# HW2

## 一、环境配置

### 1.1 conda环境创建

`conda create -n gym2 python=3.8`

### 1.2 安装依赖

numpy, matplotlib, tqdm按照文档安装即可

gym根据文档安装0.24.0版本，但是运行代码的时候报错reset返回值是一个int而不是tuple，推测是API版本过低，gym升级到最新版即可

## 二、Blank fill

### 2.1 update_q_value

```python
def update_q_value(q, q_next, reward, alpha, gamma):
    """
    TODO:
    Please fill in the blank for variable 'td_target'
    according to the definition of the TD method
    """
    td_target = reward + gamma * q_next
    return q + alpha * (td_target - q)
```

sarsa和Q-learning都是表格型时序差分方法，策略评估为更新状态-动作值函数：

SARSA：

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(R_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t))$$

Q-learning：

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(R_t + \gamma \max_{a'} Q(s_{t+1}, a'_{t+1}) - Q(s_t, a_t))$$

其中reward为奖励，gamma为折扣因子，填入即可

### 2.2 sarsa

```python
"""
            TODO:
            Please fill in the blank for variable 'next_q_value'
            according to the definition of the SARSA algorithm
            """
            next_q_value = q_values[next_state, next_action]
            q_values[state, action] = update_q_value(
                q_values[state, action],
                0 if done else next_q_value,
                reward, config.alpha, config.gamma)
```

根据SARSA的on-policy时序差分算法：

**Sarsa: An on-policy TD control algorithm**

Initialize $Q(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$
Repeat (for each episode):
  Initialize $S$
  Choose $A$ from $S$ using policy derived from $Q$ (e.g., $\epsilon$-greedy)
  Repeat (for each step of episode):
    Take action $A$, observe $R$, $S'$
    Choose $A'$ from $S'$ using policy derived from $Q$ (e.g., $\epsilon$-greedy)
    $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma Q(S',A') - Q(S,A)\big]$
    $S \leftarrow S'$; $A \leftarrow A'$;
  until $S$ is terminal

此处选择下一个状态的Q值填入即可

## 2.3 q_learning

```
"""
            TODO:
            Please fill in the blank for variable 'next_q_value'
            according to the definition of the Q-learning algorithm
            """
            next_q_value = np.max(q_values[next_state])
            q_values[state, action] = update_q_value(
                    q_values[state, action],
                    0 if done else next_q_value,
                    reward, config.alpha, config.gamma)
```

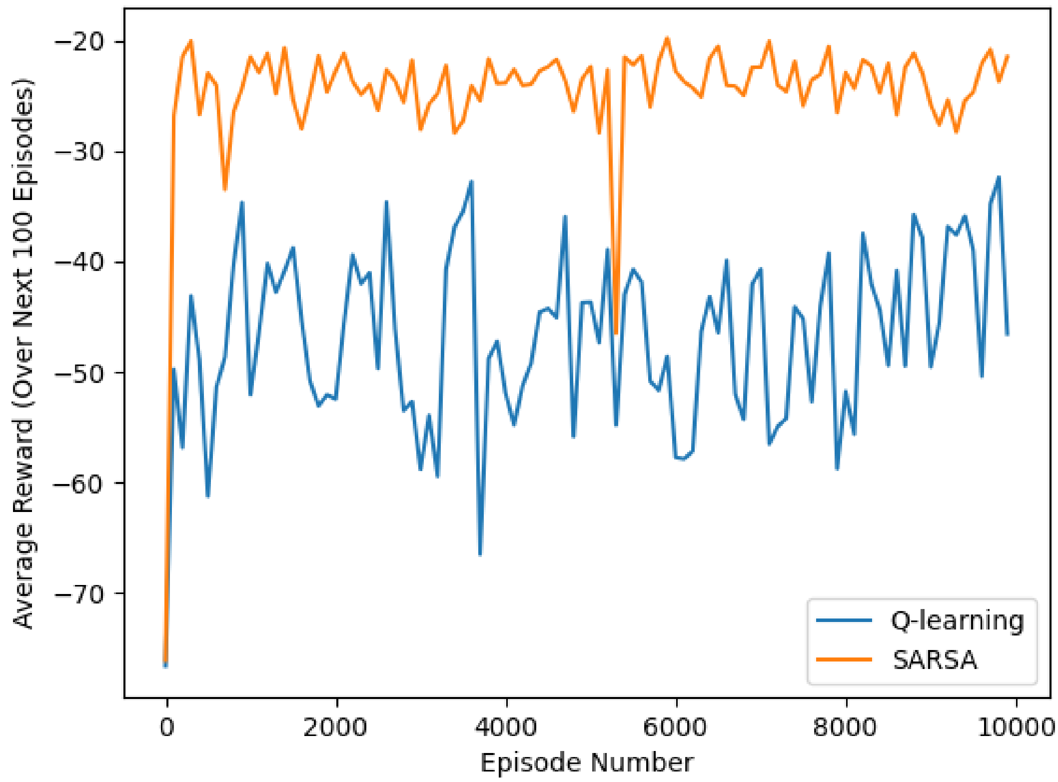
对于Q-learning算法，next_q_value是下一状态所有动作的最大Q值。

# 三、Test

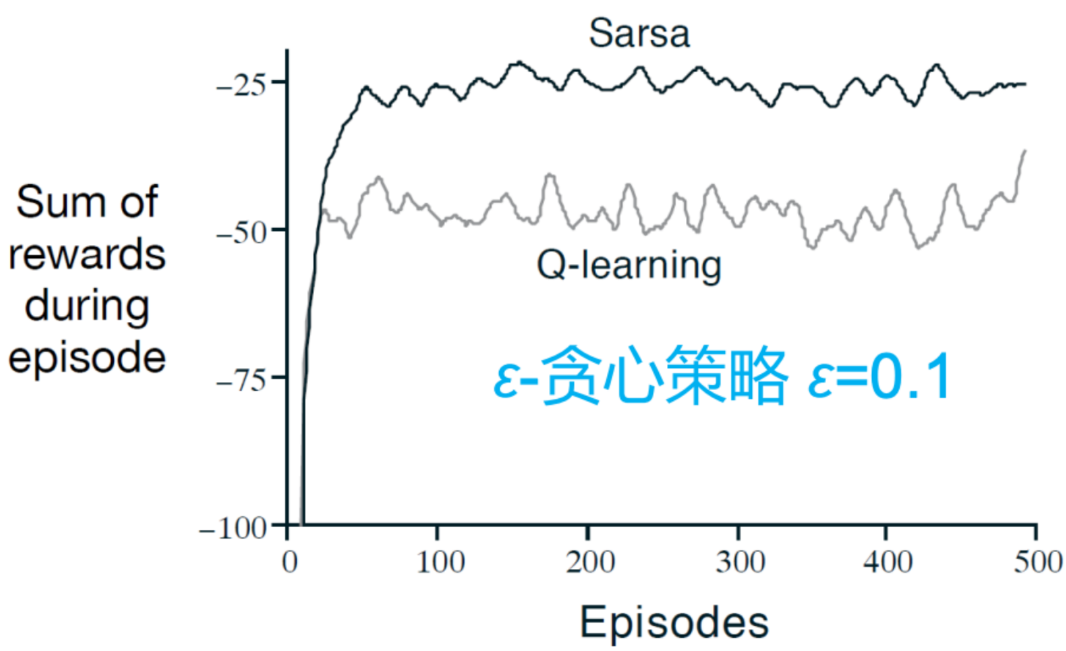
## 3.1 CliffWalking

测试结果如下：

```
(gym2) PS E:\ZJU2024-MARL-JUEWU> python .\ex2_cliff-walking.py
Discrete(4)
Discrete(48)
100%|████████████████████████████████████████| 10000/10000 [00:05<00:00, 1729.27it/s]
Best Average Reward over 100 Episodes:  -13
100%|████████████████████████████████████████| 10000/10000 [00:06<00:00, 1585.93it/s]
Best Average Reward over 100 Episodes:  -17
```




可以看到结果和教材给出的大体相同：

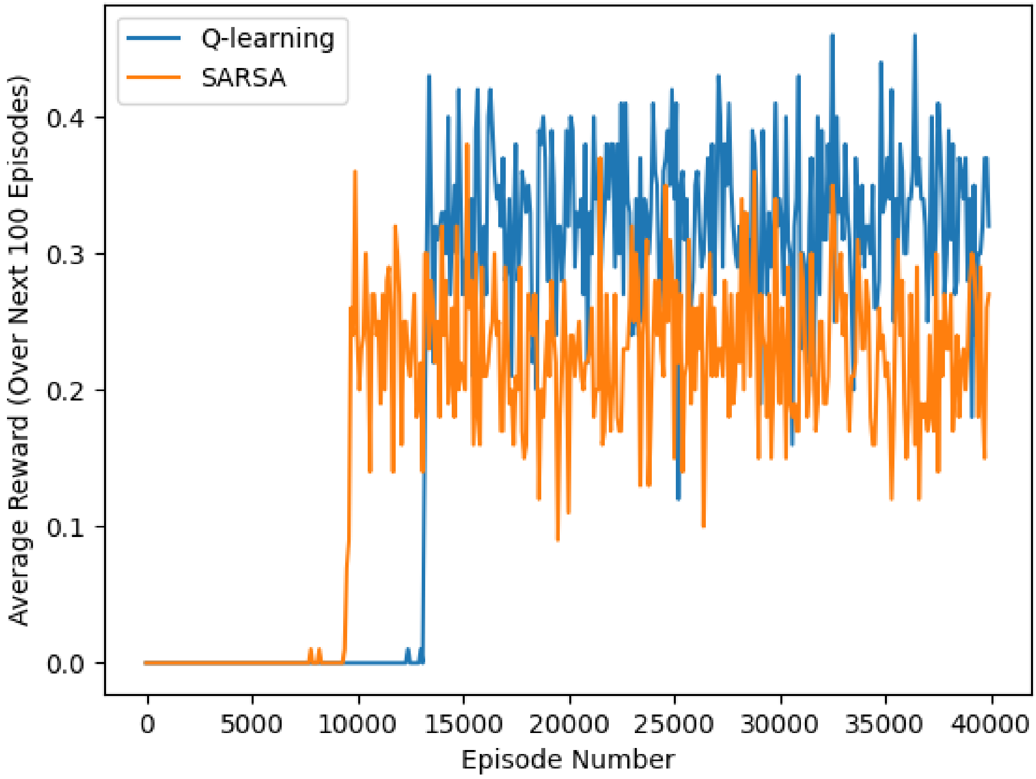



## 3.2 FrozenLake

修改代码如下：

```
env = gym.make('FrozenLake-v1')
config = Configs(env, max_timestep=200, num_episode=40000, plot_every=100)
```

测试结果如下：




```
(gym2) PS E:\ZJU2024-MARL-JUEWU> python .\ex2_cliff-walking.py
Discrete(4)
Discrete(16)
100%|████████████████████████████████████████| 40000/40000 [00:33<00:00, 1185.53it/s]
Best Average Reward over 100 Episodes:  1.0
100%|████████████████████████████████████████| 40000/40000 [00:25<00:00, 1583.88it/s]
Best Average Reward over 100 Episodes:  1.0
```