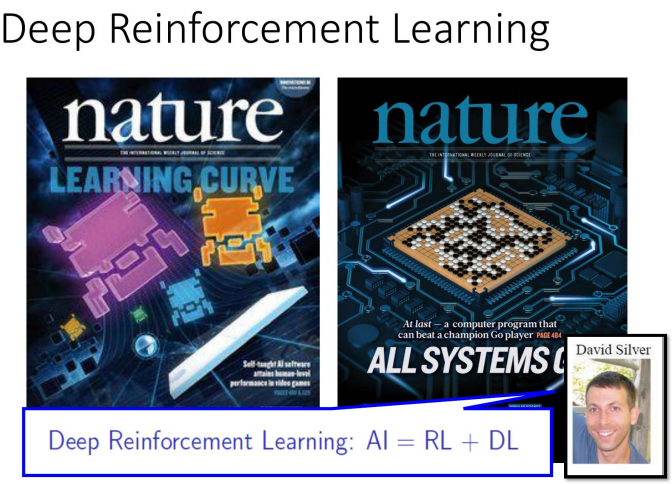
ML Lecture 23: Deep Reinforcement Learning - I

> 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/

Introduction

- 發展歷程
  - 2015 年 2 月
    - Atari 的小遊戲
    - Kreeger 在Nature 上面發了一篇用 Reinforcement Learning (RL) 的方法來玩
    - perfomance 遠超人類

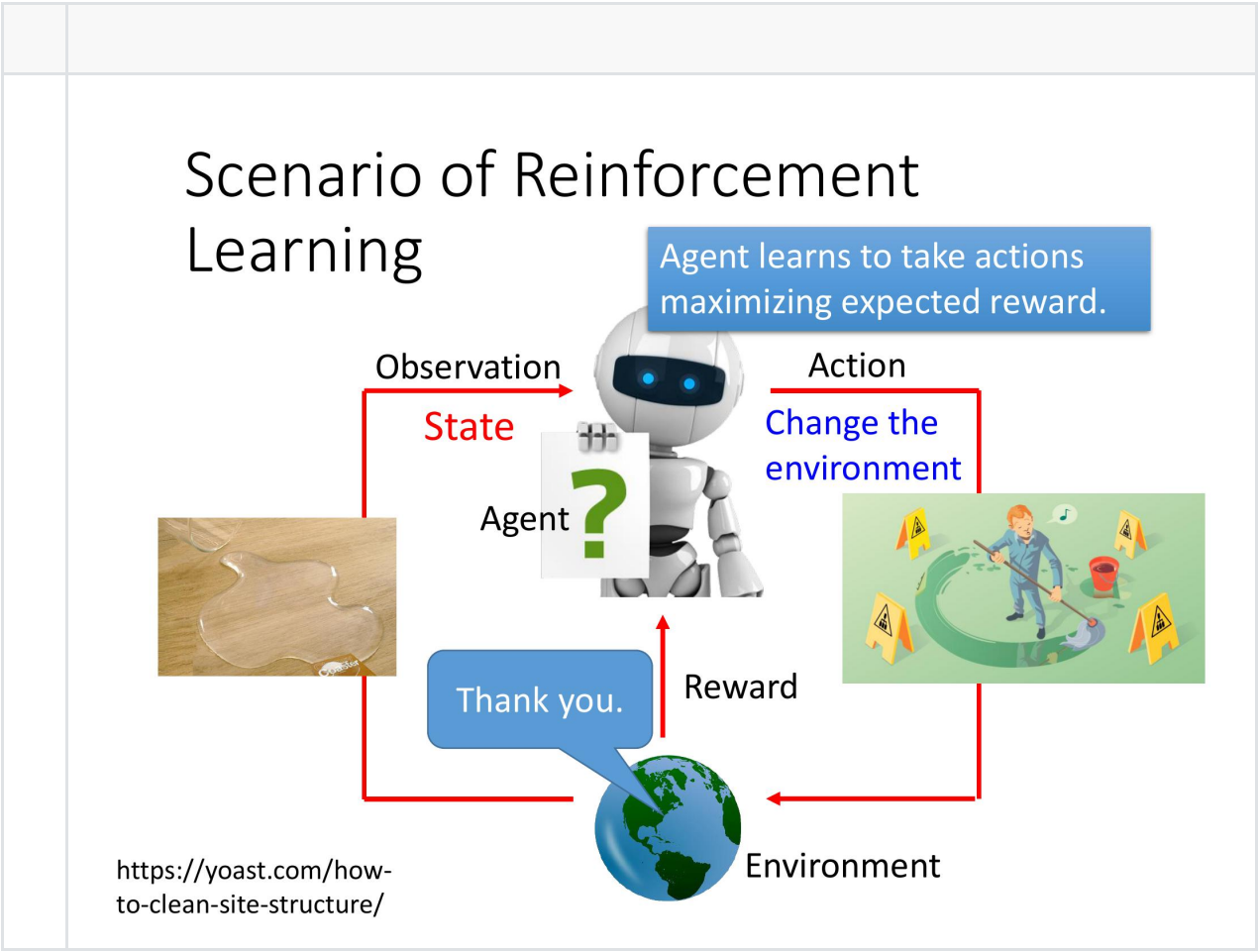


- 2016 年的春天
  - AlphaGo，perfomance 也是遠超人類
- David Silver：AI 就是 Reinforcement Learning 加 Deep learning (也就是Deep Reinforcement Learning)

Reinforcement Learning

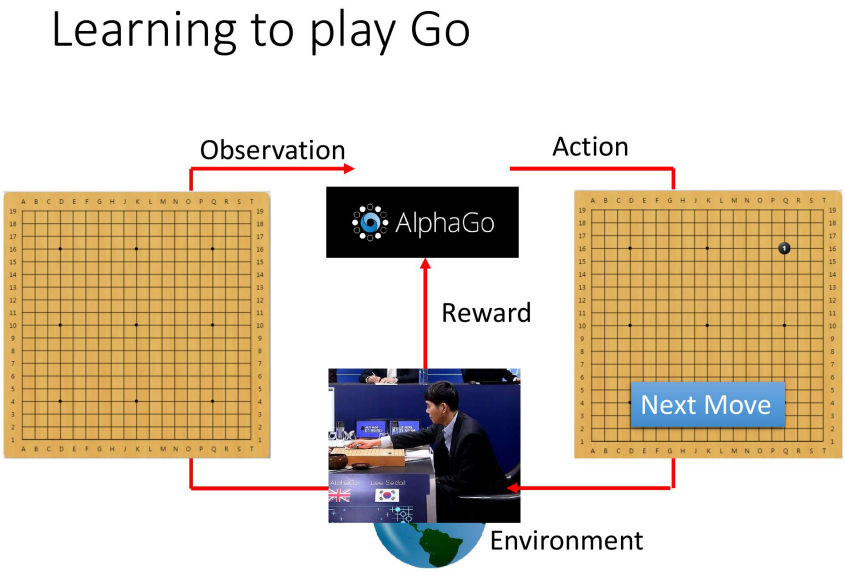
- 模型基本架構：
  - 一個 Agent，一個 Environment
  - Environment => Agent
    - Observation，可以偵測世界種種的變化
    - Observation 又叫做 State，是 **環境的狀態**，也就是 Machine 所看到的東西。
  - Agent => Environment
    - Machine 會做一些事情，做的事情就叫做 Action，Action 會影響環境。
  - Environment => Agent

- 造成影響後會得到 Reward，這 Reward 會告訴它，它的影響是好的，還是不好的。



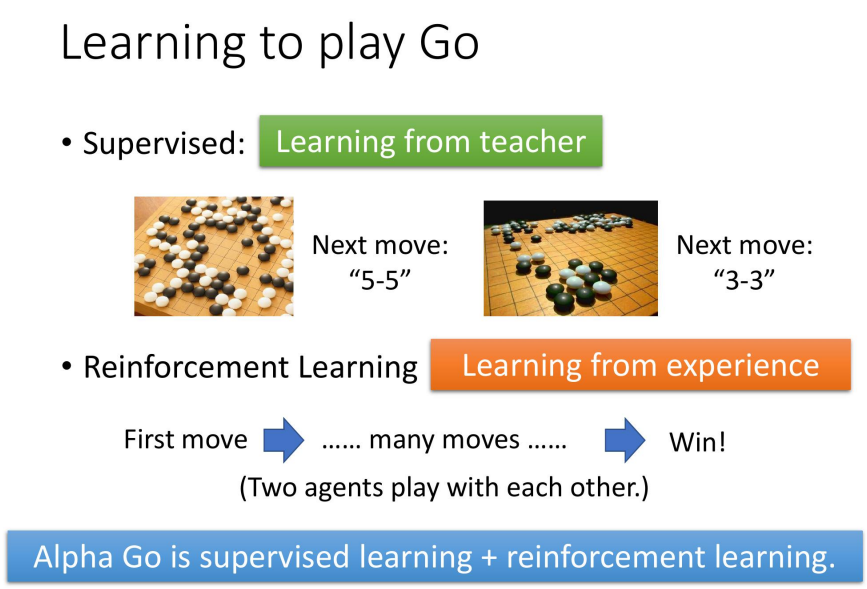
- 根據過去的得到的 positive reward 還有 negative reward，去學習要採取那些 action
  -

舉例：AlphaGo (下圍棋)



- 架構
  - Observation：棋盤，可以用一個 19*19 的 matrix 來描述它
  - Take action：落子的位置
  - Environment：你的對手，你的落子在不同的位置，會影響你的對手的反應
- 困難的原因
  - 多數的時候，得到的 reward 都是零。在你贏了，或是輸了的時候，才會得到 reward。
  - 機器怎麼在只有少數的 action 會得到 reward 的情況下，知道哪些是正確的 action
  -
- machine 要怎麼學習下圍棋?
  - 不斷地找某一個對手一直下一直下，有時候輸、有時候贏
  - 調整它看到的 observation 和 action 之間的關係，讓它得到的 rewards 最大化。

Supervised learning v.s. Un-supervised learning

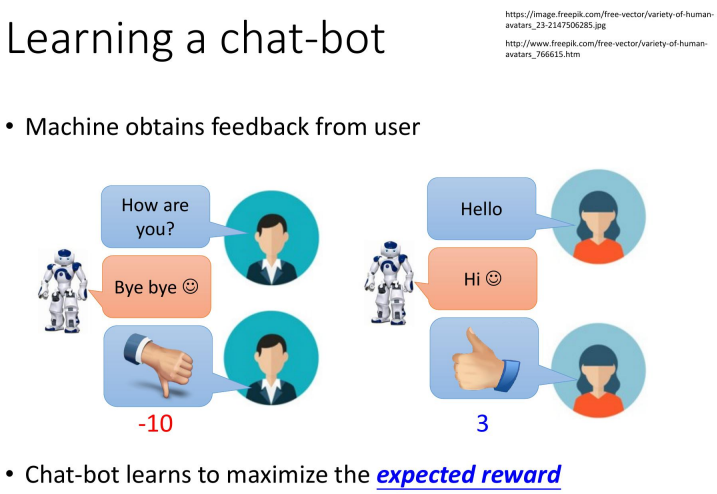


- Supervised learning
  - 概念：
    - 看到什麼盤勢，就落子在哪一個位置
  - 缺點：
    - 不足的地方是，有時候人也不一定知道正確答案
    - 棋譜上面的應對不見得是最 optimal
  - 舉例來說：
    - Supervised learning 就是 machine 從一個老師那邊學，老師會告訴它說，每次看到這樣子的盤勢，你要下在甚麼樣的位置
- Reinforcement Learning (Un-supervised)
  - 舉例來說：
    - 讓機器呢，就不管它，它就找某一個人去跟它下圍棋，從經驗中學習。

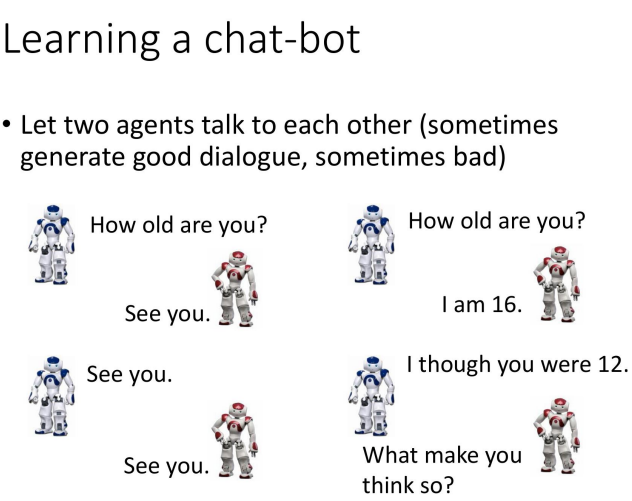
        - 在這幾百步裡面，甚麼樣的下法是好的，哪幾步是不好的，它要自己想辦法去知道。
    - 缺點：
        - 需要大量的 training 的 examples（可能要下三千萬盤以後，它才能夠變得很厲害）
- AlphaGo 的解法
    - 先做 Supervised learning，訓練一批學得不錯的machine
    - 再讓任兩個 machine它們自己互下，去做 Reinforcement Learning。

Chat-bot

- chat-bot 是怎麼做的（Supervised）：
    - learn一個sequence-to-sequence model
    - input 是一句話，output 就是機器人回答



- Reinforcement Learning 的 learn 法
    - 概念：
        - 讓 machine 胡亂去跟人講話，講一講以後，人最後就生氣了
        - Machine 要自己去想辦法發覺說，它某句話可能講得不太好
        - 用AlphaGo一樣的train法，任兩個 Agent，讓它們互講。



- 缺點：
  - 對話完以後，沒有人去告訴它說，講的好還是不好（不像圍棋有明確的輸贏）
  - 尚待克服
- 宏毅老師預言，接下來會有人用 GAN 來 learn chat-bot
  - discriminator 看真正的人的對話 和 那兩個 machine 的對話，判斷這兩個 Agent 的對話像不像人
  - 用 discriminator 自動 learn 出給 reward 的方式

更多其他應用

- 特別適合的應用就是:
  - 人也不知道怎麼做，你人不知道怎麼做 就代表著 **沒有 labeled data**

例子：

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
  - 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

- Interactive retrieval：
  - 有一個搜尋系統，Machine 跟它說想要找尋一個跟 US President 有關的事情
  - Machine 可能覺得說，這 US President 太廢了，反問它一個問題（你要找的是不是跟選舉有關的事情等等），要求它 modify
  - 用 Reinforcement Learning 的方式，讓 machine 學說，問甚麼樣的問題，可以得到最高的 reward。
    - 最後搜尋的結果，使用者覺得越好，就是 reward 越高。
    - 每問一個問題，對人來說，就是 extra 的 effort，可以定義為 negative reward
- 無人機，無人車
- 幫 Google 的 server 節電
- 造句子：summarization or translation
  - 因為 translation 有很多種，machine 產生出來的句子，可能是好的，卻跟答案不一樣。

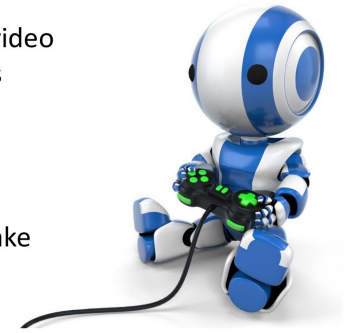
Video Game (打電玩)

Example: Playing Video Game

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

Machine learns to play video games as human players

- What machine observes is pixels
- Machine learns to take proper action itself

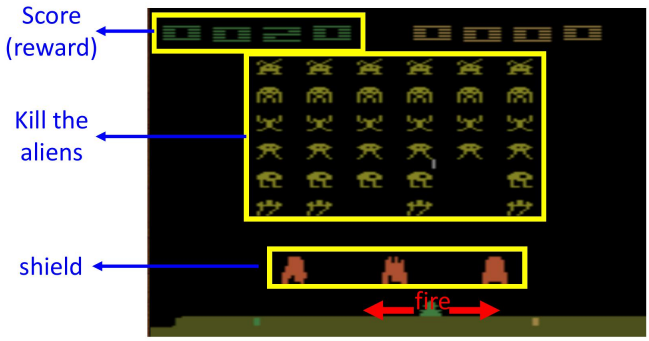


- 現成 environment (by Open AI)
  - Gym
    - 比較舊
  - Universe
    - 很多 3D 的遊戲
- 那些已經內建的 AI 遊戲不同，要讓 machine 用 Reinforcement Learning 的方法，去學玩遊戲
  - machine 學怎麼玩這個遊戲，其實是跟人一樣的
  - 看到的東西就是螢幕畫面 (pixel)，並不是從那個程式裡面去擷取甚麼東西出來
  - 看到這個畫面，要 take 哪個 action (要做甚麼事情)
- 舉例：
  - Space invader (小蜜蜂)
  -

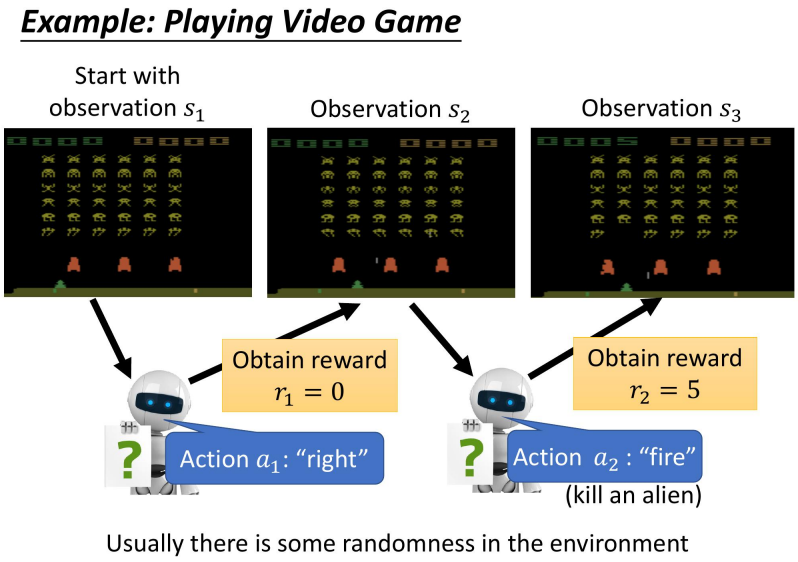
Example: Playing Video Game

- Space invader

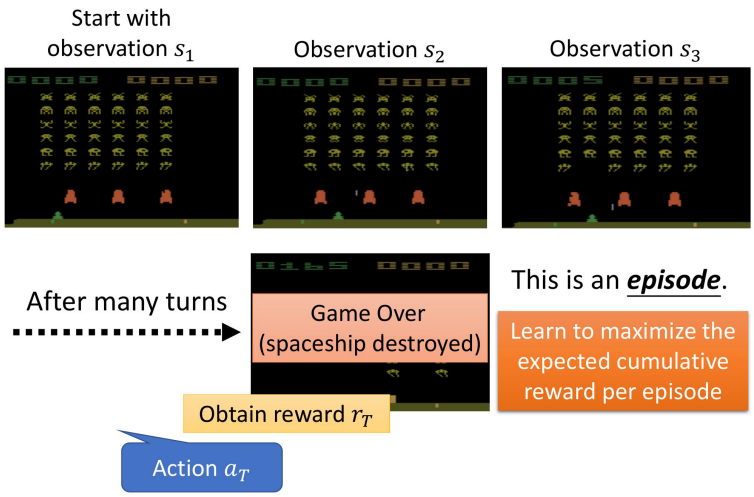
Termination: all the aliens are killed, or your spaceship is destroyed.



  - 可以 take 的 action 有三個，就是左、右移動，跟開火
  - Scenario



- 首先machine 會看到一個 observation (s1) 以後，它要決定要 take 哪一個 action
- 現在只有三個 action 可以選擇，比如說它決定要 "往右移"，會得到一個 reward
- reward 是左上角的這個分數，那往右移不會得到任何的 reward
- take 完這個 action 以後，它的 action 會影響了環境，machine 看到的 observation 就不一樣 (變成s2)
- 因為它自己往右移了，但也有環境的隨機變化（比如這些外星人，也會稍微往下移動一點，或是突然多一個子彈）
- 然後 machine 看到 s2 以後他要決定 take 哪一個 action，決定要射擊
- 假設他成功殺了一隻外星人，他就會得到一個 reward
- 他看到的 observation 就變少了一隻外星人，這個是第三個 observation (s3)
- 這個 process 就一直進行下去，直到某一天在第 T 個回合的時候，他被殺死了，就進入 **terminal state**
- 遊戲的開始到結束叫做一個 **episode**



- 對 machine 來說它要做的事情就是要不斷去玩這個遊戲，要學習在怎麼在一個 episode 裡面 maximize 它可以得到的 reward

---

Reinforcement Learning 的難點

Properties of Reinforcement Learning

- **Reward delay**
  - 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
- Agent's actions **affect the subsequent data it receives**
  - E.g. Exploration

- Reward delay (reward 的出現往往會有 delay)
  - 比如說在 Space Invader 裡面，只有開火這件事情才可能得到 reward
    - 但是如果 machine 只知道開火以後就得到 reward，它最後 learn 出來的結果它只會瘋狂開火
    - 對它來說往左移、往右移沒有任何 reward 它不想做
    - 實際上往左移、往右移這些 moving，對開火能不能夠得到 reward 這件事情是有很關鍵的影響
  - machine 需要有這種遠見 (vision)，才能夠把這些電玩完好
  - 就跟下圍棋也是一樣，有時候短期的犧牲最後可以換來最後比較好的結果
- Agent 採取的行為，會影響它之後所看到的東西
  - agent 要學會去探索這個世界
  - 比如說在 Space Invader，如果 agent 只會往左移、往右移，從來不開火，他就永遠不知道開火可以得到 reward
    - 或是它從來沒有試著去擊殺最上面這個紫色的母艦，它就永遠不知道擊殺那個東西可以得到很高的 reward
  - 要讓 machine 知道要去 **explore** 它沒有做過的行為

Reinforcement Learning 的方法

- Markov Decision Process
  - 暫不提
- Deep Q Network
  - 已經被A3C完勝
- A3C

- gym 裡面最強的那些 agent 都是用這個model

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

- 把 actor 跟 critic 加起來的方法，叫做 Actor-Critic
- 現在最強的方法就是 Asynchronous Advantage Actor-Critic，縮寫叫 A3C
- 問題：最強的 Alpha Go 是用甚麼方法
  - 各種方法大雜燴
  - 有 Policy Gradient 方法、Policy-based 方法
  - 有 Value-based 的方法
  - 有 Model-based 的方法（未提到，指一個 Monte Carlo tree search 那一段）