**DSAI - HW4: Classic control in reinforcement learning**

F74031027 李東霖

# RL 題目 : Classic control: Mountain Car

action space : 0-2

|  |  |
| --- | --- |
| 0 | move backward |
| 1 | stop |
| 2 | move forward |

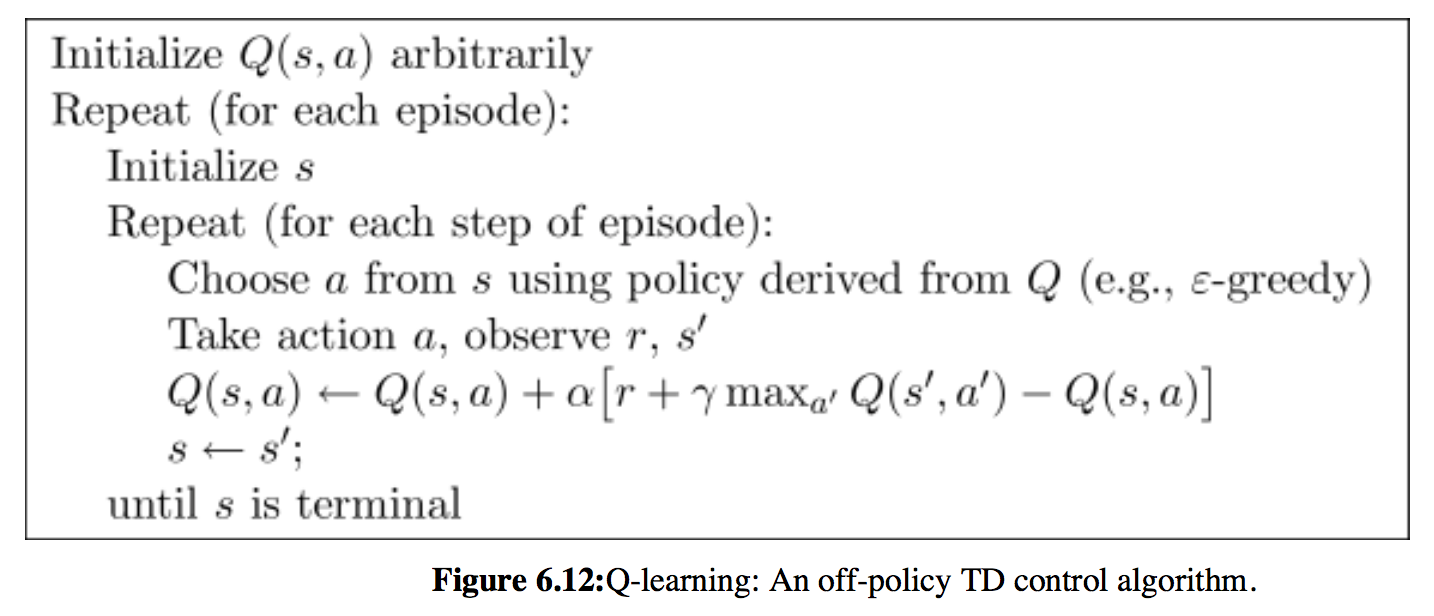
observation space : [position, velocity]

|  |  |
| --- | --- |
| position | [-1.2 , 0.6] |
| velocity | [-0.07, 0.07] |

reward : -1

done condition : position >= 0.5

# 選用的 RL 演算法 : Q-learning



## 設計 Q-learning 狀態轉換

將 position 切成 n 個區間，並考慮速度正負再分成兩個不同狀態

總共產生 n\*2 個狀態，因此狀態表為 n\*2\*3

下面先設定 n = 20

# 不同獎賞演算法的學習曲線

因為這一題剛開始看到發現 reward 都是 -1 ，第一個疑問是這樣能夠使用 RL 找到解法嗎，都是 -1 不就不知道要怎麼做才是有利的。

因此我利用設計了幾個算 reward 的方法

並把都是 -1 的 reward 稱為 origin reward

* 把 position 當作 reward 稱為 custom reward

將 position 映射到 [-1, 1] 成為 custom reward

* 當下如果是抵達終點，給予更多 reward

當到達終點，該 action 的 reward 為 5

可以用在 origin reward 或 custom reward

採用總共四種 reward 算法去看看 Q-learning 跑出來的效果

* origin reward

都是 -1

* custom reward

介於 [-1,1]

* origin reward with done reward

到達終點前的 reward 與 origin reward 相同

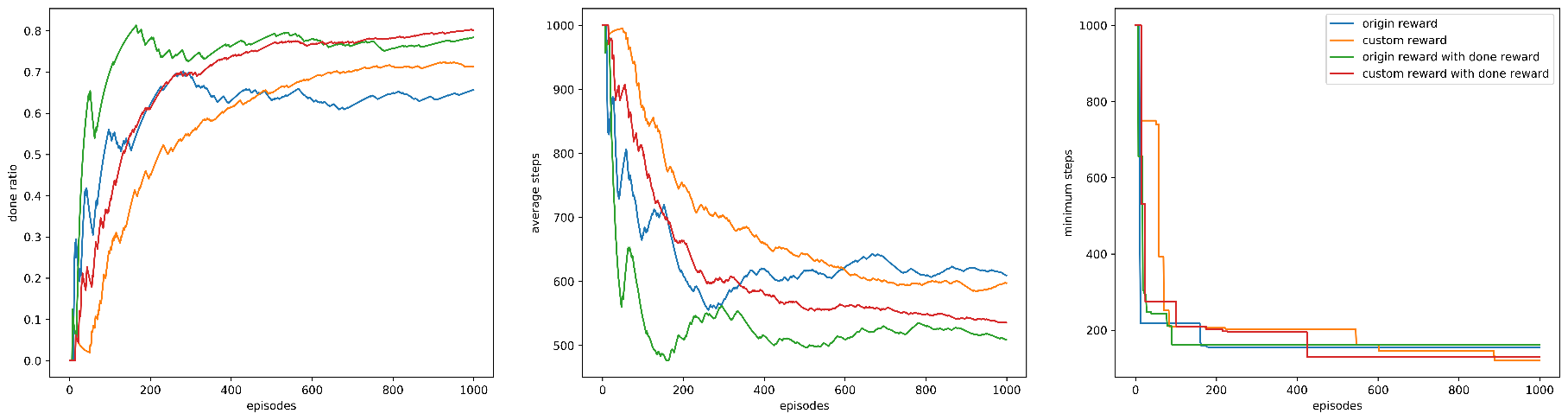
到達終點的 reward 是 5

* custom reward with done reward

到達終點前的 reward 與 custom reward 相同

到達終點的 reward 是 5

alpha 設定 0.5，gamma 設定 0.7，epsilon 設定 0.01



左圖是山車跑到終點的比率( 完成次數 / 完成 episodes 數)

中間是到目前 episodes 的總平均 step ，越少代表使用越少 step 到達終點

右圖是到目前 episodes 的最小 step，可以看出進步幅度與速度

藉由左圖發現 custom reward 跟 custom reward with done reward成長很穩定，不會突然掉下來

origin reward 跟 origin reward with done reward雖然很快就有高完成率，但後來完成率卻沒有繼續成長，而且還下降，代表後來有很多 episodes 沒辦法跑到終點

然後可以發現到在跑到終點加上數量級大的 reward 是非常有幫助的

不管是用在 origin reward 或 custom reward

中間圖發現到 origin reward with done reward有最低的平均

這個還蠻令人意外的，我想是因為都是 -1 reward 突然出現一個大的 reward 對整個 model 有非常大的幫助，因此能快速有低的 steps ，就算後面表現不佳也可以保持低平均

右圖則呈現出 custom reward 能夠確實讓 model 持續有學習到該怎麼走

不像 origin reward 到後來都沒辦法讓 step 變得更低

# 不同參數的學習曲線

基本預設參數

使用的獎賞算法是 custom reward with done reward

## C:\Users\USER\AppData\Local\Microsoft\Windows\INetCache\Content.Word\alpha比較.png學習率

看起來當 0.3 時表現較佳，應該是因為學習到更多環境回饋跟過往記憶的 reward

## C:\Users\USER\Desktop\gamma比較.png衰退因子

雖然 0.7 時，稱不上最好，可是在右圖可以看到他能更進一步的變好

## C:\Users\USER\Desktop\epsilon比較.png隨機選擇率

這裡就很有趣了，差異非常大，當 0.01 時大獲全勝

雖然在左圖看到進步神速，但是卻後繼無力，無法繼續進步

# 問題回答

1. Q: What kind of RL algorithms did you use? value-based, policy-based, model-based? why?

A: 我使用 Q-learning。是 value-based 的, 因為 Q-learning 是更新在不同 state中不同 action所得到 return value。並且在選擇 action時，選擇該state中擁有最高 return value 的 action。

1. Q: This algorithms is off-policy or on-policy? why?

A: 是 off-policy 的，因為在更新 value 的公式當中，是對即將到達的 state 進行 argmax 藉此取得之前訓練中最大 value 的 action。但不一定在下一個 step 會選擇相同的 action。

1. Q: How does your algorithm solve the correlation problem in the same MDP?

A: 在更新 value 公式當中

發現到 Q-learning 只在意當下 state 與選擇 action 到達後的 state

並沒有去處理在 MDP 的 correlation problem