HW3_109403021

Colab 連結: 連結

1. 訓練完的 Q-table

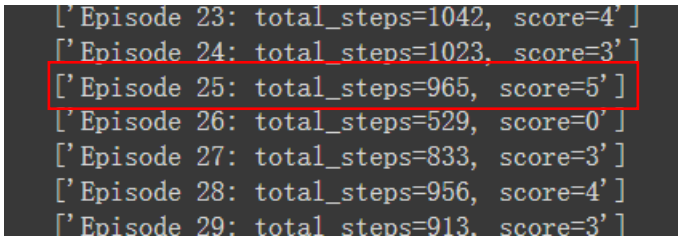
```
Q-table:

(<built-in function all>,           left      right         up       down
0   -51.365591 -40.592331 -64.096286 -40.191932
1   -40.987620 -40.962355 -52.191225 -46.223794
2   -41.295081 -41.304106 -55.470715 -55.752007
3   -41.557443 -48.437432 -49.311300 -41.522938
4     0.000000   0.000000   0.000000   0.000000
..         ...        ...        ...        ...
226   0.000000   0.000000   0.000000   0.000000
227 -32.035775 -13.749672 -23.552315 -31.353936
228 -13.578188 -25.398016 -13.815264 -34.577391
229   0.000000   0.000000   0.000000   0.000000
230   0.000000   0.000000   0.000000   0.000000

[231 rows x 4 columns])
```

2. 步數最少且寶藏數最多的截圖

Episode25: total_steps=965, socre=5



3. Reward 和參數設定

```
[66] def get_env_feedback(S, A, path):
        global SCORE, TREASURE
        if A=='right':
            if (S % N_STATES_x == N_STATES_x - 1) or (S + 1 in WALL):
                S_ = S
                R = BLOCK_R
            elif S + 1 in TREASURE:
                S_ = S + 1
                R = TREASURE_R
                SCORE += 1
            elif S + 1 in path:
                S_ = S + 1
                R = BACK_R
            else:
                S_ = S + 1
                R = MOVE_R
        elif A=='left':
            if (S % N_STATES_x == 0) or (S - 1 in WALL):
                S_ = S
                R = BLOCK_R
            elif S - 1 in TREASURE:
                S_ = S - 1
                R = TREASURE_R
                SCORE += 1
            elif S - 1 in path:
                S_ = S - 1
                R = BACK_R
            else:
                S_ = S - 1
                R = MOVE_R
```

```
[66]        elif A=='up':
                if (S // N_STATES_x == 0) or (S - 21 in WALL):
                    S_ = S
                    R = BLOCK_R
                elif S - 21 in TREASURE:
                    S_ = S - 21
                    R = TREASURE_R
                    SCORE += 1
                elif S - 21 in path:
                    S_ = S - 21
                    R = BACK_R
                else:
                    S_ = S - 21
                    R = MOVE_R
            elif A=='down':
                if S == GOAL - 21:
                    S_ = "terminal"
                    R = TERMINAL_R
                elif (S // N_STATES_x == N_STATES_y - 1) or (S + 21 in WALL):
                    S_ = S
                    R = BLOCK_R
                elif S + 21 in TREASURE:
                    S_ = S + 21
                    R = TREASURE_R
                    SCORE += 1
                elif S + 21 in path:
                    S_ = S + 21
                    R = BACK_R
                else:
                    S_ = S + 21
                    R = MOVE_R
            return S_, R
```

依序為右、左、上、下的 action 內都給予了不同情況的下一步與 reward 指示，各別都依序為遇到邊界或牆壁就留在原地並 reward 為 BLOCK_R、遇到寶藏就移動到寶藏並 reward 為 TREASURE_R、遇到走過的路或新的空個就移動過去並分別給予 reward 為 BACK_R 與 MOVE_R。而下的 action 中還給予了遇到終點並移動到終點給予 reward 為 TERMINAL_R 的情況

```
N_STATES_x=21
N_STATES_y=11
ACTIONS=['left','right','up','down']
GOAL=230
EPSILON=0.9
ALPHA=0.1
GAMMA=0.9
MAX_EPISODES=300
#FRESH_TIME=0

TREASURE=[6,79,170,212,227]
WALL=[4,5,7,9,22,23,25,30,31,35,39,43,45,47,49,50,51,53,55,57,58,59,61,65,71,74,80,85,88,

TERMINAL_R  =  -10
TREASURE_R  =  200
BLOCK_R  =  -40
BACK_R  =  -5
MOVE_R  =  -4

SCORE=0
```

參數設定上 greedy rate、learing rate 和 discount rate 我都有嘗試過使用其他數值，但效果沒有比叫好甚至常常更差，所以最後還是用回原本的。

Reward 的參數設定上終點的設為負值可以阻止寶藏都沒拿就過早到達終點，寶藏重要性很高所以 reward 設比較大，撞牆 reward 自然也要設為負值。
然後正常移動 MOVE_R 我本來試過 0 或 1 等等，但最後發現設為負的可以減少 total_steps 而設為負數。而 BACK_R 的部分我發現倍率和 MOVE_R 相差越大容易過早收斂到不撿寶藏直接去終點，所以我的 MOVE_R 和 BACK_R 從[-1:-2]、[-2:-3]、[-3:-4]試到最後設為-5 與-4。

4. 心得

這次的作業我覺得是三次裡最有趣的，感覺很像在玩遊戲，調參數上感覺有點像在幫遊戲角色配點以達到最佳，相對之前建模型更有感覺。

Reward 的 if,else 判斷那邊就差不多那樣不太花時間，主要還是花了很多時間調參數。我一開始還忘了在更新 Q table 那邊把每輪 episode 的 score 歸零，害我看到 score 一直是 5 步數還很少還白開心。後來發現不合理改掉之後才發現原來是一下子就收斂到只去終點 score 為 0 步數才少的情況，才發現不能像我本來想的終點最重要所以 reward 設最高、寶藏低一點這樣，甚至要把終點的 reward 設為負數避免沒撿到寶藏就過去，畢竟寶藏太重要了。

我也試過觀察地圖進去寶藏的路，讓上下左右一般移動獲得的 reward 不一樣促進拿寶藏，但寶藏有 5 個而且路很多所以不太可行的樣子。

我甚至還想過把貪婪度設很低讓它都隨機行動拚拚運氣，後來發現根本不實際，還是乖乖做好 Q table 更新才是對的。

雖然我最後做的蠻差的，結果還是會讓它收斂到步數少但 score=0 的情況，感覺還靠了一點運氣在前面的 episode 找到一個還可以的，但是過程確實蠻有趣，比起之前作業更會讓我想有空再試試怎麼讓結果更好甚至激發我更多研究機器學習的興趣，。