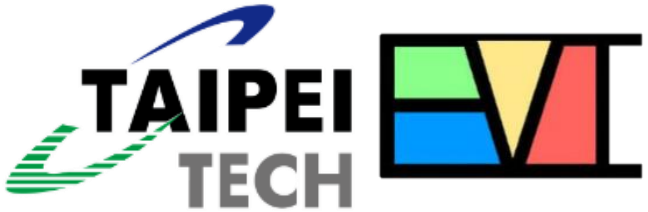


Embedded Vision Intelligent Laboratory

嵌入式智慧影像分析與實境界面
Fall 2021

Instructor：Yen-Lin Chen(陳彥霖), Ph.D.

Professor

Dept. Computer Science and Information Engineering

National Taipei University of Technology

Lecture 11

強化學習與DQN介紹

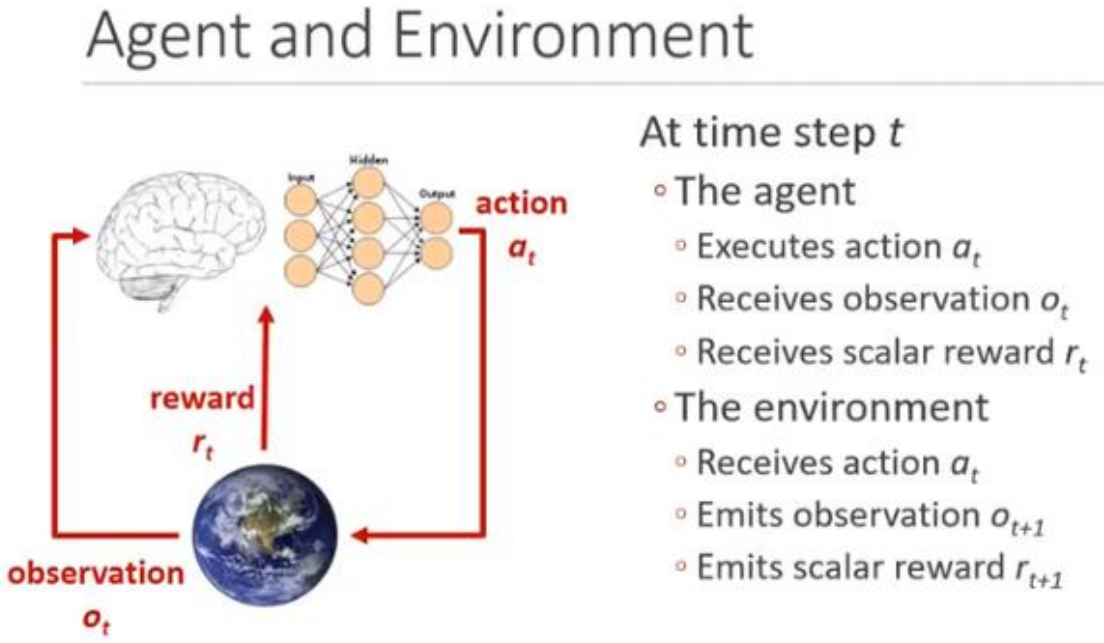
Reinforcement Learning
(強化學習)

Reinforcement Learning

- 自從 Alpha Go擊敗人類後開始，大家開始重視增強式學習演算法的能力，沒想到能透過一個 Deep learning 、 Machine learning的演算法，能擊敗最強的圍棋手。
- **強化學習**（英語： Reinforcement learning）是機器學習的一種方法，從現在的環境來決定行為，是一種互動式的學習過程。
- 強化學習其實就是訓練一個AI 可以通過每一次的錯誤來學習，就跟我們小時候學騎腳踏車一樣，一開始學的時候會一直跌倒，然後經過幾次的失敗後，我們就可以上手也不會跌倒了。

Reinforcement Learning

- Agent部份
  - 簡單來說，就是訓練出一個agent可以去適應environment
  - 會將environment環境每一個時間點的observation的集合當作環境的 state
  - 從環境的state跟reward再去選擇一個最好的action，稱為policy
- Environment部份
  - 簡單來說，就是一個環境在不同的state下會有不同的情況，並將這些情況告訴agent ，讓agent可以去學習。
  - 會接收Agent執行的action，並吐出reward跟observation給agent。
- 其目的就是找到一個最好的Policy，可以讓reward最多



MDP (Markov Decision Process)

- 因為一開始並不知道環境的狀態是怎麼樣，所以只能從以前所經歷的observation，action，reward跟現在所得到的observation, reward來去當作現在的狀態

$$s_t = f(o_1, r_1, a_1, \ldots, a_{t-1}, o_t, r_t)$$

- 當要去估計下一個狀態S_{t+1}，如果把S_1～ S_t都考慮進去，模型會變得非常巨大。Markov提出假說：“未來只取決於當前”，此假說可以將模型縮小

$$P(s_{t+1} \mid s_t) = P(s_{t+1} \mid s_1, \ldots, s_t)$$

- 強化學習想像成是MDP的一種模型，因為我們從現在的狀態來知道未來的狀態，未來知道了，相對的，我們要找到最好的動作也變得有可能了

Value function 價值函數

- 運用當前狀態對未來reward的期望來定義目前狀態跟策略的好與壞
- 狀態對未來每一個時間點的reward相加，λ是discount factor，越是未來所給的reward影響是越來越小的，當下的reward是最大的。

$$G_t = R_{t+1} + \lambda R_{t+2} + \ldots = \sum_{k=0}^{\infty} \lambda^k R_{t+k+1}$$

- 使用一個Value function來表示這個狀態未來的潛在價值

$$v(s) = \mathbb{E}[G_t | S_t = s]$$

Value function 價值函數

- Bellman方程式可以看出，value function跟下一步的reward還有下一個狀態的Value function來疊代求解

$$
\begin{aligned}
v(s) &= \mathbb{E}[G_t | S_t = s] \\
&= \mathbb{E}[R_{t+1} + \lambda R_{t+2} + \lambda^2 R_{t+3} + \ldots | S_t = s] \\
&= \mathbb{E}[R_{t+1} + \lambda (R_{t+2} + \lambda R_{t+3} + \ldots) | S_t = s] \\
&= \mathbb{E}[R_{t+1} + \lambda G_{t+1} | S_t = s] \\
&= \mathbb{E}[R_{t+1} + \lambda v(S_{t+1}) | S_t = s]
\end{aligned}
$$

Value function 價值函數

- 把Action(動作)給考慮進來，我們關心在這個State中所做動作的價值跟reward，可以定義出Action-Value function

$$Q^{\pi}(s,a) = \mathbb{E}[r_{t+1} + \lambda r_{t+2} + \lambda^2 r_{t+3} + \ldots | s, a]$$

$$= \mathbb{E}_{s'}[r + \lambda Q^{\pi}(s', a') | s, a]$$

- Policy是從state跟reward中找到一個最好的action，相對的，Policy的決定就會跟Action-Value function有關，所以上面的公式所代表的就是在策略π下的動作價值函數。

Optimal value function 優化價值函數

- 由於強化學習最重要的點在於找到一個最好的Policy，讓Reward可以最多，所以在這裡需要找到最優的Value function，Policy也會是最優的，這就是Value-based的方法
- 我們可以寫出以下公式：**最好的動作價值函數就是在所有策略下的動作價值函數的最大值**，然後把bellman function代進去，就可以找到一個公式來代表最優的動作價值函數

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

$$Q^*(s,a) = \mathbb{E}_{s'}[r + \lambda \max_{a'} Q^*(s',a')|s,a]$$

Q-learning

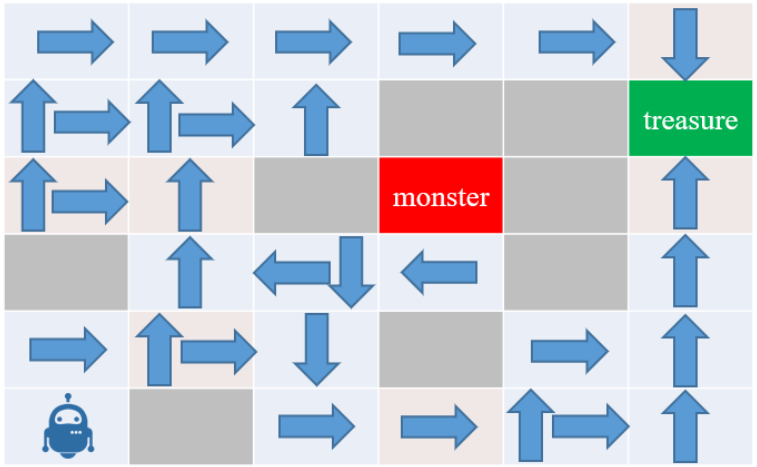
- Q-learning的想法是由Value Iteration所得到的
- 利用新得到的reward和原本的Q值來更新現在的Q值，這就是value iteration
- Q-learning的更新公式

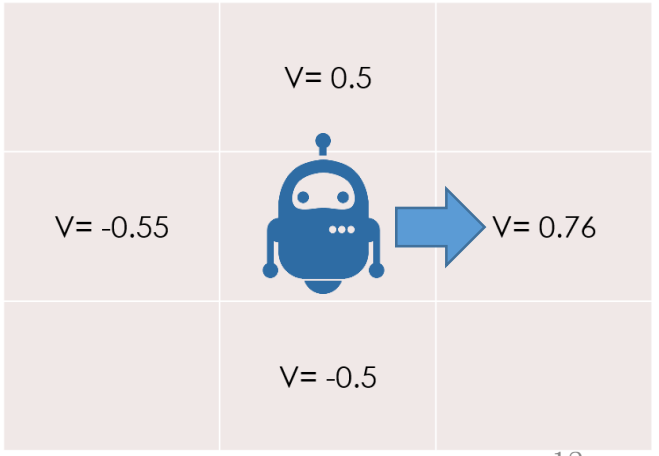
$$Q^*(s,a) = Q(s,a) + \alpha(\underbrace{r + \gamma \max_{a'} Q(s',a')}_{\text{Target Q}} - Q(s,a))$$

- 中間的bellman function(target)，一開始的值可能是錯的，但經過一次又一次的疊代，target Q也會越來越精準，如果夠長的時間，value-function就可以優化為最優的價值函數，這就是Q-learning。

Exploration and Exploitation探索跟利用

- 知道Q-learning的更新方法，但是還需要Policy去決定動作，才可以知道下一個執行的動作要是什麼，以下就有二種方法去選擇Policy：
  - Exploration(探索)：隨機生成一個動作
  - Exploitation(利用)：我們利用目前的所有的Q值來找出一個最好的動作，這個policy稱為greedy policy，舉例來說，就是假設現在我要選擇我要走的方向(上下左右)，向上的Q值0.5，向下的Q值-0.5，向左的Q值-0.55，向右的Q值0.76，因此我們取最大的Q值，所以就向右走。
- 將這二種方法結合在一起，就是所謂的ϵ-greedy policy



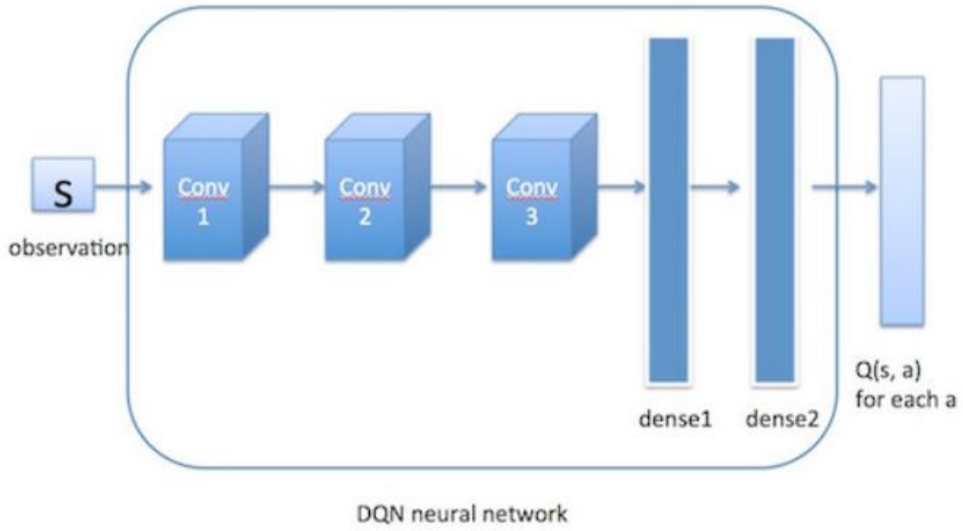


Deep Q-learning(DQN)

Deep Q-learning(DQN)

- Q-learning如果是用在狀態比較多的情況(像是Atari game那種有非常多的情況)，會導致Q-learning無法把所有狀態的Q值都計算出來
- DQN使用RL+DL的學習方法，設計出一個神經網路，只需要將狀態輸入，輸出的是每個動作的Q值，以解決記憶體爆炸的情況，這個就是**Value function approximation**

$$Q(s) \approx f(s, w)$$



Deep Q-learning(DQN)

- 神經網路的訓練就是要找一個最好的參數來優化loss function，DQN將State當作input，Target Q當作label，就可以定義出loss function進行training

$$Q^*(s,a) = Q(s,a) + \alpha(\underbrace{r + \gamma \max_{a'} Q(s',a')}_{\text{Target Q}} - Q(s,a))$$

Q-learning更新公式

$$L(w) = \mathbb{E}[(\underbrace{r + \gamma \max_{a'} Q(s',a',w)}_{Target} - Q(s,a,w))^2]$$

Loss function

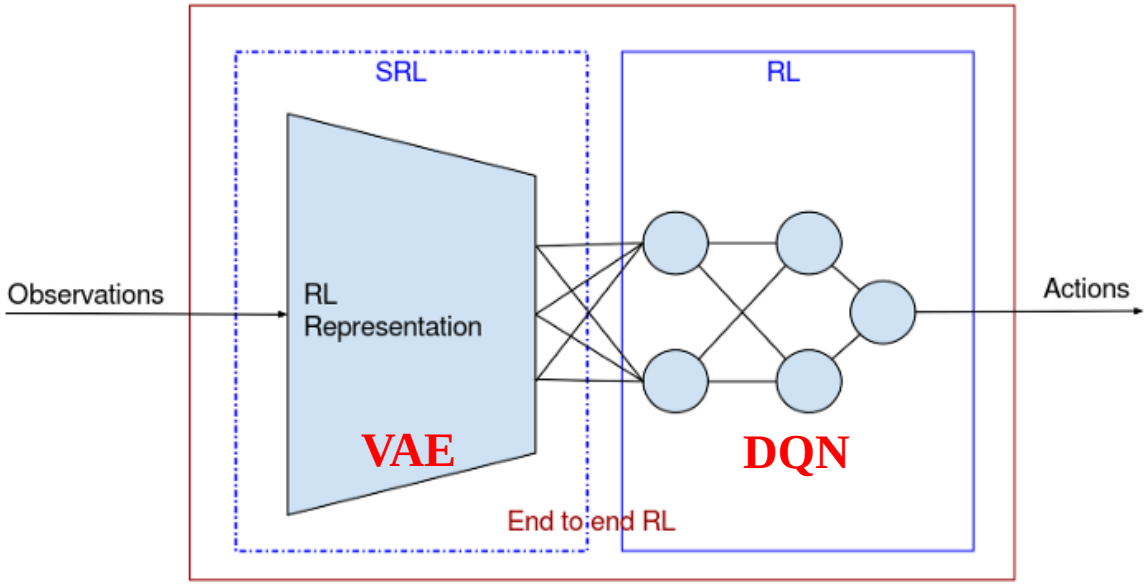
Reinforcement Learning 與自駕車應用

自駕車競賽

- 在目前的自駕車競賽中，主流有幾種作法:
  - 使用supervised learning方式進行model training
  - 使用reinforcement learning方式進行學習
- 使用Supervised Learning方式進行道路辨識，需要先進行大量的人工標記並進行data augmentation後，才能將data交給model進行訓練，並且不一定可以得到很好的效果。
- 使用reinforcement learning方式進行道路辨識，是透過與環境資料進行比對調整行為，不需要使用人工標記的數據，也可以得到較佳的效果。

VAE Feature Extractor

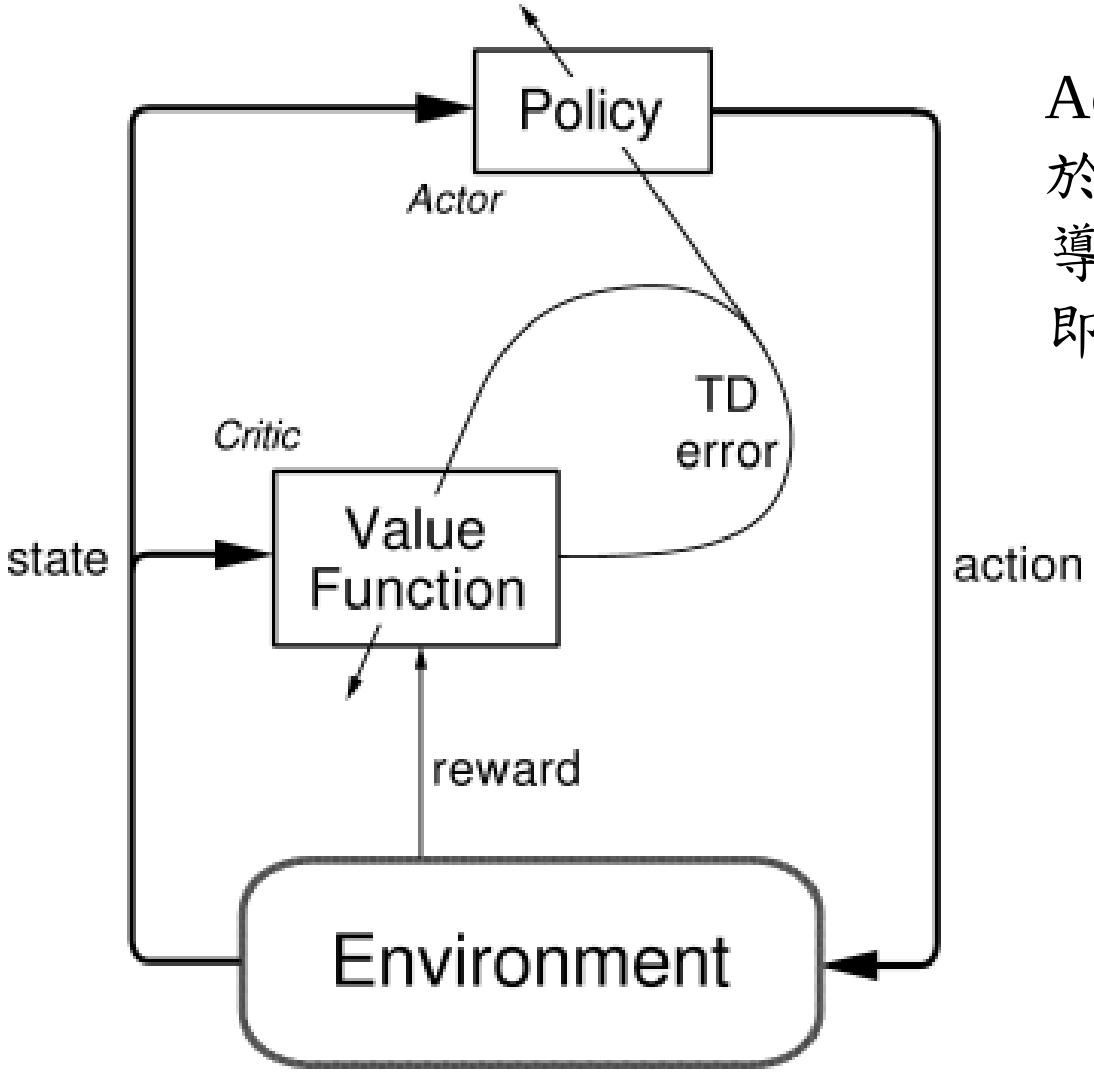
- 在Reinforcement Learning道路辨識中，VAE可以作為一個feature extractor，將影像壓縮至lower dimensional space。
- VAE model的bottleneck結構使得在重構影像時訊息會被壓縮，可以將relevant information從raw data中萃取出來。此步驟稱為State Representation Learning(SRL)。



Soft Actor-Critic (SAC)

- 一個穩定且有高效的deep RL algorithm ，適用於現實生活中的機器人技能學習
- 基於 maximum entropy reinforcement learning，希望可以將expected reward以及policy‘s entropy最大化
- 透過學習狀態映射到動作的隨機策略和估計當前策略目標值的 Q 函數，使用動態的方式對其進行最佳化，從而達成entropy最大化

SAC流程圖



Actor-Critic的字面意思是”演員-評論”，相當於演員在演戲的同時，有評論家在旁進行指導，而演員演得越來越好。
即使用Critic來預測行為的好壞：

$$Q_w(s,a) \approx Q^{\pi\theta}(s,a)$$

Soft Actor-Critic(SAC)(1/2)

- Soft actor-critic 是一個深度增強式學習的方法，用於連續區域中訓練maximum entropy，使模型逐漸收斂。
- SAC基本概念:
  - Actor(演員、使用者):為了得到盡可能更高的獎勵(reward)，需要輸入state，輸出action，藉由觀察到的狀態(state)，來決定程式做出的動作(action)。結果可以使用神經網路來靠近這個函數，剩下的任務就是如何訓練神經網路，讓自己能得到更高的獎勵(reward)。
  - Critic(評分委員):為了訓練Actor，需要知道Actor的表現到底如何，根據表現來決定對action-value函數的調整。

Soft Actor-Critic(SAC)(2/2)

- Actor-Critic的訓練:
  - Actor依照訓練目前的狀態(state)，做出一個動作(action)。
  - Critic根據狀態(state)和動作(action)兩者，對動作(actor)的表現給予獎勵。Actor依據Critic給予的獎勵，調整自己的獎勵策略（actor神經網路參數），並期望下次做得更好。
  - Critic根據系統給出的獎勵(reward)和其他Critic的獎勵（critic target）來調整自己的獎勵方式(critic神經網絡參數)，類似Ground truth的概念。
- Actor-Critic的基本流程:
  - 取樣 > 更新Critic參數 > 根據Critic計算最佳化函數 > 更新Actor參數
- 一開始Actor隨機表演，Critic隨機獎勵。由於有獎勵(reward)機制的存在，使的Critic獎勵越來越準確，Actor表現越來越好。

參考資料

參考資料

- [機器學習 ML NOTE] Reinforcement Learning 強化學習(DQN原理) | by GGWithRabitLIFE | 雞雞與兔兔的工程世界 | Medium
- Deep Q-learning (DQN) 原理說明 @ 我的小小AI 天地 :: 痞客邦 :: (pixnet.net)
- [Day-28] 增強式學習 (Reinforcement learning) 介紹 - iT 邦幫忙::一起幫忙解決難題，拯救 IT 人的一天 (ithome.com.tw)
- Google AI Blog: Soft Actor-Critic: Deep Reinforcement Learning for Robotics (googleblog.com)
- Learning to Drive Smoothly in Minutes | by Antonin RAFFIN | Towards Data Science