電子商務技術-作業2

110423078何仲霖 資管碩一

作業連結：

https://colab.research.google.com/drive/1DC2kkBa1I_eXwfoFtXKcima5hPDgZYQ9?usp=sharing

一、 訓練完Q-table

```
Q-table:

(<built-in function all>,           left      right         up       down
0    -14.814894   -9.924175 -14.769811   -9.924009
1     -9.924488   -9.924033 -14.758292 -14.693053
2     -9.924521   -9.924042 -14.773547 -14.626113
3     -9.924424 -14.674890 -14.594736   -9.924161
4      0.000000    0.000000    0.000000    0.000000
..          ...         ...         ...         ...
226    0.000000    0.000000    0.000000    0.000000
227   -0.600000   -0.528679   -0.600000   -0.600000
228   -0.413305   -0.580252   -0.407089   -1.159368
229    0.000000    0.000000    0.000000    0.000000
230    0.000000    0.000000    0.000000    0.000000

[231 rows x 4 columns])
```

二、步數最少且寶藏數最多的截圖（步數 + score分數）

```
['Episode 43: total_steps=914 Score=5']
```

三、Reward 設定截圖並說明

1. 初始設定

```
N_STATES_x = 21
N_STATES_y = 11
ACTIONS = ['left','right','up','down']
GOAL = 230
EPSILON = 0.8
ALPHA = 0.1
GAMMA = 0.9
MAX_EPISODES = 350
FRESH_TIME = 0

SCORE = 0
```

先設定迷宮size以及相關資訊，Max_episodes設為350次，Epsilon設為0.8

```
Walls = (4,5,7,9,22,23,25,30,31,35,39,43,45,47,49,50,51,53,55,57,58,59,61,65,
         71,74,80,85,88,90,94,97,100,101,102,104,109,110,111,113,114,119,120,
         127,128,129,132,134,136,141,142,143,145,151,153,155,157,158,164,166,
         169,172,176,178,181,183,186,187,190,191,193,195,196,206,211,214,226,229)

Treasures = (6, 79, 170, 212, 227)
```

迷宮相關資訊，如牆壁、寶藏，用數值方式表達。

2. Reward設定

(1) 往右走

```
def get_env_feedback(S, A, path):
    # 設定Reward
    Win, Stop, Normal, Get = 70, -6, -1, 7
    global SCORE

    # 往右
    if A == 'right':
        # Goal
        if S == GOAL - 1:
            S_ = "terminal"
            R = Win
        # Wall
        elif S % N_STATES_x == N_STATES_x - 1 or S + 1 in Walls:
            S_ = S
            R = Stop
        # Treasure
        elif S + 1 in Treasures and S + 1 not in path:
            SCORE += 1
            S_ = S + 1
            R = Get
        # normal
        else:
            S_ = S + 1
            R = Normal
```

(2) 往左走

```
# 往上
elif A == 'up':
    # Goal
    if S - N_STATES_x == GOAL:
        S_ = "terminal"
        R = Win
    # Wall
    elif S // N_STATES_x == 0 or S - N_STATES_x in Walls:
        S_ = S
        R = Stop
    # Treasure
    elif S - N_STATES_x in Treasures and S - N_STATES_x not in path:
        SCORE += 1
        S_ = S - N_STATES_x
        R = Get
    # normal
    else:
        S_ = S - N_STATES_x
        R = Normal
```

(3) 往上走

```python
# 往上
elif A == 'up':
    # Goal
    if S - N_STATES_x == GOAL:
        S_ = "terminal"
        R = Win
    # Wall
    elif S // N_STATES_x == 0 or S - N_STATES_x in Walls:
        S_ = S
        R = Stop
    # Treasure
    elif S - N_STATES_x in Treasures and S - N_STATES_x not in path:
        SCORE += 1
        S_ = S - N_STATES_x
        R = Get
    # normal
    else:
        S_ = S - N_STATES_x
        R = Normal
```

(4) 往下走

```python
# 往下
elif A == 'down':
    # Goal
    if S + N_STATES_x == GOAL:
        S_ = "terminal"
        R = Win
    # Wall
    elif S // N_STATES_x == N_STATES_y - 1 or S + N_STATES_x in Walls:
        S_ = S
        R = Stop
    # Treasure
    elif S + N_STATES_x in Treasures and S + N_STATES_x not in path:
        SCORE += 1
        S_ = S + N_STATES_x
        R = Get
    # normal
    else:
        S_ = S + N_STATES_x
        R = Normal
return S_, R
```

每次不管是要走上、下、左、右，都要考慮到四個狀況，是否下一步是否抵達終點、撞到牆壁、獲得寶藏，以及正常的路線，並且設定每個狀況所可以得到的Reward，包括獎勵及懲罰，以及更新目前的狀態。

四、心得

這次的作業花了很多時間在思考該如何設計Reward才能讓學習的效果更好，因為給予適當的Reward才能幫助電腦在學習過程中知道哪些是正確的決定，在參數的設定上也試過很多種組合，才能夠有效率的有效降低抵達終點所需要的總步數，並且也考慮到為了找到更多寶藏，稍微繞一點路市值得的資訊，也嘗試給予不同的Epsilon，確保一定的隨機性，都可能會影響到其學習的結果。這次的作業很有挑戰性，但過程中也學到很多，知道好的Reward設計會很直接影響結果，對於強化學習的應用也更有概念，只能透過不斷的嘗試，以找到最佳的方式來進行訓練。