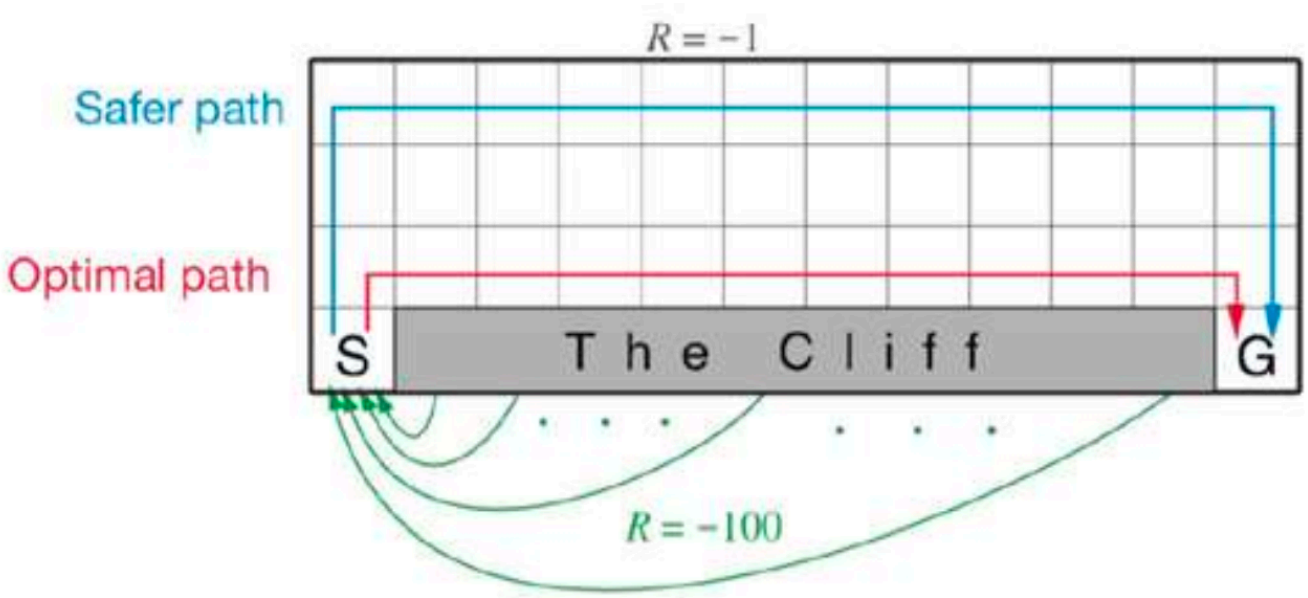
Assignment 2

21307035 邓栩瀛

1、实验内容

基于Cliff Walk例子实现SARSA、Q-learning算法

- Consider the gridworld shown below
- This is a standard undiscounted, episodic task, with start and goal states, and the usual actions causing movement up, down, right, and left.
- Reward is -1 on all transitions except those into the region marked as Cliff. Stepping into this region incurs a reward -100 and sends the agent instantly back to the start.



2、核心思路

在该问题中，对每个状态进行编号，并建立相应的Q-table，在每次的迭代中，初始化状态为角色位于起点，每个状态有四个动作（上下左右），每一步转移的reward都设为-1，若采取某一动作后若到达了边界则待在原地，若到达了悬崖则返回起点，reward设为-100，并设置一个符号变量flag来判断某次迭代是否终止，若到达了终点则该次迭代终止，保留Q-table，然后重新开始。

2-1 Q-learning

Q-learning算法，该算法的伪代码如下

Algorithm 1 Q-learning

Initialize $Q(s, a)$ arbitrarily
Initialize s
Loop
$a \leftarrow$ probabilistic outcome of behavior policy derived from $Q(s, a)$
Take action a, observe reward r and next state s'
$Q_{t+1}(s, a) \leftarrow Q_t(s, a) + \alpha(r + \gamma max_{a'} Q_t(s', a') - Q_t(s, a))$
$s \leftarrow s'$
End Loop
$\forall_{s \in S}\, \pi(s) \leftarrow \arg\max_a Q_t(s, a)$

- Q(s, a)：Q值函数，用于表示状态 s 下采取动作 a 的价值估计。初始时可以任意设定（通常设为零），随着算法迭代，Q值将逐渐更新以接近最优值。
- s：当前的状态，表示智能体在环境中的位置或条件。初始时随机选择一个起始状态。
- a：在当前状态 s 下选择的动作。根据行为策略（例如ε-贪心策略）从Q值中选出动作 a 。
- r：执行动作 a 后获得的即时奖励。它反映了该动作带来的短期收益，可能为正、负或零。
- s′：在状态 s 下执行动作 a 后到达的下一状态。即从当前状态过渡到的新的状态。
- α：学习率，控制Q值更新的步长。值在0到1之间，越接近1表示更依赖新信息，越接近0表示更依赖旧信息。
- γ：折扣因子，用于控制未来奖励的权重。值在0到1之间，越接近1表示更看重长期回报，越接近0表示更关注即时回报。
- $max_{a\prime}Q(s\prime, a\prime)$：在下一状态 $s\prime$ 下可能采取的所有动作中，选择能够产生最高Q值的动作 $a\prime$ 的Q值。
- $\pi(s)$：策略，表示在每个状态 s 下选择的动作。通过选择使 Q(s, a) 最大化的动作来构建最终策略。

2-2 SARSA

SARSA算法公式

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)\right] \tag{1}$$

相应的伪代码如下

Algorithm 2: SARSA Learning Algorithm

Initialize Q arbitrarily, $Q(_{\text{terminal}}) = 0$
repeat
    Initialize s_t
    choose a ε − greedily
    repeat
        Take action a_{t+1}, *go to* s_{t+1}, observe r_{t+1}
        Choose a_{t+1} ε − greedily
        $Q(s_{t+1}, a_{t+1}) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$
        $s_t \leftarrow s_{t+1}, a_t \leftarrow a_{t+1}$
    until s_t is terminal
until convergence

- Q：Q值函数，用于表示状态-动作对的价值估计。初始时可以任意设定，通常为零；终止状态的Q值设为0。
- s_t：当前状态，表示智能体在环境中的位置或条件。在每个新的回合中，智能体从某一初始状态 s_t 开始。
- a_t：当前动作，智能体在状态 s_t 下选择的动作。此动作由ε-贪心策略选择，这意味着在探索和利用之间进行平衡，通常随机选择一个动作（ε的概率）或选择当前Q值最大的动作（1-ε的概率）。
- r_{t+1}：即时奖励，智能体在执行动作 a_{t+1} 并转移到新状态 s_{t+1} 后获得的奖励。

- s_{t+1}：下一状态，在执行动作后智能体到达的状态。
- a_{t+1}：在新状态 s_{t+1} 下选取的下一动作，使用ε-贪心策略选择。
- α：学习率，控制Q值更新的步长，取值范围在0到1之间。较大的α值让算法更快适应新的奖励，而较小的α值使算法更依赖历史数据。
- γ：折扣因子，控制未来奖励的权重。范围在0到1之间，γ越接近1，表示未来的奖励对当前决策的影响越大。

3、关键代码

3-1 Q-learning

```
def q_learning():
    Q_table = initialize_q_table()  # 初始化Q值表
    clif_map = np.zeros((rows, cols))  # 创建一个地图表示矩阵，记录智能体的路径
    reward_list = []  # 用于存储每一回合的总奖励

    for _ in range(episodes):  # 迭代多轮训练
        pos, total_reward = (3, 0), 0  # 初始化智能体的位置为起始位置，累计奖励为0
        end = False
        clif_map.fill(0)  # 每次新回合开始时，重置地图

        while not end:  # 在到达终止状态之前
            clif_map[pos[0], pos[1]] = 1  # 标记当前智能体所在位置
            action = epsilon_greedy(Q_table, pos)  # 选择动作，基于ε-贪心策略
            next_pos = move(pos, action)  # 执行动作，得到下一个位置
            next_pos, reward, end = calculate_reward(next_pos)  # 根据新的位置计算奖励和是否结束

            max_q_next = max_q_value(Q_table, next_pos)  # 获取下一状态的最大Q值
            idx = pos[0] * cols + pos[1]  # 计算当前状态在Q表中的索引
            # 更新Q值
            Q_table[idx, action] += alpha * (reward + gamma * max_q_next - Q_table[idx, action])

            pos = next_pos  # 更新当前状态
            total_reward += reward  # 累加奖励

        reward_list.append(total_reward)  # 记录每回合的总奖励
    return clif_map, Q_table, reward_list  # 返回路径地图、Q表、奖励列表
```

3-2 SARSA

```
def sarsa():
    Q_table = initialize_q_table()  # 初始化Q值表
```

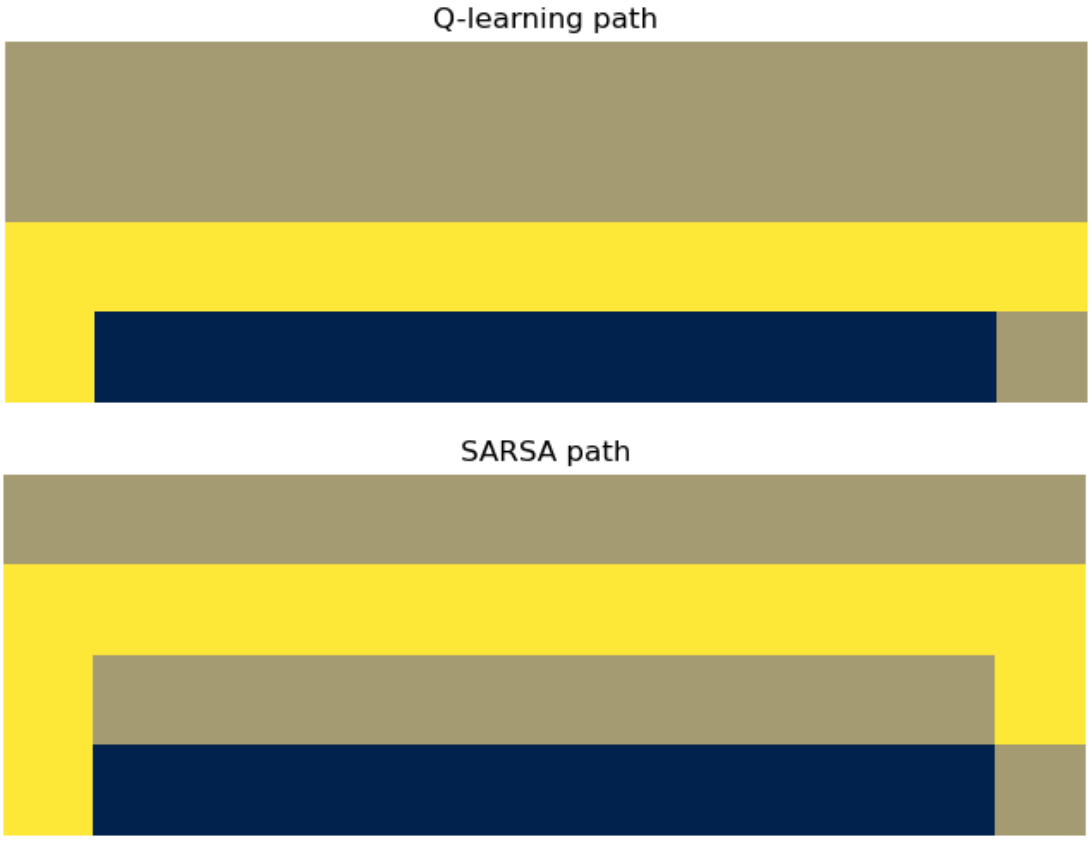
```
    clif_map = np.zeros((rows, cols))  # 创建一个地图矩阵，记录智能体的路径
    reward_list = []  # 用于存储每回合的总奖励

    for _ in range(episodes):  # 迭代多轮训练
        pos, total_reward = (3, 0), 0  # 初始化智能体的位置为起始位置，累计奖励为0
        end = False
        clif_map.fill(0)  # 每回合开始时重置地图
        action = epsilon_greedy(Q_table, pos)  # 使用ε-贪心策略选择初始动作

        while not end:  # 在到达终止状态之前
            clif_map[pos[0], pos[1]] = 1  # 标记当前智能体所在位置
            next_pos = move(pos, action)  # 执行动作，得到下一个位置
            next_pos, reward, end = calculate_reward(next_pos)  # 根据新位置计算奖励和是否结束

            next_action = epsilon_greedy(Q_table, next_pos)  # 根据ε-贪心策略选择下一个动作
            idx = pos[0] * cols + pos[1]  # 计算当前状态在Q表中的索引
            # 使用SARSA更新Q值
            Q_table[idx, action] += alpha * (
                reward + gamma * Q_table[next_pos[0] * cols + next_pos[1], next_action]
- Q_table[idx, action]
            )

            pos, action = next_pos, next_action  # 更新当前状态和动作
            total_reward += reward  # 累加奖励

        reward_list.append(total_reward)  # 记录每回合的总奖励
    return clif_map, Q_table, reward_list  # 返回路径地图、Q表、奖励列表
```

4、实验结果

4-1 实验参数设置

```
episodes = 1000 # 迭代次数
epsilon = 0.05 # 选择随机方向的概率
alpha = 0.3 # 学习率
gamma = 0.99 # 衰减值
```

4-2 路径图



黄色为路径，蓝色为悬崖，其余部分为可走区域

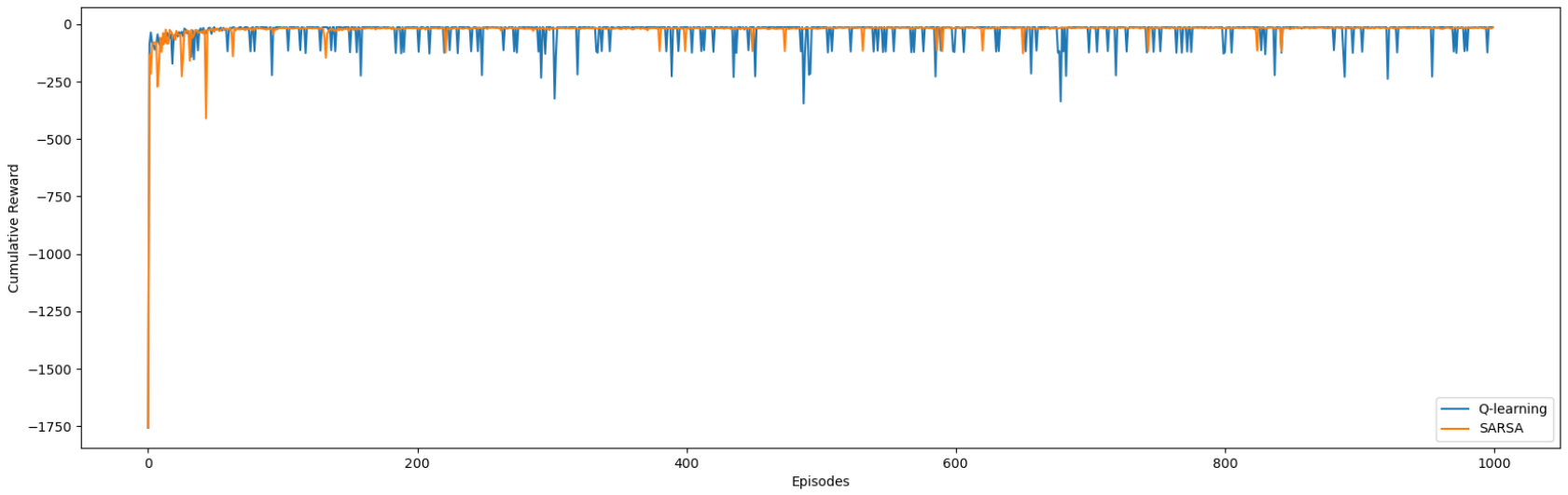
4-3 Q-table

```
Q-learning Q-table:
 [[ -11.1837634   -11.39226316  -11.05712282  -11.14376226]
 [ -10.69089307  -10.72556909  -10.95356164  -10.67450069]
 [ -10.12290449  -10.35549532  -10.3288953   -10.09875288]
 [  -9.43920544   -9.49200327   -9.63897875   -9.40203454]
 [  -8.83141349   -8.83545609   -8.97886086   -8.65380117]
 [  -7.83494684   -8.0363192    -8.24335924   -7.87355957]
 [  -7.08178638   -7.1878763    -7.19021217   -7.09778549]
 [  -6.39619938   -6.38310865   -6.91921073   -6.27557612]
 [  -5.54868518   -5.4434487    -5.54659773   -5.46521152]
 [  -4.68160898   -4.655566     -4.76838465   -4.64596667]
 [  -3.83056644   -3.81881979   -3.90662564   -3.81496501]
 [  -2.95982231   -2.94514172   -3.22746339   -2.95982231]
 [ -11.52736961  -11.61612655  -11.71609013  -11.64177   ]
 [ -11.04674716  -11.13333087  -11.19504663  -11.08307028]
 [ -10.55441085  -10.31897717  -10.75592951  -10.321617  ]
 [  -9.61749281   -9.50871042   -9.51778304   -9.49352948]
 [  -8.77373291   -8.62868288   -8.95507047   -8.6218661 ]
 [  -7.94704395   -7.71415314   -7.86316875   -7.71508911]
 [  -7.17651703   -6.78940825   -6.83649421   -6.78799856]
 [  -6.28187695   -5.84983951   -6.18652769   -5.84921858]
 [  -5.1606632    -4.89974033   -5.28879852   -4.89992748]
 [  -4.07020296   -3.94015516   -3.95134962   -3.9401473 ]
 [  -3.57080873   -2.97007127   -3.09626665   -2.97007545]
 [  -2.32786739   -1.98999999   -2.14622461   -2.34786363]
 [ -12.1802783   -13.11073834  -12.24166426  -11.36151283]
 [ -11.74940622 -110.91893313  -12.2426801   -10.46617457]
```

```
 [ -11.00596729 -111.27024705  -11.35321794   -9.5617925 ]
 [ -10.28400143 -108.66261511  -10.42746294   -8.64827525]
 [  -9.48638087 -111.98984973   -9.55644511   -7.72553056]
 [  -8.60117026 -110.39946593   -8.64629749   -6.79346521]
 [  -7.69418904 -110.48183111   -7.72418809   -5.85198506]
 [  -6.72152868 -111.84062496   -6.78117254   -4.90099501]
 [  -5.75716305 -112.07198367   -5.83479942   -3.940399  ]
 [  -4.88117255 -111.88890999   -4.8663336    -2.9701    ]
 [  -3.93475705 -110.98515286   -3.93281816   -1.99      ]
 [  -2.89377855   -1.           -2.94509184   -1.98735669]
 [ -12.2478977   -13.11771617  -13.12076527 -112.0543145 ]
 [   0.            0.            0.            0.        ]
 [   0.            0.            0.            0.        ]
 [   0.            0.            0.            0.        ]
 [   0.            0.            0.            0.        ]
 [   0.            0.            0.            0.        ]
 [   0.            0.            0.            0.        ]
 [   0.            0.            0.            0.        ]
 [   0.            0.            0.            0.        ]]
SARSA Q-table:
 [[ -15.22935803  -15.0655896   -15.28812187  -14.41846303]
 [ -14.32796734  -14.90934442  -14.66380236  -13.14583061]
 [ -13.37657761  -13.99689765  -14.18946761  -12.22295392]
 [ -12.08071654  -13.40082485  -13.36679494  -11.16552985]
 [ -10.82880926  -11.69885498  -12.32963863   -9.99412344]
 [ -10.08745793   -9.53193281  -11.54932237   -8.97697425]
 [  -8.994509     -8.85416048  -10.28837743   -7.91982666]
 [  -8.00787281   -6.88496881   -9.51636023   -7.7551152 ]
 [  -7.331607     -6.16996942   -8.17329627   -6.74168314]
 [  -5.92271648   -5.13273774   -7.0135004    -5.42760407]
 [  -4.97360794   -4.73816558   -5.92481551   -4.28301133]
 [  -4.40744743   -3.02475479   -4.73025355   -4.06230291]
 [ -15.46869118  -16.68128873  -15.66167004  -15.11153429]
 [ -14.23372858  -15.33059598  -15.53157345  -13.98187279]
 [ -12.89167555  -14.4762983   -14.75971697  -14.28441048]
 [ -11.79839454  -20.07749951  -13.33851702  -13.22964934]
 [ -10.93971485  -14.20407024  -11.99314927  -12.02551375]
 [  -9.73820601  -11.71874688  -10.32317406   -8.5504373 ]
 [  -8.39400063   -9.44475567   -8.65288601   -7.2254021 ]
 [  -7.91087619   -8.04719725   -8.49297128   -5.88892715]
 [  -7.32017368   -7.50458964   -7.63086201   -4.91181913]
 [  -6.09330011   -6.24799221   -6.5491248    -3.94278386]
 [  -5.18398211   -4.39564317   -4.86414834   -2.97053229]
 [  -4.22612377   -2.56547844   -4.51726705   -3.04603415]
 [ -16.0107241   -17.49193216  -16.5016158   -32.40733774]
 [ -14.73423151  -84.95266663  -15.15206648  -15.07512239]
 [ -13.59667073  -52.36035357  -14.15466879  -14.42014097]
```

```
[ -11.6362963   -81.97509401  -11.98237107  -11.73351931]
[ -11.51650942  -55.07772684  -11.5433213   -23.65858438]
[  -9.56814959  -73.54623616   -9.88043202   -9.218615  ]
[  -8.51383587  -52.81769413   -8.67690399   -8.25155976]
[  -7.76776711  -30.57550608   -7.96548046   -7.20397676]
[  -6.61417151  -30.63170167   -7.58505404  -16.3768677 ]
[  -5.15346952  -79.03042798   -6.04402231  -11.58834522]
[  -2.76168906  -55.88440585  -16.339139     -1.99451966]
[  -3.77274621   -1.           -3.01790872   -2.03675106]
[ -16.83546912  -17.52049002  -17.73508366 -116.14078579]
[   0.            0.            0.            0.        ]
[   0.            0.            0.            0.        ]
[   0.            0.            0.            0.        ]
[   0.            0.            0.            0.        ]
[   0.            0.            0.            0.        ]
[   0.            0.            0.            0.        ]
[   0.            0.            0.            0.        ]
[   0.            0.            0.            0.        ]
[   0.            0.            0.            0.        ]
[   0.            0.            0.            0.        ]]
```

4-4 reward总和变化图



在on-line训练中，Q-learning的表现要比SARSA差，但Q-learning最后得到的结果比SARSA要好，通过分析可以看出两个算法的区别，SARSA是一种on-policy算法，Q-learning是一种off-policy算法。SARSA选取的是一种保守的策略，在更新Q值的时候已经为未来规划好了动作，对错误和死亡比较敏感。而Q-learning每次在更新的时候选取的是最大化Q的方向，而当下一个状态时，再重新选择动作，Q-learning是一种鲁莽、大胆、贪婪的算法，对于死亡和错误并不在乎。