

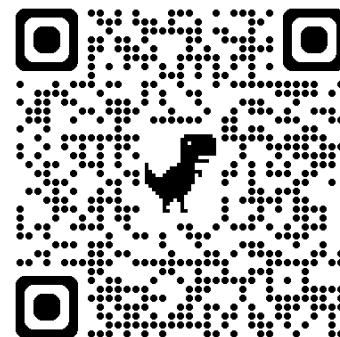


# 深度學習

# Deep Learning

Reinforcement Learning  
強化學習

Instructor: 林英嘉 (Ying-Jia Lin)  
2025/04/28



[Course GitHub](#)



[Slido # DL\\_0428](#)

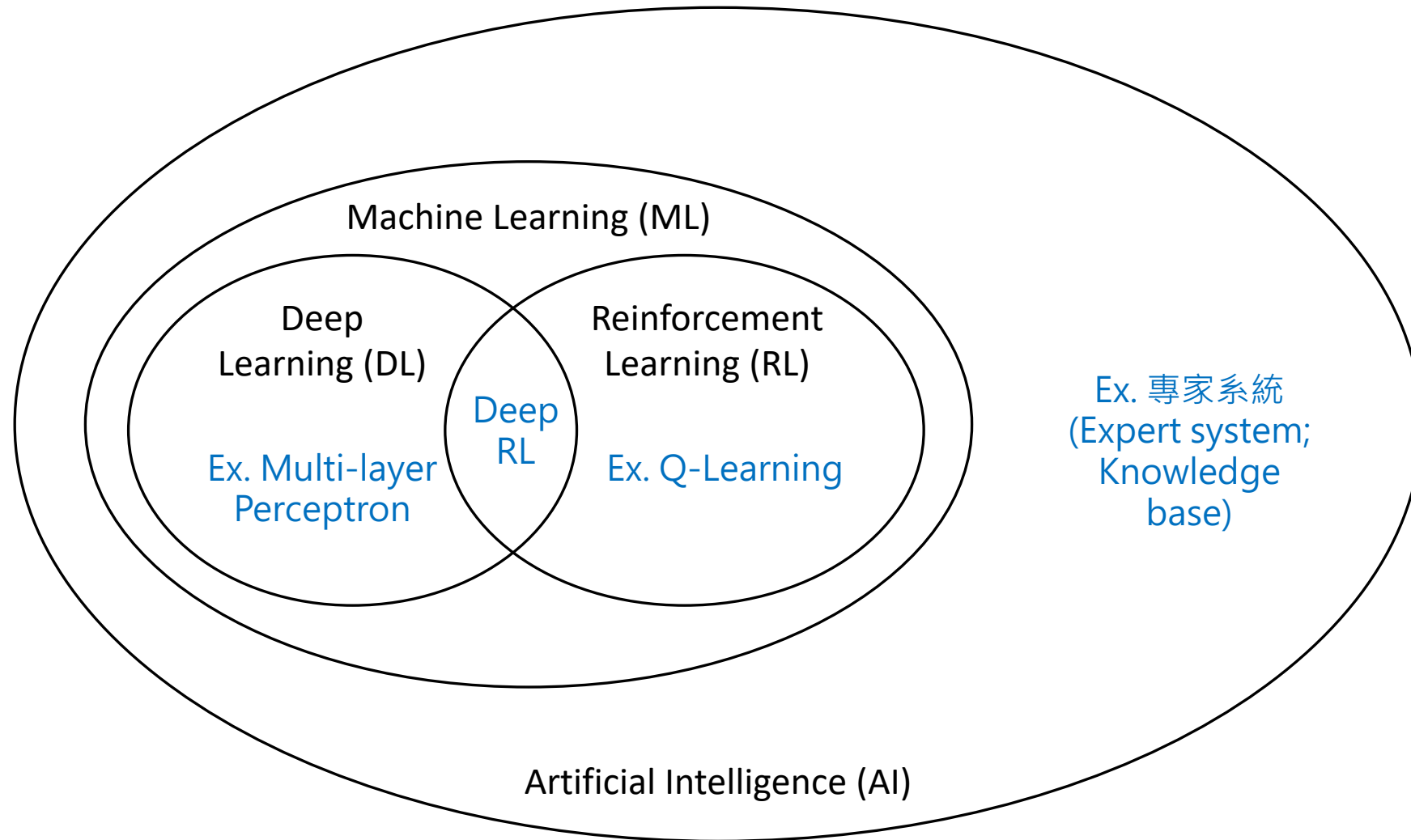
# Outline

---

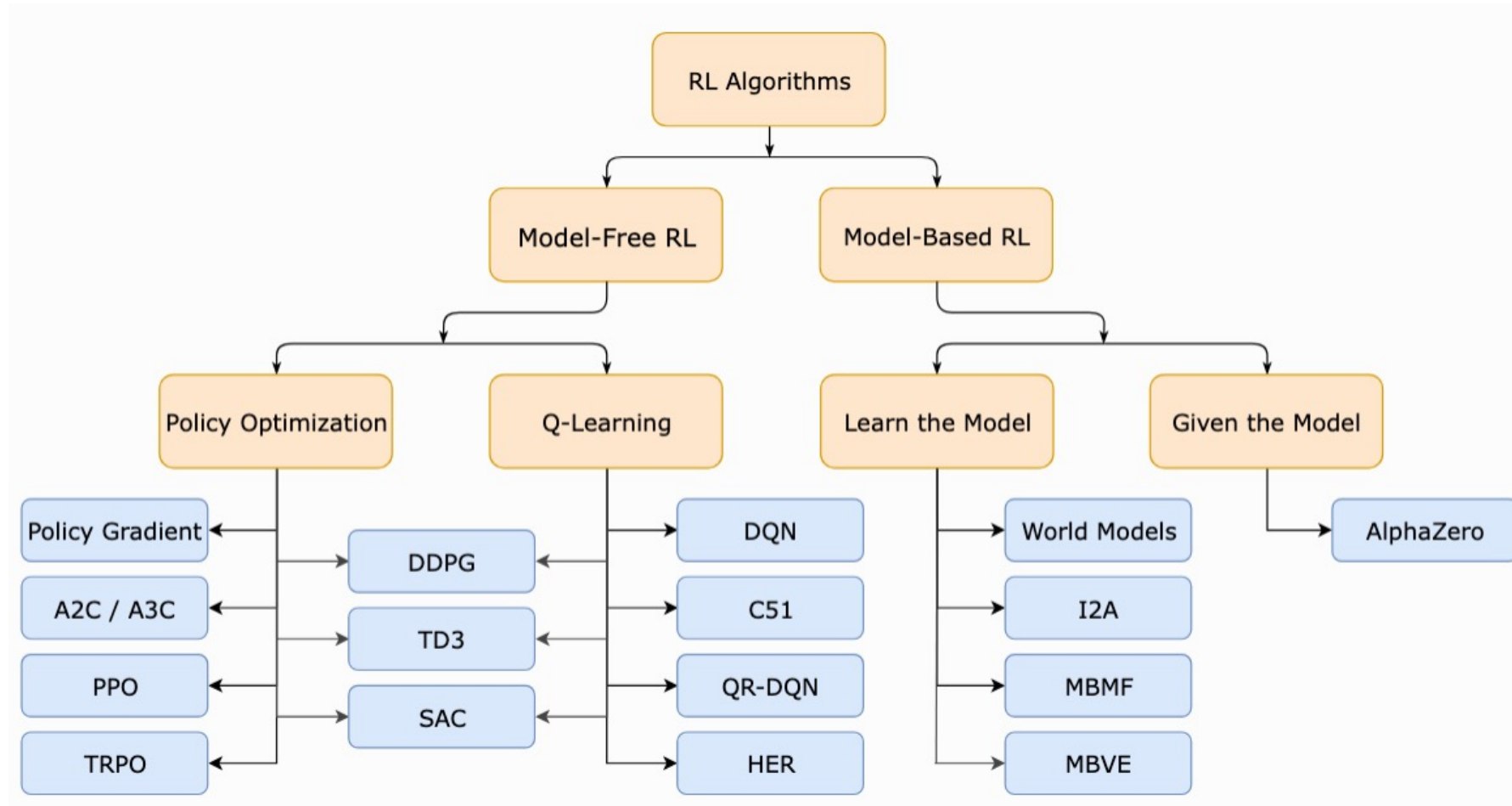
- Reinforcement Learning (RL) [100 min]
- Quiz [20 min]



# Reinforcement Learning (RL) 歸類?



# A Taxonomy of RL Algorithms



# 為什麼我們要學 RL (1): RL is everywhere

---



AlphaGo vs. Lee Sedol (2016)



- Paper: Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., ... & Hassabis, D. (2016). Mastering the game of Go with deep neural networks and tree search. *nature*, 529(7587), 484-489.
- Figure source: <https://www.newyorker.com/tech/annals-of-technology/alphago-lee-sedol-and-the-reassuring-future-of-humans-and-machines>



# 為什麼我們要學 RL (1): RL is everywhere

Ouyang, Long, et al. "Training language models to follow instructions with human feedback." NeurIPS 2022.

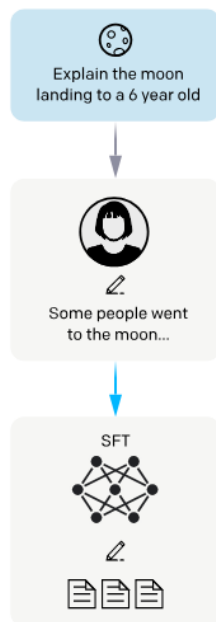
Step 1

**Collect demonstration data, and train a supervised policy.**

A prompt is sampled from our prompt dataset.

A labeler demonstrates the desired output behavior.

This data is used to fine-tune GPT-3 with supervised learning.



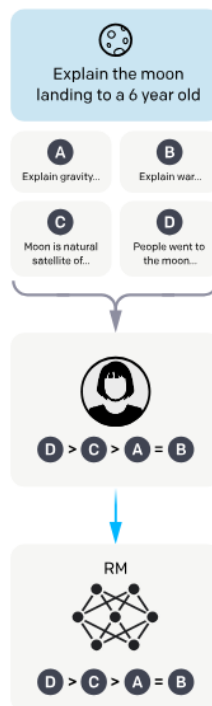
Step 2

**Collect comparison data, and train a reward model.**

A prompt and several model outputs are sampled.

A labeler ranks the outputs from best to worst.

This data is used to train our reward model.



Step 3

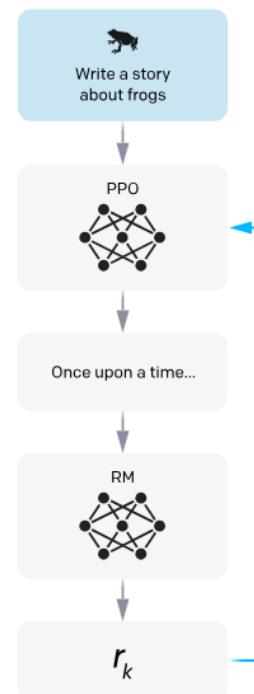
**Optimize a policy against the reward model using reinforcement learning.**

A new prompt is sampled from the dataset.

The policy generates an output.

The reward model calculates a reward for the output.

The reward is used to update the policy using PPO.



# 為什麼我們要學 RL (1): RL is everywhere

Using reinforcement learning.

Code: <https://github.com/jiseongHAN/Super-Mario-RL>



# Atari

---

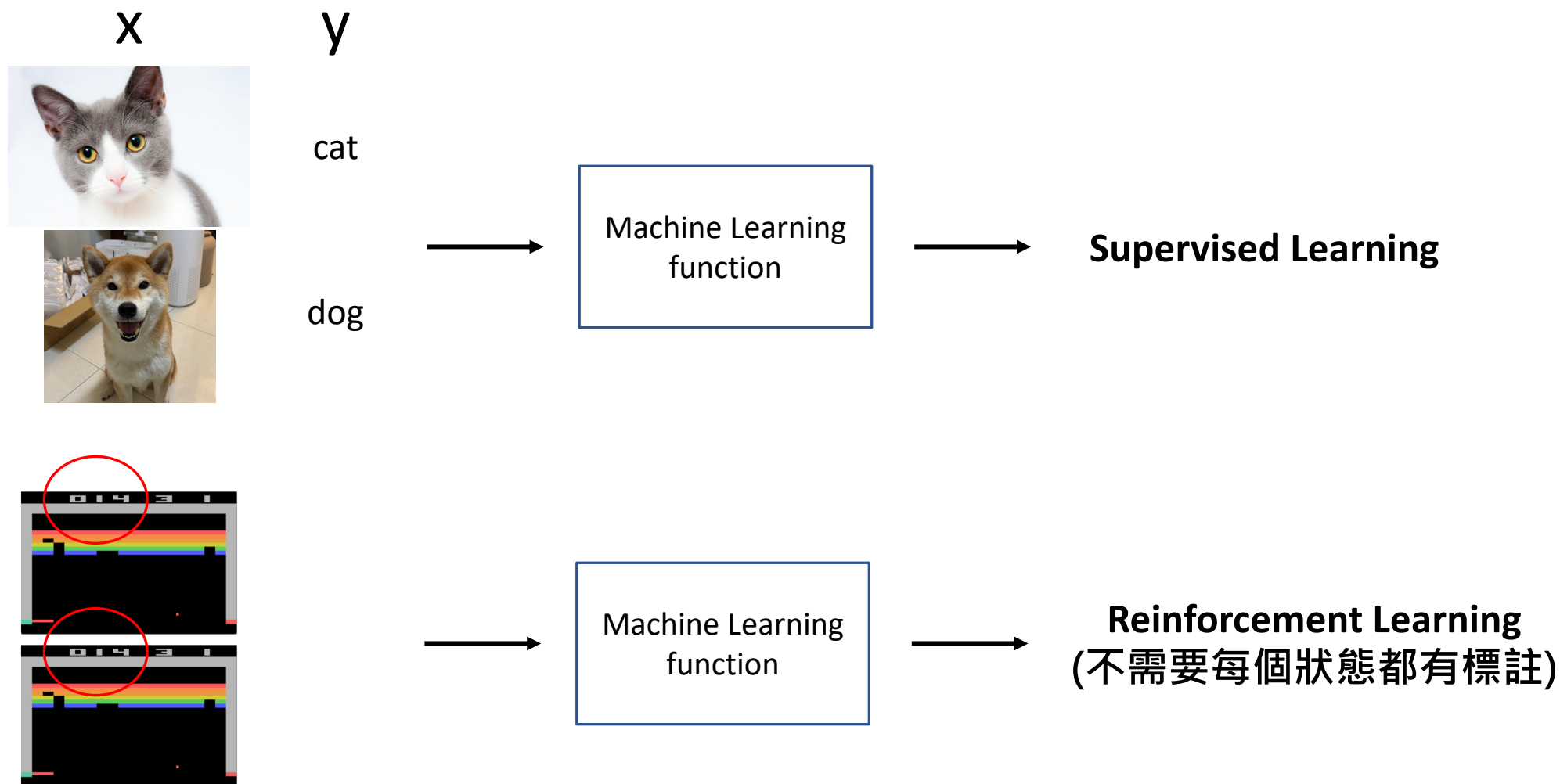


Figure from: Mnih, Volodymyr, et al. "Playing atari with deep reinforcement learning." NIPS (2013).





# 為什麼我們要學 RL (2): 訓練方式的不同



# Supervised Learning vs. Reinforcement Learning

---

- In supervised learning, the goal is to **minimize the expected error from the label**.
- In reinforcement learning, the goal is to maximize **sum of reward**.

Self-supervised learning 也是一種 supervised learning!



(Simplified)  
Path of Deep  
Reinforcement  
Learning

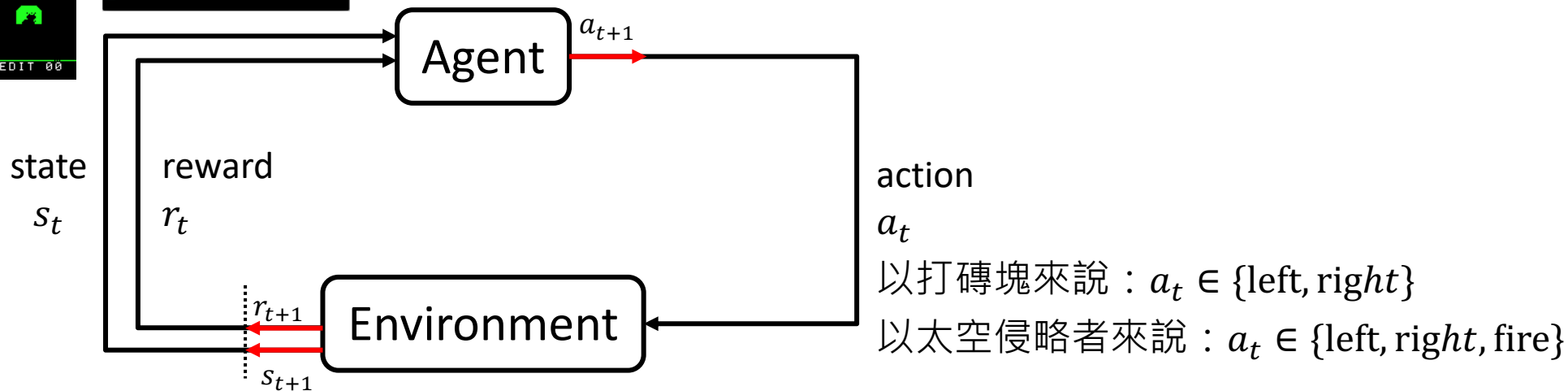
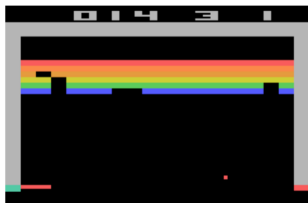
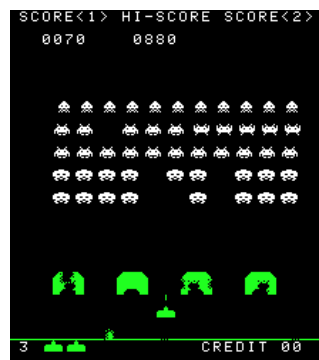
Elements of RL

Policy Gradient Methods

Actor-Critic

On-policy and Off-policy

# [Illustration] RL 的元素



Rennie, Steven J., et al. "Self-critical sequence training for image captioning."  
*Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017.

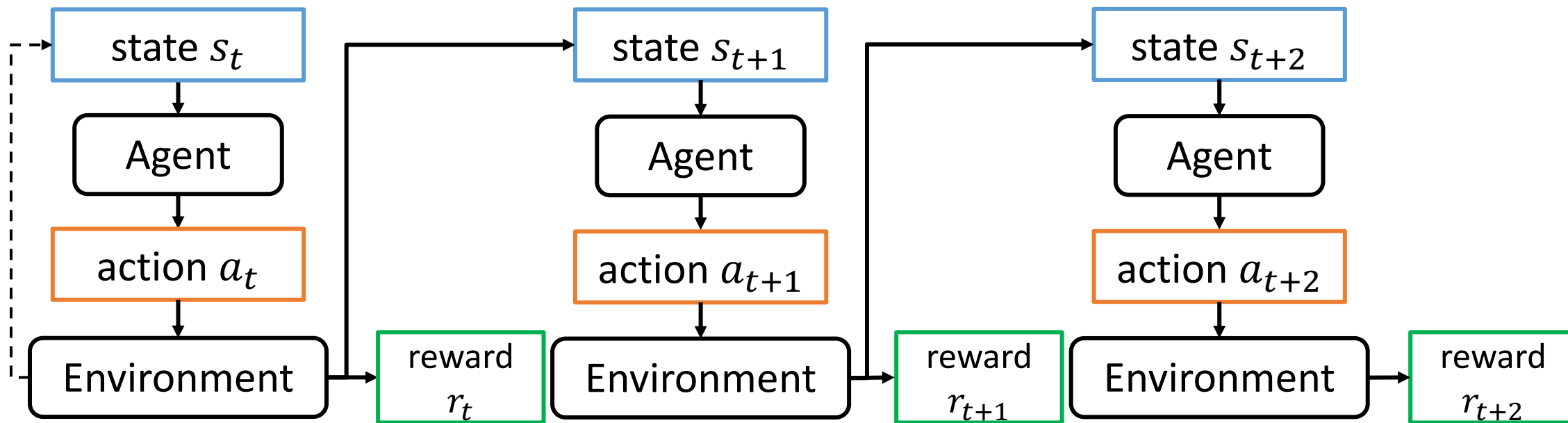


# [Definitions] RL 的元素

符號	名稱	意義
$t$	時間點	狀態或 agent 執行動作的最小單位
$s_t$	狀態 (state)	在 $t$ 時間點，agent 所觀察到的狀態，由環境 (environment) 所提供
$a_t$	動作 (action)	agent 在狀態 $s_t$ 下選擇的動作
$r_t$	即時回饋 (reward)	agent 執行 $a_t$ 後，由環境給的分數



# [Illustration] RL 的元素



# [Definition] Trajectory and Episode

---

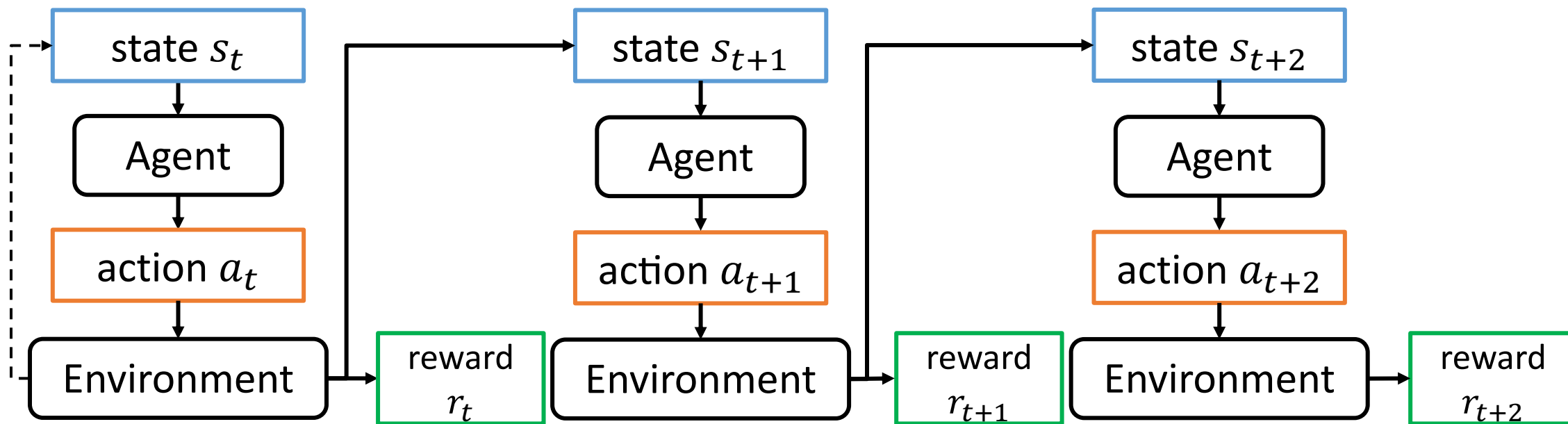
- Trajectory (軌跡, 通常 symbol 為  $\tau$ ) 是指 agent 從環境中互動時，依序經歷的一連串狀態、動作和得到的回饋。

$$\tau = (s_1, a_1, r_1, s_2, a_2, r_2, \dots, s_T, a_T, r_T)$$

- Episode (回合, 通常無固定 symbol) 是指從「開始遊戲」到「遊戲結束」(或者達到某種終止條件，比如死亡、完成任務) 這段完整的互動過程
- 一條 trajectory，也就是一個 episode



# [Illustration] RL 的元素 (假設 $\tau$ 只有3個點)



$$\pi_{\theta}(\tau) = \cancel{\rho(s_1)} \times \pi_{\theta}(a_1|s_1) \times \pi_{\theta}(a_2|s_2) \times \pi_{\theta}(a_3|s_3) = \prod_{t=1}^3 \pi_{\theta}(a_t|s_t)$$





# [Definition] Cumulated reward

對 agent 來說：

輸入

$s_t$

$a_t$

得到

$r_t$

reward  
 $r_1$

reward  
 $r_2$

reward  
 $r_3$

...

reward  
 $r_T$

$$G_1 = r_1 + r_2 + r_3 + \cdots + r_T$$

$$G_2 = r_2 + r_3 + \cdots + r_T$$

$$G_3 = r_3 + \cdots + r_T$$

$$G_T = r_T$$

$G_t$ : cumulated reward  
since the time step  $t$



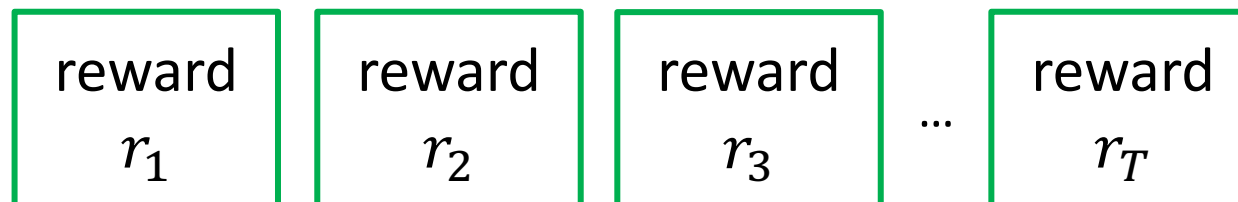
# [Definition] Discounted cumulated reward

對 agent 來說：

輸入  $s_t$   $a_t$

得到  $r_t$

$\gamma$ : discount factor  
(通常  $\gamma < 1$ )



$$G_1 = r_1 + \gamma r_2 + \gamma^2 r_3 + \cdots + \gamma^{T-1} r_T$$

$$G_2 = r_2 + \gamma r_3 + \cdots + \gamma^{T-2} r_T$$

$$G_3 = r_3 + \gamma r_4 + \cdots + \gamma^{T-3} r_T$$

$\vdots$

$$G_T = r_T$$

$G_t$ : cumulated reward  
since the time step  $t$



(Simplified)  
Path of Deep  
Reinforcement  
Learning

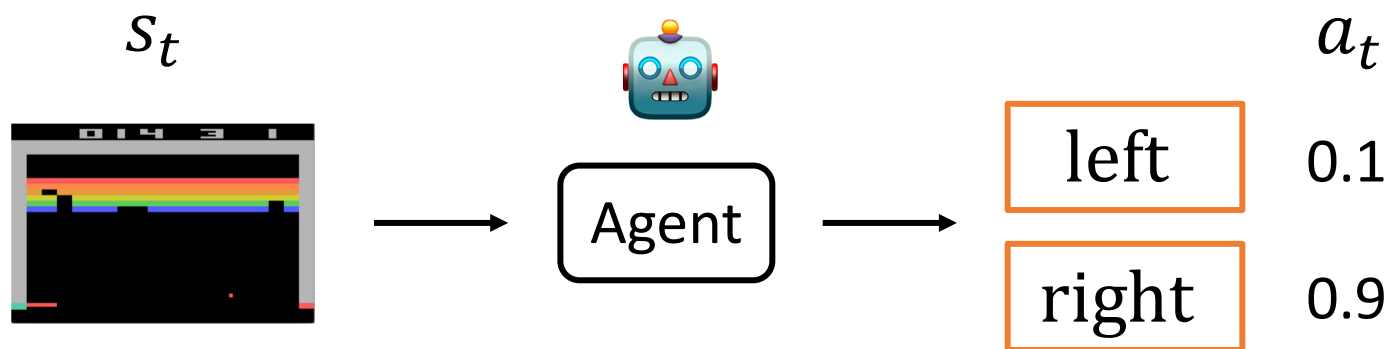
Elements of RL

**Policy Gradient Methods**

Actor-Critic

On-policy and Off-policy

# [Definition] What is policy?



以打磚塊來說： $a_t \in \{\text{left}, \text{right}\}$

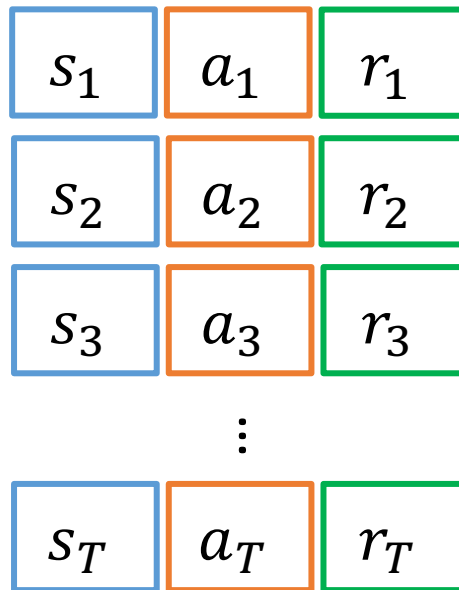
- Policy (策略) 是 agent 根據當前狀態來決定要採取哪個動作的行為準則
  - actor 是其中一種 agents
- Policy 通常以  $\pi$  為符號，在 DRL 中屬於可以被訓練的模型



# 如何訓練一個RL模型？

- Policy gradient

Step1: 收集資料



Step2: 計算 reward 總和  $\sum_{t=1}^T r_t$

Step3: 更新模型

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

累積的 reward 越大越好



# 如何訓練一個RL模型？(3-1)

Log-derivative trick:  
[http://www.math.ncu.edu.tw/~yu/smr/cal100\\_1/boards/lec28\\_sc\\_100.pdf](http://www.math.ncu.edu.tw/~yu/smr/cal100_1/boards/lec28_sc_100.pdf)

- Policy gradient and gradient descent

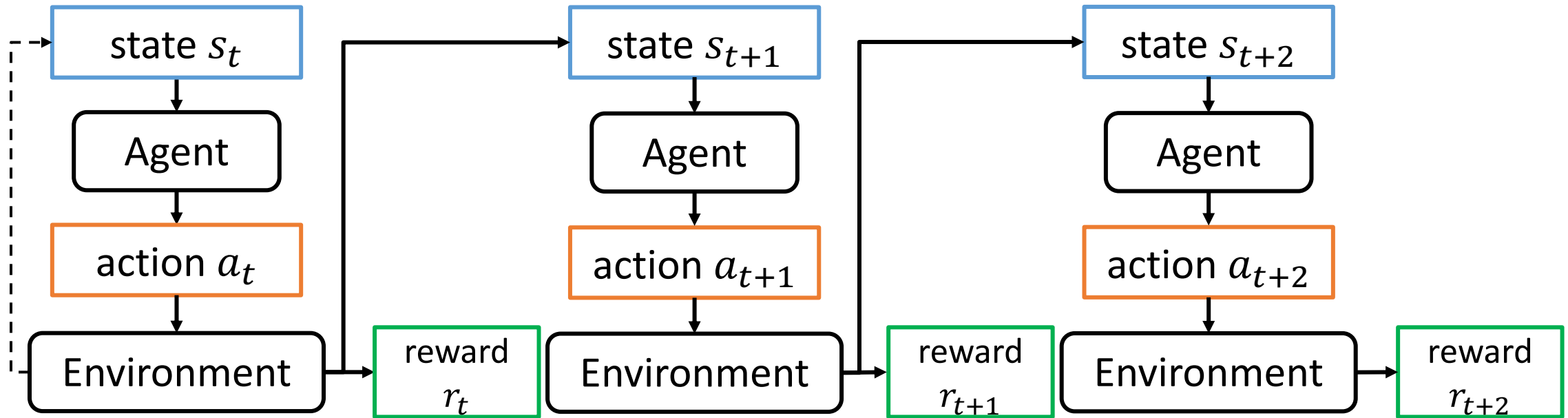
Step3: 更新模型

$$\begin{aligned} L &= \sum_{t=1}^T r_t = R(\tau) \\ \nabla L &= \sum_{\tau} R(\tau) \cdot \nabla \pi_{\theta}(\tau) \\ &= \sum_{\tau} R(\tau) \cdot \pi_{\theta}(\tau) \cdot \nabla \log \pi_{\theta}(\tau) \quad \text{Log-derivative trick} \\ L &= \sum_{\tau} R(\tau) \cdot \pi_{\theta}(\tau) \quad \text{期望值的概念} \\ &= \mathbb{E}_{\tau \sim \pi_{\theta}(\tau)} [R(\tau) \cdot \nabla \log \pi_{\theta}(\tau)] \end{aligned}$$

從  $\pi_{\theta}(\tau)$  機率分佈中採樣 (sampled) 的  $\tau$   
想成有可能你玩很多個回合，現在探討的是這組  $\tau$



# [Illustration] RL 的元素 (假設 $\tau$ 只有3個點)



$$\pi_{\theta}(\tau) = \cancel{\rho(s_1)} \times \pi_{\theta}(a_1|s_1) \times \pi_{\theta}(a_2|s_2) \times \pi_{\theta}(a_3|s_3) = \prod_{t=1}^3 \pi_{\theta}(a_t|s_t)$$



# Log Derivative Trick

---

假設有一function  $p_{\theta}(x)$

$$\frac{\log p'_{\theta}(x)}{p(x)} = \frac{1}{p(x)} p'(x) \quad \Rightarrow \quad \frac{p'(x)}{p(x)} = \log p'_{\theta}(x)$$

function取log的一次微分 (梯度)  
對照上一頁：  $\nabla \log \pi_{\theta}(\tau)$

function的一次微分 (梯度)  
對照上一頁：  $\nabla \pi_{\theta}(\tau)$





# 如何訓練一個RL模型？(3-2)

- Policy gradient and gradient descent

Step3: 更新模型

$$\pi_{\theta}(\tau) = \prod_{t=1}^T \pi_{\theta}(a_t | s_t) \quad \text{帶入動作和狀態}$$

$$\begin{aligned} \log \pi_{\theta}(\tau) &= \log \prod_{t=1}^T \pi_{\theta}(a_t | s_t) \\ &= \sum_{t=1}^T \log \pi_{\theta}(a_t | s_t) \end{aligned}$$

$$\nabla L = \mathbb{E}_{\tau \sim \pi_{\theta}(\tau)} [R(\tau) \cdot \nabla \log \pi_{\theta}(\tau)]$$

$$= \mathbb{E}_{\tau \sim \pi_{\theta}(\tau)} \left[ R(\tau) \cdot \nabla \sum_{t=1}^T \log \pi_{\theta}(a_t | s_t) \right]$$

$$= \mathbb{E}_{\tau \sim \pi_{\theta}(\tau)} \left[ \sum_{t=1}^T R(\tau) \cdot \nabla \log \pi_{\theta}(a_t | s_t) \right]$$



# Why Log Derivative Trick?

還沒有使用 **log derivative trick** 前:

$$\nabla L = \sum_{\tau} R(\tau) \cdot \nabla \pi_{\theta}(\tau)$$

$$\pi_{\theta}(\tau) = \rho(s_1) \times \pi_{\theta}(a_1|s_1) \times \pi_{\theta}(a_2|s_2) \times \cdots \times \pi_{\theta}(a_T|s_T) = \prod_{t=1}^T \pi_{\theta}(a_t|s_t)$$

$$\log \pi_{\theta}(\tau) = \log[\pi_{\theta}(a_1|s_1) \times \pi_{\theta}(a_2|s_2) \times \cdots \times \pi_{\theta}(a_T|s_T)] = \sum_{t=1}^T \log \pi_{\theta}(a_t|s_t)$$

**相加** -> easier



# 如何訓練一個RL模型？(3-3)

- Policy gradient and **gradient descent**

Step3: 更新模型

$$\nabla L = \mathbb{E}_{\tau \sim \pi_{\theta}(\tau)} \left[ \sum_{t=1}^T R(\tau) \cdot \nabla \log \pi_{\theta}(a_t | s_t) \right]$$

更新w：

$$w_N = w_{N-1} - \eta \frac{\partial L}{\partial w_{N-1}} \overset{-\nabla L}{\boxed{\quad \quad \quad}}$$

更新b：

$$b_N = b_{N-1} - \eta \frac{\partial L}{\partial b_{N-1}} \overset{-\nabla L}{\boxed{\quad \quad \quad}}$$

- $\eta$  代表 learning rate



# 把 $G_t$ (Cumulated reward) 考慮進去

$$\nabla L = \mathbb{E}_{\tau \sim \pi_{\theta}(\tau)} \left[ \sum_{t=1}^T \frac{R(\tau)}{\gamma^{t-1}} \cdot \nabla \log \pi_{\theta}(a_t | s_t) \right]$$

Replace total reward  $R(\tau)$  with timestep-specific cumulative reward  $G_t$

$$G_t = \sum_{t'=t}^T \gamma^{t'-t} r_{t'}$$
$$\begin{aligned} G_1 &= r_1 + \gamma r_2 + \gamma^2 r_3 + \cdots + \gamma^{T-1} r_T \\ G_2 &= r_2 + \gamma r_3 + \cdots + \gamma^{T-2} r_T \\ G_3 &= r_3 + \gamma r_4 + \cdots + \gamma^{T-3} r_T \\ &\vdots \\ G_T &= r_T \end{aligned}$$

$$\nabla L = \mathbb{E}_{\tau \sim \pi_{\theta}(\tau)} \left[ \sum_{t=1}^T G_t \cdot \nabla \log \pi_{\theta}(a_t | s_t) \right]$$



# [Important Notes] Training 細節

- 通常一個 episode 更新一次 agent
- 每次 agent 更新後，資料要重新收集

Step1: 收集資料



$s_1$	$a_1$	$r_1$
$s_2$	$a_2$	$r_2$
$s_3$	$a_3$	$r_3$
$\vdots$		
$s_T$	$a_T$	$r_T$



(Simplified)  
Path of Deep  
Reinforcement  
Learning

Elements of RL

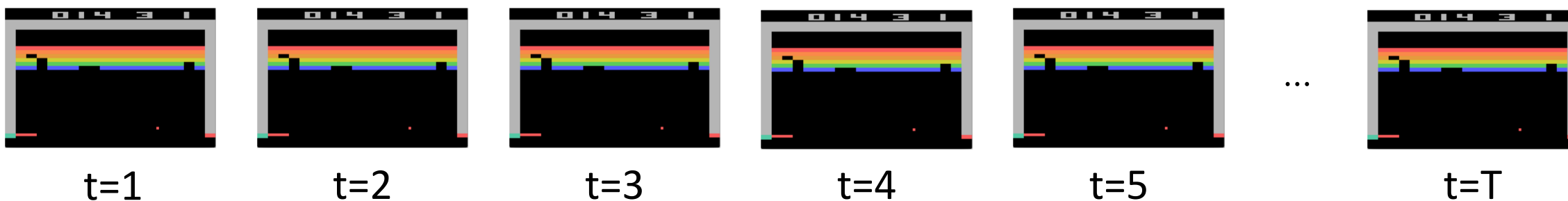
Policy Gradient Methods

**Actor-Critic**

On-policy and Off-policy

# General Problem (1)

通常一個回合你會玩很久 ... (?)



$a_1$

每個 $t$ 時間點的動作都會影響未來的 $s$ 跟 $a$ ，導致每個回合的 $G_t$ 都有可能出現很大差異

$a_1$ 做什麼？	左	右	不動
$G_t$	10	100	0



# General Problem (2)

---

$$\nabla L = \mathbb{E}_{\tau \sim \pi_{\theta}(\tau)} [\underline{R(\tau)} \cdot \nabla \log \pi_{\theta}(\tau)]$$



代表玩完整場遊戲才能知道 reward 高不高 (過程中的 actions 到底好不好)

有沒有可能每個  $t$  時間點都能算出一個 reward 分數？





# [Definition] Critic

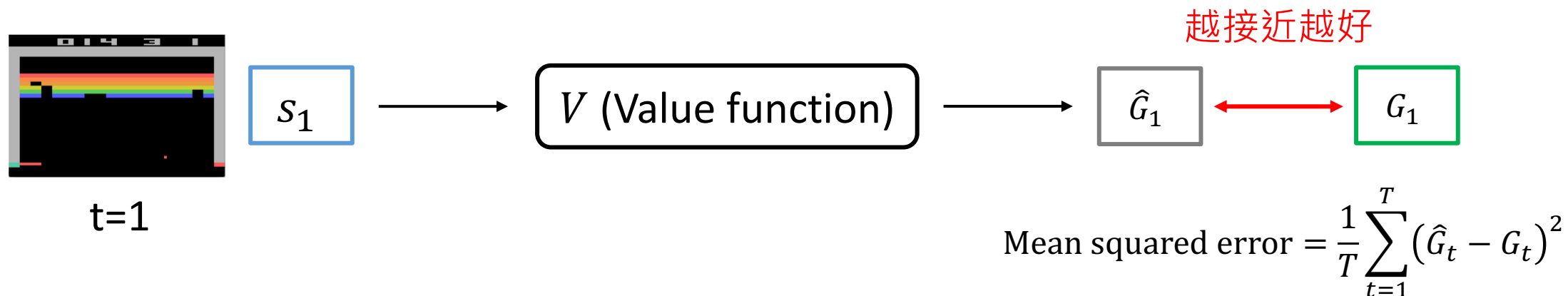
---

- Critic: 用來評價 state (或 state, action) 的函數，可以是個模型
  - 用來評價 state 的函數：Value Function (通常 symbol 為  $V$ )
  - 用來評價 state, action 的函數：State-action Value Function (通常 symbol 為  $Q$ )
- Critic 就是 value function，功能為未卜先知，預測「未來總 reward」



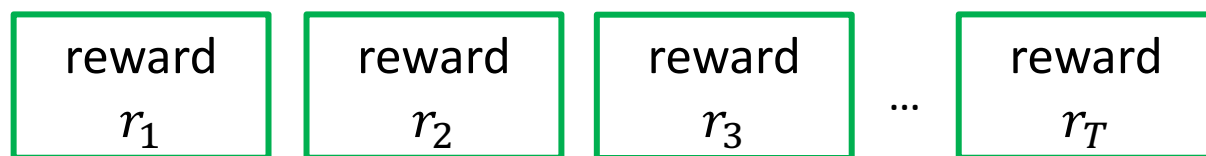
# 訓練 $V$ 的方法 (1): Monte Carlo

- 直接 train 下去



# 訓練 $V$ 的方法 (2): Temporal Difference (TD)

- [Recap] Discounted cumulated reward



$$G_1 = r_1 + \gamma r_2 + \gamma^2 r_3 + \cdots + \gamma^{T-1} r_T$$

$$G_2 = r_2 + \gamma r_3 + \cdots + \gamma^{T-2} r_T$$

$$G_3 = r_3 + \gamma r_4 + \cdots + \gamma^{T-3} r_T$$

$\vdots$

$$G_T = r_T$$

$$G_1 = \gamma G_2 + r_1$$

$$G_2 = \gamma G_3 + r_2$$



$$G_t = \gamma G_{t+1} + r_t$$

$$G_t - \gamma G_{t+1} = r_t$$

任兩個時間點的  $G$  的關係

$G_t$ : cumulated reward  
since the time step  $t$

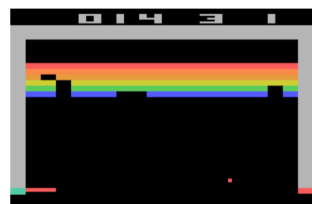


# 訓練 $V$ 的方法 (2): Temporal Difference (TD)

任兩個時間點的  $G$  的關係：

$$G_t - \gamma G_{t+1} = r_t$$

➡ 以相鄰的兩個 steps 的差異 train 下去



time step: t

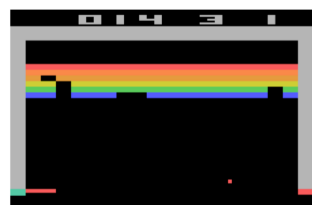
$s_t$

$V$  (Value function)

$\hat{G}_t$

$$\hat{G}_t = V(s_t)$$

模型訓練目標： $V(s_t) - \gamma V(s_{t+1})$  越接近  $r_t$  越好



time step: t+1

$s_{t+1}$

$V$  (Value function)

$\hat{G}_{t+1}$

$$\hat{G}_{t+1} = V(s_{t+1})$$



# $V$ 如何幫助 actor 訓練？

$$\nabla L = \mathbb{E}_{\tau \sim \pi_{\theta}(\tau)} \left[ \sum_{t=1}^T G_t \cdot \nabla \log \pi_{\theta}(a_t | s_t) \right]$$

Advantage function (優勢函數)

$$A_t = G_t - V(s_t)$$

- Monte Carlo

$$\nabla L = \mathbb{E}_{\tau \sim \pi_{\theta}(\tau)} \left[ \sum_{t=1}^T (G_t - \underbrace{V(s_t)}) \cdot \nabla \log \pi_{\theta}(a_t | s_t) \right]$$

$V(s_t)$  學到預估的累積 reward

只要 actor 試著將  $G_t - V(s_t)$  變大，就能產生更好的 action



# V 如何幫助 actor 訓練？

$$\nabla L = \mathbb{E}_{\tau \sim \pi_{\theta}(\tau)} \left[ \sum_{t=1}^T G_t \cdot \nabla \log \pi_{\theta}(a_t | s_t) \right]$$

Advantage function (優勢函數)

$$A_t = G_t - V(s_t)$$

- Temporal Difference (TD)

訓練V時：  $V(s_t) - \gamma V(s_{t+1})$  越接近  $r_t$  越好

訓練 actor 時：  $r_t - V(s_t) + \gamma V(s_{t+1})$  越大越好，因為  $V(s_t) - \gamma V(s_{t+1})$  只是預估值

$$\nabla L = \mathbb{E}_{\tau \sim \pi_{\theta}(\tau)} \left[ \sum_{t=1}^T \underbrace{(r_t + \gamma V(s_{t+1}) - V(s_t))}_{\text{TD}} \cdot \nabla \log \pi_{\theta}(a_t | s_t) \right]$$

TD 是看即時 reward



(Simplified)  
Path of Deep  
Reinforcement  
Learning

Elements of RL

Policy Gradient Methods

Actor-Critic

**On-policy and Off-policy**

# [Recap, Important Notes] Training 細節

- 通常一個 episode 更新一次 **actor**
- 每次 agent 更新後，資料要重新收集

Question: 舊的資料是否能沿用呢？

Step1: 收集資料



$s_1$	$a_1$	$r_1$
$s_2$	$a_2$	$r_2$
$s_3$	$a_3$	$r_3$
$\vdots$		
$s_T$	$a_T$	$r_T$





# [Definition] On-policy and Off-policy

---

- On-policy：被訓練的 agent 跟資料是來自同一個 policy
  - 訓練資料來自目前的 policy (現在訓練  $\pi_{\theta}$ ，資料來自  $\pi_{\theta}$ )
- Off-policy：被訓練的 agent 跟資料不是來自同一個 policy
  - 訓練資料來自不同的 policies (例如：現在訓練  $\pi_{\theta_{\text{new}}}$  但資料來自  $\pi_{\theta_{\text{old}}}$ )



# On-policy and Off-policy

Hung-yi Lee: <https://youtu.be/OAKAZhFmYol?si=zmFUM46UJ4ONKchQ&t=90>

On-policy: 阿光下棋

Off-policy: 佐為下棋、阿光在旁邊看



On-policy: 自己打game



Off-policy: 看實況主打game，自己偷學

<https://www.youtube.com/watch?v=wmabM0dFGWU>



# Off-policy 的問題與 PPO

TRPO: Schulman, John, et al. "Trust region policy optimization." *International conference on machine learning*. 2015.  
PPO: Schulman, John, et al. "Proximal policy optimization algorithms." arXiv preprint arXiv:1707.06347 (2017).

- 資料 (來自  $\pi_{\theta_{\text{old}}}$ ) 是舊的，但模型 ( $\pi_{\theta_{\text{new}}}$ ) 是新的
  - $\pi_{\theta_{\text{new}}}$  可能已經變強了，可能會被舊的資料誤導
- 有些 paper (如 TRPO) 會採用 importance sampling 的方法來限制  $\pi_{\theta_{\text{new}}}(a_t|s_t)$  的變化

$$\frac{\pi_{\theta_{\text{new}}}(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$$

- 但在  $\pi_{\theta_{\text{new}}}(a_t|s_t)$  和  $\pi_{\theta_{\text{old}}}(a_t|s_t)$  差異很大的情況下，訓練不穩定
  - > 近代作法 PPO (Proximal Policy Optimization) 用範圍更加限制 importance sampling 的變化：

$$1 - \epsilon \leq \frac{\pi_{\theta_{\text{new}}}(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} \leq 1 + \epsilon$$

- $\epsilon$  為超參數，PPO 論文中設為 0.2



# Summary

---

- Policy gradient: 讓 policy 透過 maximize reward 方向更新
- Value function: 預測一個 state 的「未來總 reward」
- On-policy vs. Off-policy: 模型「自己收資料」或「看舊資料學習」的差別
- PPO: 透過限制 policy 來小步更新幅度，慢慢變好



# 學習資源

---

- Sutton, R. S., & Barto, A. G. (2018). *Reinforcement Learning: An Introduction* (2nd ed.), Chapter 5: Monte Carlo Methods.
  - <http://incompleteideas.net/book/the-book-2nd.html>
- 李宏毅老師 Deep Reinforcement Learning, 2018 播放清單
  - [https://youtube.com/playlist?list=PLJV\\_el3uVTsODxQFgzMzPLa16h6B8kWM\\_&si=qN41KpUMwMBVipiA](https://youtube.com/playlist?list=PLJV_el3uVTsODxQFgzMzPLa16h6B8kWM_&si=qN41KpUMwMBVipiA)
- OpenAI Spinning Up in Deep RL
  - <https://spinningup.openai.com/en/latest/algorithms/ppo.html>



# Thank you!

Instructor: 林英嘉

 yjlin@cgu.edu.tw

TA: 林君襄

 becky890926@gmail.com