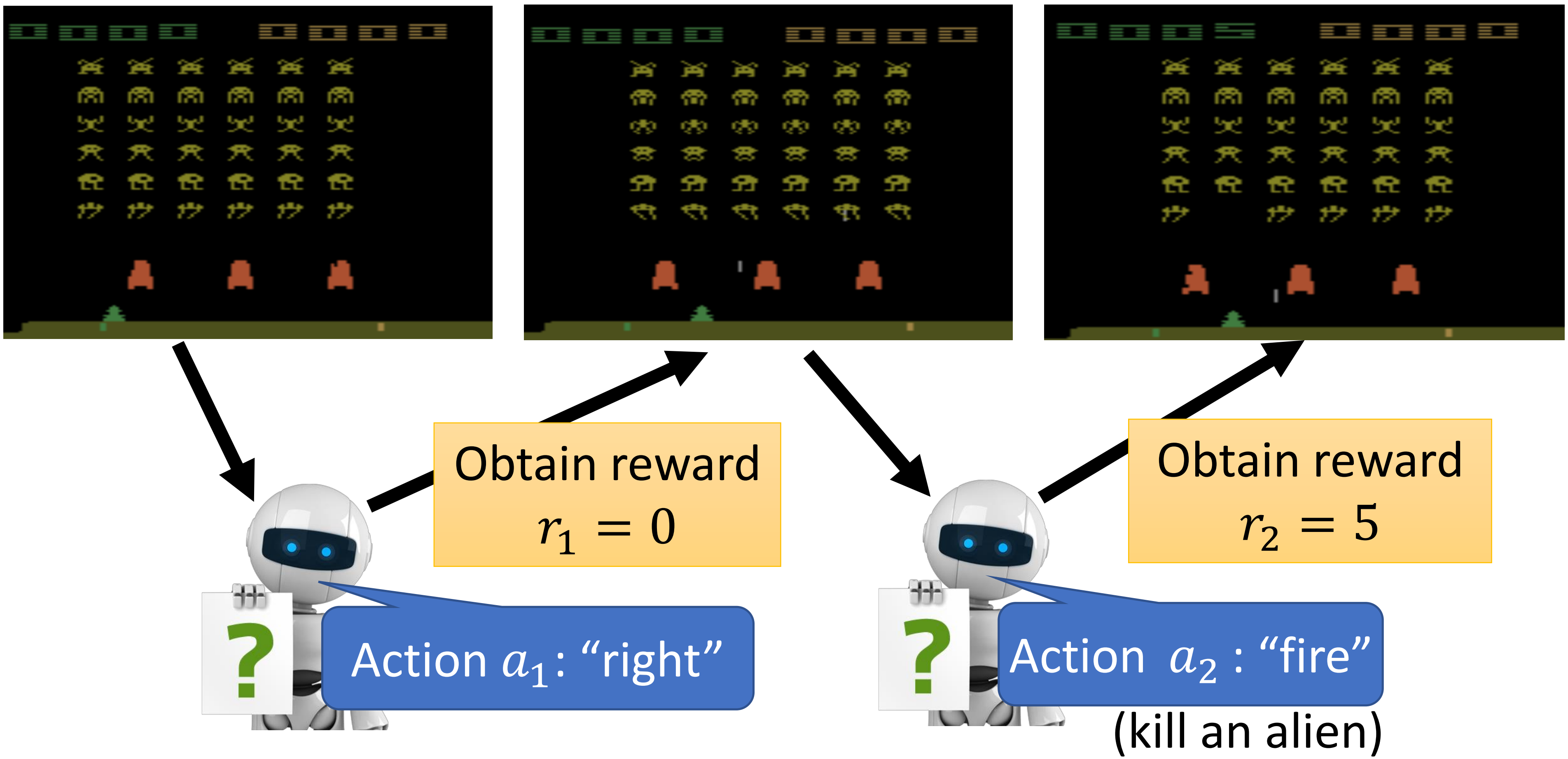
Example: Playing Video Game

為了讓machine看到現在的observation時也能記得之前發生的過程，我們可以將input改成輸入前一段時間所看到的畫面

Start with observation s_1

Observation s_2

Observation s_3

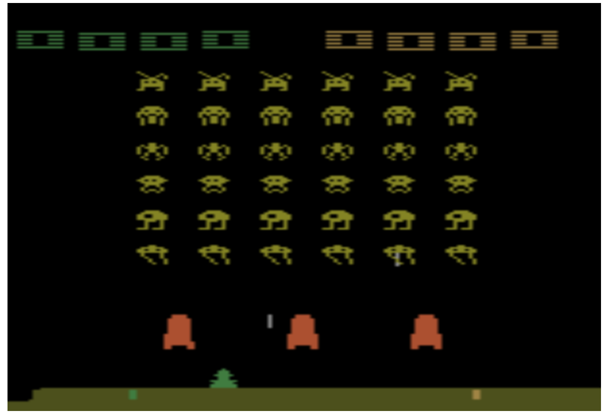


Usually there is some randomness in the environment

Example: Playing Video Game

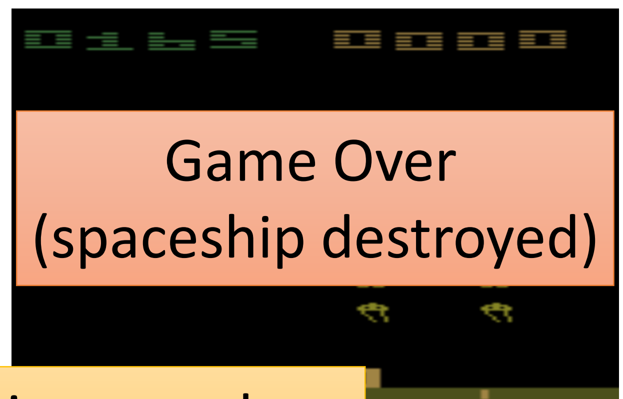
Start with observation s_1

Observation s_2



Observation s_3

After many turns



Game Over
(spaceship destroyed)

Obtain reward r_T

Action a_T

一場遊戲

This is an ***episode***.

Learn to maximize the expected cumulative reward per episode

Properties of Reinforcement Learning

- **Reward delay**
  在這個遊戲中，只有開火才能得到reward
  - In space invader, only “fire” obtains reward
    - Although the moving before “fire” is important
  - In Go playing, it may be better to sacrifice immediate reward to gain more long-term reward
  犧牲短期利益換取最後最大的reward（策略）
- Agent’s actions **affect the subsequent data it receives**
  - E.g. Exploration

不同action會影醒不同的結果，因此在學習的時候要考量到不同可能的action initial讓他去學習

Outline

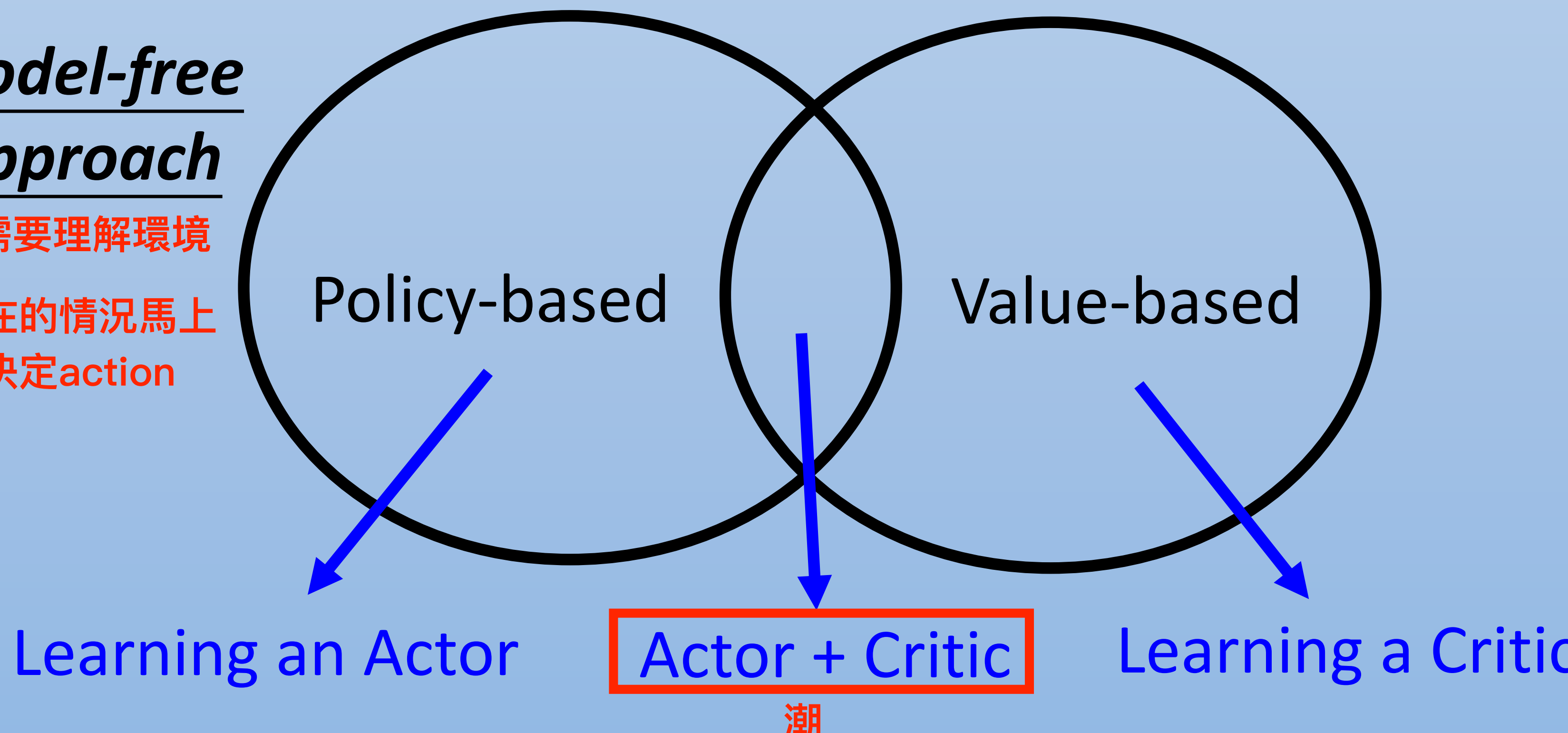
RL: model–based，Model–free

Alpha Go: policy-based + value-based + model-based

Model-free Approach

不需要理解環境

看到現在的情況馬上就可以決定action



Model-based Approach

已經知道環境的狀況，並且訓練一個model，可以對未來狀況直接做預測

Policy-based Approach

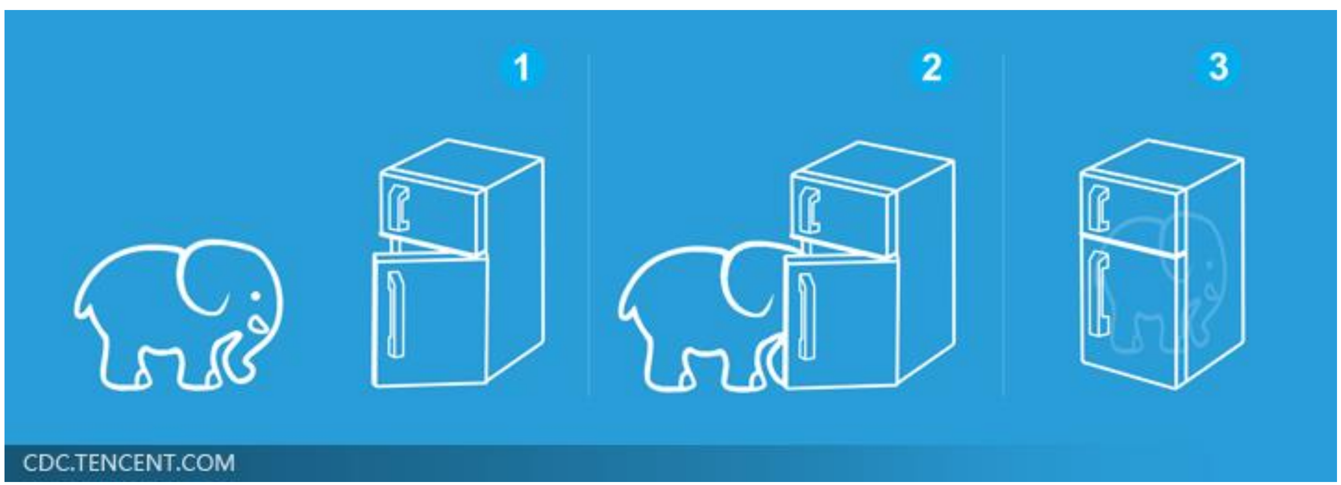
Learning an Actor

Three Steps for Deep Learning

define function set

Step 1: Neural Network as Actor → Step 2: goodness of function → Step 3: pick the best function

Deep Learning is so simple ……

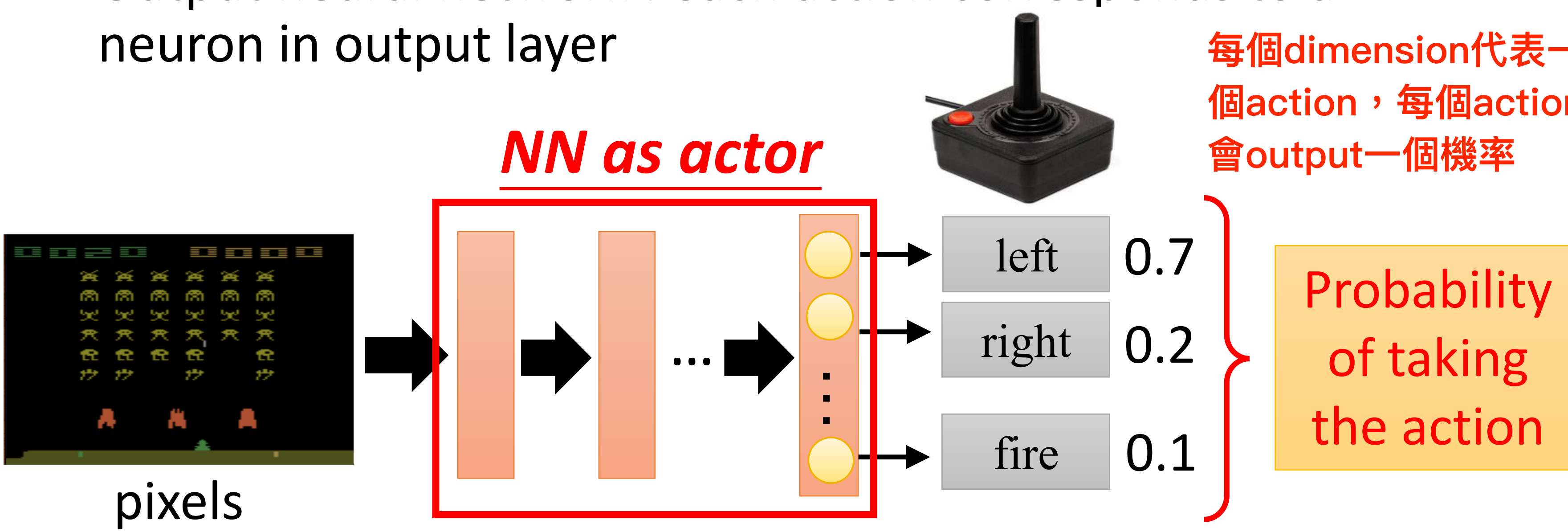


Neural network as Actor

RL之所以厲害是將所要找的function用DNN取代之

- Input of neural network: the observation of machine represented as a vector or a matrix
- Output neural network : each action corresponds to a neuron in output layer

每個dimension代表一個action，每個action會output一個機率



What is the benefit of using network instead of lookup table?

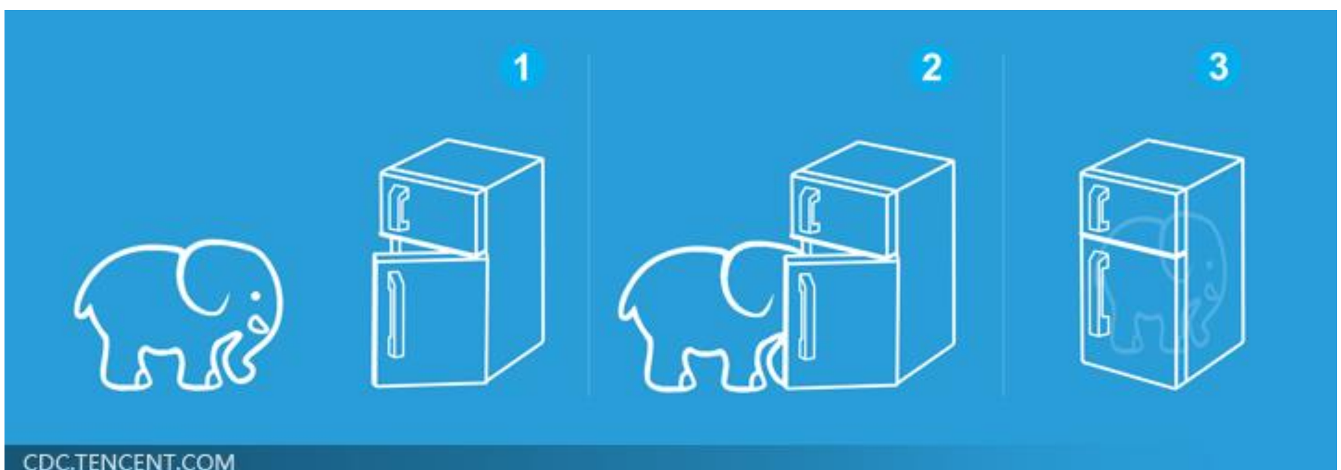
把本來要找的function（可能是table），用NN取代，好處是generalization（一般

generalization

Three Steps for Deep Learning

Step 1: Neural Network as Actor → Step 2: goodness of function → Step 3: pick the best function

Deep Learning is so simple ……



Goodness of Actor

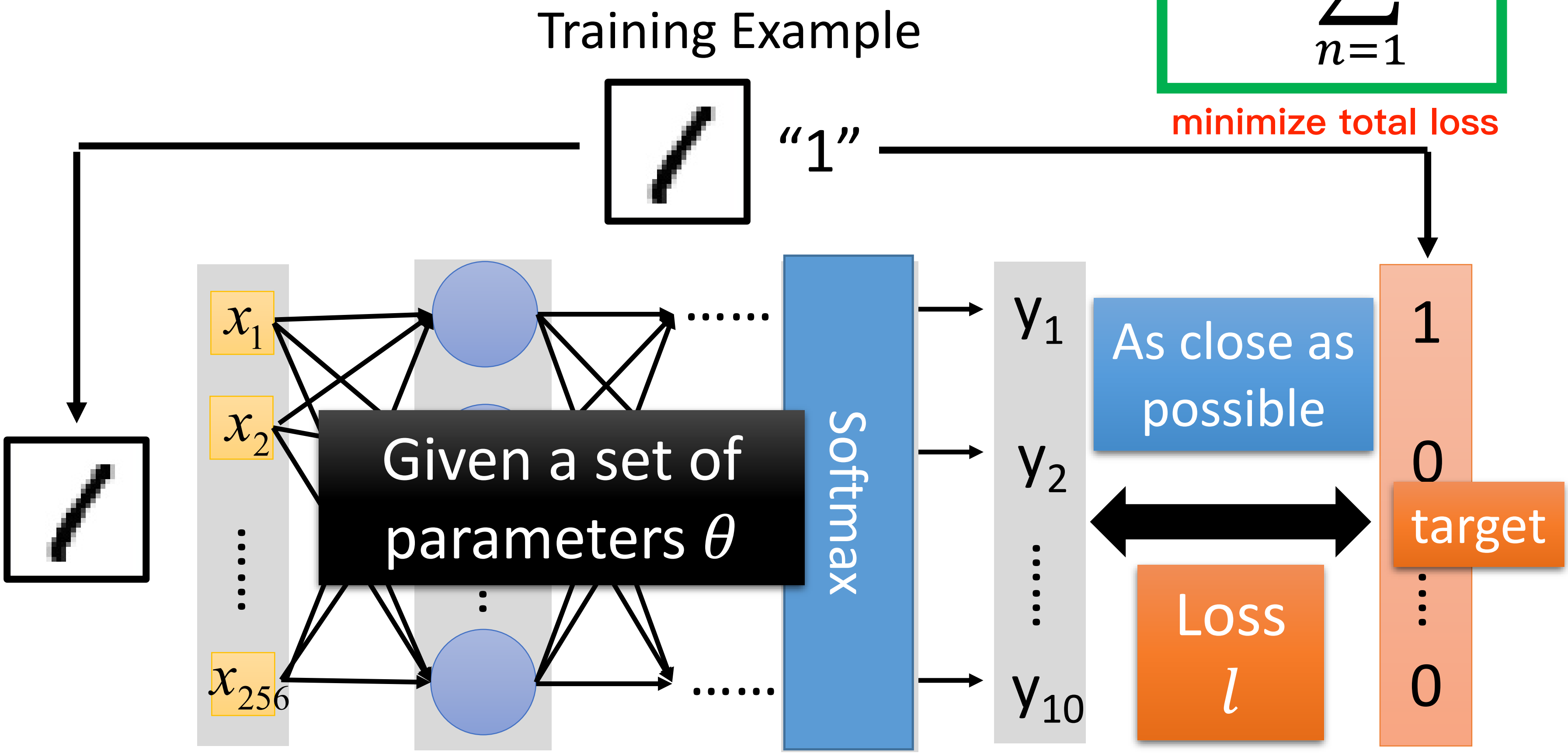
評估一個actor的好壞

- Review: Supervised learning

Total Loss:

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

minimize total loss



一模一樣的action可能會產生不一樣的reward，因為環境有隨機性的，舉例敵人不一定每次移動方向都一樣

Goodness of Actor

所以要計算的不是一個actor玩一次遊戲的結果，而是玩很多次後得到的reward之期望值

- 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 *expected value* of R_θ

算玩很多次game的total reward之期望值

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

Goodness of Actor

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

trajectory，s1,s2..是環境的變動機率（遊戲決定），a1,a2是機器決定之actor，而r1r2為reward，由遊戲決定

- $\tau = \{s_1, a_1, r_1, s_2, a_2, r_2, \cdots, s_T, a_T, r_T\}$

$P(\tau|\theta) =$

機器決定　遊戲決定　機器決定　遊戲決定

$$p(s_1)p(a_1|s_1,\theta)p(r_1,s_2|s_1,a_1)p(a_2|s_2,\theta)p(r_2,s_3|s_2,a_2)\cdots$$

得到trajectory的機率

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

雖然機率是固定的，但condition會變化

not related to your actor

Control by your actor π_θ

$p(a_t = \text{"}fire\text{"}|s_t,\theta) = 0.7$

s_t → Actor π_θ

left → 0.1

right → 0.2

fire → 0.7

Goodness of Actor

theta: 某一個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: $P(\tau|\theta)$

窮舉所有的τ 每個τ之機率 無法窮舉，因此改用sample逼近

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

Sum over all possible trajectory

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

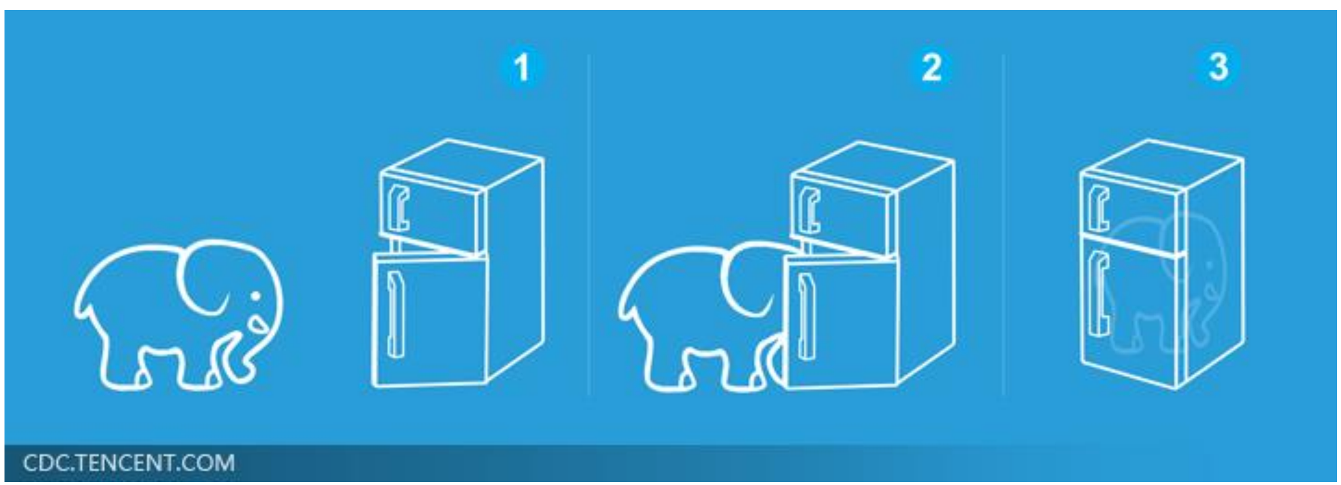
Sampling τ from $P(\tau|\theta)$ N times

實作上無法窮舉所有可能之trajectory，因此改讓actor玩N遍遊戲後從中sample

Three Steps for Deep Learning

Step 1: Neural Network as Actor → Step 2: goodness of function → Step 3: pick the best function

Deep Learning is so simple ……



Gradient Ascent

- Problem statement

過去都用gradient ascent

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

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

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

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

只要$P(\tau|\theta)$ 可微分即可，但是微分後無法用sample取代summation

Policy Gradient

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

summation碰到P($\tau|\theta$)才能取sample，為此這邊做了一個trick

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

$R(\tau)$ do not have to be differentiable 因為θ與他無關，不需微分

It can even be a black box.

$$= \boxed{\sum_\tau} R(\tau)\boxed{P(\tau|\theta)}\nabla logP(\tau|\theta) \qquad \boxed{\frac{dlog(f(x))}{dx} = \frac{1}{f(x)}\frac{df(x)}{dx}}$$

$$\approx \boxed{\frac{1}{N}\sum_{n=1}^{N}} R(\tau^n)\underline{\nabla logP(\tau^n|\theta)}$$

換成sample

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

Policy Gradient

$$\nabla logP(\tau|\theta) = ?$$

- $\tau = \{s_1, a_1, r_1, s_2, a_2, r_2, \cdots, s_T, a_T, r_T\}$

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

$$logP(\tau|\theta)$$

所有與θ無關的項可以刪去

$$= \cancel{logp(s_1)} + \sum_{t=1}^{T} logp(a_t|s_t,\theta) + \cancel{logp(r_t, s_{t+1}|s_t, a_t)}$$

$$\boxed{\nabla logP(\tau|\theta) = \sum_{t=1}^{T} \nabla logp(a_t|s_t,\theta)}$$

Ignore the terms not related to θ

Policy Gradient

$$\nabla \log P(\tau|\theta) = \sum_{t=1}^{T} \nabla \log p(a_t|s_t,\theta)$$

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

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

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

What if we replace $R(\tau^n)$ with r_t^n ……

If in τ^n machine takes a_t^n when seeing s_t^n in

$R(\tau^n)$ is positive → Tuning θ to increase $p(a_t^n|s_t^n)$

$R(\tau^n)$ is negative → Tuning θ to decrease $p(a_t^n|s_t^n)$

一定要看完全程的reward在學習，不然以遊戲為例子，只有在開火的時候才有機會得到reward

It is very important to consider the cumulative reward $R(\tau^n)$ of the whole trajectory τ^n instead of immediate reward r_t^n

這樣機器只會學到瘋狂開火

实际实作

Policy Gradient

Given actor parameter θ

τ^1: (s_1^1, a_1^1) $R(\tau^1)$

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

$\vdots$ $\vdots$

τ^2: (s_1^2, a_1^2) $R(\tau^2)$

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

$\vdots$ $\vdots$

Update Model

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

$$\nabla \bar{R}_\theta =$$

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

Data Collection

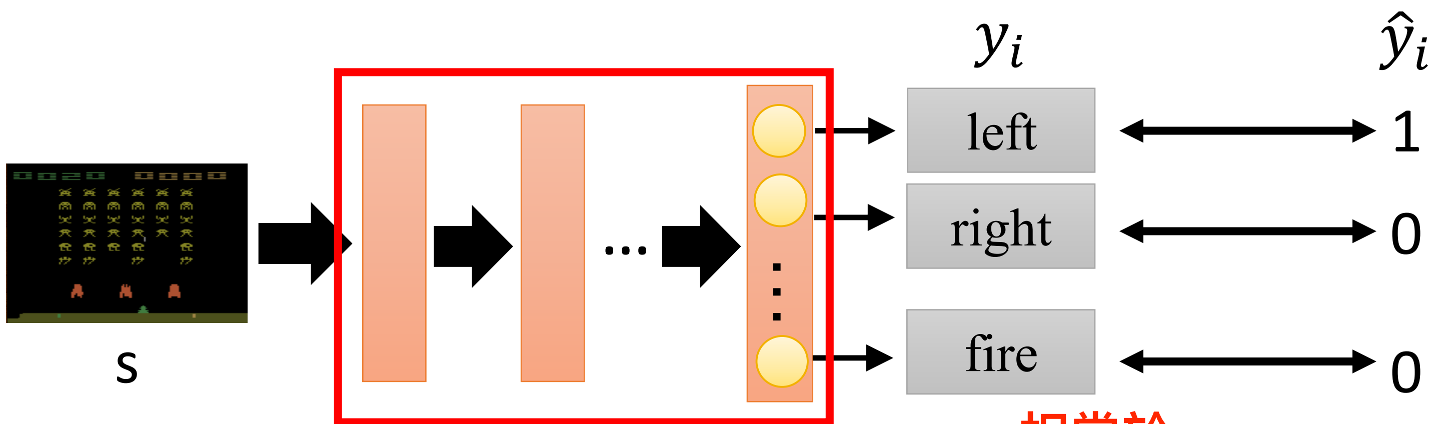
Policy Gradient

把它當成分類問題

Considered as Classification Problem

minimize cross entropy

Minimize: $-\sum_{i=1}^{3} \hat{y}_i log y_i$



相當於

Maximize: $log y_i =$

$logP("left"|s)$

$$\theta \leftarrow \theta + \eta \nabla logP("left"|s)$$

解一個分類的問題，希望network的output越接近我們定義好的output越好

Policy Gradient

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

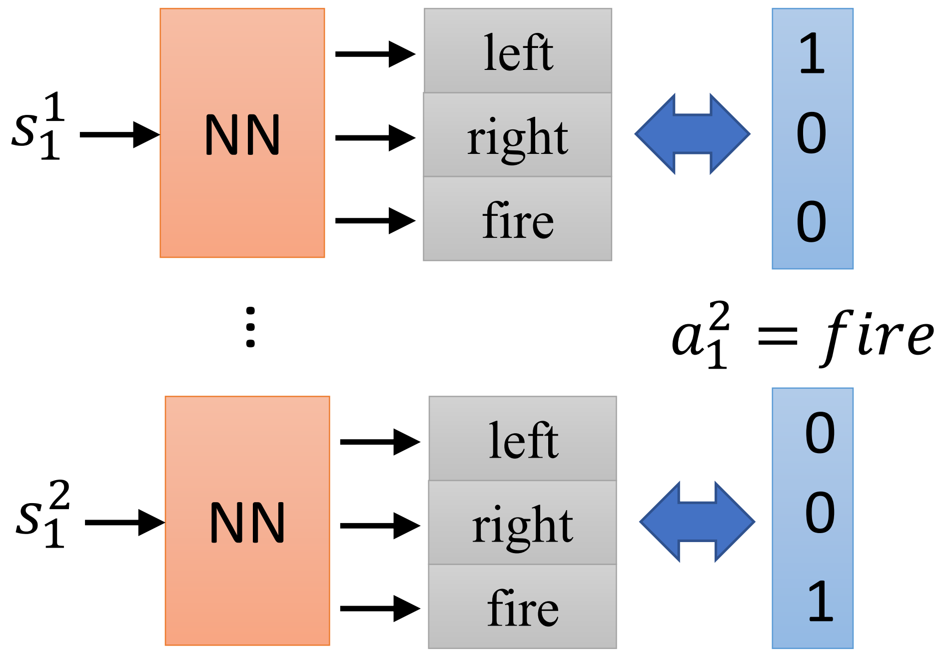
$$\nabla \bar{R}_\theta = \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n} \nabla \log p(a_t^n | s_t^n, \theta)$$

先拿掉reward來看的話就是一個很基本的分類問題

Given actor parameter θ

τ^1:	(s_1^1, a_1^1)	$R(\tau^1)$
	(s_2^1, a_2^1)	$R(\tau^1)$
	$\vdots$	$\vdots$
τ^2:	(s_1^2, a_1^2)	$R(\tau^2)$
	(s_2^2, a_2^2)	$R(\tau^2)$
	$\vdots$	$\vdots$

$a_1^1 = left$



$a_1^2 = fire$

Policy Gradient

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

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

Given actor parameter θ

τ^1: (s_1^1, a_1^1) $R(\tau^1)$ 2

(s_2^1, a_2^1) $R(\tau^1)$ 2

⋮ ⋮

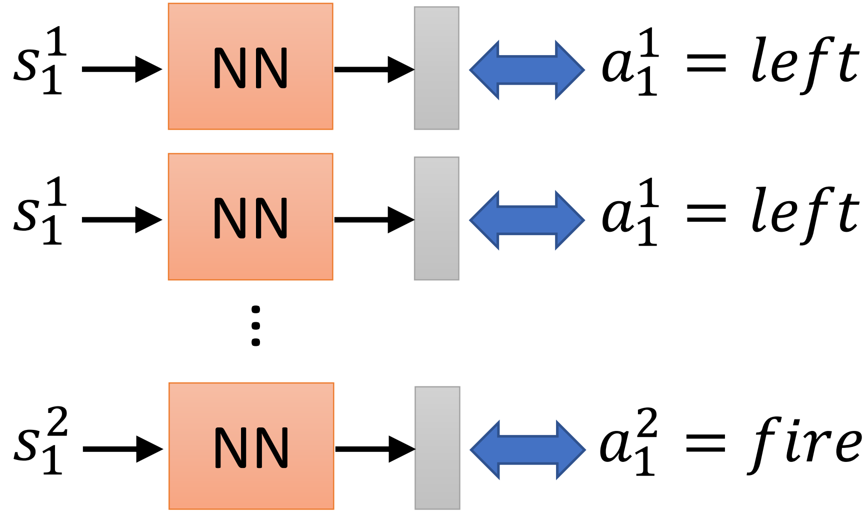
τ^2: (s_1^2, a_1^2) $R(\tau^2)$ 1

(s_2^2, a_2^2) $R(\tau^2)$ 1

⋮ ⋮

Each training data is weighted by $R(\tau^n)$

因為reward是2因此複製兩次



因為reward是1因此複製1次

Add a Baseline

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

reward有可能都是正的，ideally是沒問題的但是因為有可能沒smple到因此讓機器改變各自的出現機率

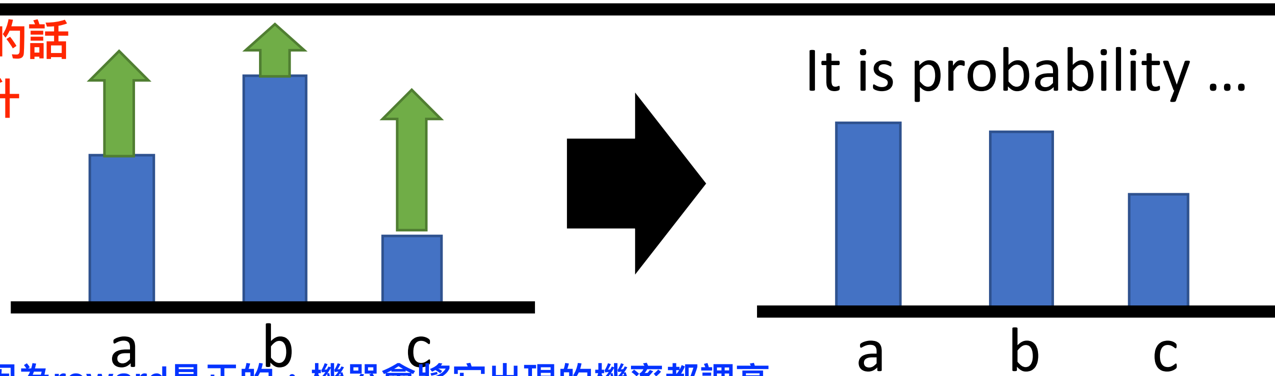
$$\theta^{new} \leftarrow \theta^{old} + \eta \nabla \bar{R}_{\theta^{old}}$$

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

baseline使得reward有時候是正的有時候是負的

如果全部都有sample的話每項會根據reward上升

Ideal case

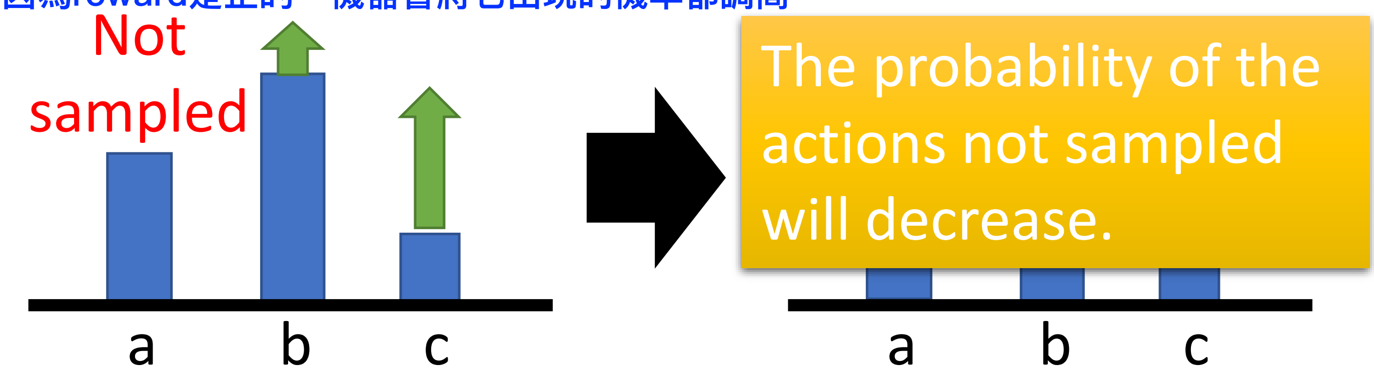


因為reward是正的，機器會將它出現的機率都調高

如果只有一部份有sample到，對於沒sample到的項目相對於下降

Sampling

……



上述問題容易在reward都是正的或是負的情況都有可能會發生

Value-based Approach

Learning a Critic

Critic

他並不是輸出一個actor，critic只是評估一個actor好或是不好

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

但是actor可以從critic中推出

An actor can be found from a critic.

e.g. Q-learning

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

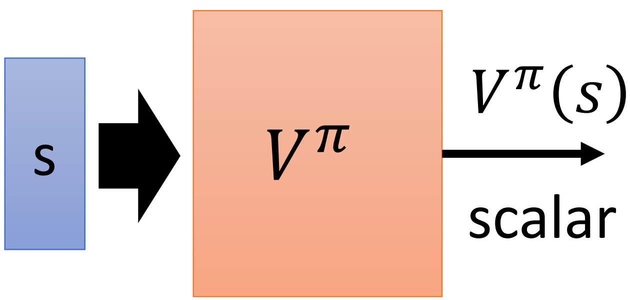
Critic

s：state

output：今天agent看到這個state後output的cumulative reward有多大

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

不同的actor即使在同一個state，接下來得到的reward有可能不一樣（depends on π...actor）



$V^{\pi}(s)$ is large

$V^{\pi}(s)$ is smaller

假設actor pi很會玩遊戲則Vpi在這情形會很大，因為有很多外星人殺

Critic

actor

$$V^{以前的阿光}(大馬步飛) = \text{bad}$$

$$V^{變強的阿光}(大馬步飛) = \text{good}$$

actor

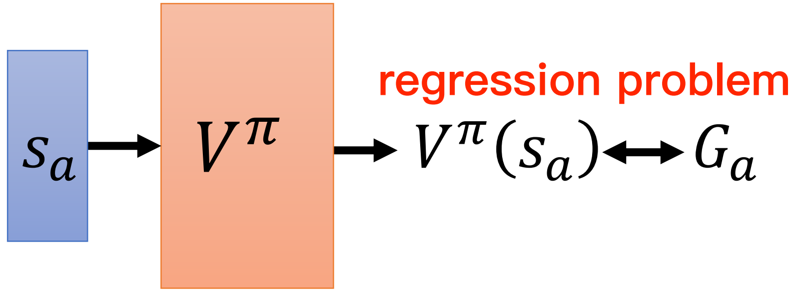
How to estimate $V^{\pi}(s)$

value function

- Monte-Carlo based approach (**MC**)
  - The critic watches π playing the game

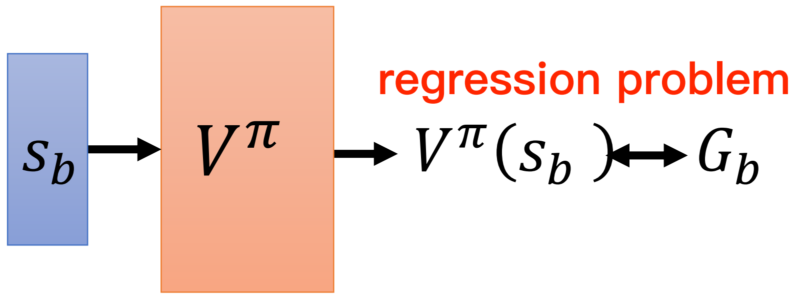
After seeing s_a,

Until the end of the episode, the cumulated reward is G_a



After seeing s_b,

Until the end of the episode, the cumulated reward is G_b



How to estimate $V^{\pi}(s)$

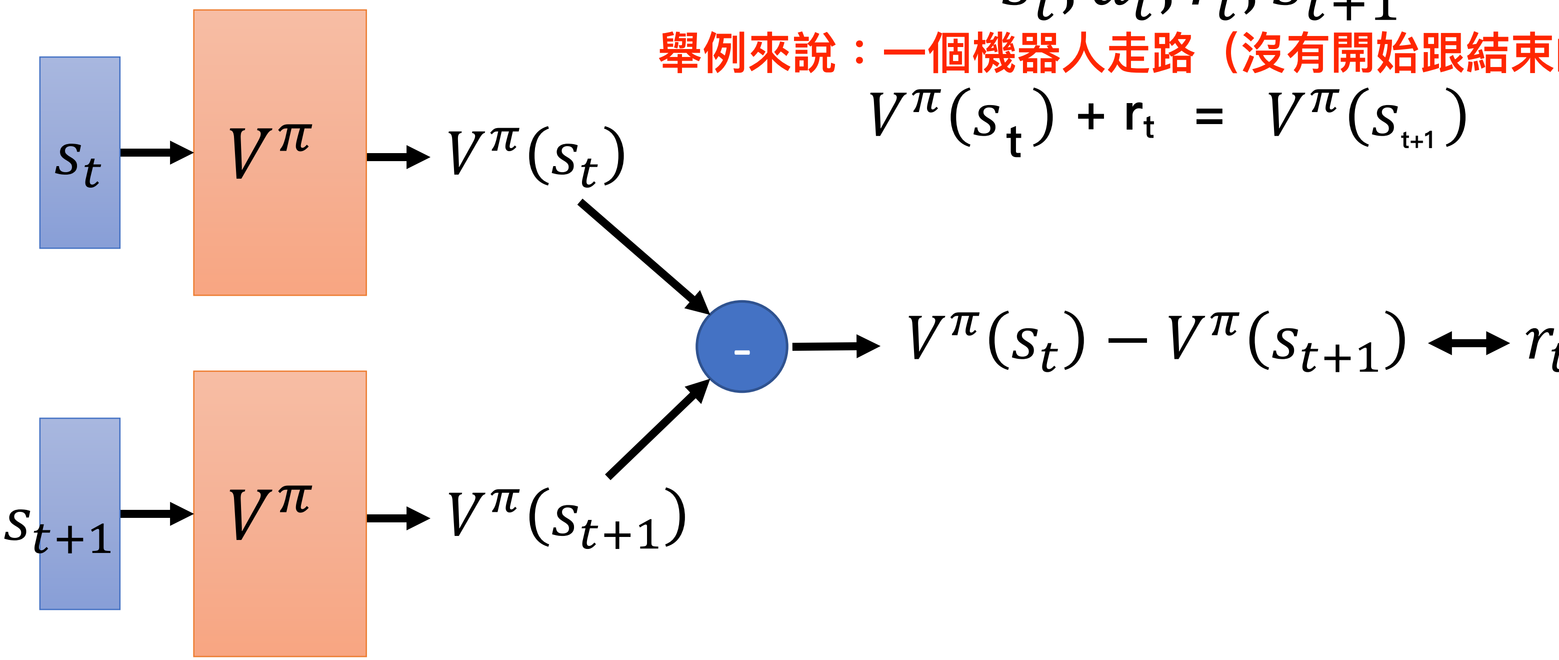
- Temporal-difference approach (TD)

假設只觀察到這些

$$\cdots s_t, a_t, r_t, s_{t+1} \cdots$$

舉例來說：一個機器人走路（沒有開始跟結束的）

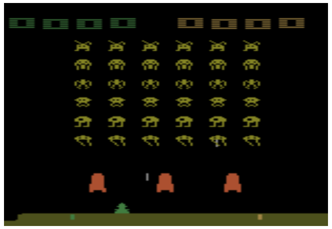
$$V^{\pi}(s_t) + r_t = V^{\pi}(s_{t+1})$$



Some applications have very long episodes, so that delaying all learning until an episode's end is too slow.

蒙地卡羅

MC v.s. TD



…


每個時間點是隨機的因此Ga variance很大

$s_a \rightarrow V^\pi \rightarrow V^\pi(s_a) \leftrightarrow G_a$

Larger variance

unbiased

走很多次Sa每次output都不一樣，最後取平均

unbias但是variance很大（因為每次走都不一樣，累積起來有很大的隨機性）

target

$s_t \rightarrow V^\pi \rightarrow V^\pi(s_t) \longleftrightarrow r + V^\pi(s_{t+1}) \longleftarrow V^\pi \longleftarrow s_{t+1}$

Vπ不一定估測的準確

Smaller variance

某一個state得到的reward

May be biased

MC v.s. TD

[Sutton, v2, Example 6.4]

- The critic has the following 8 episodes
  - $s_a, r = 0, s_b, r = 0$, END
  - $s_b, r = 1$, END
  - $s_b, r = 1$, END
  - $s_b, r = 1$, END
  - $s_b, r = 1$, END
  - $s_b, r = 1$, END
  - $s_b, r = 1$, END
  - $s_b, r = 0$, END

平均得到的reward: 6/8

$$V^\pi(s_b) = 3/4$$

$$V^\pi(s_a) = ?$$ 0? 3/4?

Monte-Carlo: $V^\pi(s_a) = 0$

踩過Sa，得到的reward=0

Temporal-difference:

$V\pi$ (St) $-V\pi$(St+1) <=> r

$$V^\pi(s_b) + r = V^\pi(s_a)$$

3/4 0 3/4

TD: 有可能是sample次數不夠因此才會是0，當sample比較多次後會得到0.75（假設Sb不會因為Sa而改變）

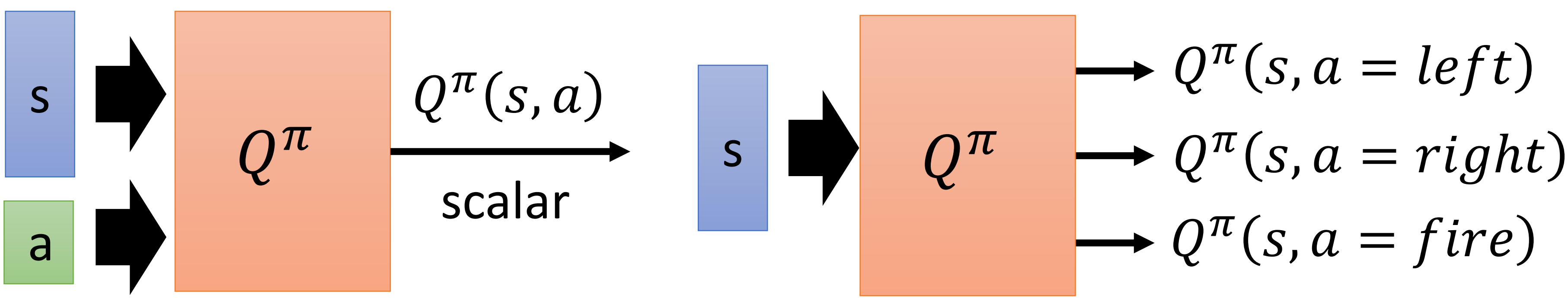
(The actions are ignored here.)

Another Critic

Q–function

- 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

state

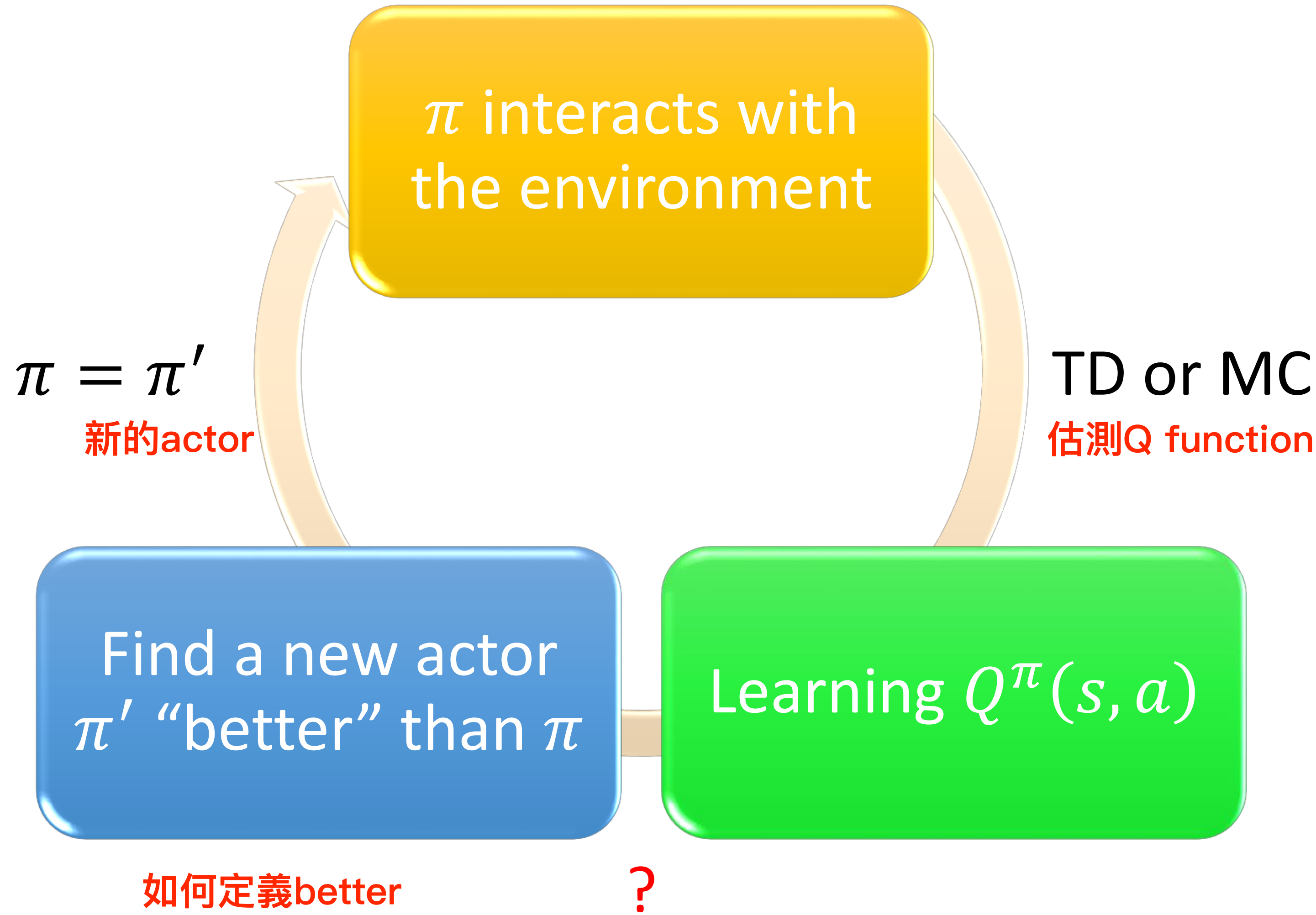


action

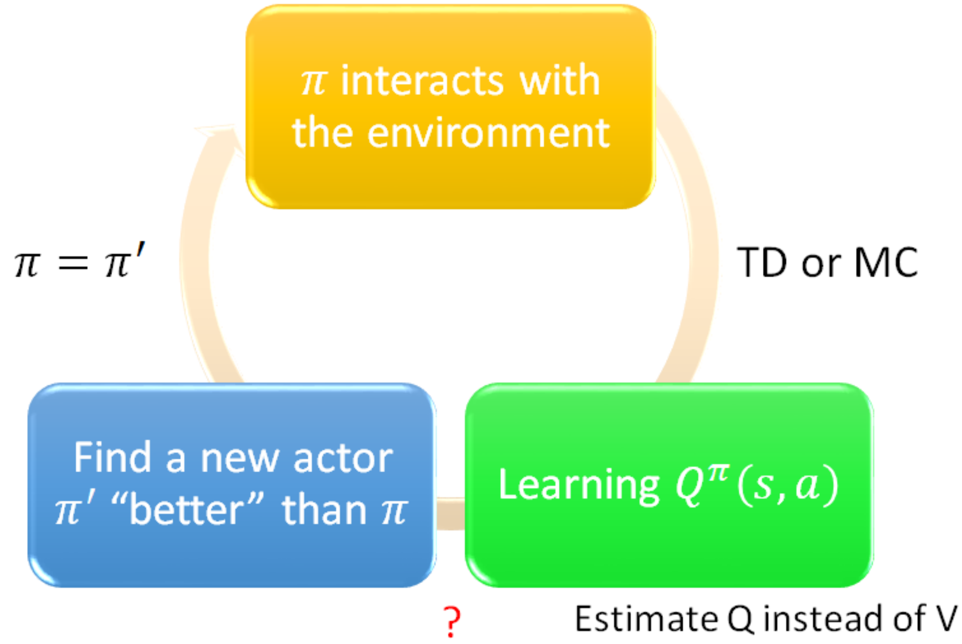
for discrete action only

根據現在的state以及採取的action接下來的得到的cumulative reward是多少

Q-Learning


π interacts with the environment
TD or MC
估測Q function
Learning $Q^{\pi}(s,a)$
?
Find a new actor π' "better" than π
如何定義better
$\pi = \pi'$
新的actor

Q-Learning



假設action可能很少則可以窮舉找出最的直的action，然而當action為continuous則要用gradient descent去找是不實在的，因此Q function只適用於discrete函式

- 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

對所有可能的state而言，pi'總是優於pi

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

找出可以讓Q function最大值的action後即為pi'（depends on Q function）

- ➢ π' does not have extra parameters. It depends on Q
- ➢ Not suitable for continuous action a

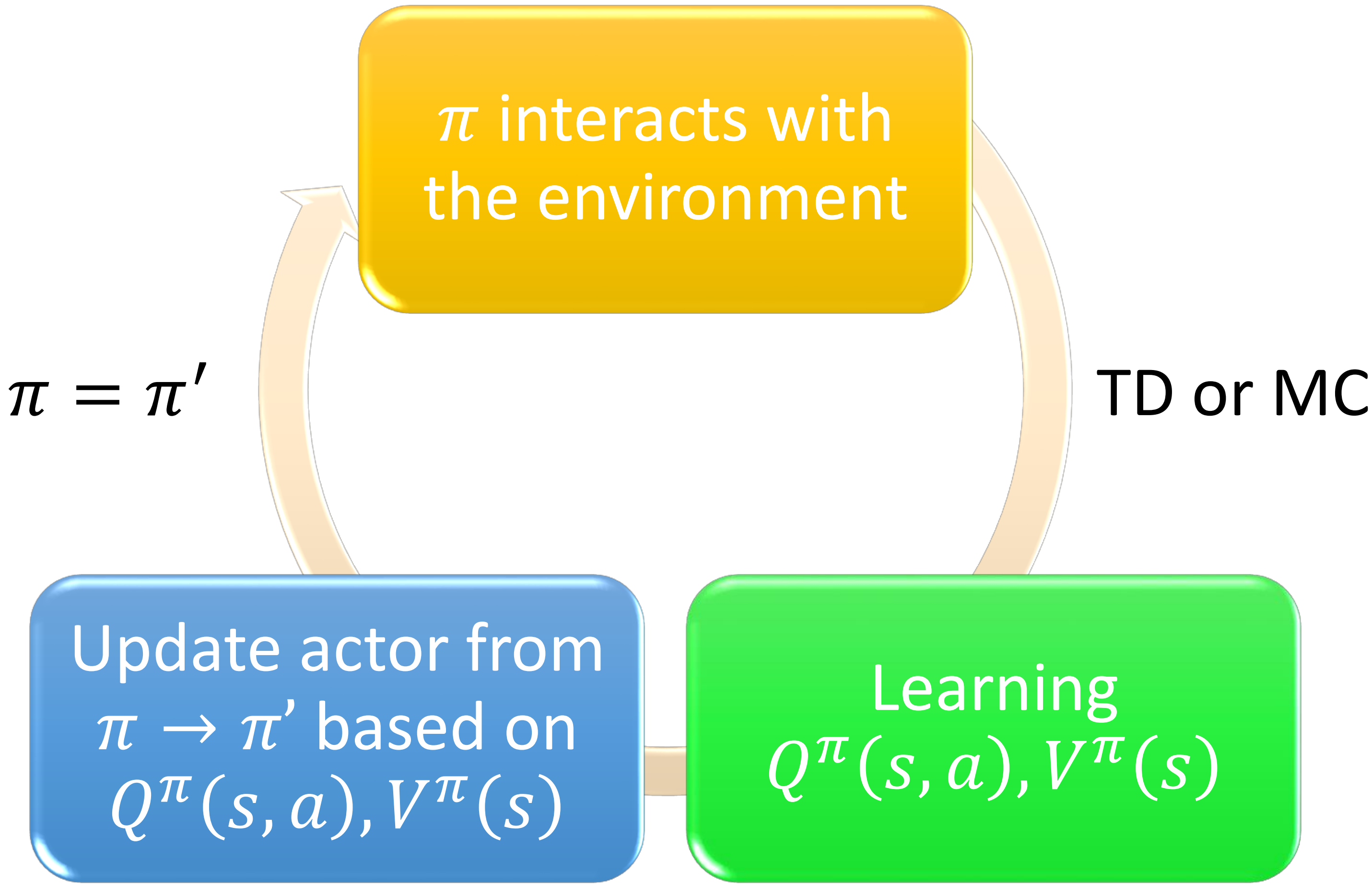
Deep Reinforcement Learning

Actor-Critic

A3C即為其中一種方法

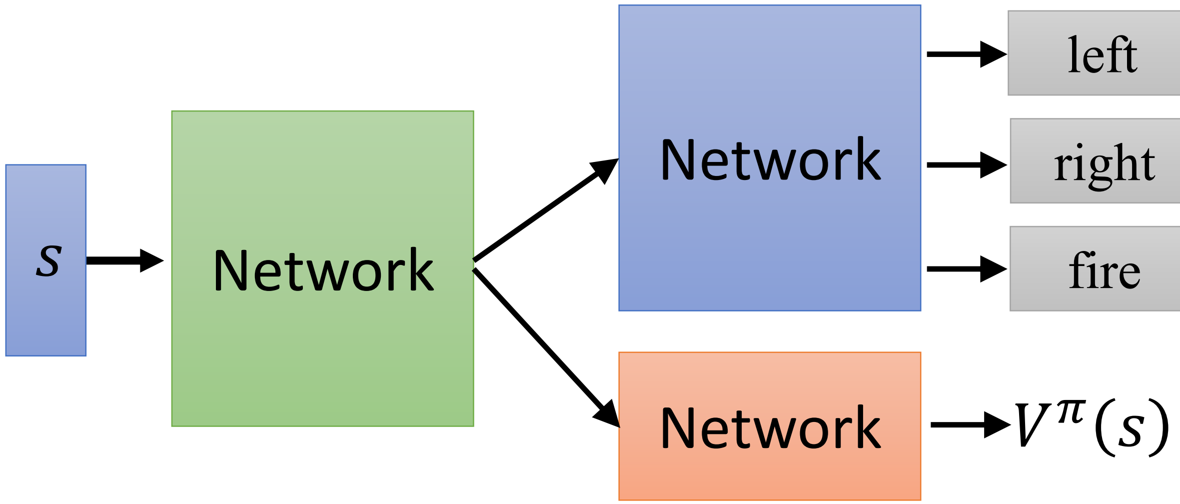
Actor-Critic

Q function中實際上不存在π’，導致我們無法使用在continuous的action上。但actor-critic真的有一個function叫做π，希望π的輸出可以maximize Q function，因此可以解continuous的action



Actor-Critic

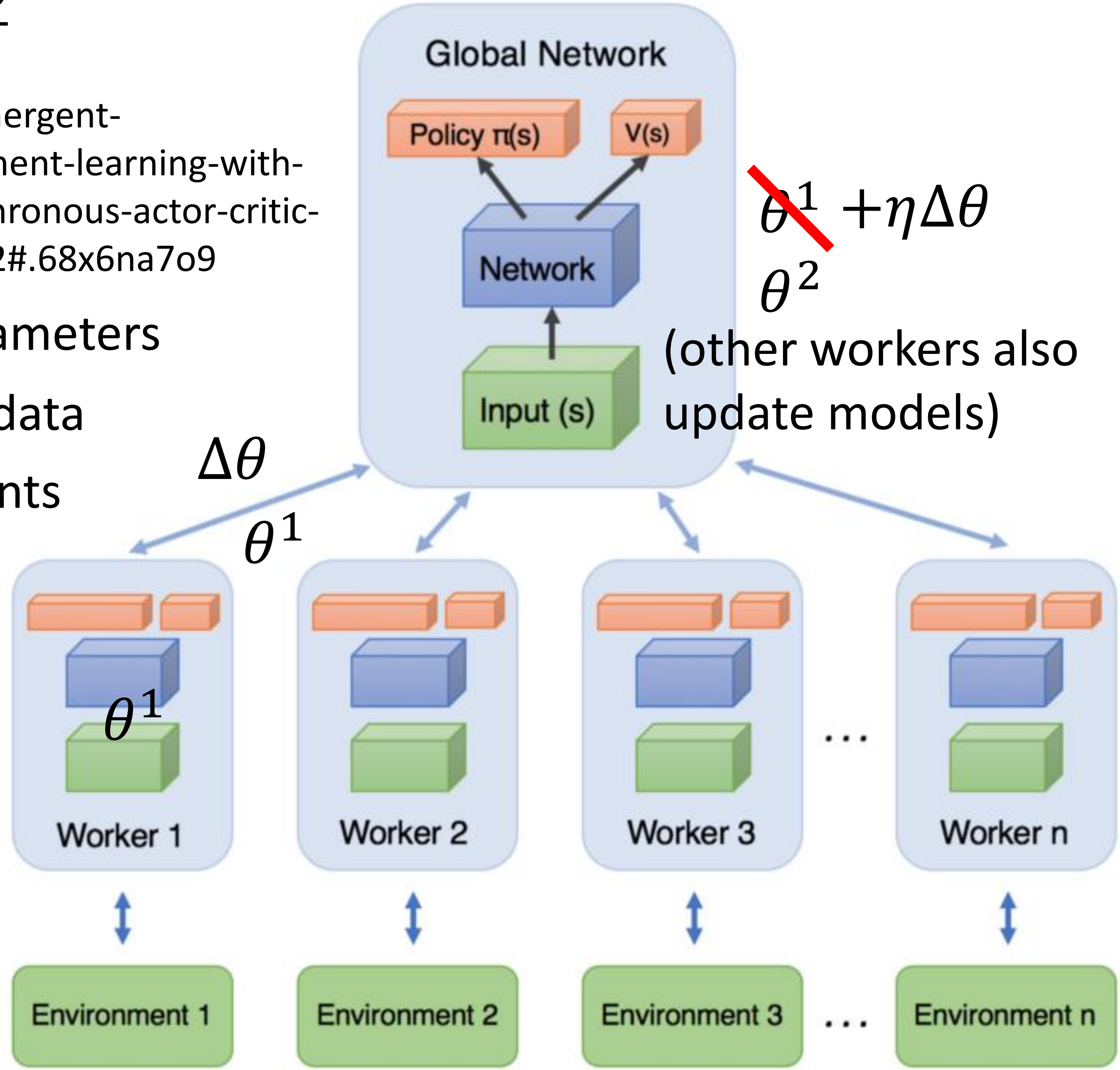
- Tips
  - The parameters of actor $\pi(s)$ and critic $V^\pi(s)$ can be shared



Asynchronous

Source of image: https://medium.com/emergent-future/simple-reinforcement-learning-with-tensorflow-part-8-asynchronous-actor-critic-agents-a3c-c88f72a5e9f2#.68x6na7o9

1. Copy global parameters
2. Sampling some data
3. Compute gradients
4. Update global models



$\cancel{\theta^1} + \eta\Delta\theta$

θ^2

(other workers also update models)

$\Delta\theta$

θ^1

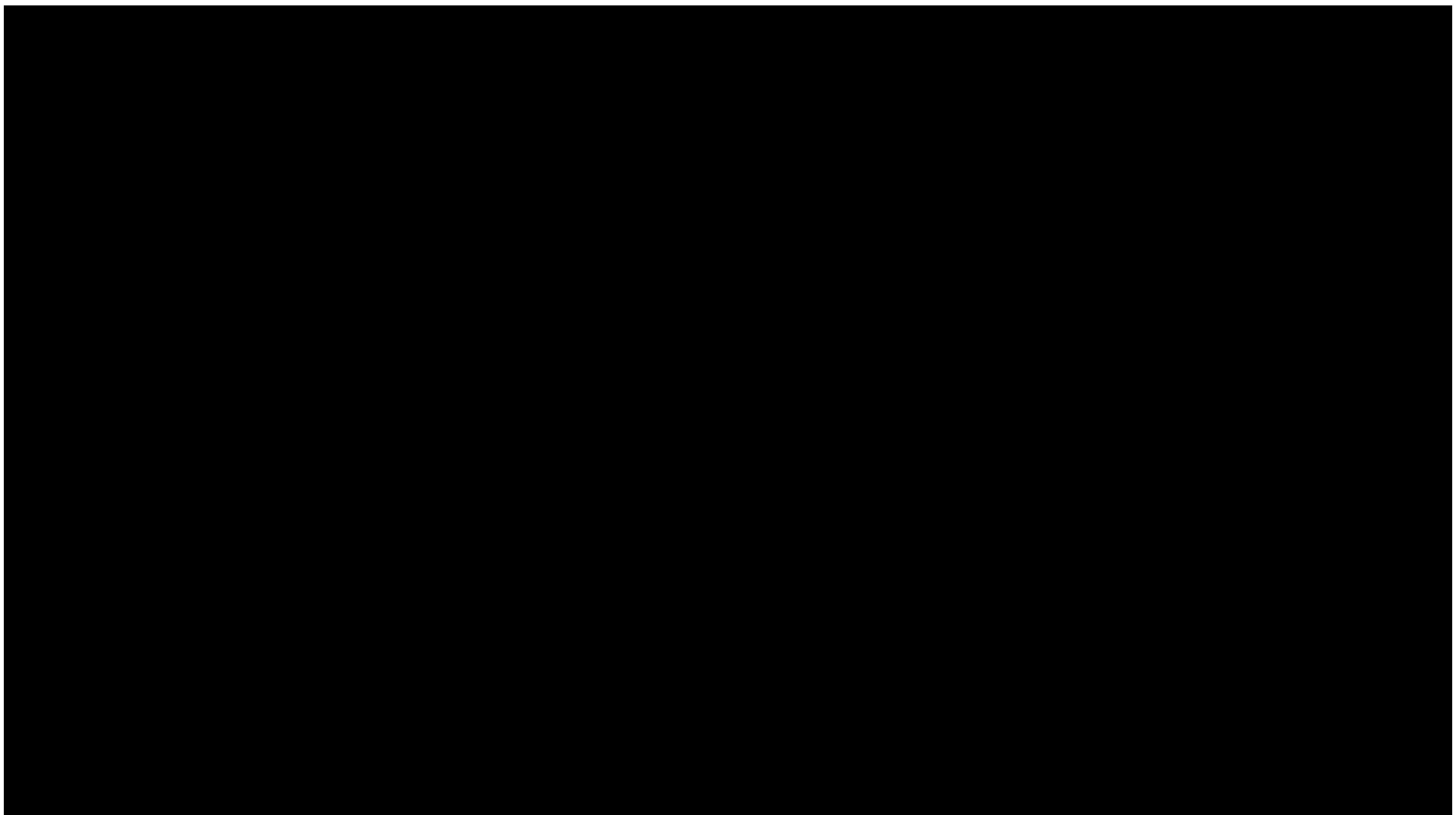
θ^1

$\Delta\theta$

Demo of A3C

https://www.youtube.com/watch?v=0xo1Ldx3L5Q

- Racing Car (DeepMind)



Demo of A3C

- Visual Doom AI Competition @ CIG 2016
- https://www.youtube.com/watch?v=94EPSjQH38Y

Concluding Remarks