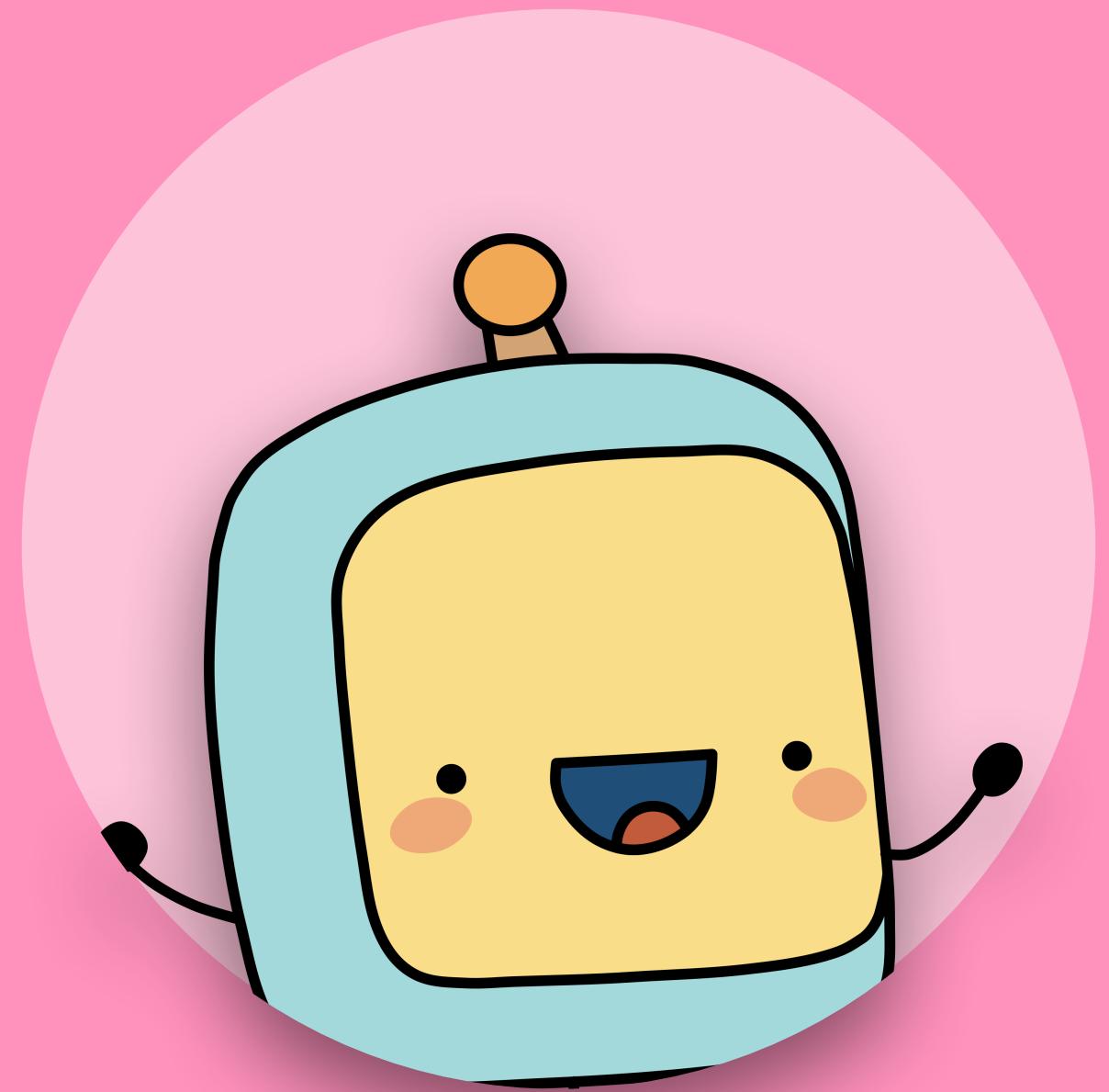


13.  
強化學習和 LLM  
的「對齊」

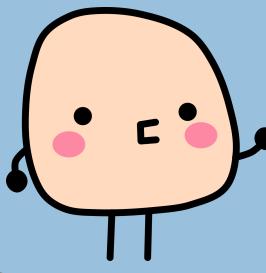


蔡炎龍  
政治大學應用數學系



01.

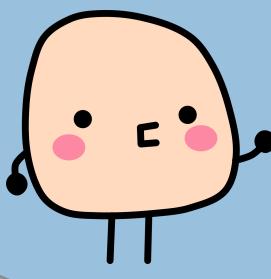
# 強化學習基礎



# 強化學習 Reinforcement Learning

## 打造 AlphaGo 的神奇魔法





## RL 的重要實例: AI 玩電動

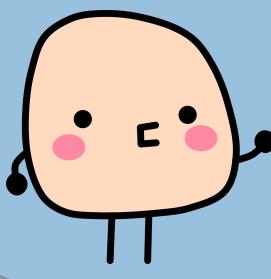


DeepMind

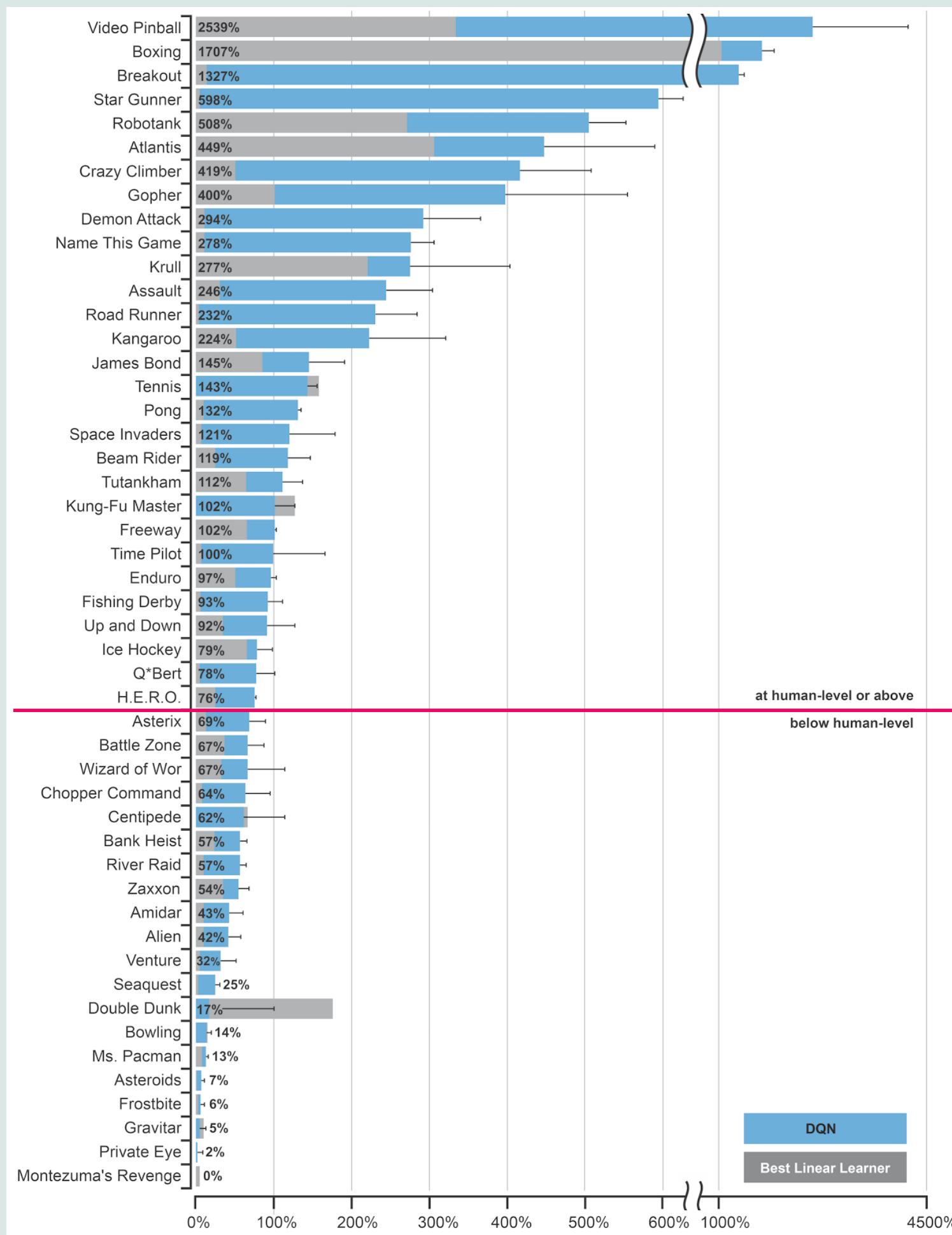
2015 年 Nature 出現一篇

“Human-level Control Through Deep Reinforcement learning”

為題的論文，基本上就是教電腦玩 Atari 的遊戲。



# RL 的重要實例: AI 玩電動

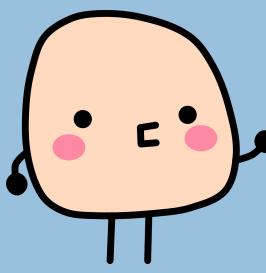


AI 勝過人類  
(超過 50%)



電腦好會玩!





# DeepMind (Google AI 研究團隊)

2011

Google Brain

Google 「原生」AI 研究團隊，由 Andrew Ng (吳恩達)、Jeff Dean 等人創立。曾有包括 Geoffrey Hinton 等重量級大師坐陣。

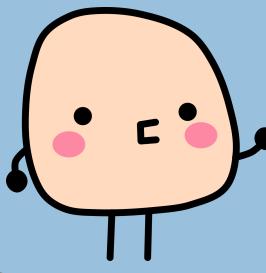
2010

DeepMind

最初在英國成立的 AI 實驗室，2014 年被 Google 收購。2016 年以強化學習訓練出來的 AlphaGo 擊敗世界棋王李世乭，2024 年 AlphaFold 獲得諾貝爾化學獎。

Google DeepMind

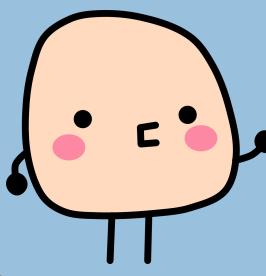
2024/04



# DeepMind 是一家很愛研究的公司



發表了超過 1,000 篇論文，  
在 Nature 或 Science 上  
的有 13 篇！



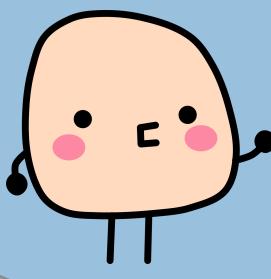
所以合併背後有一些八卦!!

Q

一直有傳聞 DeepMind 自認為是研究單位，  
之前他們有想要獨立離開 Google, 或是爭取  
更自主的經營研究, 這是真的嗎?



<https://www.facebook.com/diveai>

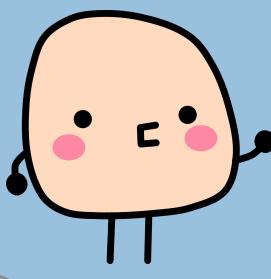


## 重要實例: AlphaGo



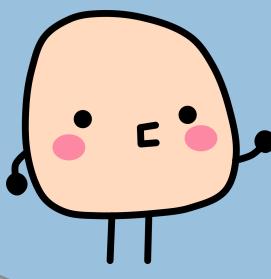
2017 年台灣人工智慧年會  
AlphaGo 創始人之一黃士傑博士演講

博士班時期開發 Erica,  
拿到電腦圍棋世界冠軍!

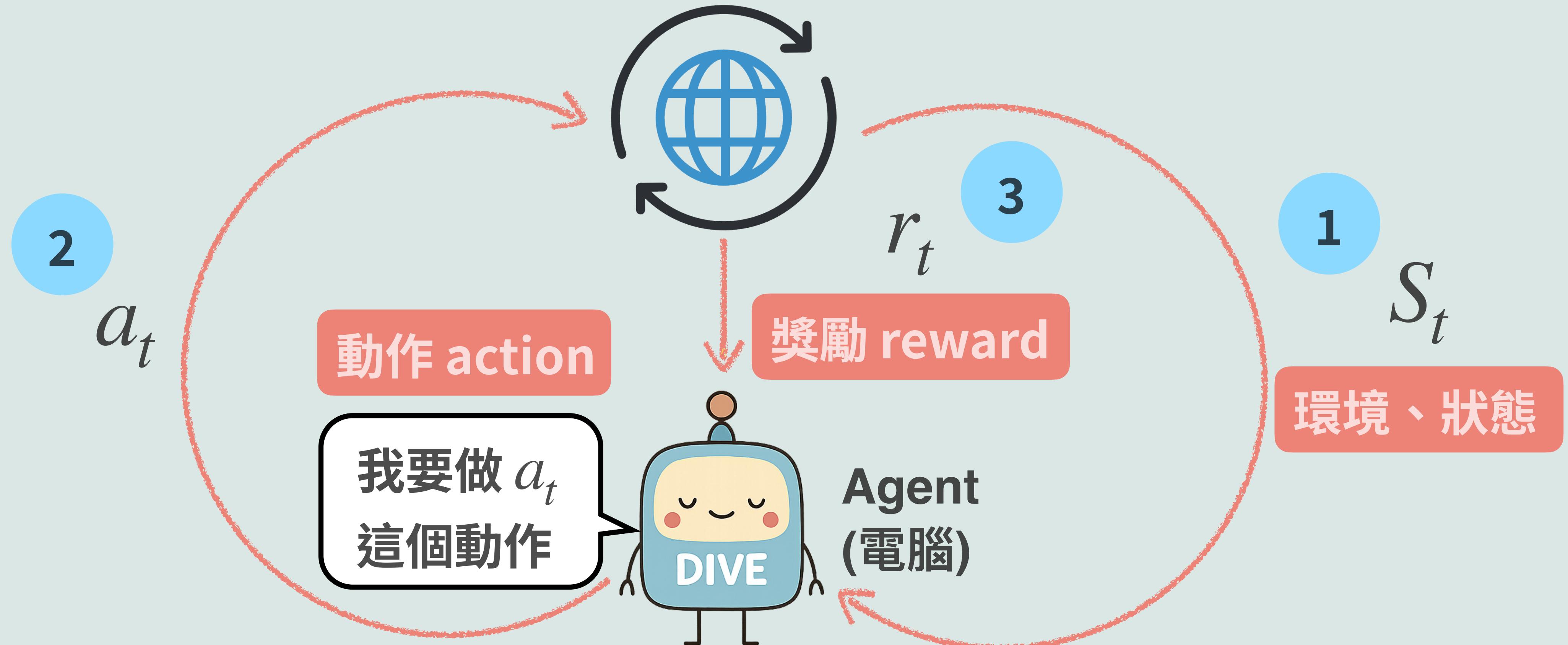


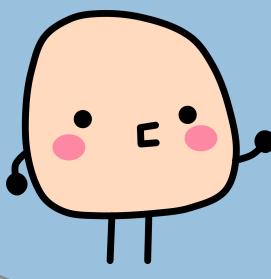
## 重要實例: AlphaGo



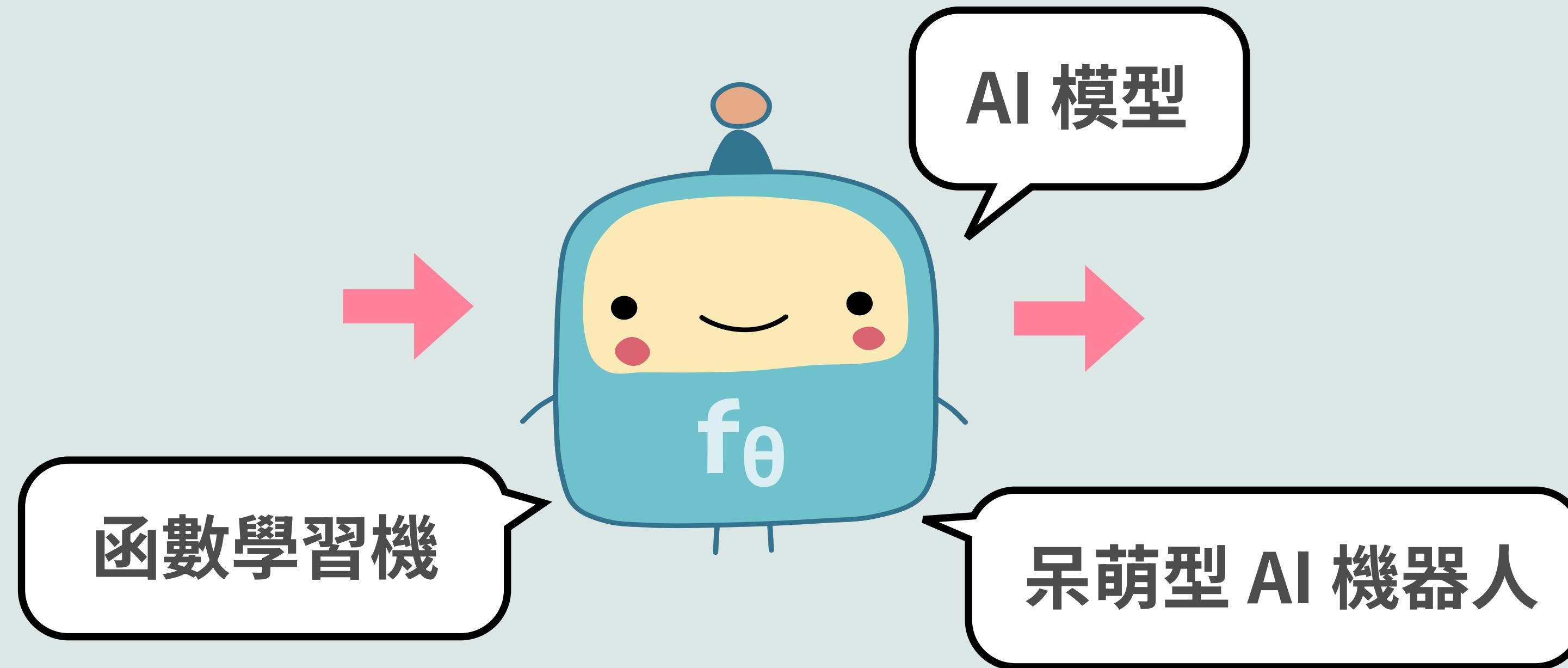


# 目標就是拿到最多獎勵的強化學習!

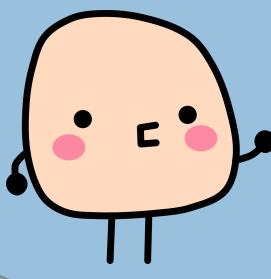




# 要打造的呆萌型 AI 機器人是什麼呢？



就是知道輸入是什麼、輸出是長什麼樣子



# 要學哪個函數呢？

我們以玩打磚塊為例

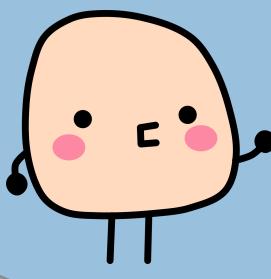


主要有兩  
種想法！

policy based

value based





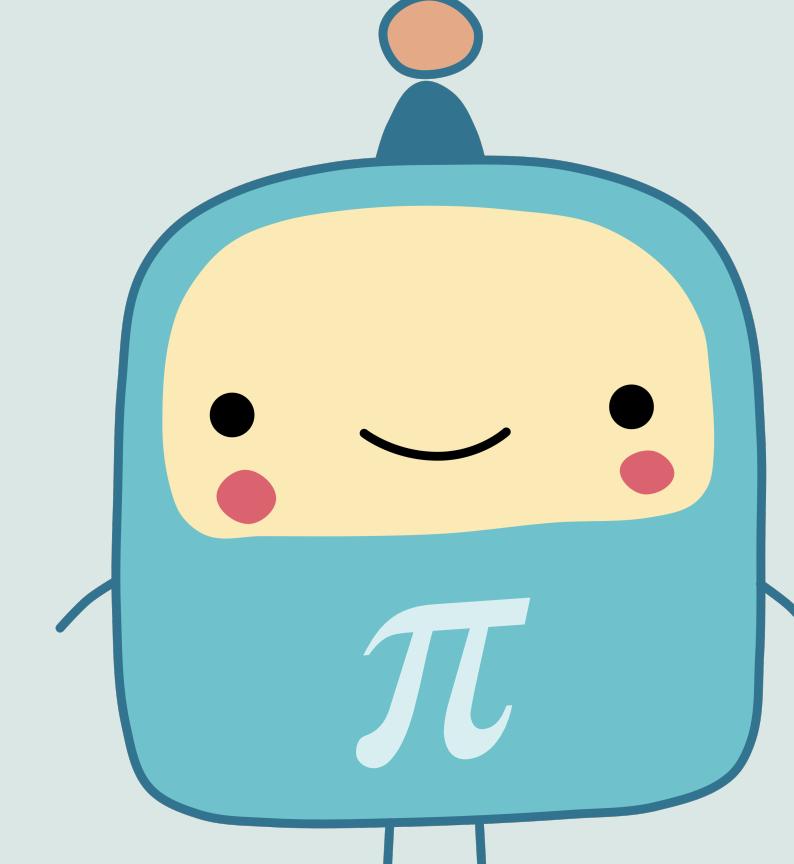
# 「自然的」選擇 Policy Based

1

## Policy Based



狀態  $S_t$



policy function

左

or

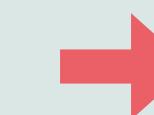
右

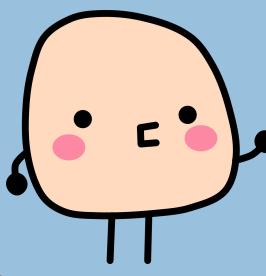
or

不動

動作  $a_t$

前面說過，這函數可能很難準備訓練資料！





## Value Based

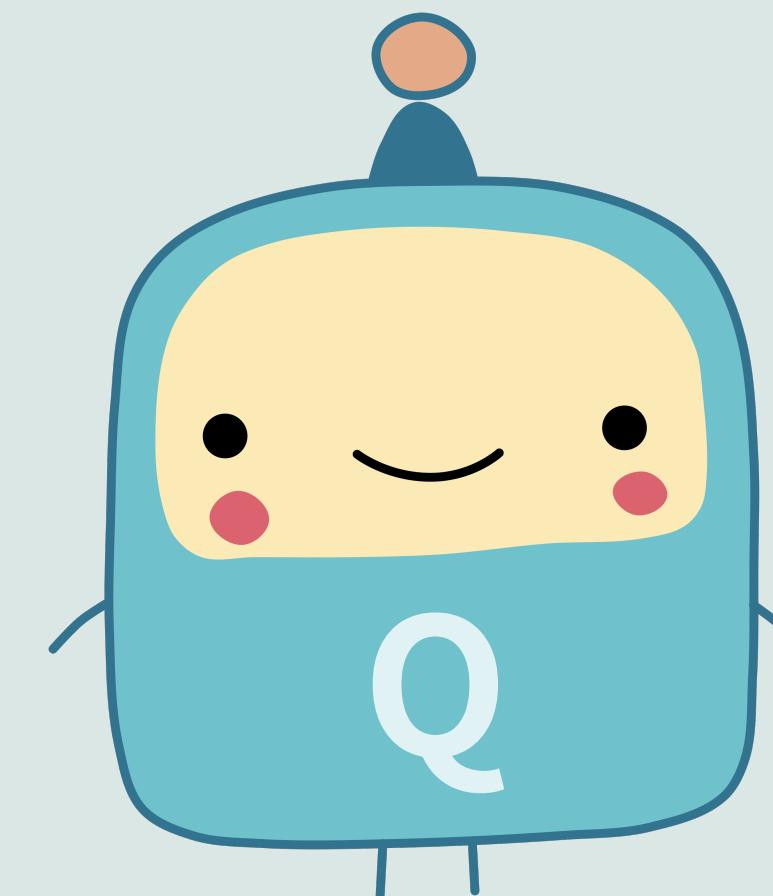
2

## Value Based



+

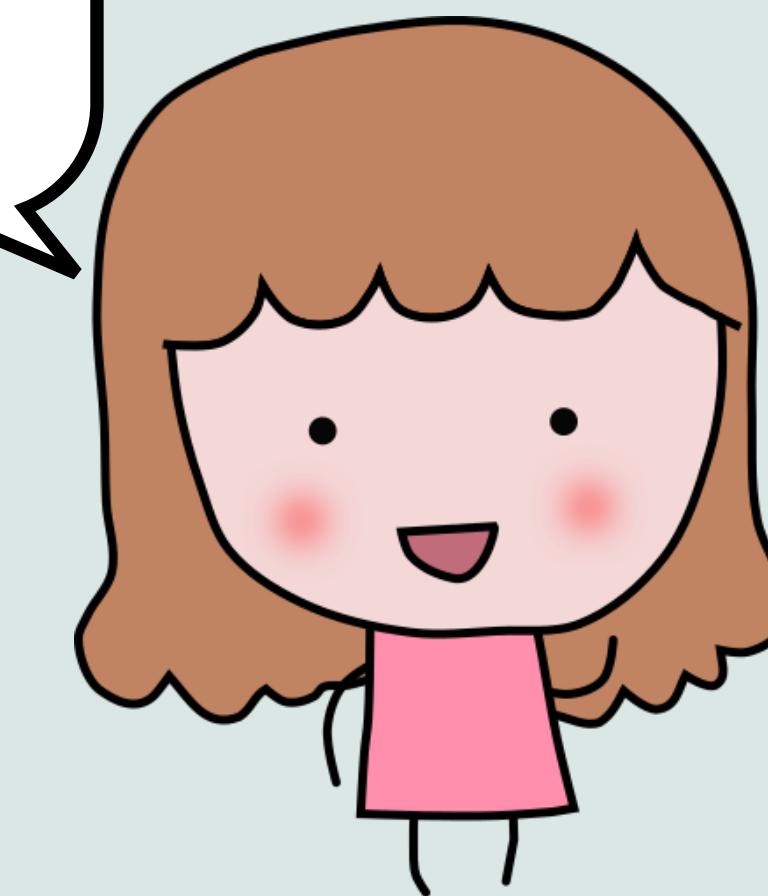
動作

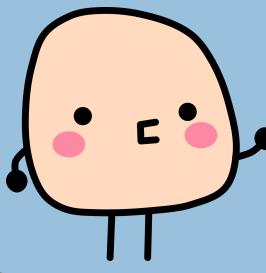


Value function

真的學成了，我們  
也會知道最好的  
動作是什麼！

→ 評分  
(通常估計  
reward)





## Value Based

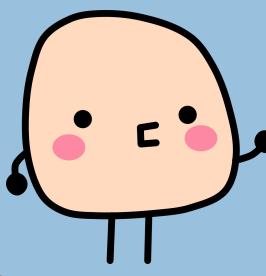
$$\pi(S) = \operatorname{argmax}_{a \in \mathcal{A}} Q_\theta(S, a)$$

假設某個狀態下的 Q 值

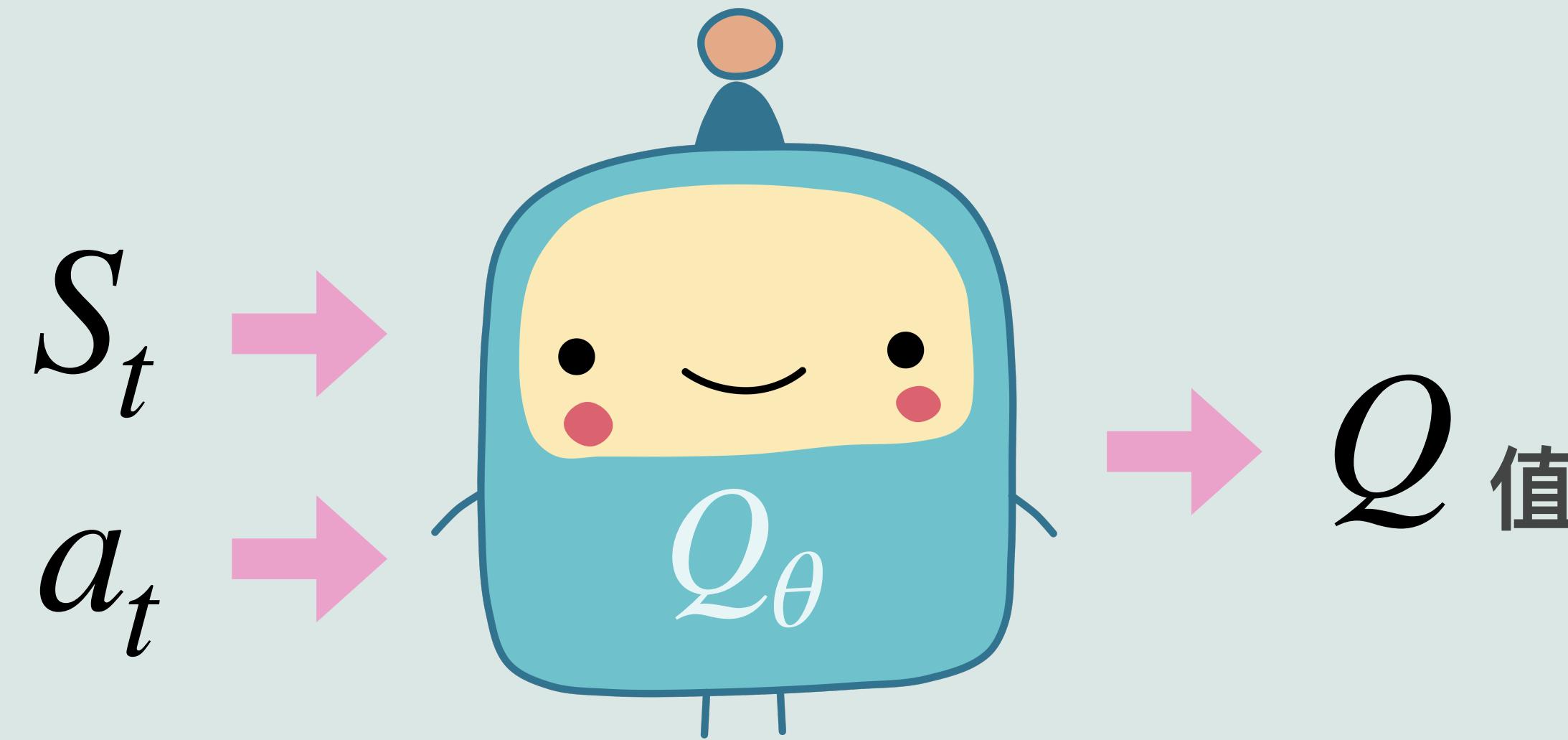


就是把所有的動作  
都帶入  $Q$  函數，看  
哪個最高分！

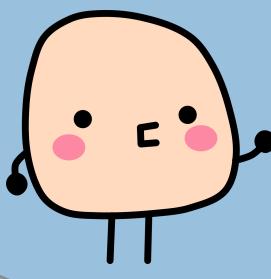




## Deep Q-Learning

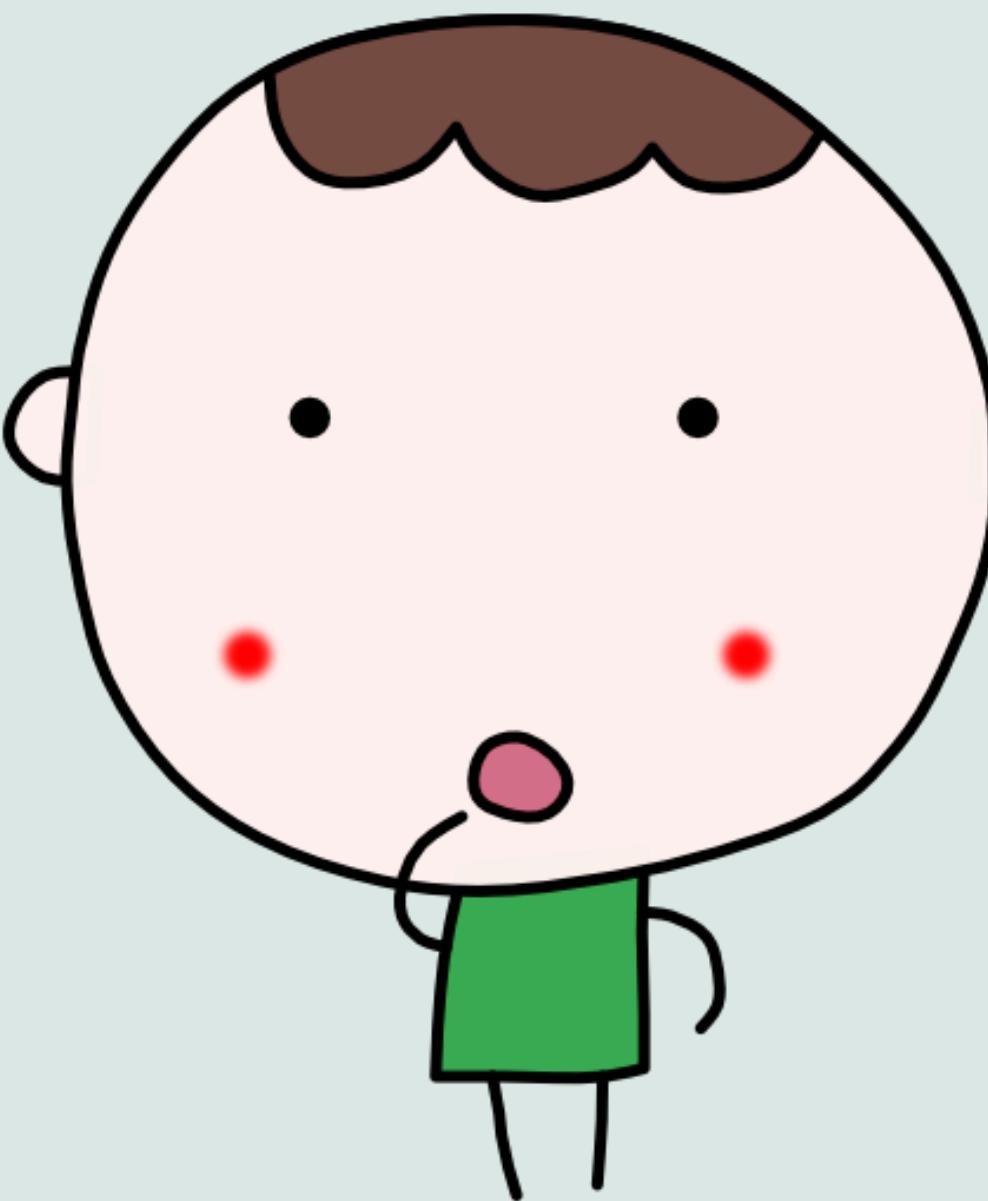


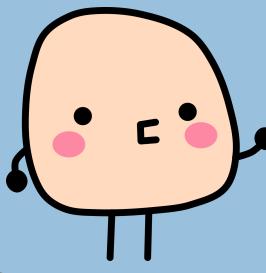
深度強化學習最常見的 Deep Q-Learning,  
就是用神經網路把  $Q$  函數學起來!



# 大問題

訓練資料怎麼來？





## Deep Q-Learning

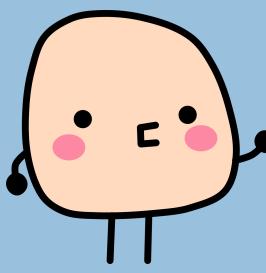
簡單的說，就是我們會讓電腦自己去玩，然後**部份的情況我們知道  $Q$  值是多少**。(雖然開始玩得很差，這  $Q$  值可能很沒用)

然後用這些**知道的  $Q$  值當訓練資料**，完整的  $Q$  函數**用深度學習的方式學起來**！

簡單的說就是自己生訓練資料自己學，所以應該叫 **self-supervised learning!**

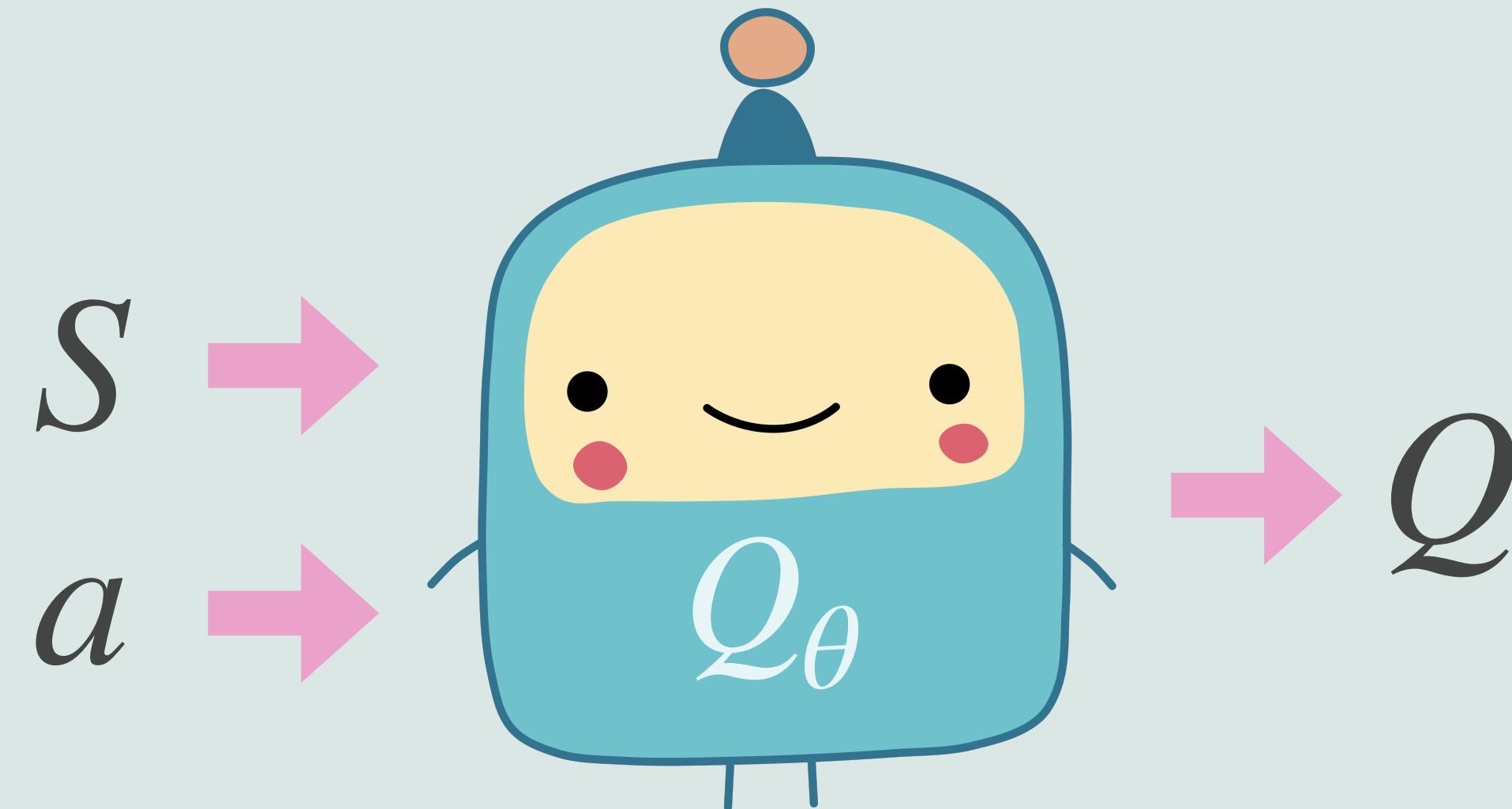


LeCun



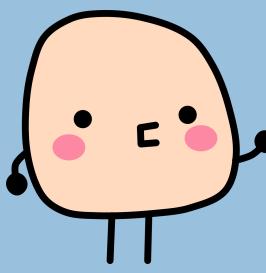
## 我們要想辦法打造出訓練資料

$[(S_1, a_1), Q_1], [(S_2, a_2), Q_2], \dots$



雖然大部份的情況  
我們都沒碰過，不過就用一個神經網路把它學起來！





## 訓練資料的 Q 值

$$Q(S_t, a_t) = r_t + r_{t+1} + \cdots + r_T$$

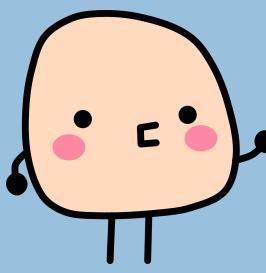
常常寫成  $G_t$

缺點是要完整的 episode  
(完整玩一次的過程)

認真想想，訓練資  
料的 Q 值不太好  
算...



蒙地卡羅 Monte-Carlo (MC) 法



## 訓練資料的 Q 值

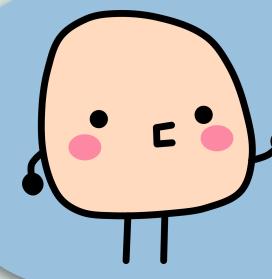
$$Q(S_t, a_t) = r_t + \gamma \cdot r_{t+1} + \cdots + \gamma^{T-t} \cdot r_T$$

這樣不管在級數的收斂啦，對未來比較沒有那麼確定的狀況啦等等，都考量進去。

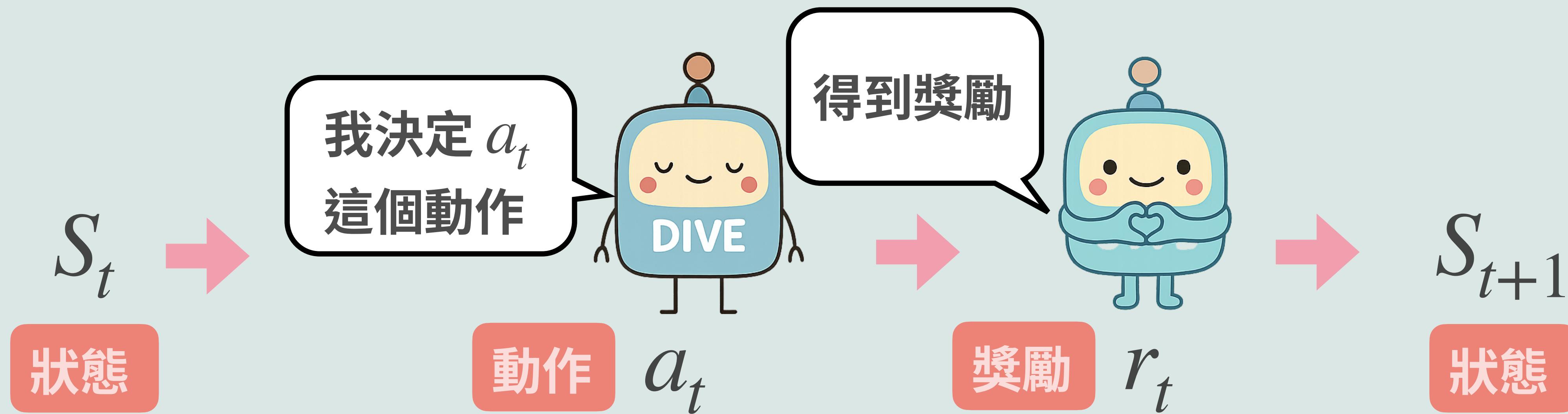
我們常常會乘上一個 **discount  $\gamma$** 。

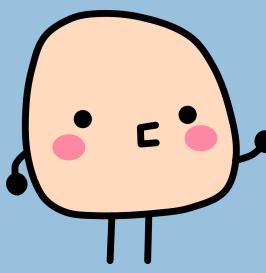


蒙地卡羅 Monte-Carlo (MC) 法



# 有沒有可能玩一步就有一筆訓練資料？





## 訓練資料的 Q 值

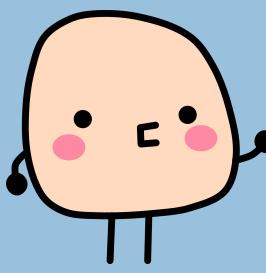
$$Q(S_t, a_t) = r_t + \gamma \cdot \max_{a \in \mathcal{A}} \{Q(S_{t+1}, a)\}$$

認真想想,  $t$  這個時間點的  $Q$  值, 基本上就是立即得到的  $r_t$ , 加上下一次的  $Q$  值。

這裡假設我們後面都用「最佳的」玩。



Temporal-Difference (TD) 法



## 訓練資料的 Q 值

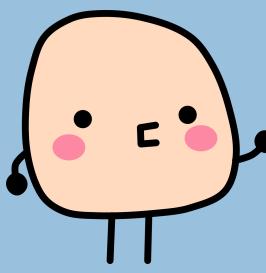
$$Q(S_t, a_t) = r_t + \gamma \cdot \max_{a \in \mathcal{A}} \{Q(S_{t+1}, a)\}$$

有一次我們玩到  $S_t$  的狀態，做了  $a_t$  的動作，只需要再知道當下的 reward  $r_t$ ，還有下一次的狀態  $S_{t+1}$  是什麼，就可以更新  $Q$  值！

$(S_t, a_t, r_t, S_{t+1})$

於是我們可以做  
**Exprience Replay**。



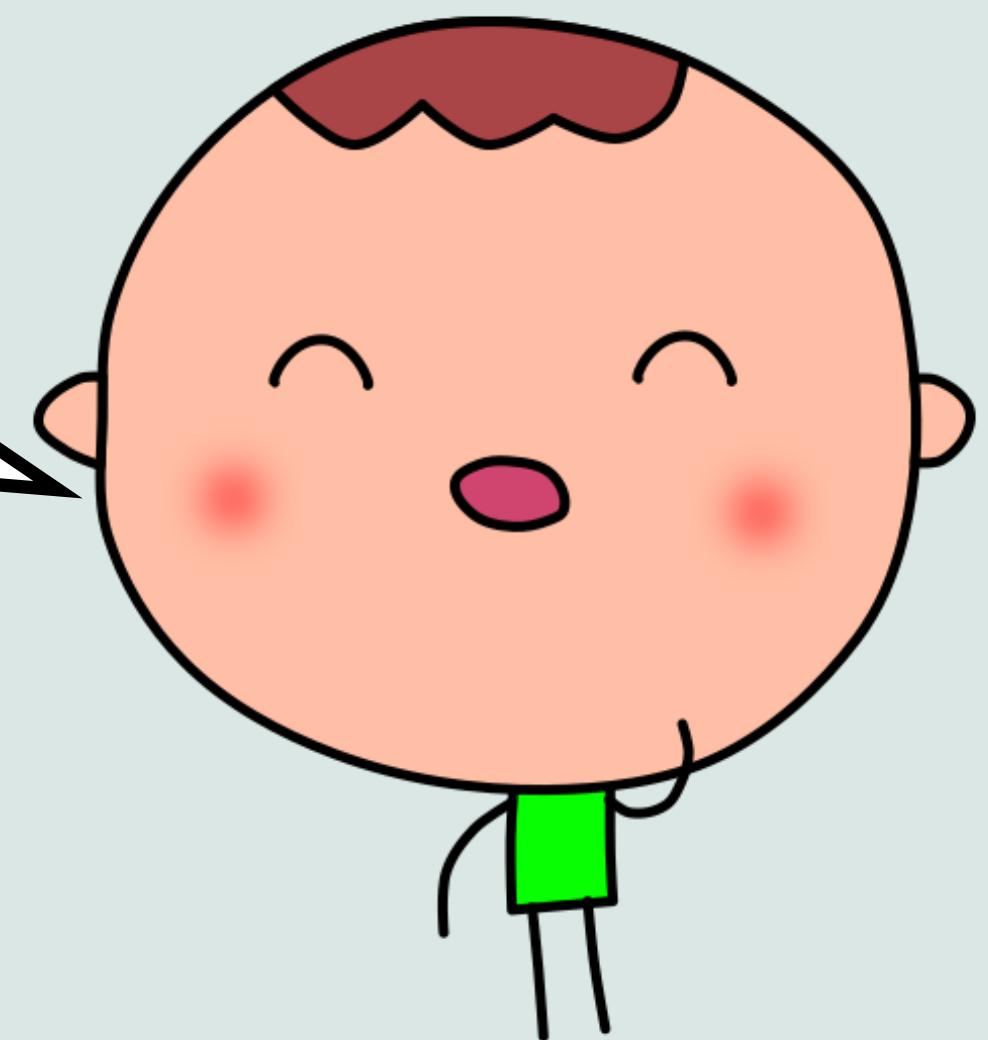


## 訓練資料的 Q 值

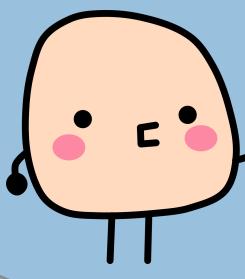
$$Q(S_t, a_t) = r_t + \gamma \cdot \max_{a \in \mathcal{A}} \{Q_\theta(S_{t+1}, a)\}$$

我們只要每步都收集到這小小片段, 就可以準備訓練  $Q$  的資料。

因此就是讓電腦一直玩一直玩!

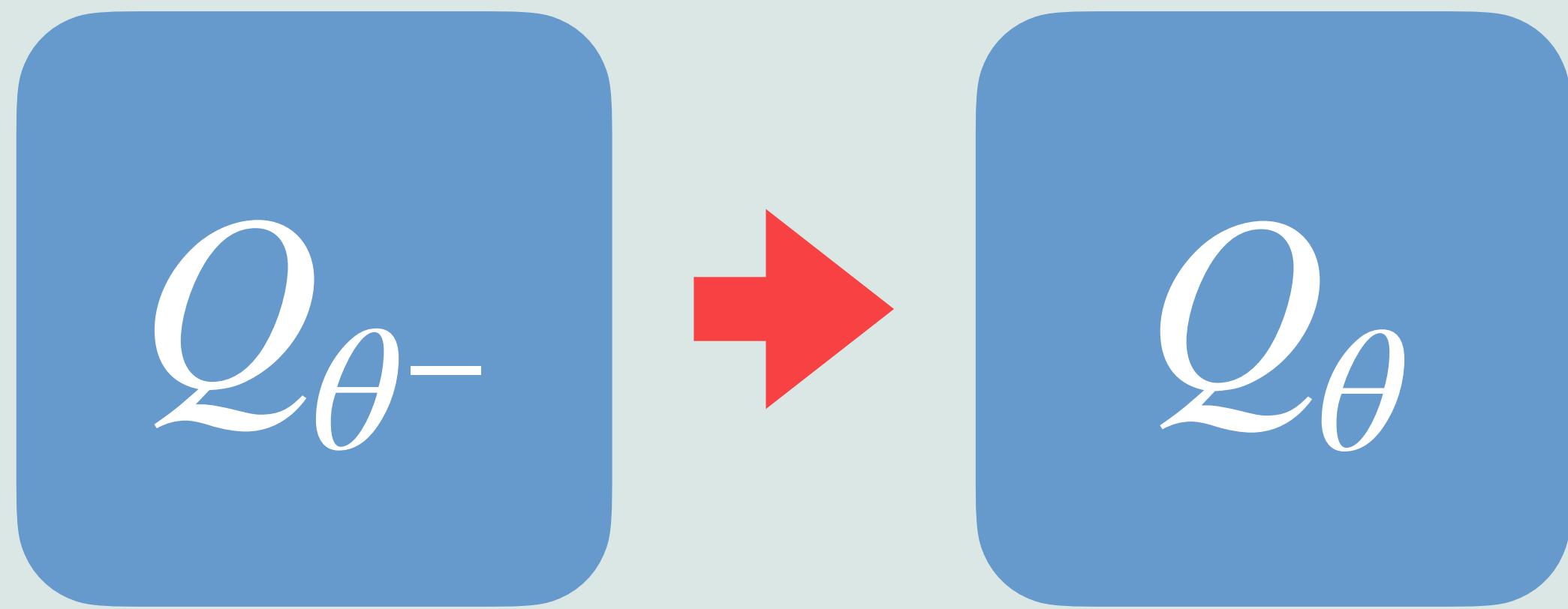


$(S_t, a_t, r_t, S_{t+1})$



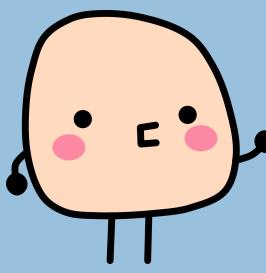
仔細想想，這是「自己學自己」的過程

為了方便，我們把上次參數的狀態叫  $\theta^-$ ，現在正要更新的叫  $\theta$ 。

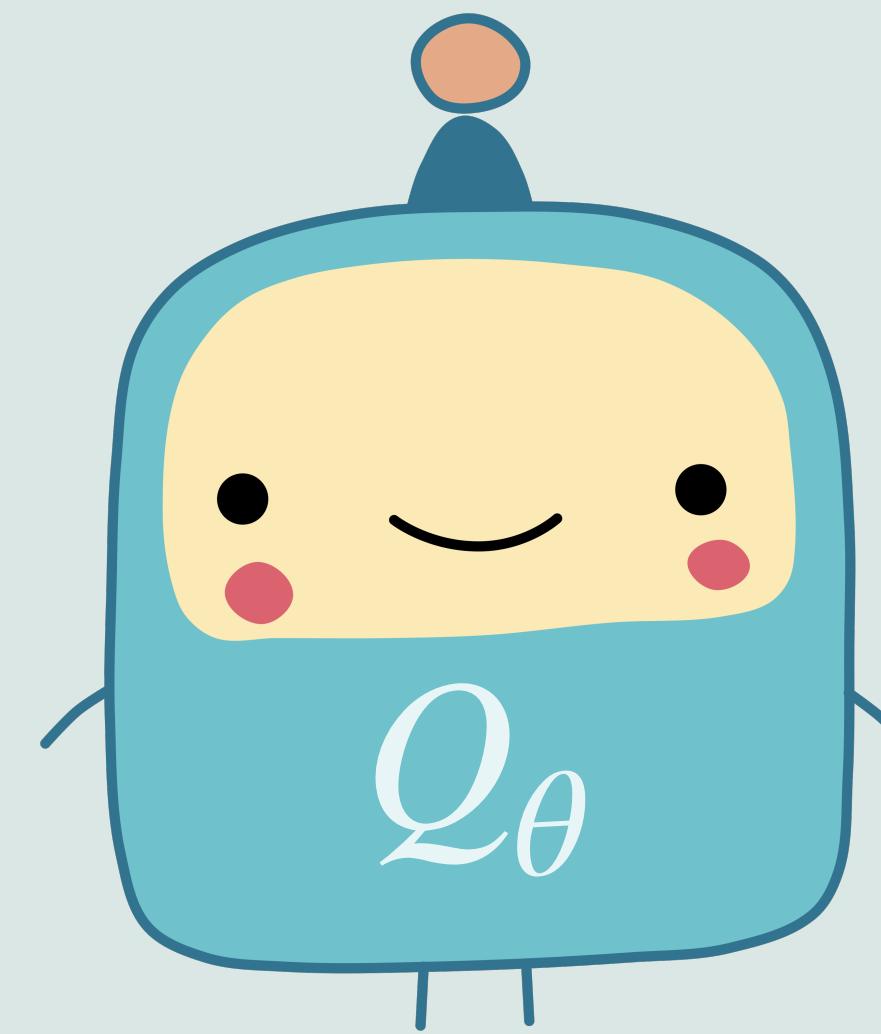


舊版

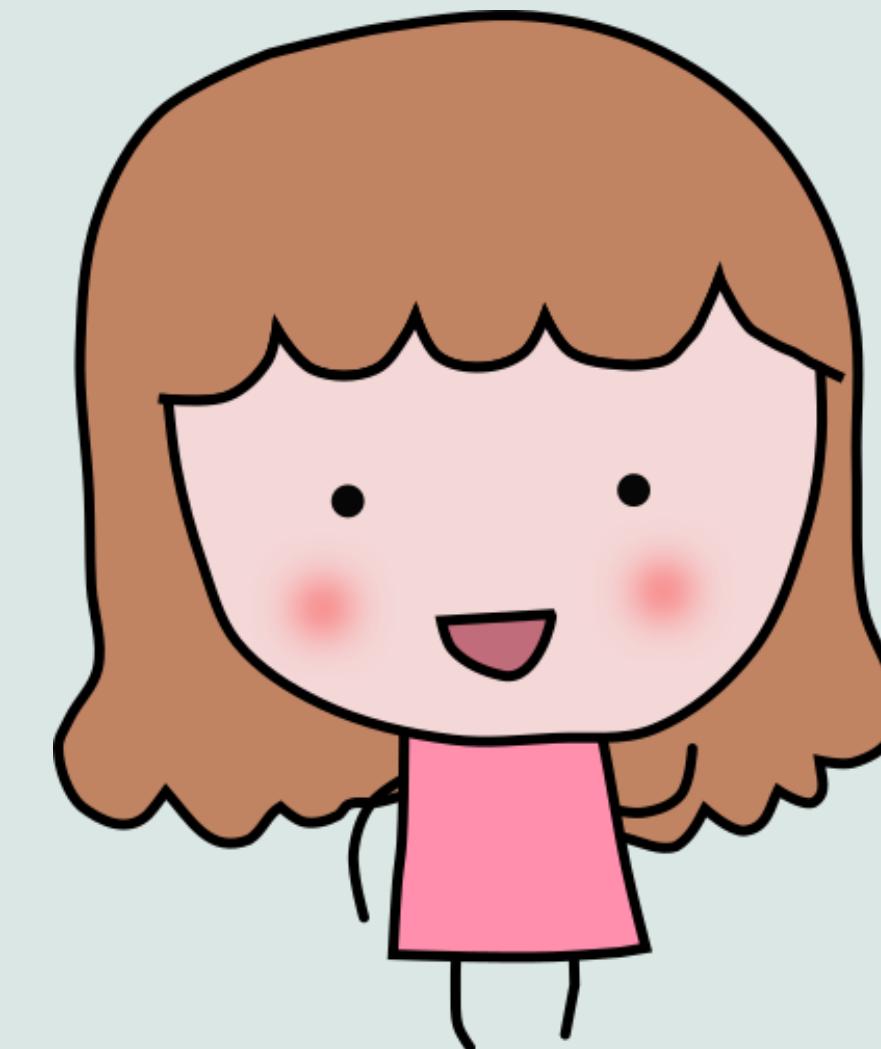
更新版



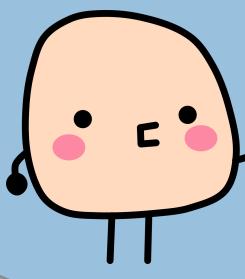
仔細想想，這是「自己學自己」的過程



有這些訓練資料，我們就可以好好去訓練  
我們的神經網路了！

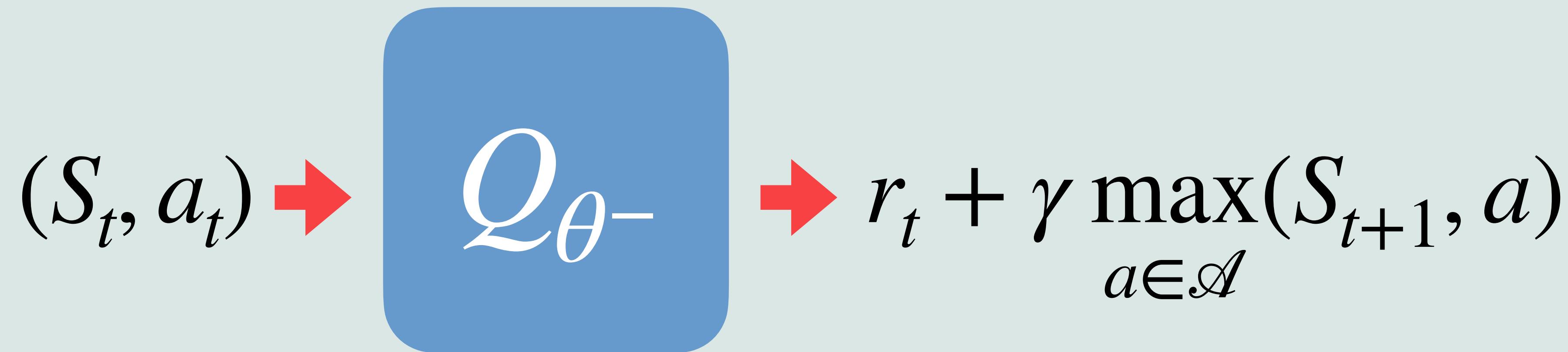


做 gradient descent 時就是  
用舊版  $Q$  生出的訓練資料！

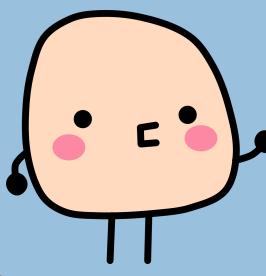


仔細想想，這是「自己學自己」的過程

有了一筆經驗  $(S_t, a_t, r_t, S_{t+1})$ ，我們會有一筆訓練資料是這樣...



再一次， $\gamma$  值是我們自己定的 discount，通常是 0 到 1 間的一個數。

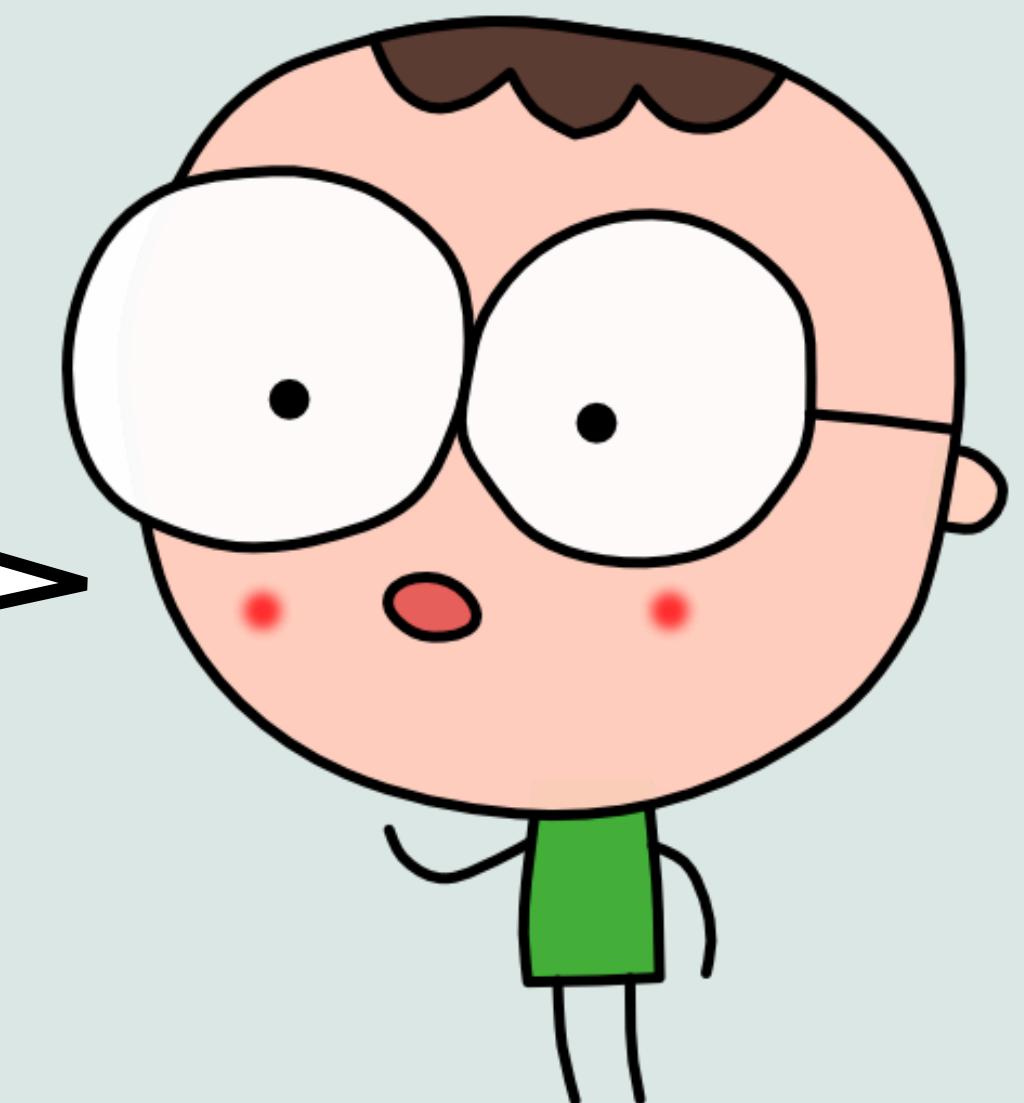


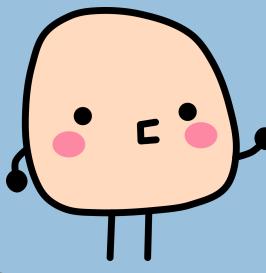
## Greedy Policy

$$\pi(S) = \arg \max_a Q(S, a)$$

當  $Q$  函數學成時，我們完完全全讓電腦依  $Q$  函數決定最好的動作，這叫 **greedy policy**！

讓電腦去玩，生訓練資料時其實還有件事要注意。





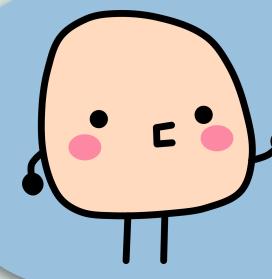
## $\varepsilon$ -Greedy Policy

這裡還有個問題...

開始的時候，電腦要依什麼規則玩？

當然可以用我們目前的  $Q_\theta$  選最高分的動作，但不要忘了我們的  $Q$  函數還爛得不得了啊...





## $\varepsilon$ -Greedy Policy

取一個  $\varepsilon \in [0,1]$ , 我們每次要做個動作時, 取一個 0 到 1 間的亂數  $r$ 。

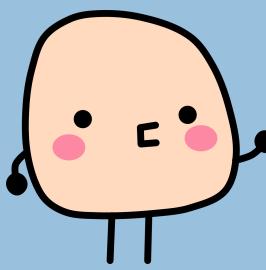
開始的時候  $\varepsilon$  設大一點。

$$\begin{cases} r > \varepsilon & \text{Greedy Policy (用 } Q \text{ 函數決定)} \\ r \leq \varepsilon & \text{亂亂玩!} \end{cases}$$





02.  
Policy Based 才是 LLM  
主流

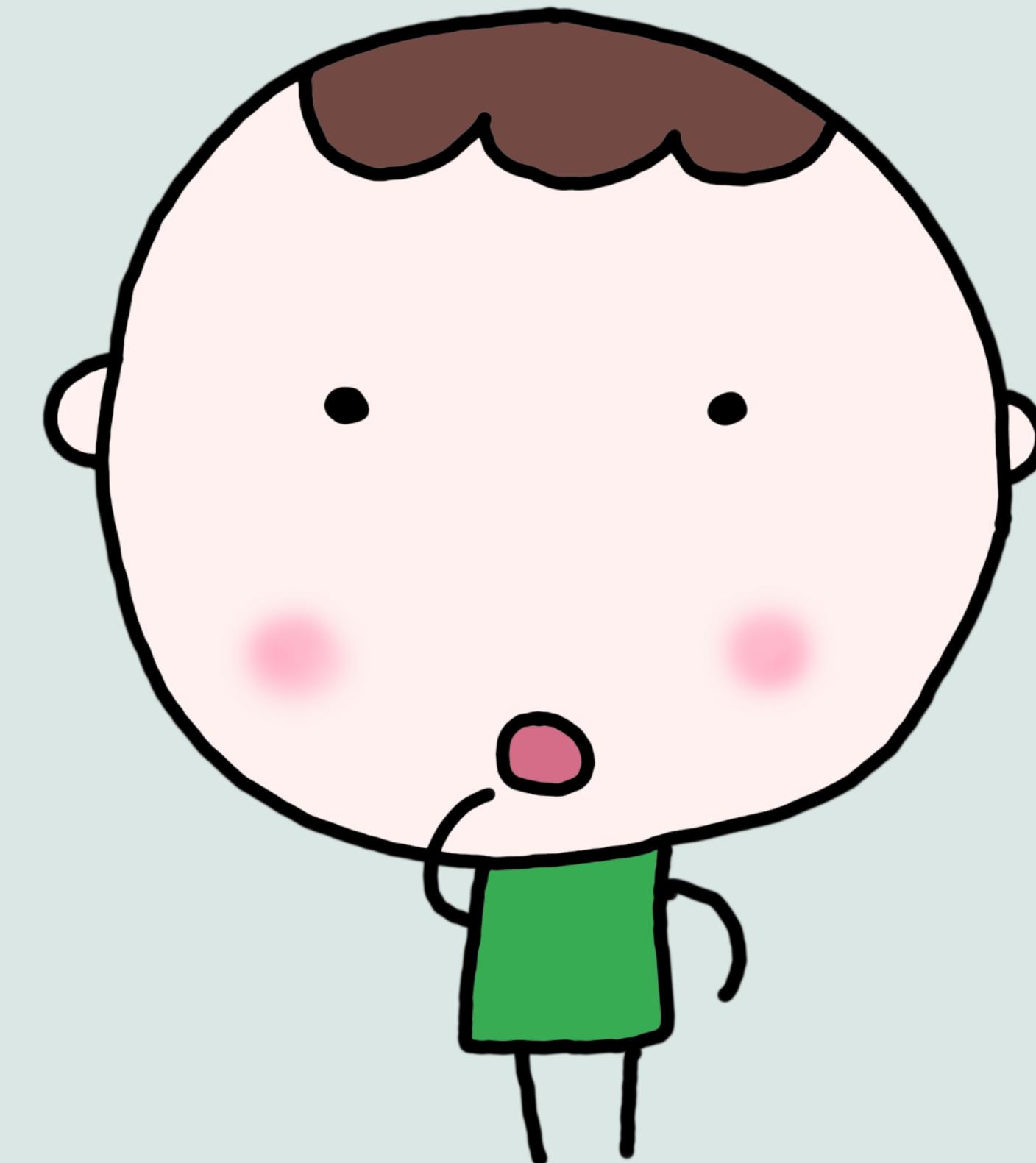


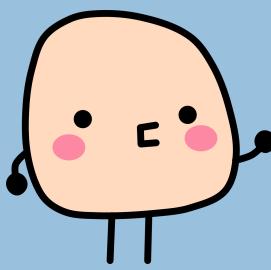
## 我們真的不能直接學 Policy 嗎？

畢竟 Q-Learning 還是有個缺點，就是無限多個（比如連續的）動作選擇，就很難做到。

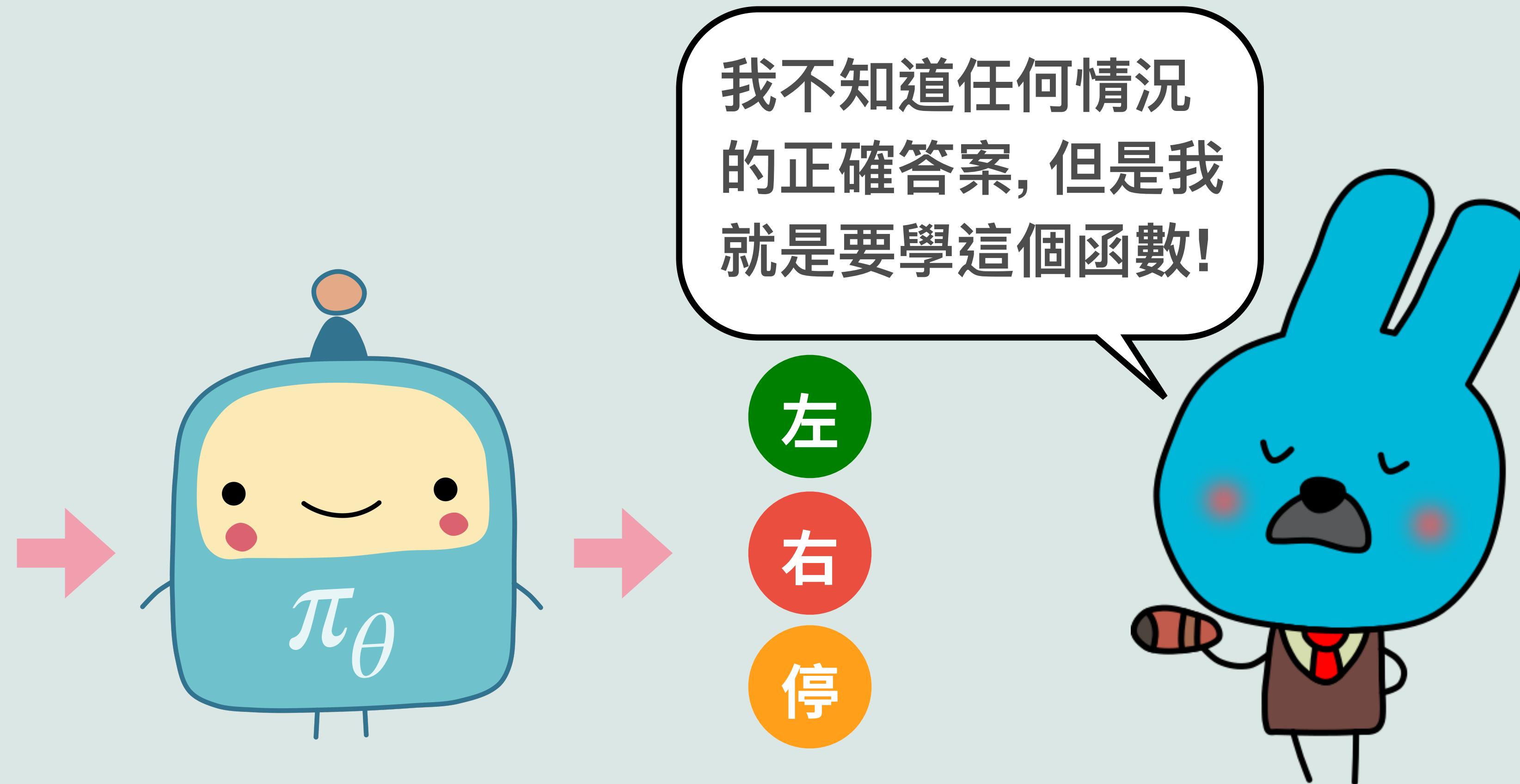
（注意是不是不是  $Q$  函數本身不能訓練，而是很難用來選最佳動作。）

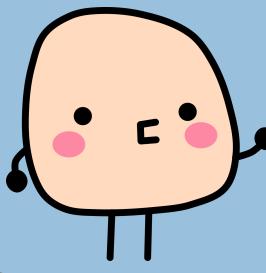
然後 LLM 輸出可能雖然是有限的，但是數量非常大，用 Q-Learning 計算量太大。





# 我們真的不能直接學 Policy 嗎？



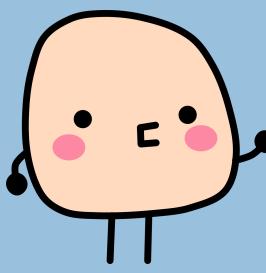


# 我們真的不能直接學 Policy 嗎？

想想其實我們目標很清楚 (要極大化 reward), 是不是可以把我們的目標, 放在... 呃, 目標函數?

## Gradient Policy



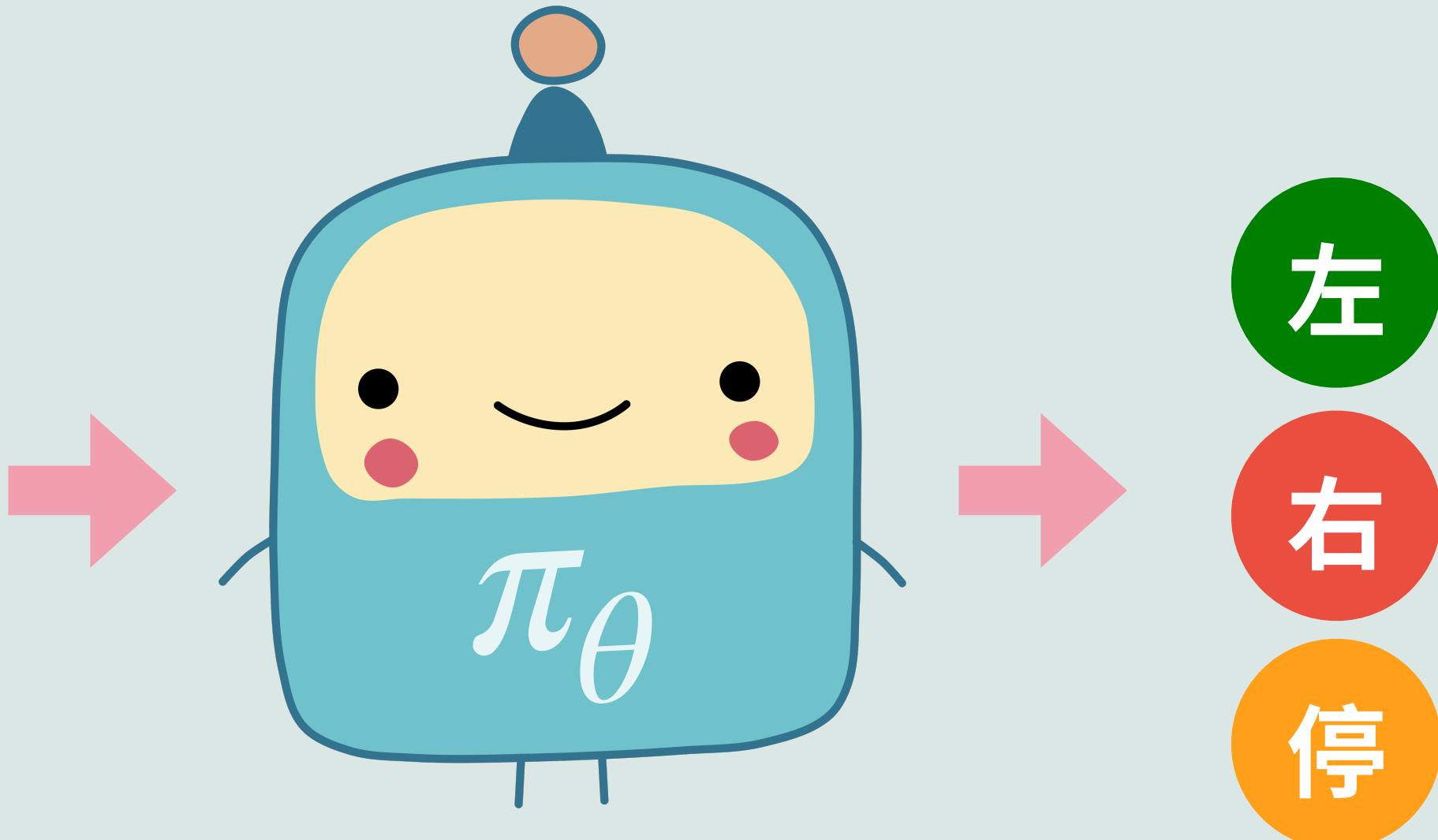


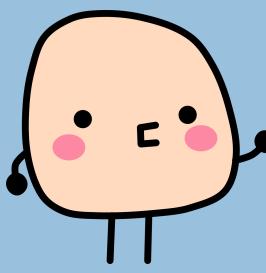
## Gradient Policy



就是設計個和  $\pi$   
有關的目標函數！

$$J(\theta)$$



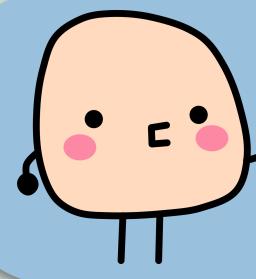


## Gradient Policy

$$J(\theta) = V_{\pi_\theta}$$

簡單啦，就是最大化我們的 state value!





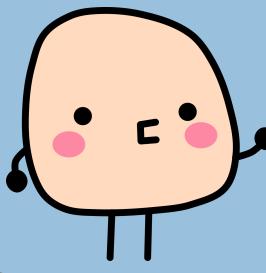
# Gradient Policy

問題是...

$$J(\theta) = V_{\pi_\theta}$$

我不會算啊...

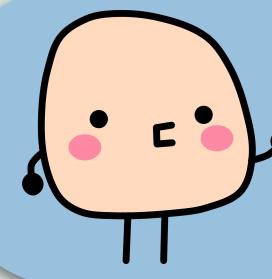




## Gradient Policy

如果說我們有個遊戲的過程片段 trajectory (常常就是一個 episode, 但 episode 一般是要完整從頭到尾)。

$$\tau = \{S_1, a_1, r_1, S_2, a_2, r_2, \dots, S_T, a_T, r_T\}$$



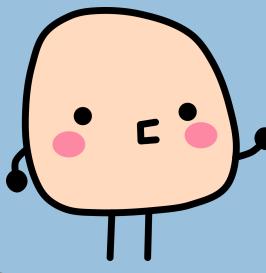
## Gradient Policy

我們好像就會算這系列動作的得分，目標是要最大化...

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

可是，這和我們的  $\pi$  無關啊！





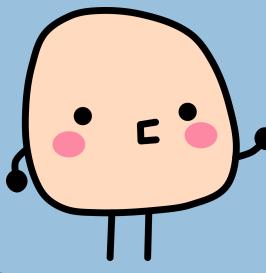
## Gradient Policy

一個想法是，我們神經網路做得和真的一樣，那會得這麼多分。所以得分高就越像越好...

然後就可以做  
gradient descent!



$$R(\tau) \cdot \sum_{t=1}^T -\log \pi_\theta(a_t | S_t)$$



## Gradient Policy

最後我們好幾個實際的 trajectory 就加總。另外就是我們可以考慮計分更細膩。

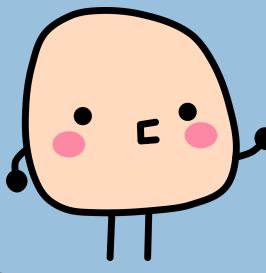
$$R(\tau) \cdot \sum_{t=1}^T -\log \pi_\theta(a_t | S_t)$$





03.

讓唬爛王不要那麼唬爛



## LLM 訓練大致上有兩個部份

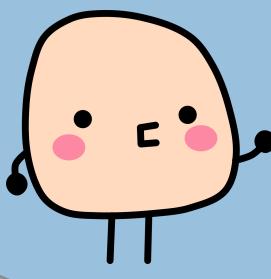
1

基礎模型訓練: 標準預測下一個 token

2

對齊 (alignment)

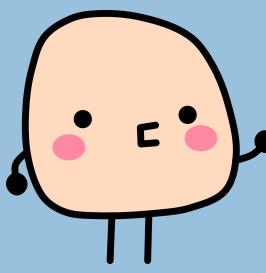




## 「對齊」是對齊什麼呢？



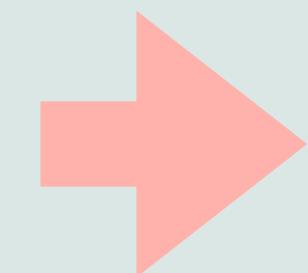
- 對齊「人類價值」 (Value Alignment)
- 對齊「人類意圖」 (Intent Alignment)
- 對齊「工具角色」 (Role Alignment)



## 以訓練的過程來看

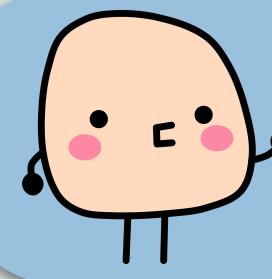
pre-training

預測下一個 token, 基  
礎語言模型的訓練

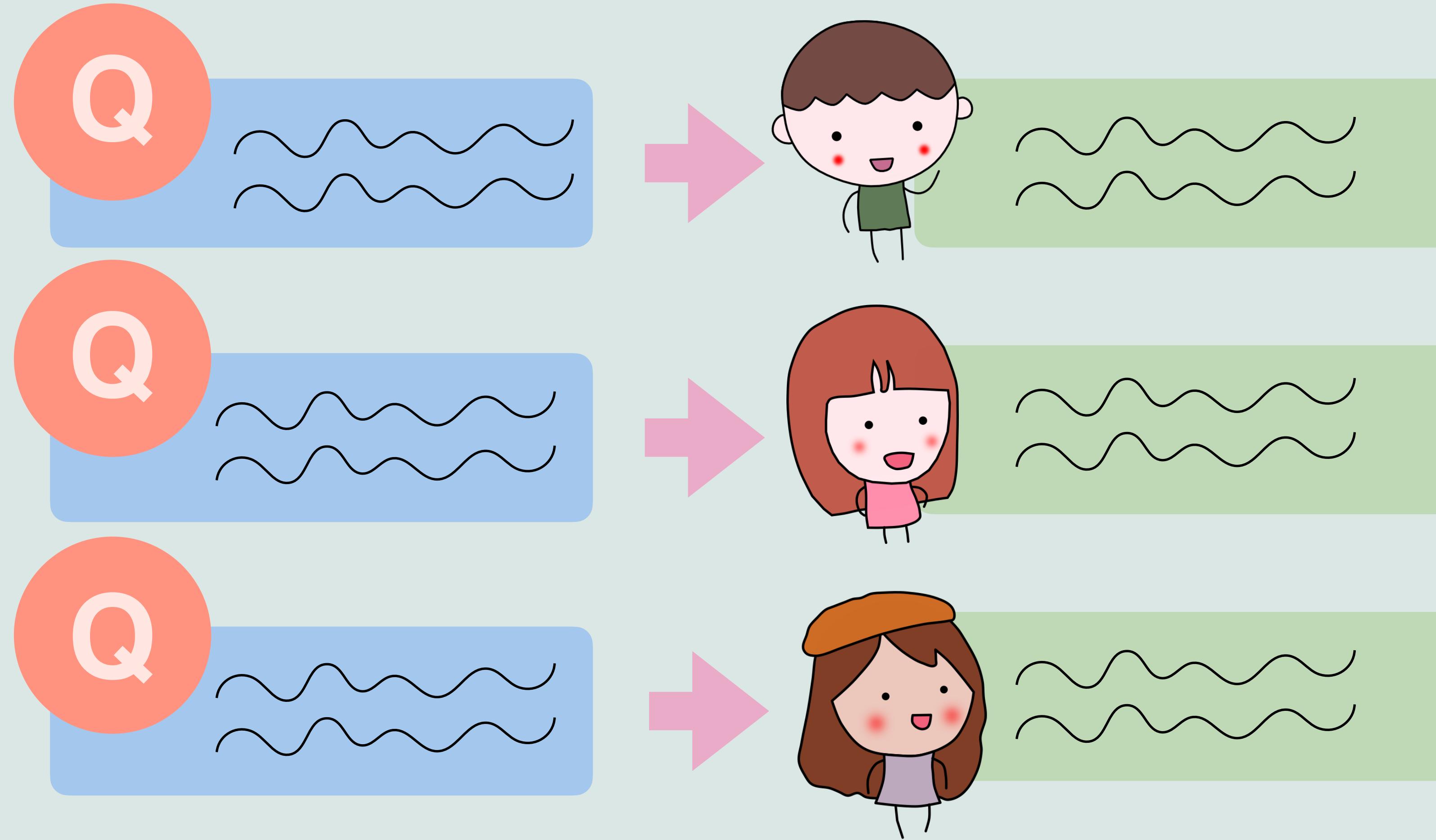


post-training

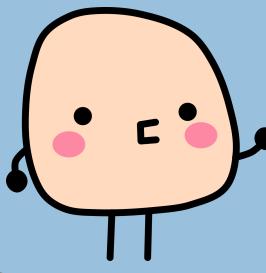
用 SFT, RLHF 等方法  
去「對齊」



## Step 1. SFT (Supervised Fine-Tuning) 真人手把手教學



由真人來回答，做為範例。去 fine tune LLM。

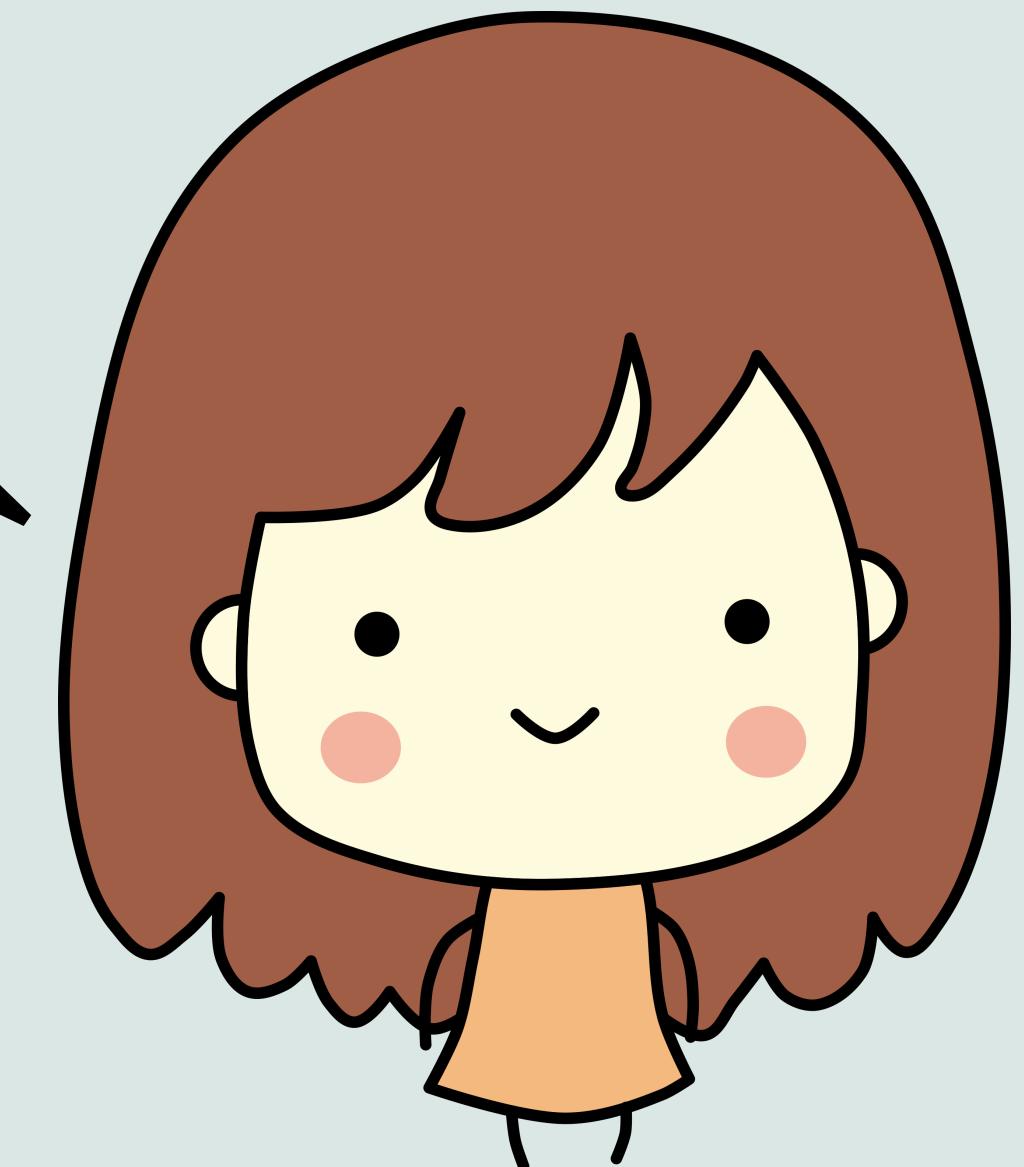


## SFT 的目標

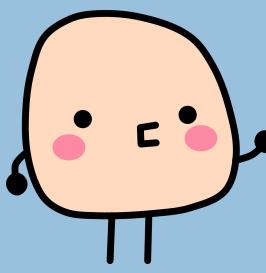
SFT 真的想教 LLM 的是：

- 風格
- 任務格式 (task format)
- 步驟拆解方式
- 語氣
- 系統性的回覆架構

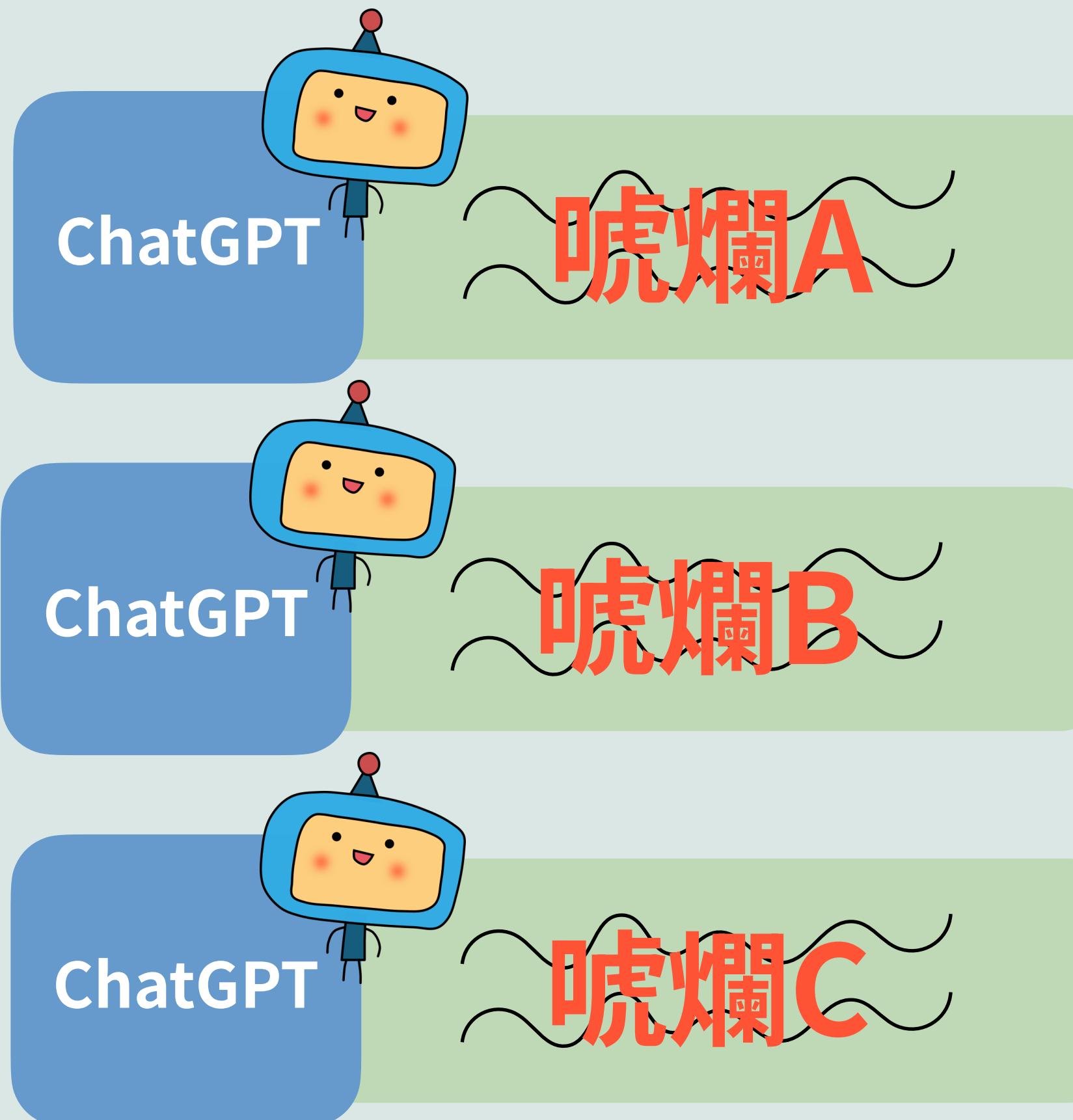
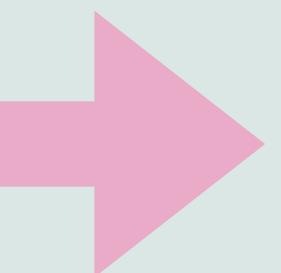
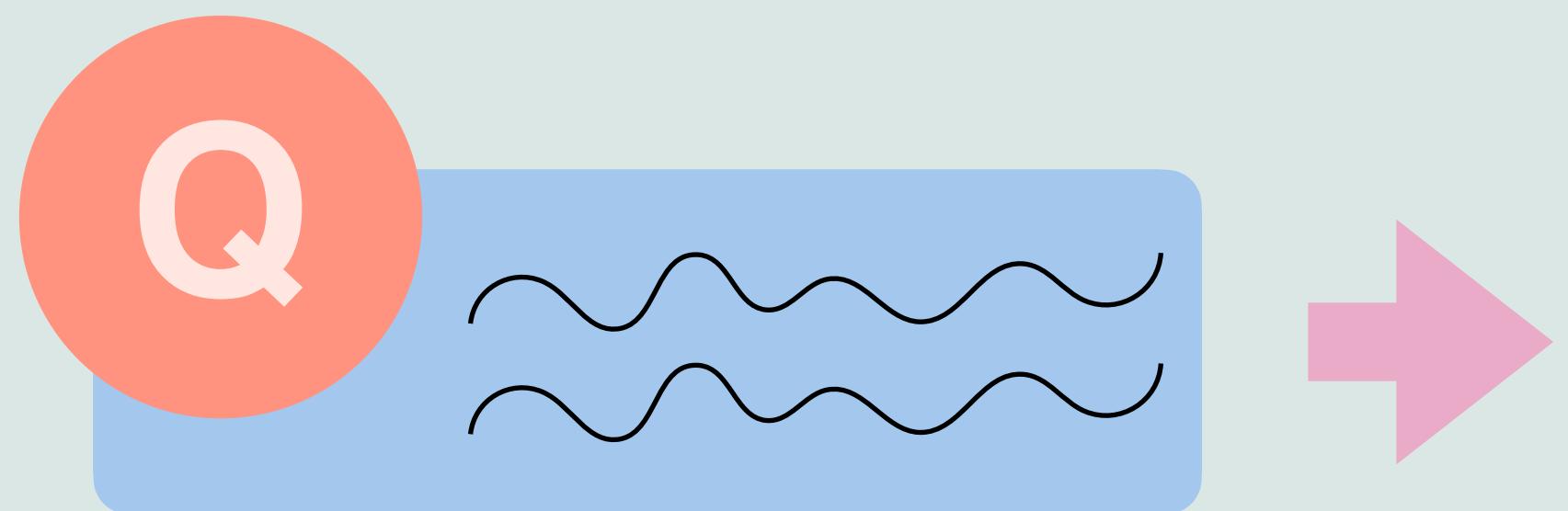
SFT 並不是真的要  
教 LLM 問題的正確  
答案 (新知識)。



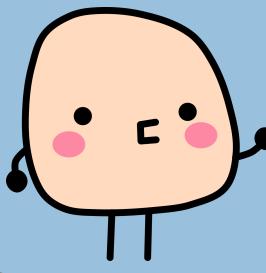
訓練集比資料量更重要的是**示範品質與風格一致性**。



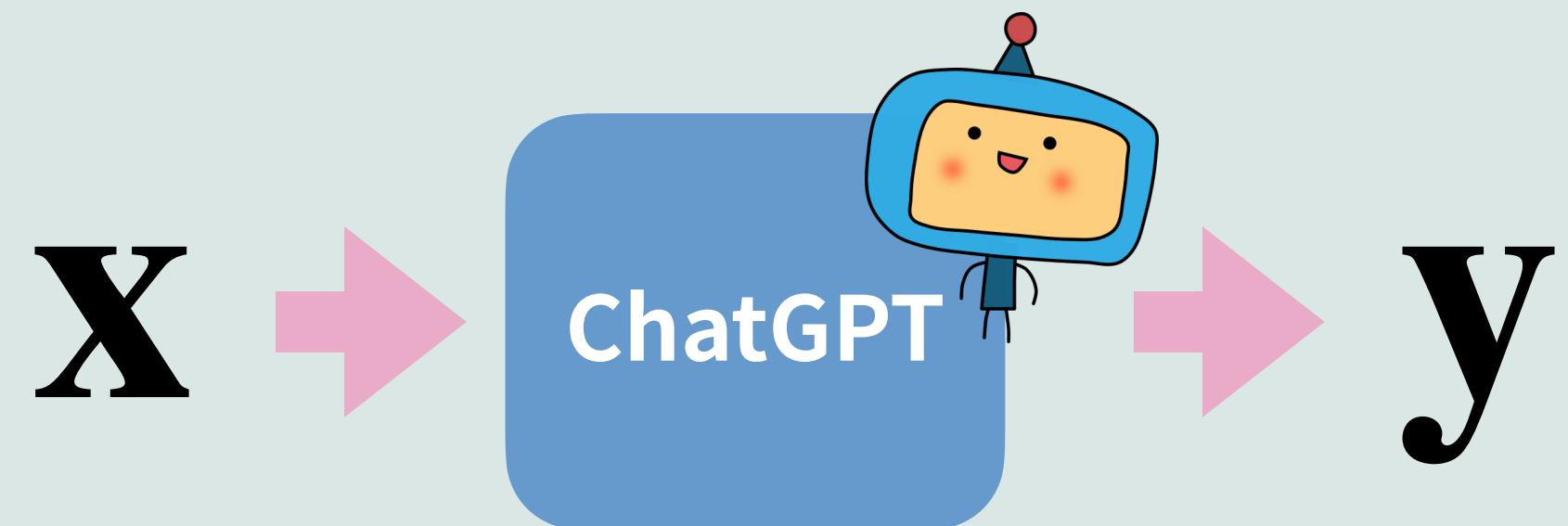
## Step 2. 打造評分系統 (reward function)



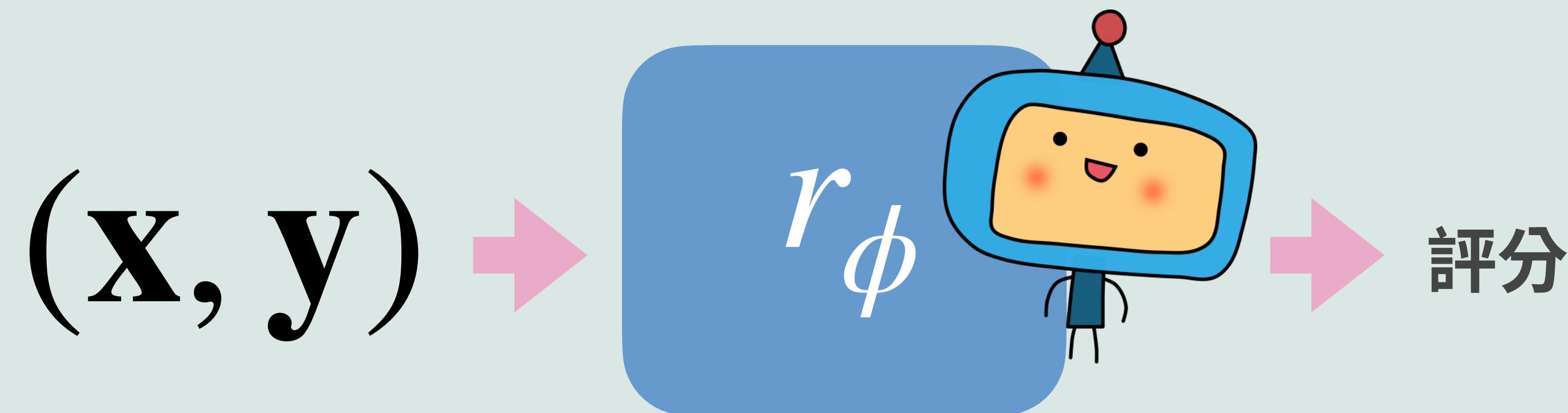
人類評分, 比如  $A > B = C$

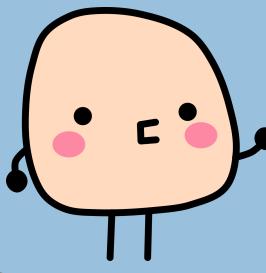


## Step 2. 打造評分系統 (reward function)



問題是  $x$ , 嘟爛王 ChatGPT 說答案是  
 $y$ , 我們想知道答得如何?





## Step 2. 打造評分系統 (reward function)

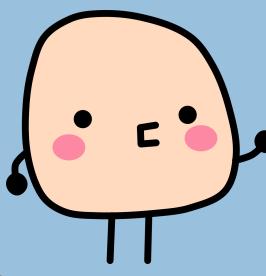
假設有個問題  $x$ , ChatGPT 說了  $y_w$  和  $y_\ell$  兩個答案, 而人類標記  $y_w$  是比較好的答案。

$$\ell(\phi) = -\log(\sigma(r_\phi(x, y_w) - r_\phi(x, y_\ell)))$$

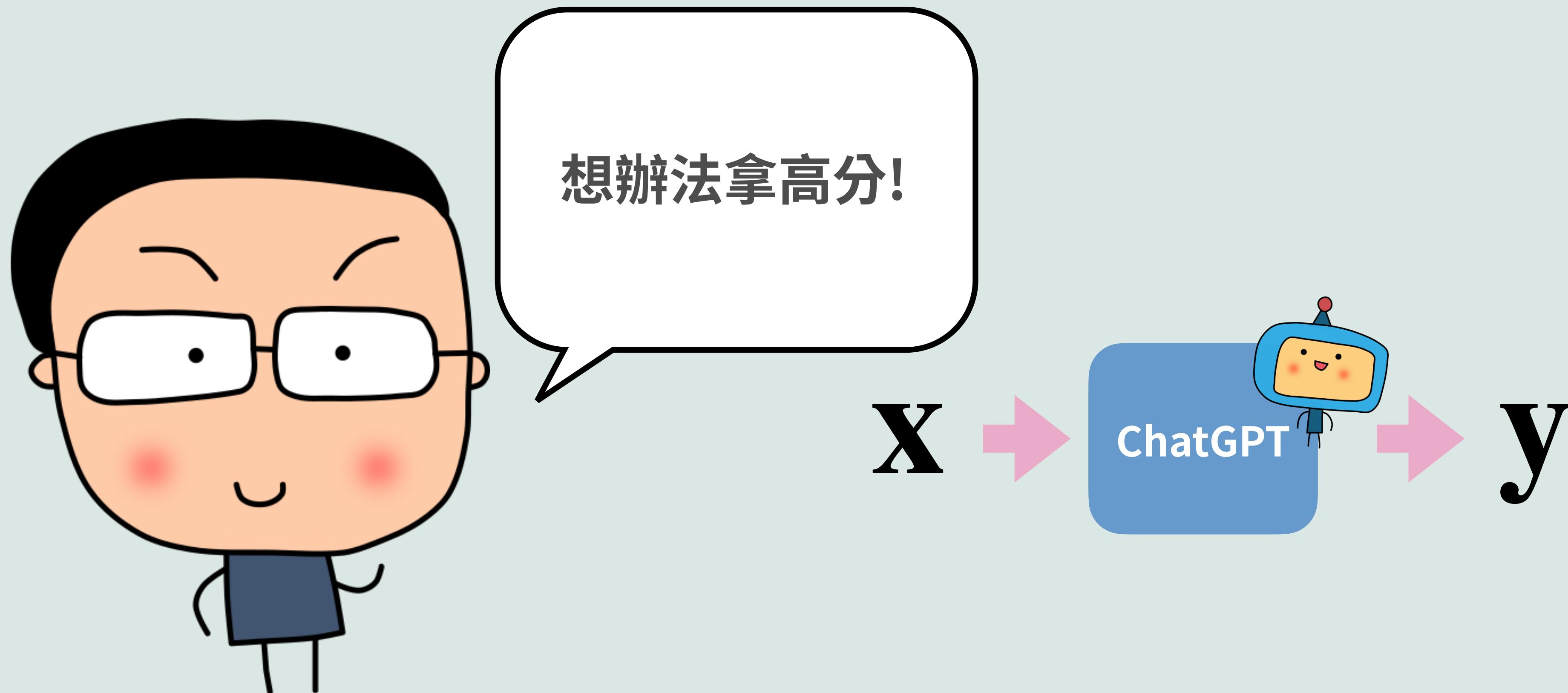
這要比較高分

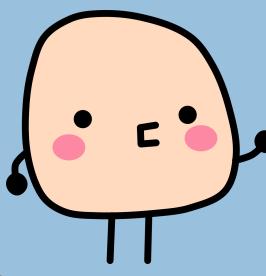
這要比較低分

評分方向對了, sigmoid 值會趨近 1, 取 log 就會接近 0。



## Step 3. 最後, 用強化學習!





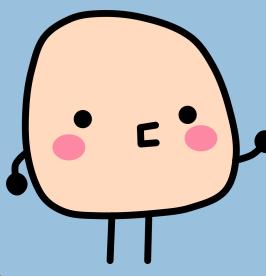
RLHF

這種由人類回饋來  
做強化學習的就叫  
**RLHF**。



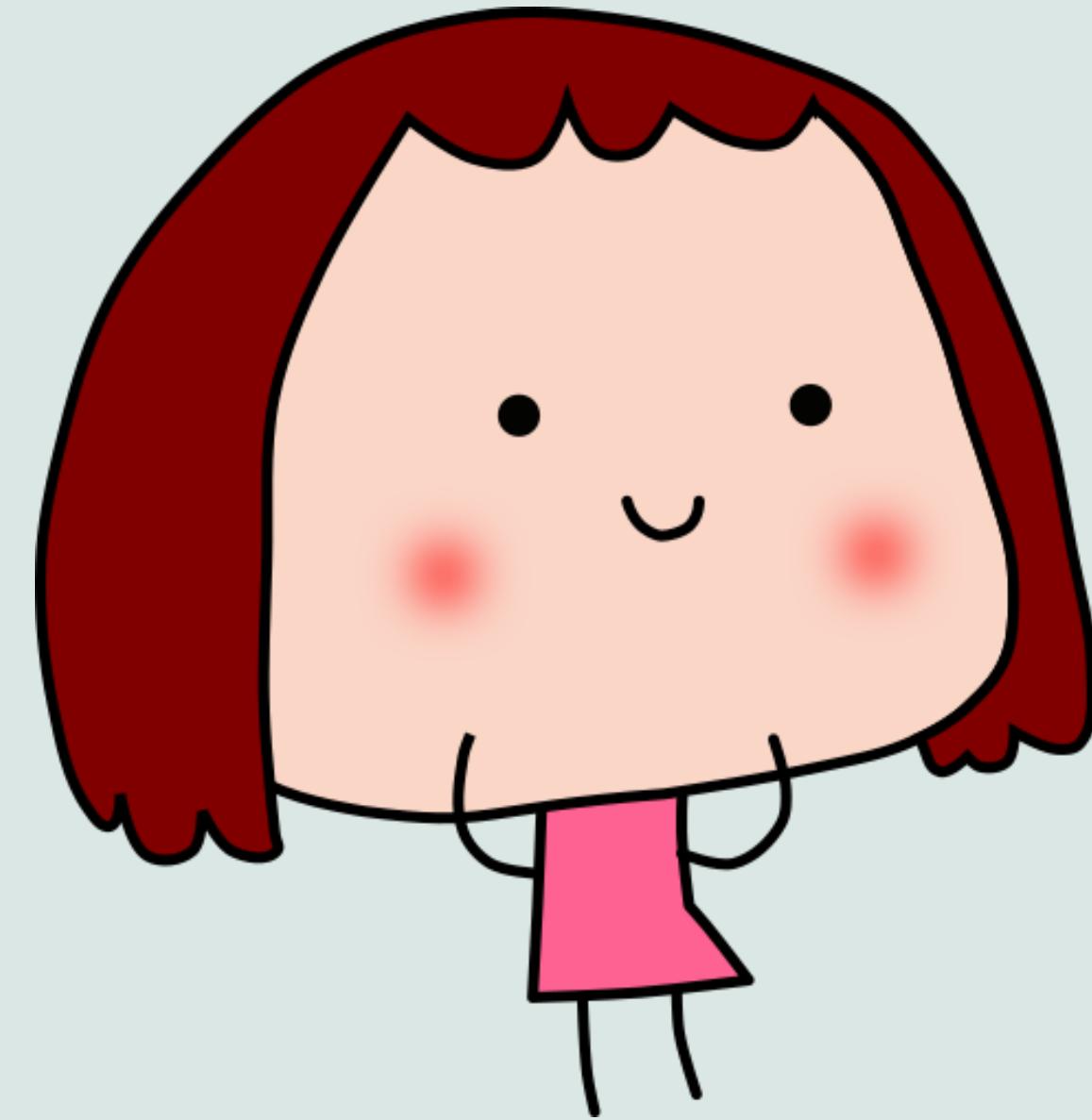
**R**einforcement **L**earning from **H**uman **F**

常用 PPO (Proximal Policy Optimization) 方法訓練。



## 等等, PPO 又是什麼?

簡單的說, 是「溫和版」的 Policy-based learning。

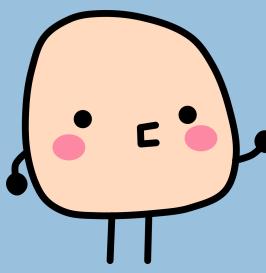


這是避免修正太快,  
毀了整個模型。



04.

## 訓練我們的模型會推理



# DeepSeek 專注在產生 thoughts

請問什麼是 AI Agents 呢？請用最白話的方式，  
但要讓人知道怎麼做自己的 AI Agent。

<think>

嗯，用戶問的是“什麼是AI Agents”，並且希望用台灣習慣的中文來回答，還要求用最白話的方式，同時還要告訴怎麼做自己的AI Agent。

首先，我需要明確AI Agents是什麼。AI Agents通常指的是具備一定自主性和智能的軟件程序，能夠執行特定任務，比如自動化操作、數據處理或提供服務。它們可以根據設定的規則或學習到的模式來決策和行動。

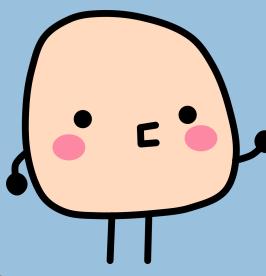
接下來，要用白話解釋，讓用戶容易理解。我可以把AI Agents比作聰明的助手，它們能幫我們完成各種任務，像設置提醒、分析數據或者自動回復信息等。這樣比較貼近生活，容易讓人明白。

⋮

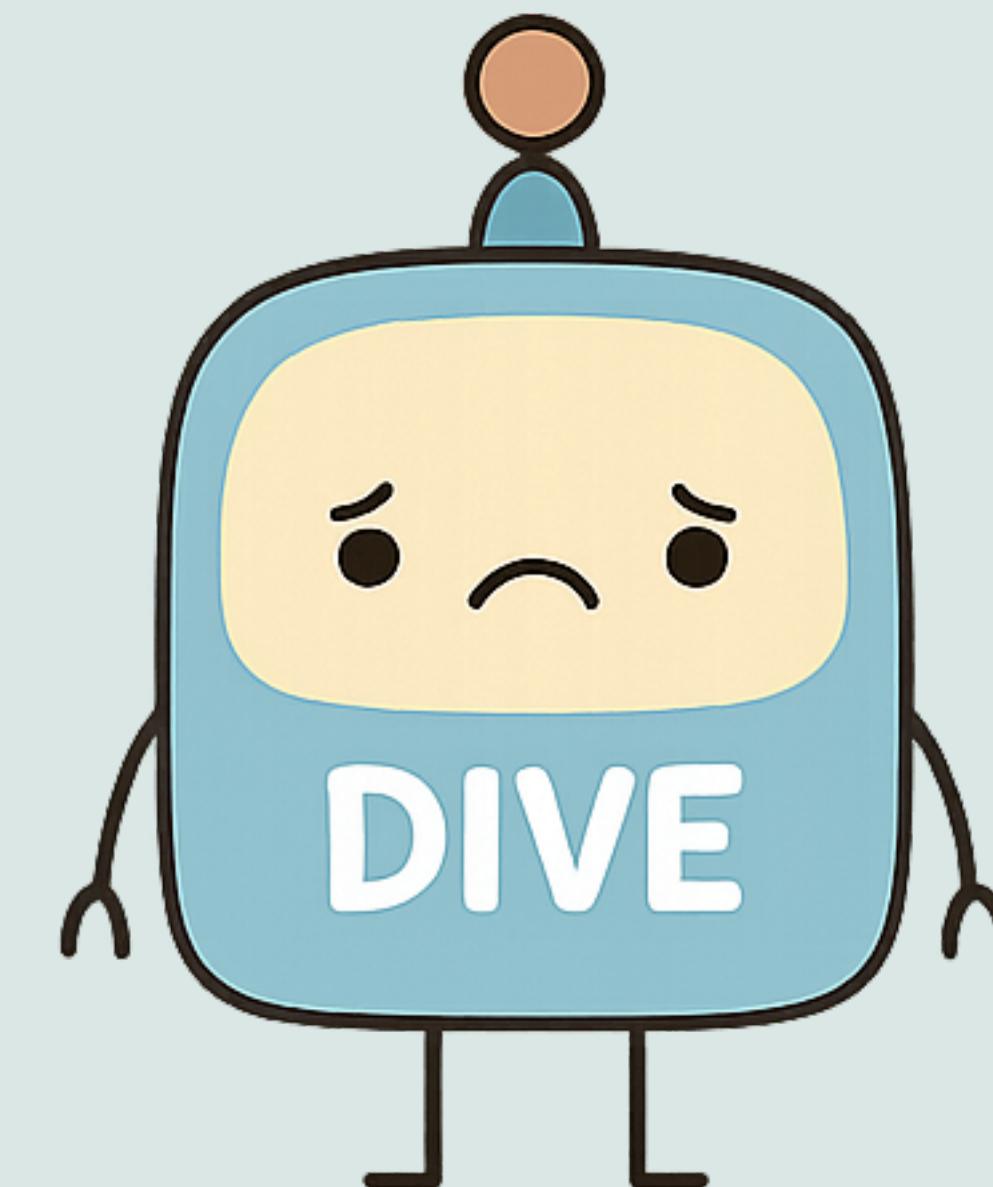
總結一下，我需要用簡單明瞭的語言解釋AI Agents，並提供一個清晰的步驟指南，幫助用戶理解和製作自己的AI Agents。

</think>

DeepSeek 專注在產生很好的 thoughts  
(<think>) 而不是直接回應。

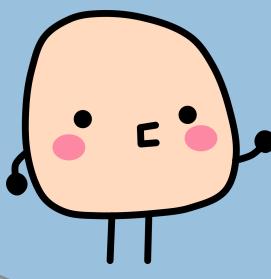


這當然可以用 RLHF

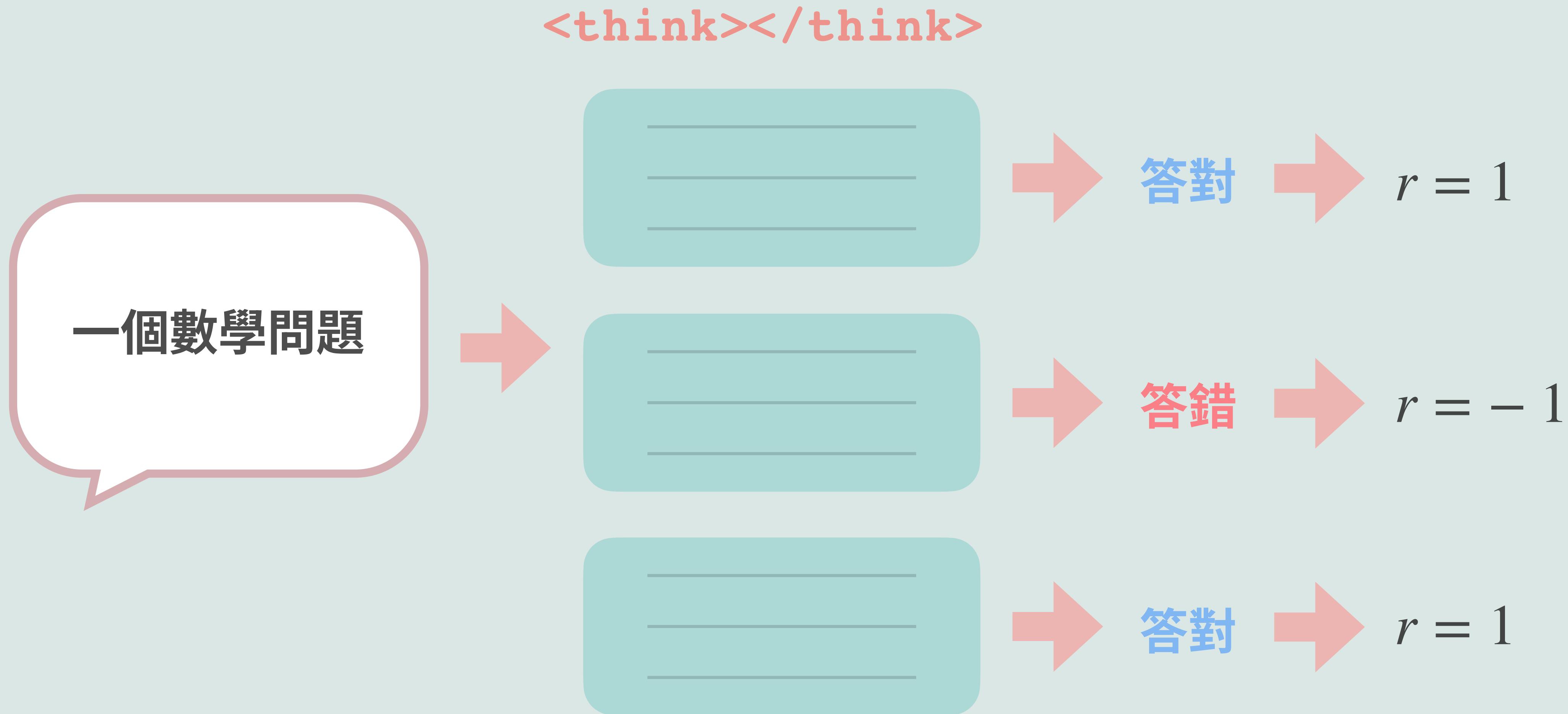


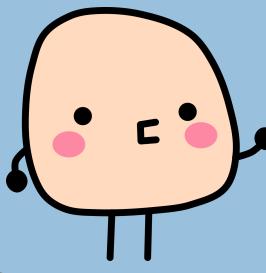
這樣要等很多人  
回應，我生出來  
的東西怎麼樣。

就是不想學  $r_\phi$  可以嗎？



## 「自動」判斷答案好還是不好



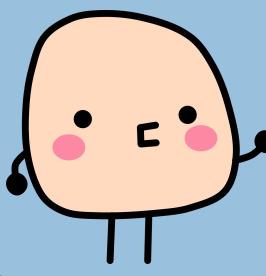


當然，也可以直接用 SFT 學人類的推理



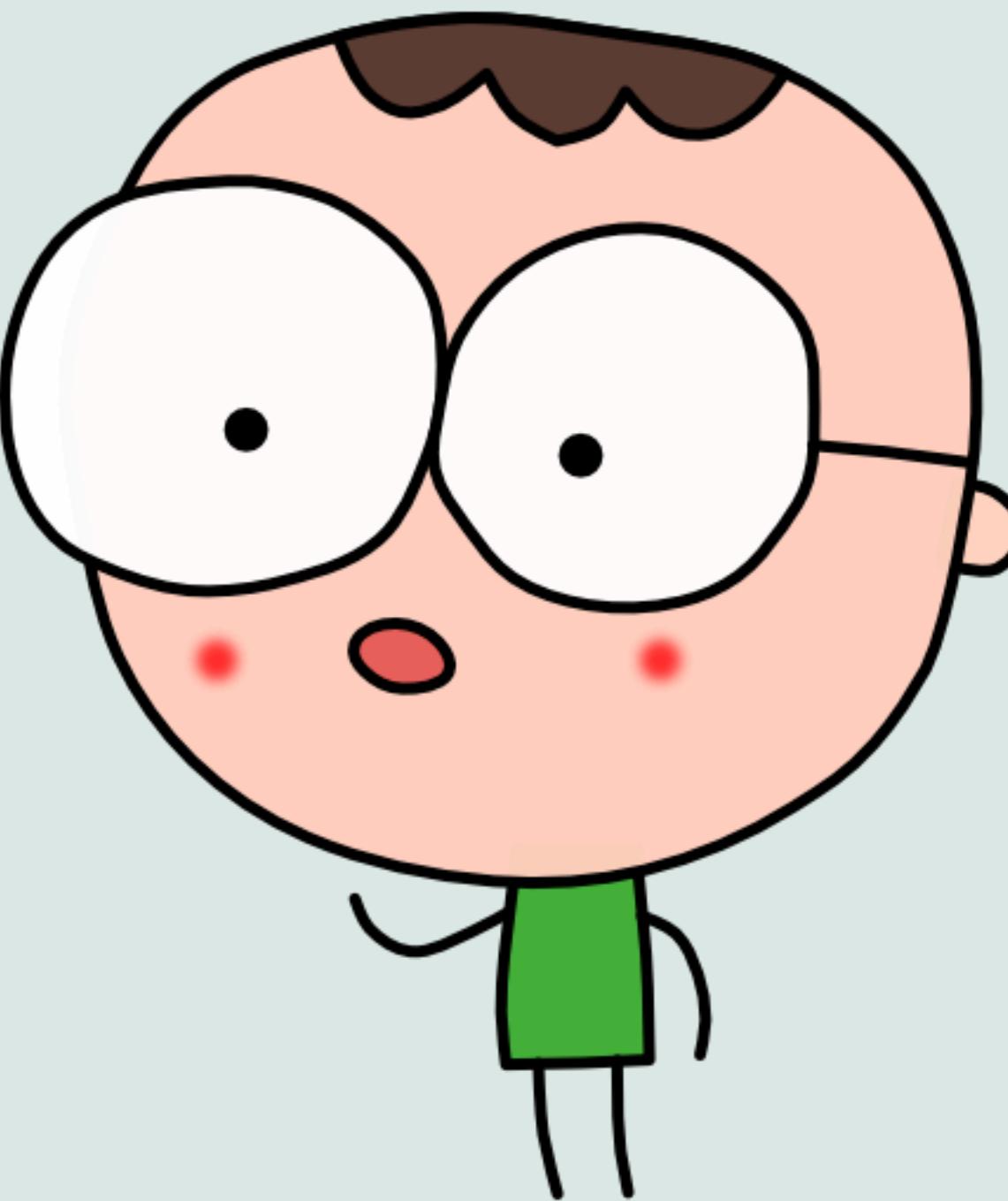
看到某問題，直  
接讓 LLM 學我  
的推理方式。

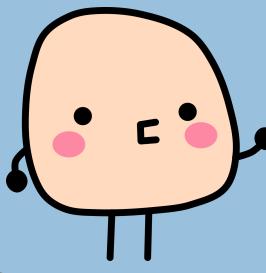
然後我們會發現，不同問題可能要  
用不同推理方式。



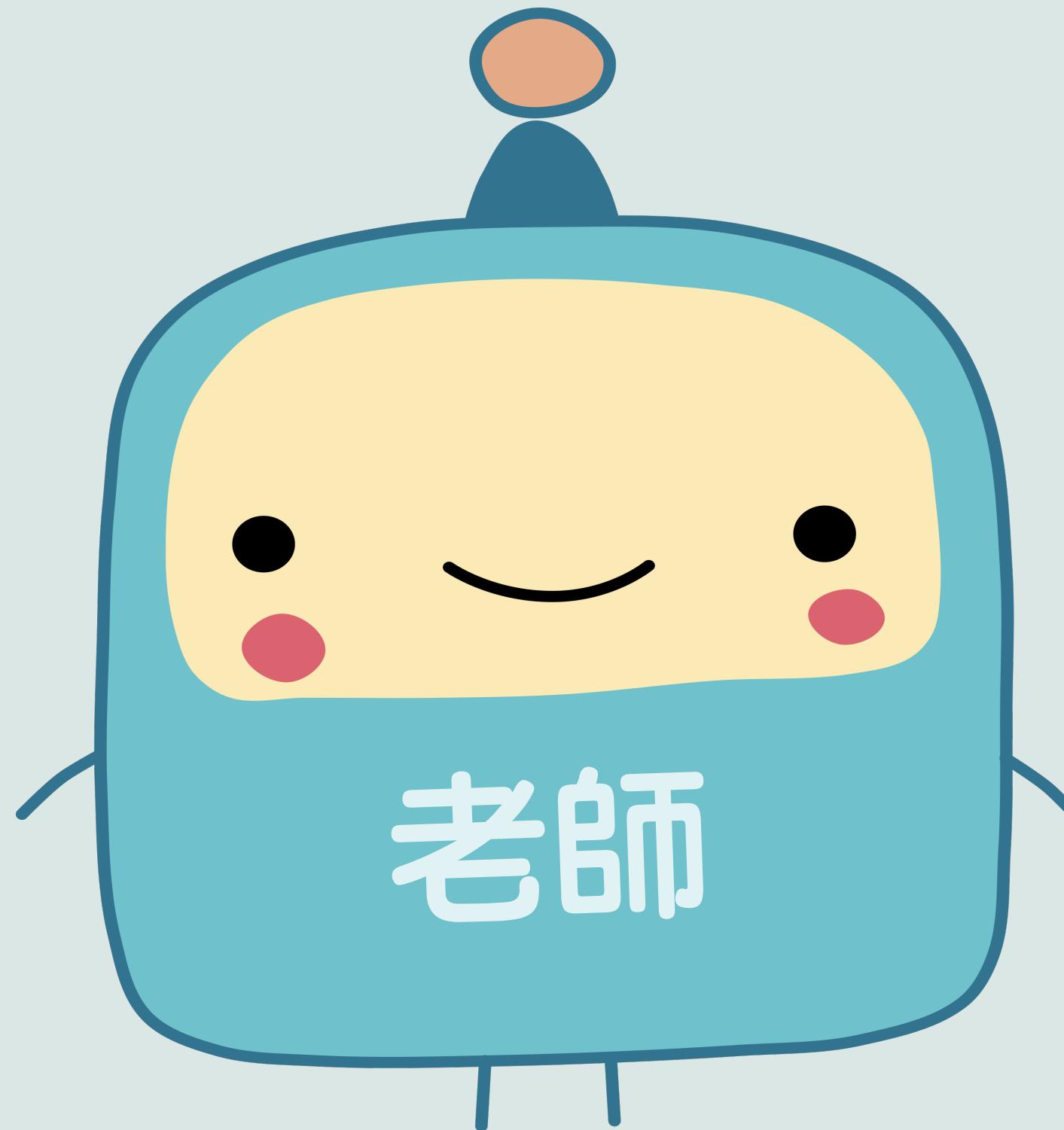
## 推理可能要很大量的資源

常常一個「會推  
理」的模型，都  
非常大..

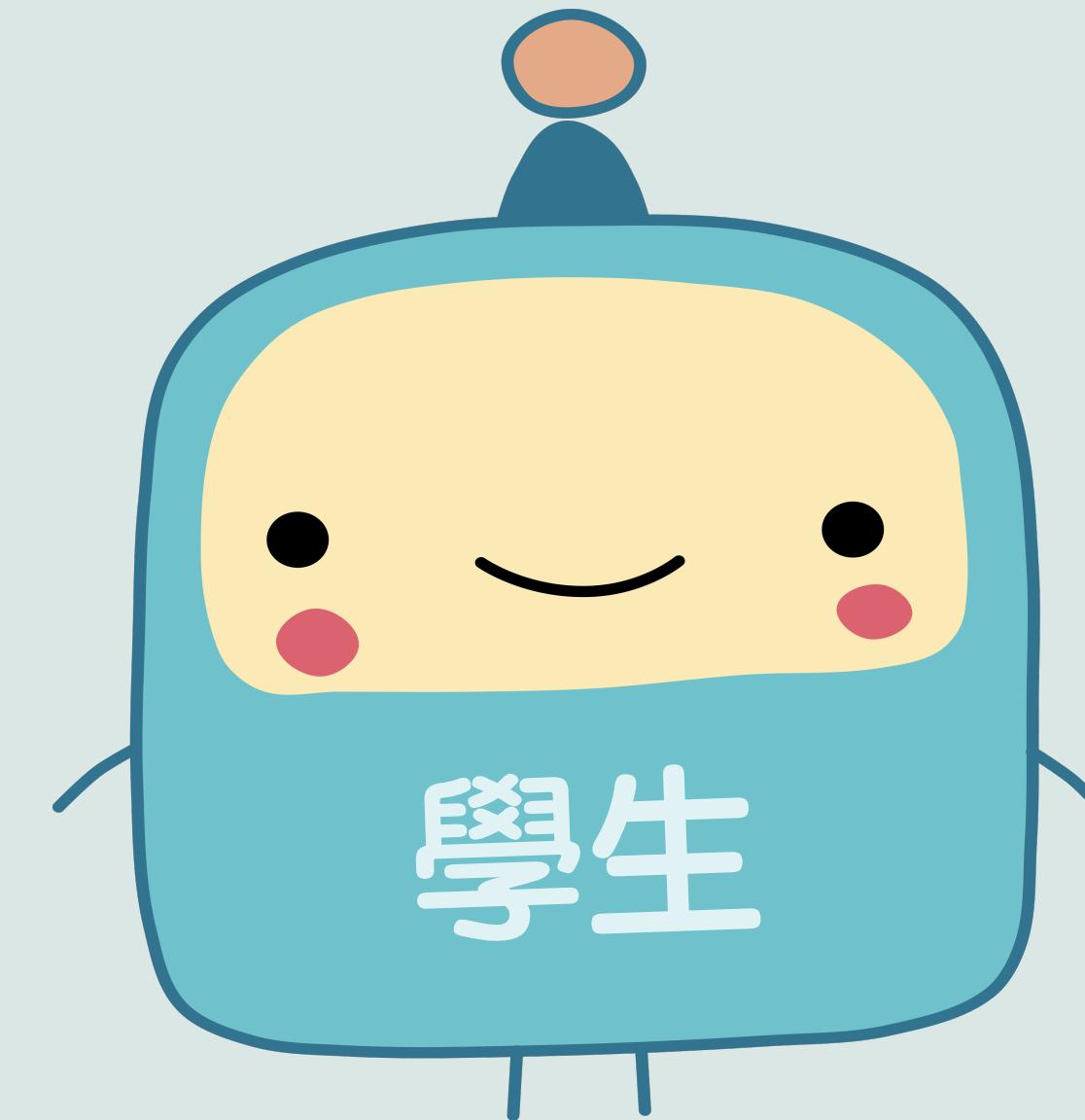


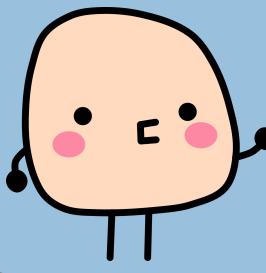


因此，我們常想訓練一個「學生模型」

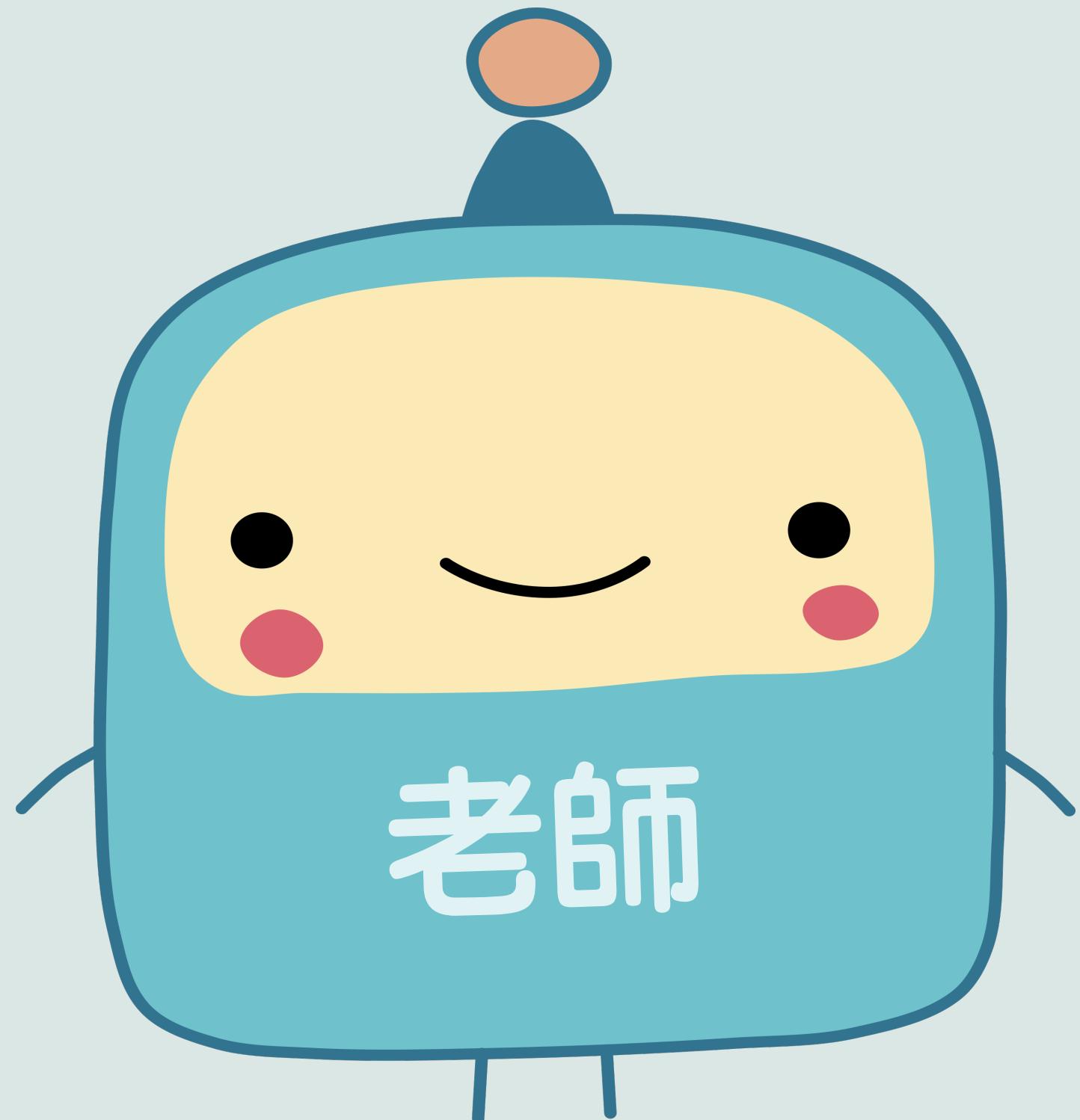


蒸餾





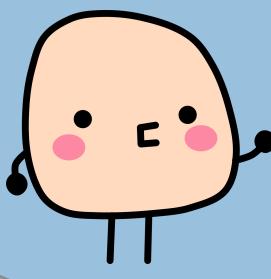
## Reinforcement Distillation (RD)



先全力訓練可能超大、會用超多資源的老師模型。然後用這個模型去訓練學生。



05.  
作業



# 用 Nano Banana Pro 來創作!

The screenshot shows the Gemini tab of the Nano Banana Pro app. A red box highlights the "建立圖像" (Create Image) option, which is associated with the banana icon. Other visible options include "Deep Research", "製作影片 (Veo 3.1)", "Canvas", and "引導式學習". The interface includes a "PRO" button, a user profile icon, and a microphone icon for voice input.

en-Lung

Gemini

PRO

Deep Research

製作影片 (Veo 3.1)

建立圖像

Canvas

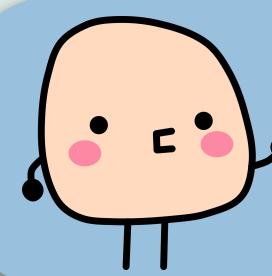
引導式學習

問問

+

思考型

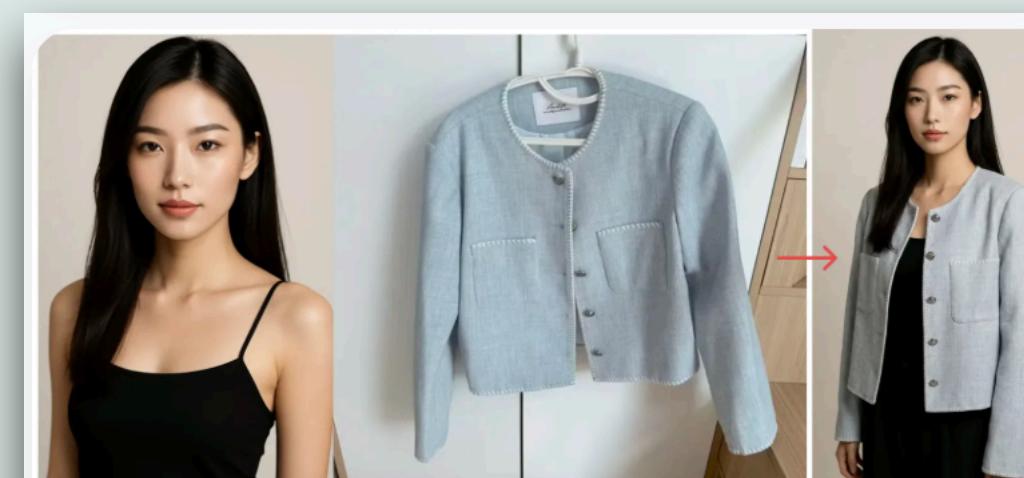
mic



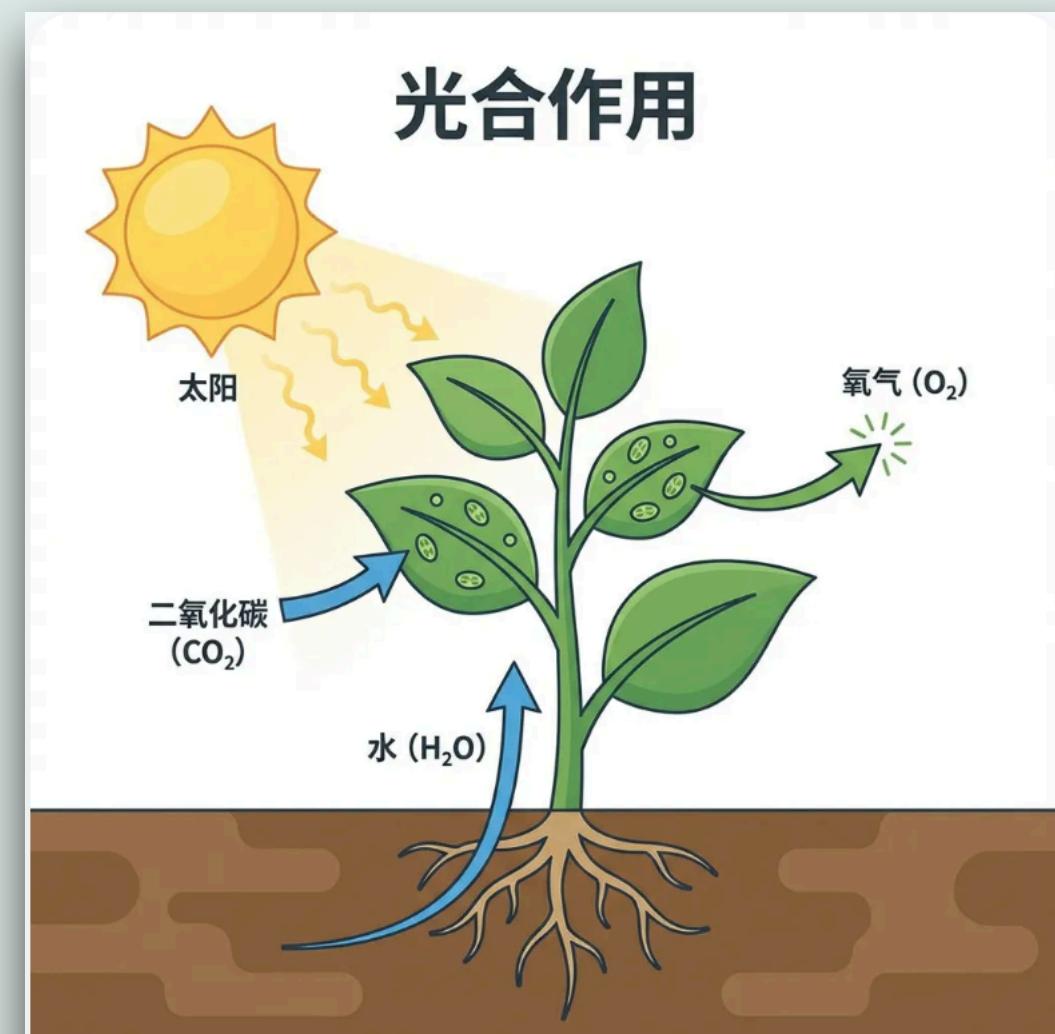
# Nono Banana Prompts 網頁



Based on the uploaded 2D floor plan, generate a professional interior design presentation board in a single image. Layout : The final image should be a collage with one large main image at the top, and several smaller images below it. Content of Each Panel : 1. Main Image (Top) : A wide-angle perspective view of the main living area , showing the connection between the living room and dining area. 2. Small Image (Bottom Left) : A view of the Master Bedroom , focusing on the bed and window. 3. Small Image (Bottom Middle) : A view of the Home Office / Study room . 4. Small Image (Bottom Right) : A 3D top-down floor plan view showing the furniture layout. Overall Style : Apply a consistent Modern Minimalist style with warm oak wood flooring and off-white walls across ALL images. Quality : Photorealistic rendering, soft natural lighting.

[Edit Image](#)[Copy](#)

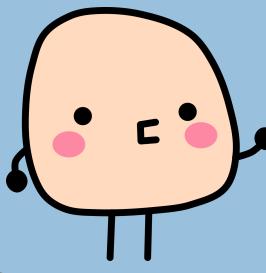
Using Image 1 (the garment) and Image 2 (the model), create a hyper-realistic full-body fashion photo where the model is wearing the garment. Crucial Fit Details : The [T-shirt/Jacket] must drape naturally on the model's body, conforming to their posture and creating realistic folds and wrinkles . High-Fidelity Preservation : Preserve the original fabric texture, color, and any logos from Image 1 with extreme accuracy. Seamless Integration : Blend the garment into Image 2 by perfectly matching the ambient lighting, color temperature, and shadow direction . Photography Style : Clean e-commerce lookbook, shot on a Canon EOS R5 with a 50mm f/1.8 lens for a natural, professional look.

[Edit Image](#)[Copy](#)

Create an educational infographic explaining [Photosynthesis] . Visual Elements : Illustrate the key components: The Sun, a green Plant, Water ( $H_2O$ ) entering roots, Carbon Dioxide ( $CO_2$ ) entering leaves, and Oxygen ( $O_2$ ) being released. Style : Clean, flat vector illustration suitable for a high school science textbook. Use arrows to show the flow of energy and matter. Labels : Label each element clearly in Simplified Chinese .

[Edit Image](#)[Copy](#)

```
{ "prompt_breakdown": { "subject_parameters": { "identity_constraint": "Preserve facial features from reference image", "hair": "Long, ash-brown, textured wolf cut", "expression": "Relaxed, dreamy, looking to the side" }, "apparel": { "top": "Sleeveless cropped black hoodie with high collar", "bottom": "Faded black denim shorts, vintage distressed style, raw hem with frayed fibers", "footwear": "Onitsuka Tiger Mexico 66 sneakers (Yellow and Black)" }, "pose_and_action": { "body_position": "Lying relaxed inside an open car trunk, legs bent and crossed", "arms": "Right arm stretched upward, left hand holding a lit cigarette" } }
```



## 說明



① 可以參考：

<https://nanobananaprompt.org/prompts/>

② 說明想要做什麼樣的作品，創作理念、動機等等。

③ 說明怎麼做出來的，比如參考上面網站、但修改了什麼，寫一個簡單的 prompt、用 LLM 擴增再修改，畫出一張滿意的圖之後再用 Fooocus 擴大等等。