Quantum Reinforcement Learning

Group 12 吳育丞(B08901027)、吳冠緯(B08901019)、陳宥辰(B08901048)

Abstract

這次的 project 我們以量子強化學習為主題，並以 Quantum agents in the Gym: a variational quantum algorithm for deep Q-learning [1]作為參考對象，其核心概念在於使用量子模型作為 Agent 而非目前主流的 DNN 模型。Project 的內容包括重現 paper 裡面的結果以及其他延伸實驗。

第一步我們 reproduce 作者使用的量子模型，並仿照作者以 Q-learning 的方式訓練 openAI gym 裡面的 cart Pole 這個任務，並跟一般 DNN agent 的表現進行比較，得到與 paper 吻合的結果。另外，我們在看 paper 的過程中分別對模型本身和作者有關 Q value 的論述產生了兩個疑惑點，所以進行了兩個額外的小實驗，並分別對實驗結果做出以下推論：

1. 環境給出的 State vector 中 features 的順序會影響訓練收斂的速度。
2. Agent 學到正確的 Q value 大小關係比學到正確的 Q value 數值更重要。

最後我們有嘗試使用相同的模型架構套用在其他 openAI gym 裡面的任務包含 Mountain Car 和 Lunar Lander，但是並沒有訓練成功，目前只能推測此量子模型的通用度還是有所限制，是日後還能繼續探討的方向。

Motivation

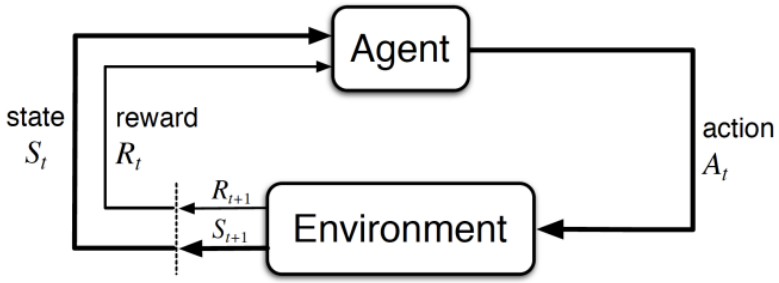
量子深度學習(QNN)是這幾年時常聽到的一個概念，雖然之前已經對一般的深度學習有一定的認識，但是礙於未接觸過量子領域，所以對量子深度學習實際上能有怎麼樣的優勢毫無頭緒。經過這學期前半有關量子的課程之後，實際去看前人設計的量子模型以及實驗結果，發現在不同任務普遍都能以大幅減少的參數量達到類似的表現，這樣的特性也使得模型訓練的收斂速度更快。不過由於深度學習的模型和任務非常多樣化，因此量子模型是否能被廣泛運用還是只適用在特例的架構仍然是未知數。

近幾年強化學習(Reinforcement Learning)的應用蓬勃發展，此方法主要是想在具有不確定性的環境中訓練出一個決策者(Agent)，已達成特定目標。但是強化學習的訓練難度相當高，因此若能使用量子模型在強化學習的任務上達到不錯的效果，會是非常有說服力的結果。

Introduction to Reinforcement Learning and Q-Learning

強化學習的訓練可拆分成許多 episodes，每個 episode 包含 n 個非連續的 timestep。每個 timestep agent 會根據環境(environment)目前給出的狀態(state)來決定接下來要在環境採取的行動(action)，而 agent 做決策的 mapping function (from state space to action space)則稱為 policy。環境接著便會根據此行動給出 reward 並移動到下一個狀態，這樣的互動會重複循環直到滿足環境的終止條件，如圖(1)所示。

本次實驗我們仿照作者採用 Q-learning，Q-learning 是一種使用 Value-based policy 的強化學習方法。顧名思義，Agent 在每個 timestep 會根據當下環境呈現的 state 去計算所有可執行的 action 對應的 expected reward(定義如圖(2))，也就是 Q-value，然後 agent 便會採取 Q-value 最大的 action 來行動，以次類推。在使用 Q-learning 進行訓練時通常還會定義一個常數 epsilon([0,1])，代表 agent 不使用 greedy policy(選 max Q-value)而是隨機採取其他 action 的機率，主要是讓 agent 的訓練不要受限於 local minima 而是能探索到其他可能性。由於這次的 project 我們並沒有著重在 epsilon 的部分，所以所有實驗的 epsilon scheduling 都是相同的(由 1 等比遞減至 0.01 為止)。



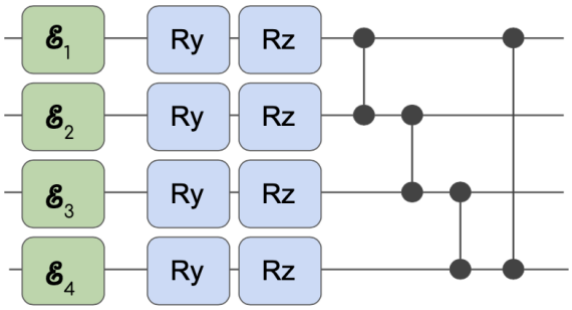
圖(1)

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

圖(2) Q-learning expected reward 的 Bellman function，gamma 為常數，本實驗仿照 paper[1]固定使用 0.99。

Quantum Agent

Paper[1]中使用的量子模型如圖(3)所示，綠色的部分是 encoder，剩下的部分則被稱為一層 variational layer。在圖(3)總共有 4 條線，每一條代表一個 qubit，而之所以是 4 個 qubits 是因為 cart pole 這個任務的 state space dimension 為 4，所以這些 state features 就會分別被一一 encode 到相同數量的 qubits 上。



圖(3)

1. Encoder

   Encoder 的作用在於將 state features 轉換成 qubit 能使用的資訊，每個 qubit 會需要一個可以訓練的 weight，其流程如下：

   (1) 將每個 state feature 乘上可訓練的 weight，其作用在於讓模型能夠調整 feature 的數值分佈，尤其當數值普遍都接近 0 的時候，下一步要做的 arctan function 會非常 sensitive，因此這個設計能以加入少量參數的方式有效提升模型表現。

   (2) 對前一步計算出的數值做 arctan 並得出介在[-pi/2, pi/2]的角度。

   (3) 對每個 qubit 加上 RX gate 並使用前一步得到的角度。

   雖然在一般的 DNN 模型中普遍只會稱第一層或前幾層的 layer 為 encoder，但是包括 paper[1]和其他多篇研究都指出在每一層 variational layer 前面都加上 encoder 會讓模型的表現更好，又稱作 data re-uploading，因此我們的實驗也沿用了這個設計。

2. Variational layer

   Variational layer 的部分就是如圖(3)所示，會對每個 qubit 加上一個 RY gate 和 RZ gate，兩者的角度是可以訓練調整的，然後再從最上面的 qubit 開始依序在兩個相鄰的 qubit 間加上 CZ gate，直到最下面的 qubit 連回第一個 qubit 為止，這樣的結構可以重複數次，類似於 DNN 模型中的 hidden layer。

3. Output (Measurement)

   Measurement 的部分根據不同 state space 和 action space 的大小會呈現 case by case 的情形，不過原則就是會在某個 basis 作測量來得到 Q-value。以此模型在 cart pole 上的應用為例，就是加上一個 Pauli-Z gate 之後進行測量。至於前面提到 case by case 的原因可以歸類成主要兩種情況：

   (1) 當 state space dimension 大於 action space dimension 且又不是倍數時應該使用幾個 qubit 的 tensor product 作為一個 action 的 Q-value，舉例來說當(state space dim., action space dim.) = (8,3)，究竟要把 qubits 分成(3,3,2)還是捨棄其中 2 個 qubit 只測量另外 6 個 qubits？不過在 cart pole 實驗中 dimension 剛好是 4 對 2 所以前兩個 qubits 和後兩個 qubits 測量到的數值很自然能對應到 move left 和 move right。

   (2) 當 state space dimension 比 action space 還要小時，又應當如何設計 measurement？這個問題在我們嘗試將模型套用在 mountain car 這個任務時就遇到了，他的 state space dimension 和 action space dimension 分別是 2, 3。因為目前還沒有看到任何研究在處理這個問題，所以我們有初步嘗試將 state vector 重複三次並 encode 到 6 個 qubits 上面，然後再分成前中後各兩個 qubits 做測量，不過並沒有成功。但是這也有可能是因為 mountain car 是屬於 win all or lose 的任務，本身就比較難訓練，這

個特性在實驗的部分會再提到。

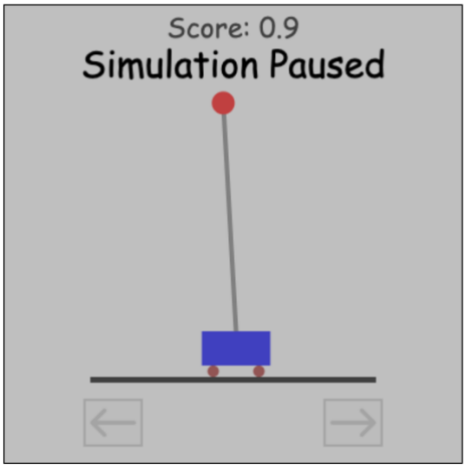
做完 measurement 之後，作者還賦予每個 qubit 一個可以訓練的 weight，需要將兩者相乘才會是最終結果。作者的觀點在於 measurement 本身是存在數值上限的，所以當理論上的 expected reward，也就是 Q-value 超出範圍時，模型將永遠不可能學會如何根據 state 得出正確對應各 action 的 Q-value。雖然這個論述本身沒有問題，不過我們不認爲這是造就此設計如此有效的關鍵，這部分在我們做的延伸實驗會更詳細說明。

Experiment

這次 project 我們在 cart pole 上做的實驗都是比較兩個不同模型或設置的表現。表現的定義方式是模型訓練完成所需的 episode 數，而訓練只有在過去 100 次 episode 中獲得平均 195 分以上的成績時才會終止，我們每種模型或設定會訓練三次取平均成績。由於這次的 project 我們並沒有著重在訓練技巧的部分，所以模型訓練的 loss function 及 optimizer 分別固定成較穩定的 smooth l1 loss 和 Adam。

Cart pole:

Cart pole 這個環境主要包含一台可以左右移動的臺車以及車上一根不會滑動但是會傾倒的竿子，如圖(4)。state 是一個四維連續的向量，包括 cart position、cart velocity、pole angle、pole angular velocity。Agent 可以選擇的 action 始終是兩種，左移和右移，移動一次的加速度跟時間都是固定的，agent 無法決定。在每個 episode 中，agent 的目標就是平衡竿子越久越好，每個 timestep 如果竿子沒有傾倒，agent 就會獲得一分的 reward，反之如果竿子傾倒而碰到邊框，此次 episode 便結束，同樣的情況發生在當臺車移出所在平臺的範圍或是 agent 已累加到 200 分的 reward。



圖(4)

1. QRL vs DRL

第一個實驗我們想仿照 paper[1]比較量子模型和一般 DNN 模型的表現。量子模型的部分，我們使用 paper 中表現最好的五層架構，DNN 模型的部分同樣使用五層。至於 DNN 模型 hidden layer 的 neuron 數我們首先同樣使用 paper 中表現最好的 30，不過我們後來也有嘗試多種更小或更淺的架構，表現都相差很多，所以我們最終就以 hidden layer 有 30 個 neuron 的模型跟量子模型做比較，以下表格(1)為比較結果：

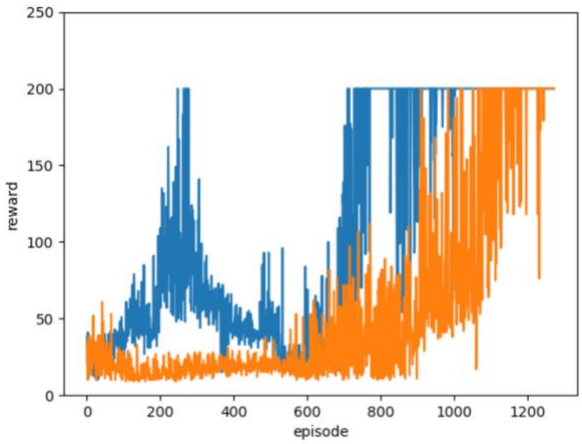
Method	Number of parameters	Number of episodes
DRL	1172	1666
QRL	**62**	**1038**

表格(1)

由結果可見量子模型所需要訓練及儲存的參數遠小於 DNN 模型，而訓練所需要的 episode 也更少，在目前 model to edge device 的趨勢下這樣的 compact model 非常具有優勢。

2. Order of state features

另外在看 paper 的過程中我們發現量子模型的 variational layer 其實不像 fully connected layer 那樣對稱(每個 neuron 間都存在可訓練的 weight)，每一層 layer 的 CZ gates 只會將相鄰的 qubits 和頭尾的 qubits 相連，因此讓我們不禁好奇 state vector 內 features 的順序是否會影響模型訓練的收斂速度。以 cart pole 為例，state feature 的預設順序是 cart position、cart velocity、pole angle、pole angular velocity，前兩者彼此和後兩者彼此看似是比較相關的，於是我們將 cart velocity 和 pole angle 的順序調換進行實驗，以下是訓練過程 reward 的折線圖(圖(5))和結果：

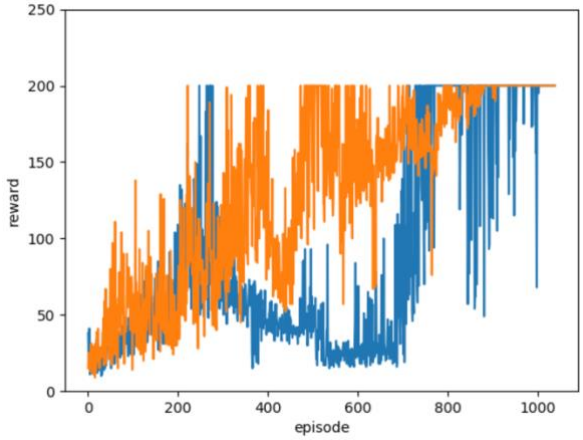


圖(5) **藍(預設順序): number of episodes = 1038**
橘(調換順序): number of episodes = 1272

從以上結果可以看出 state features 的順序確實會稍微影響訓練收斂的速度，但是不至於讓模型出現訓練失敗的情形，加上 features 間的相關性也非常難在訓練前量化，所以頂多就是以一些比較 heuristic 的方法做順序上的安排，很難統整出有系統且有效的方法。

3. Effect of output weight design

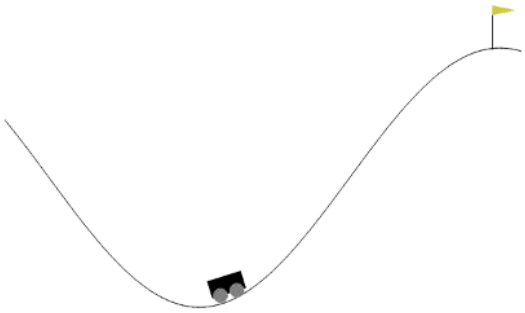
在前面 measurement 的部分我們有提到說作者設置 trainable weight 的原因在於避免發生實際 expected reward 超出 measurement 上限的情況，此論點雖然合理，但是我們認為並沒有解釋到為何此設置能進一步加快收斂速度。我們之所以會這樣想是因為作者其實也有嘗試直接在 output 乘上固定常數讓 measurement 的上限恰巧會是理論上 Q-value 的上限，但是此方法依然明顯輸給在 output 設置 trainable weight 的方法。因此我們認為 trainable weight 最主要的優勢在於更容易讓模型在訓練前期就學習出正確的 Q-value 大小關係，而不是單純為了接近實際的 Q-value。我們為此做了兩種訓練設置，分別讓 output trainable weight 的初始狀態設置成 mean 為 1、std 為 1 的常態分佈和 mean 為理論 Q-value 上限、std 為 1 的常態分佈。以下是訓練過程 reward 的折線圖(圖(6))和結果：



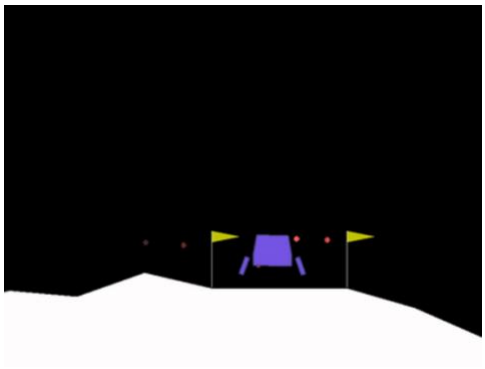
圖(6) **藍(mean 為 Q-value 理論上限): number of episodes = 1038**
橘(mean 為 1): number of episodes = 916

從以上結果可以看出如果將 output trainable weight 的起始狀態設定成 mean 為 1、std 為 1 的常態分佈，模型能以更快的速度收斂，主要的原因在於使用較小的平均值作為起始點，output trainable weight 每更新一次參數造成的倍率變化是比較明顯的，也證實了學會 Q-value 正確的大小關係，即便還尚未達到理論的 Q-value，也對模型訓練的收斂很有幫助。

除了 cart pole 相關的實驗之外，我們原本想嘗試另一個 openAI gym 裡面的任務 mountain car，其任務目標是讓車子從山谷移動到指定山頂(如圖(7))，整個任務只有在抵達山頂時才會獲得正的 reward，反之每個 timestep 若無法抵達山底則會持續獲得負的 reward。這樣 win all or lose 的任務跟 cart pole 每個 timestep 只要竿子不倒就能獲得 reward 的任務在 exploration 的部分難度差異很大，也因此成為我們想要測試的對象。不過就如同前面所述，mountain car 的 state space dimension 比 action space dimension 還要大，因此無法直接沿用 paper[1]的模型。後來嘗試將 state vector 複製三次並 encode 到 6 個 qubits，最後再以兩個 qubits 為一組做測量，不過這個方法也以失敗告終。於是我們又再找到了一個 win all or lose 的任務 lunar lander，其目標是讓火箭能以兩隻腳而非其他部位在指定範圍著陸(如圖(8))，而且 state space dimension 和 action space dimension 分別是 8 和 4，也代表能直接將 cart pole 模型擴大成兩倍就能使用，但是在嘗試多種訓練方法之後最終還是很遺憾的沒有成功，也意味著目前 paper[1]的量子模型的通用難易度可能還有優化的空間，因為在實驗一使用的 DNN 模型是能成功在 lunar lander 訓練成功的。



圖(7)



圖(8)

Conclusion

1. 在 cart pole 的環境下，quantum agent 能以 DNN agent 接近 1/20 的參數量完成訓練，並且只花了 0.6 倍的 episodes。
2. 使用 paper[1]的量子模型時，state feature 的順序會影響訓練的收斂速度。
3. Paper[1]量子模型使用的 output trainable weight 除了讓量子模型輸出的 Q-value 不再受限之外最主要的優勢在於讓模型更容易學會 Q-value 正確的大小關係，加快訓練的收斂速度。
4. Paper[1]的量子模型目前還無法輕易應用到其他的強化學習任務。

Reference

[1] Quantum agents in the Gym: a variational quantum algorithm for deep Q-learning. Andrea Skolik, Sofiene Jerbi, and Vedran Dunjko

分工

吳育丞：QRL for Cart Pole, experiment for output weight initialization in QRL, QRL for Mountain Car, QRL for Lunar Lander, report.

吳冠緯：DNN for Cart Pole, experiment for state features' order in QRL, training process visualization, report

陳宥辰：QRL for Cart Pole, QRL for Lunar Lander, DNN for Lunar Lander, report