Application of reinforcement learning on Cartpole problem

作者廖育捷(s06210043)*
指導教師姓名 吳桂光 副教授
*Email: s60210043@thu.edu.tw

摘要

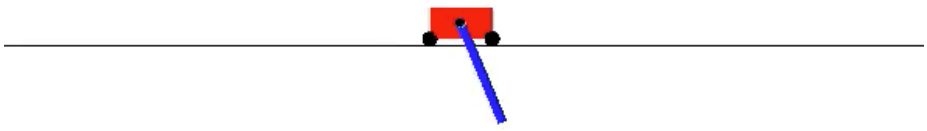
在這份報告中我們討論兩種強化學習的演算法，用來控制 Cartpole 系統，利用的算法有 Q-learning 何結合深度學習的 Deep Q Network(DQN)，去比較兩者的效果，在一開始的時候會稍微簡介 Open AI Gym 和 Cartpole 系統，之後利用 Q-learning 的方式去控制，接著是利用 DQN 的方式，來比較這兩種方法的優劣和差異，最後是將 DQN 的算法帶入 Cartpole Swing Up 來看看在這種方法有沒有辦法應用在動作是連續的問題。

(一) 研究動機與研究問題

強化學習是機器學習的一個領域，強調透過環境決定行為，在不透過監督者提供行為準則，透過獎勵告訴計算機這個行為的好壞，從經驗中學習在任何狀態下如何選擇行為，以達到最大獎勵為目的，在強化學習中有幾個重要的元素，包含代理人、環境、狀態、獎勵和動作，我們透過代理人來選擇動作，輸入於環境改變狀態，同時環境也給代理人獎勵，來告訴代理人這個動作好或壞[1]，Open AI Gym[2]是一個有很多強化學習的環境資料庫，裡面除了有很多知名遊戲外，還有很多類似物理系統的問題，Cartpole system[2]就是其中之一，如圖(一):

(a) (b)

圖(一) A Cartpole system: (a)平衡狀態,(b)不平衡態



圖(二)Cartpole Swing up

這個控制系統的主要結構是一台滑軌車在滑軌上，車子的正中間有一個培林，連接一根桿子，目的就是要使桿子保持在正上方，當桿子不和車子保持垂直，桿子就會受重力作用，產生一個與傾倒方向反向的力，傳統上會透過解析力學的方式寫下微分方程來計算需要施加多少力來達成目標，這系統的微分方程沒有辦法用手解，需透過數值模擬的方法，但透過強化學習可以不用解許多的微分方程就可以讓計算機從失敗

中學習經驗，來達到長時間維持目的。

（二） 文獻回顧

人工智慧在近年的應用範圍相當廣泛，隨著摩爾定律的發展，計算機也有了飛躍的進步，生活中處處可以看見人工智慧的蹤影，生活中常見的人工智能有車牌辨識技術、生物特徵辨識或是人工智能秘書等等，改善了許多生活上的不便，也提升了做事情的效率，最廣為人知的強化學習例子是2016年google deepmind團隊提出的 Alphago[3]打敗職業圍棋選手李世乭的新聞，但我本身不會下棋，所以讓我對強化學習感興趣的不是Alphago，而是在2019年時一樣由google deepmind所提出的Alphastar，starcraft2，是一款即時戰略遊戲，相較於圍棋，starcraft是不完整資訊的遊戲，意旨有著未知的部分需要去探索，局勢不像圍棋一樣一目瞭然，動作的選擇也比圍棋多的很多，在如此困難的情況下，AI依然可以擊敗歐洲的職業選手，而論文也刊登在《Nature》期刊中[4]。

而Deep Q Learing(DQN)一樣是由Google deepmind提出，最成功的例子是在2013年，該團隊利用DQN的演算法讓計算機玩Atari2600[5、6]，裡面有七個簡單的小遊戲，這些遊戲沒有一個明顯的狀態來表示遊戲的進展，所以利用捲積神經網絡(CNN)，透過影像分析的方法找出畫面的特徵，而動作也都是簡單的上下左右和攻擊，都是離散的，很適合用DQN的演算法達成目標[5]，DQN也很適合應用在Cartpole problem動作的選擇是離散的。

（三） 研究方法及步驟

在一般的Cartpole時動作只有向左推或向右推，力的大小是固定的，但在Cartpole swing up的問題中，動作變成了-1~+1之間，裡面包含了方向和大小，我們可以透過切割將其變成離散的問題；強化學習中最入門的算法是Qlearning，將狀態和動作製作成表格Q表，如下圖:

$$R=\begin{array}{c} \\ \text{State} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array}\begin{array}{c} \begin{array}{cccccc} \text{Action} & & & & & \\ 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} -1 & -1 & -1 & -1 & 0 & -1 \\ -1 & -1 & -1 & 0 & -1 & 100 \\ -1 & -1 & -1 & 0 & -1 & -1 \\ -1 & 0 & 0 & -1 & 0 & -1 \\ 0 & -1 & -1 & 0 & -1 & 100 \\ -1 & 0 & -1 & -1 & 0 & 100 \end{bmatrix} \end{array}$$

圖(三)Q-table的形式

表格內的值代表的是在該狀態下選擇該動作內涵的價值Q值，Q值越大表示在該狀態下這個動作最優，我們可以透過下列算式來更新表格

$$Q(s,a) \leftarrow Q(s,a) + \alpha[\text{Reward} + \gamma * \max Q(s',a') - Q(s,a)]$$

其中α是學習率，介於0～1之間[2]，值越大表示越依賴此方法更新，我們定義$\text{Qtraget} = \text{Reward} + \gamma * \max Q(s',a')$，裡面包含了當下的獎勵與下個狀態最好的動作的值，透過目標和舊的Q值的差距來更新Q表，γ指的是對未來獎勵的遞減率，看Qtarget的部分，把a移除我們可以把式子拆解成$Q(s_1) = R_2 + \gamma * Q(s_2)$ 再將$Q(s_2)$拆解成跟$Q(s_3)$有關的形式，一直拆解到只剩下最後時，我們可以把式子整理成$Q(s_1) = R_2 + \gamma * [R3 + \gamma * (R_4 + \gamma \ldots)]$再整理一下，$Q(s_1) = R_2 + \gamma * R_3 + \gamma^2 * R_4 + \gamma^3 * R_5 \ldots\ldots$ ，γ=1時，對於每一次的獎勵都很重要，當γ=0時計算機只看的到當下最大的獎勵，如果在0~1之間越後面的獎勵就越不重要。

透過這樣的方式來更新方法來達到目的，但如果動作選擇太多，狀態不是很明確，或狀態太複雜時，這時用表格更新對計算機而言非常的吃力[5]，但在深度學習中的神經網路就可以將這件事情做得很完美，我們就可以將表格替換成神經網路，雖然透過表格更新的方法比較收斂，但神經網路裡有很多參數，一樣可以Fit出很好的函數，而此方法利用了深度學習的神經網路，也因此稱此方法為Deep Q Learning[5、6]。

我們透過ε- greedy來決定要如何選擇動作，當 ε =0.7時，代表有七成的狀況是隨機亂選，三成的情況是照著訓練的模組去決定，所以我們可以在一開始時 ε =1，讓計算機可以學習更多經驗，隨著回合數變多 ε 也跟著遞減，這樣子到後面就會大部分都是照著訓練的模組選擇行為了。

而我採用的方法是透過Keras[7]套件，建立多層感知器，並輸入狀態當input，Qtarget值當作output，這樣可以使Qfunction更逼近Qtarget[1]，來訓練神經網絡，而神經網絡的架構為輸入層，兩個128個神經元的隱藏層，一個64神經元的隱藏層，和一個輸出層所構成。再另外將每一次的各項數據儲存起來，從裡面的資料隨機抽樣來學習，會這樣做的原因是因為強化學習是有時序性，也就是說資料前後有關聯性，有可能導致模型沒有辦法正常訓練，但透過隨機抽樣就不會有關聯性的問題，希望可以透過此方法來達到將桿子甩上去並保持平衡。

一、實驗環境

State:[車子的位置,車子的速度,桿子的角度,桿子的角速度]，向右為正向左為負，起始值為[0,0,-0.03,0]，範圍:[-4.8~4.8,-0.5~0.5,-12°~12°,-24°~24°]
Action:[0、1](0 是向左推 10N，1 是向右推 10N)
Reward: 每過一幀就是+1
Gamma:0.95

Epsilon:從 1 開始，每過一回合減去$\log[\frac{回合數+1}{25}]$
回合數:2000
每回合有多少時間:200
成功條件:每一回合需撐過 199，達成的回合要 120 個

二、建立神經網路與更新算法

透過 Keras 套件建立全連接的神經網路，架構如下:

1. 輸入層與隱藏層 1(共有 128 個神經元，輸入層有四個神經的輸入，激活函數:relu)
2. 隱藏層 2(128 個神經元，激活函數:relu)
3. 輸出層(2 個神經元，激活函數:linear)

損失函數：Mean-Square Error(mse)
優化器：Adam

算法更新[1]:

- 首先初始化 Memory D，它的容量為 N;
- 初始化 Q 網路，隨機生成權重 ω;
- 初始化 target Q 網路;
- 迴圈遍歷 episode =1, 2, …, M:

- 初始化 initial state S1;
- 迴圈遍歷 step =1,2,…, T:
  - 用 ϵ-greedy 策略生成 action a_t：以 ϵ 概率選擇一個隨機的 action，或選擇 a_t=argmaxQ(S_t,a;ω)；(amax=沿著一個軸找最大值是在第幾列，我們找的是在 S_t 下 a 值最大的列數)
  - 執行 action a_t，接收 reward r_t 及新的 state S_{t+1};
  - 將 transition 樣本 (S_t,a_t,r_t,S_{t+1})存入 D 中；
  - 從 D 中隨機抽取一個 minibatch 的 transitions (S_j,a_j,r_j,S_{j+1})；
  - 令 y_j=r_j，如果 j+1 步是 terminal 的話，否則，令 y_j=r_j+γ argmaxQ(S_{t+1},a’ ;ω^-)；
  - 對$(y_j-Q(S_t,a_j;\omega))^2$關於 ω 用梯度下降法進行更新；

每隔C steps更新target Q網路，ω^-=ω。

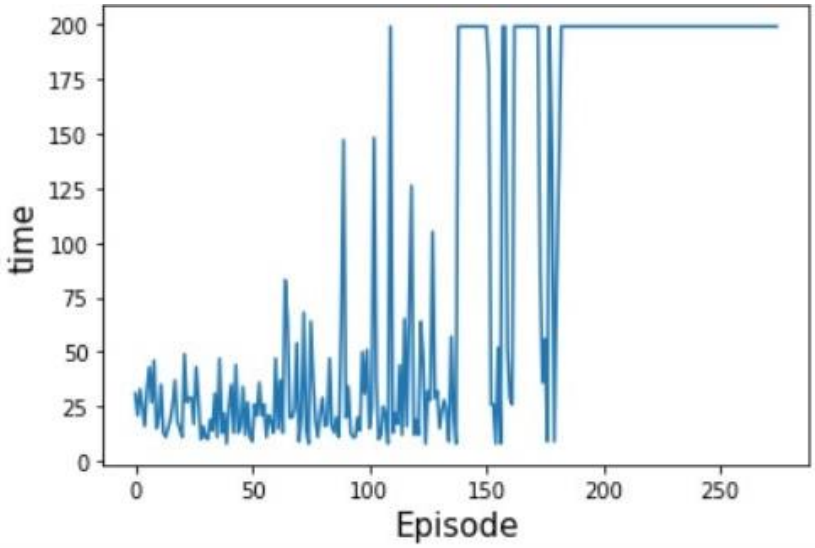
實驗方法

先利用Q Learning(透過表格)的方式來學習，之後再透過將表格取代成神經網路，並將每回合維持多久和回合數作圖，來比較何種訓練方法較優。

(四) 結果及討論

一、Q-learning

將實驗環境帶入Q Learning 架構，並把狀態範圍切割成離散的數，位置和速度不進行分割，角度切割成 6 等分，角速度切割成 3 等分，每個狀態都可以轉換成[0,0,0~5,0~2]的向量，每項都為整數，從 0 開始是因為 python 的初始值為 0，並將該回合的時間與回合數作圖，如下圖:

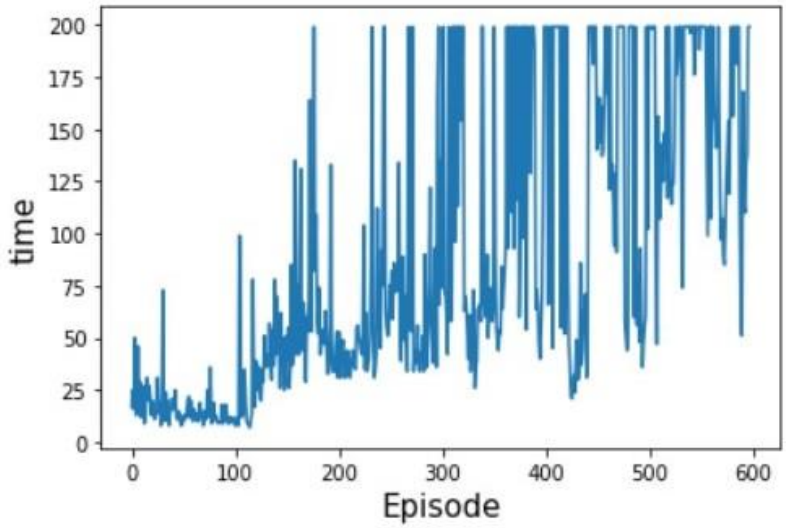


圖(四)，Q-learning，橫軸為回合數，縱軸為桿子維持平衡的時間

可以從圖中一開始確實維持時間很低，因為不清楚要如何做動作才能得到獎勵，隨著回合的增長 ε 開始減少，也開始學習到比較多成功的經驗，可以連續幾回合都是成功的，最後面也隨著 α 的降低，確定了表格的樣子就可以每回合達成目的。

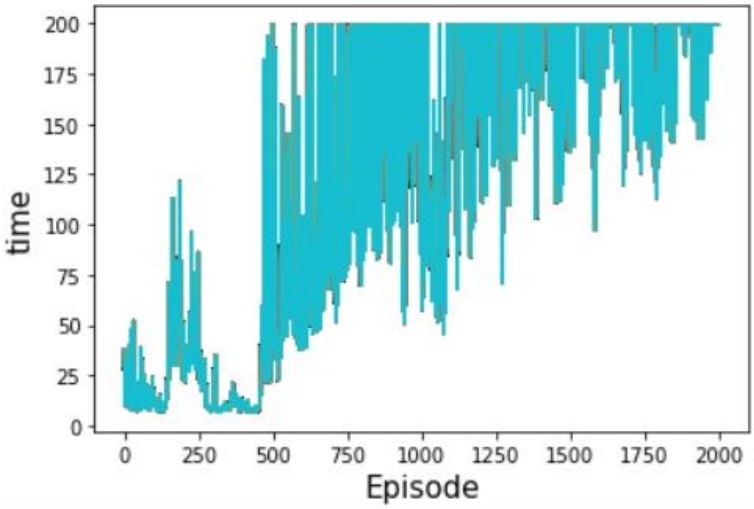
二、DQN

再來是利用神經網路去學習，此時的狀態就不需轉換成離散的向量，只需要透過神經網路去尋找狀態和動作之間的關係即可，所以可以用在更多的情況下，若沒有明確的狀態也可以用影像分析的方法當作狀態輸入，結果如下圖:



圖(五)DQN，橫軸為回合數，縱軸為桿子維持平衡的時間

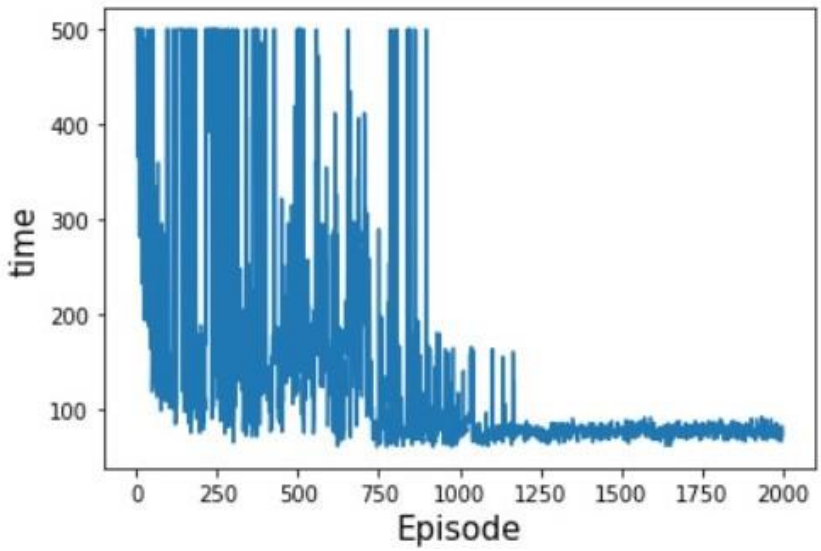
比較起來其實利用表格學習的 Q Learning 學習效果更收斂效果更好，會造成此原因的關係有可能是因為更新神經網路的回合數不夠多，只要隨著回合的增加也可以訓練的很好，如下圖:



圖(六):DQN 訓練 2000 回合，橫軸為回合數，縱軸為桿子維持平衡的時間

三、Cartpole swing up

接著觀察一下透過 Deep Q Learning 的方法有沒有辦法解決 Cartpole Swing Up 的問題，因為環境不同，我們必須先將狀態和動作重新定義，狀態基本上大同小異，只是將角度換成正弦和餘弦的值，動作的變化較多，是由[-10N~+10N]的連續浮點數，我們必須將動作動點手腳，將它分成 11 個動作，將-10~10 以間隔為 2 分成 11 個數，神經網路的輸入層變成 5 個神經元，輸出則有 11 個結果如下，獎勵也有些變動是以角度的餘弦值當作獎勵，當桿子一直朝下那獎勵就為負的，成功擺動至上方就可以拿到較高的獎勵，而我們也將把桿子擺動上去的時間記入起來與回合作圖，不用畫面維持的時間是因為我們沒有達成將桿子維持平衡在正上方的目標。



圖(六)把桿子擺動上正上方需花的時間和回合作圖

將桿子擺動上去的時間有明顯的下降，但是這樣的結構實驗結果還是很差，因為每一回合的時間沒有到太長，每回合大約是70~90，但將桿子擺動上去上方的時間又明顯的縮短到後期將桿子擺動上去的時間大約40~60，有學習到得到最大獎勵的方法，但還沒完成維持最大獎勵。

（五） 個人貢獻主要項目及比重

全部

（六） 結論

預期上我們希望DQN的學習效果應該比Q-learning好，在參考文獻的第一篇裡也有做相同的比較，其結果是DQN的學習成效較佳，或許是我們的神經網路不夠好導致訓練的結果不一樣，未來會繼續調整神經網路來達到最好的結果。

在Cartpole Swing up的問題中可以看到，使用Deep Q Learning處理連續動作空間的問題並不是這麼有利，而且連續動作空間的問題中，把動作空間分為離散並不是個好選擇，DQN在離散動作空間中可以發揮得很好，但多數的真實情況是以連續的動作為主，離散的動作可能只適合在電玩遊戲的控制。

未來打算利用可以處理連續動作的強化學習方法，像是Policy Gradient、Actor-Critic、DDPG (Deep Deterministic Policy Gradient)…等方法來處理連續動作空間的方法，使Cartpole Swing Up的問題一樣可以透過強化學習的方式來解決。

以及利用以學習過的演算法完成更多open ai gym上的物理問題，像是平衡複擺、登入月球……等問題。

（七） 參考文獻

[1] Kumar(2020),Balancing a CartPole System with Reinforcement Learning -- A Tutorial, eprint arXiv:2006.04938。

[2] OpenAI Gym. Toolkit for developing and comparing reinforcement learning algorithms. https://gym.openai.com/.

[3] David Silver(2016), Mastering the game of Go with deep neural networks and tree search,Nature volume 529, pages484-489

[4] Oriol Vinyals(2019), Grandmaster level in StarCraft II

using multi-agent reinforcement learning, Nature volume 575, pages350–354
[5] Beakcheol Jang、Myeonghwi Kim、Gaspard Harerimana、ong Wook Kim, (2019) Q-learning Algorithms: A Comprehensive Classification and Applications, IEEE Access, DOI: 10,109。
[6] Mnih(2013),Playing Atari with Deep Reinforcement Learning, et al, arXiv:1312.5602。
[7] A. Gulli and S. Pal(2017). Deep learning with Keras. Packt Publishing Ltd。