中山大学计算机学院

人工智能本科生实验报告

课程名称：Artificial Intelligence

教学班级	网安软工合班	专业（方向）	网络空间安全
学号	20337251	姓名	伍建霖

一、实验题目

实验任务

- 在给定迷宫环境中实现Q-learning和Sarsa算法。

报告提交要求

- 提交一个压缩包。压缩包命名为：“学号_姓名_作业编号”，例如：20220525_张三_实验10。
- 压缩包包含三部分：code文件夹和实验报告pdf文件
  - Code文件夹：存放实验代码
  - Pdf文件格式参考发的模板
- 如果需要提交新版本，则在压缩包后面加_v1等。如“学号_姓名_作业编号_v1.zip”，以此类推。
- 截止日期：2022年6月15日23点59分
- 提交邮箱：zhangyc8@mail2.sysu.edu.cn

二、实验内容

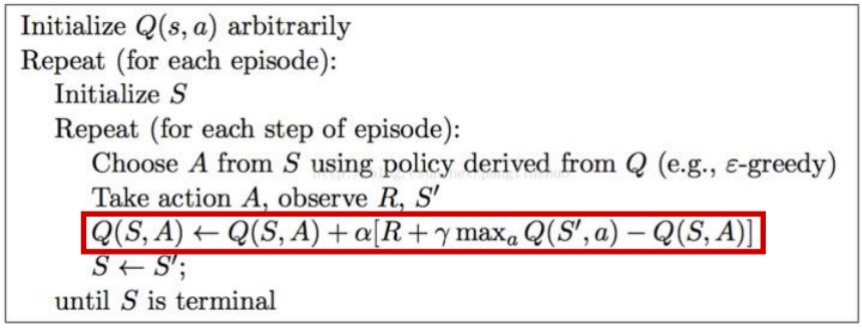
1.算法原理

choose_action

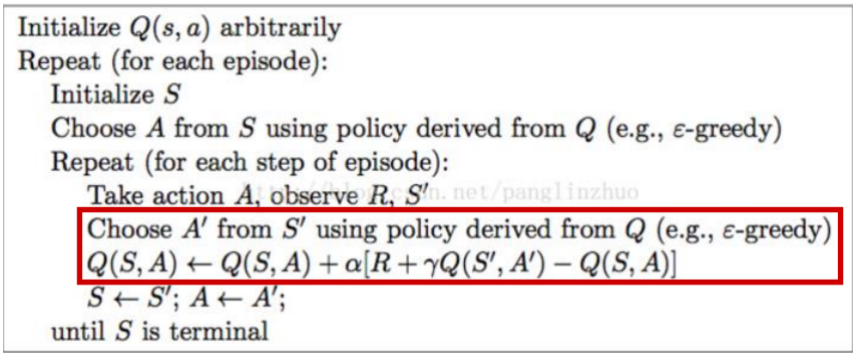
qlearning和sarsa都是根据$\epsilon-greedy$策略来选择动作，有ϵ的概率随机选择一个动作，有1-ϵ的概率选择当前状态下的最优动作

Q-learning_learning部分

qlearning根据greedy策略来更新Q表，不过我更愿意用offpolicy来表示qlearning的Q表更新策略，qlearning用下一状态的最优动作的Q值来代替下一状态的V值，而不像Sarsa，用和当前相同的策略(即用choose_action函数)来求出下一步的动作(onpolicy)，最后用贝尔曼方程更新Q表



Sarsa_learning部分



2. 伪代码

```
def qlearning:
    初始化Q表
    重复episode次:
        初始化state
        重复step次:
            choose_action()
            走出这一步，获得下个状态以及奖励值R
            Q(S, A)<-Q(S, A)+a[R+gmax(Q(S'+a))-Q(S, A)]
            swap S和S'
        若state为terminal则break

def sarsa:
    初始化Q表
    重复episode次:
        初始化state
        A = choose_action(cur_state)
        重复step次:
            走出这一步，获得下个状态以及奖励值R
            A' = choose_action(next_state)
            Q(S, A)<-Q(S, A)+a[R+gQ(S'+A')-Q(S, A)]
            swap S和S', A和A'
    若state为terminal则break
```

3.关键代码展示（带注释）

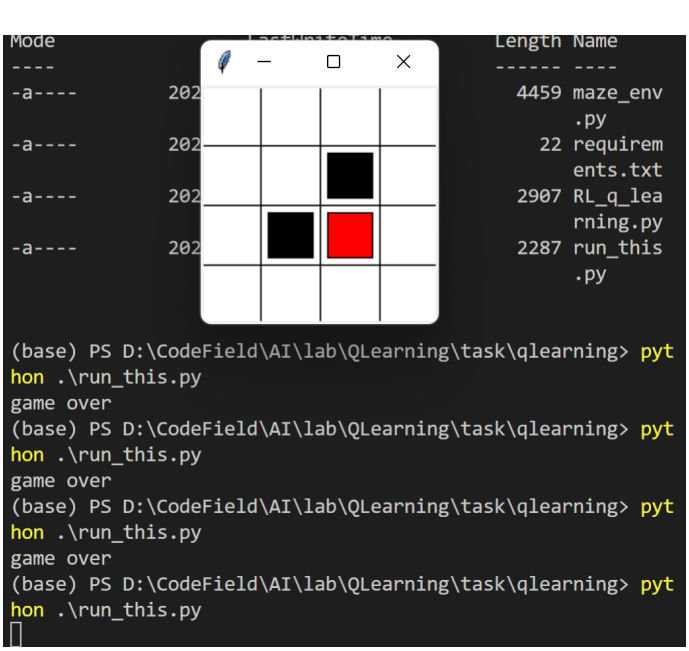
```
def choose_action(self, observation):
        ''' choose action from q table '''
        ############################

        # YOUR IMPLEMENTATION HERE #
        # 判断当前state是否在已知的状态中
        self.check_state_exist(observation)
```

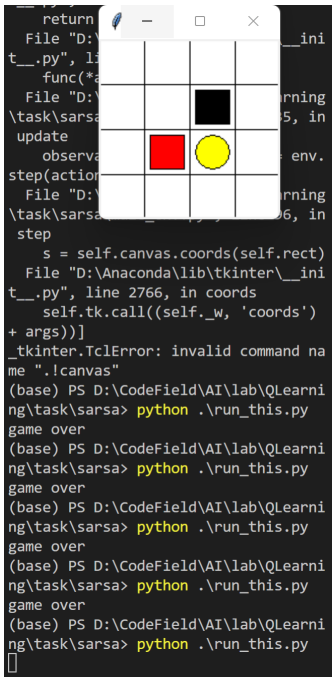
```
        """
        使用epsilon-greedy策略选择动作:
        epsilon的概率选择最优动作
        1-epsilon的概率随机选择动作
        """
        if np.random.uniform() < self.epsilon:
            state_action = self.q_table.loc[observation, :]
            """
            可能存在多个动作相同Q值
            则从这些动作中随机选择一个即可
            """
            action = np.random.choice(
                state_action[state_action == np.max(state_action)].index)
        else:
            action = np.random.choice(self.actions)
        return action

# qlearning
def learn(self, s, a, r, s_):
        ''' update q table '''
        ############################

        # YOUR IMPLEMENTATION HERE #
        """
        使用greedy策略
        off-policy, 从表的最大值中选
        """
        # 判断当前state是否在已知的状态中
        self.check_state_exist(s_)
        if s_ != 'terminal':
            # 未达到最终状态
            # qtarget = R + g*max(Q(S', a))
            q_target = r + self.gamma * self.q_table.loc[s_, :].max()
        else:
            # 达到最终状态
            q_target = r
        # 使用贝尔曼方程更新
        self.q_table.loc[s, a] = self.q_table.loc[s, a] + \
            self.lr * (q_target - self.q_table.loc[s, a])

# sarsa
def learn(self, s, a, r, s_, a_):
        ''' update q table '''
        ############################

        # YOUR IMPLEMENTATION HERE #
        """
        使用epsilon-greedy策略
        on-policy, 使用和上一步相同的策略
        """
        self.check_state_exist(s_)
        if s_ != 'terminal':
            # 未达到最终状态
            # qtarget = R + g*Q(S', a')
            q_target = r + self.gamma * self.q_table.loc[s_, a_]
        else:
            q_target = r
        # 使用贝尔曼方程更新
```

```
            self.q_table.loc[s, a] = self.q_table.loc[s, a] + \
                self.lr * (q_target - self.q_table.loc[s, a])
```

三、实验结果及分析

1. 实验结果展示示例（可图可表可文字，尽量可视化）





2.评测指标展示及分析（其它可分析运行时间等）

Qlearning

当epsilon为0.3时，最后难以到达奖励值最高的地方

当epsilon为0.5和0.9时，基本都能到奖励值最高的地方

Sarsa

当epsilon大于0.4时，最后的行为绝大部分都是左右横跳

当epsilon小于0.3时，最后大部分都能走到奖励值最高的地方

四、参考资料

怎样正确理解马尔科夫链？ - 知乎 (zhihu.com)

如何理解强化学习中的Q值和V值？ - 知乎 (zhihu.com)

[理论篇]怎样直观理解Qlearning算法？ - 知乎 (zhihu.com)

pd.Series()函数解析（最清晰的解释）_我是管小亮的博客-CSDN博客pd.series()

https://morvanzhou.github.io/tutorials/