电子科技大学

计算机科学与工程学院

标 准 实 验 报 告

（实验）课程名称 人工智能

电子科技大学教务处制表

电 子 科 技 大 学

实 验 报 告

学生姓名：朱若愚　　学 号：2022150501027　　指导教师：段立新

实验地点：A2-413-1　　实验时间：2024.5.25

一、实验室名称：　　计算机学院实验中心

二、实验项目名称：MDP 实验

三、实验学时：5 学时

四、实验原理：

（1）迷宫游戏说明

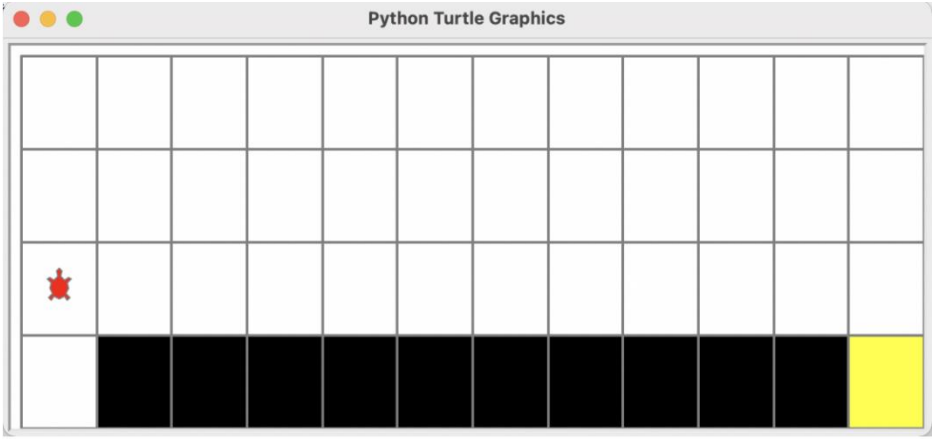


图 1 迷宫游戏示例

如上图所示，本实验所研究的迷宫由一个二维表格构成。其中白色区域是可行走区域，黑色区域是陷阱区域，黄色区域是终点。规则如下：

1） 智能体从左下角出发，到达黄色区域即游戏成功。
2） 智能体每次可选择上、下、左、右四种移动动作，每次动作得到-1 奖励。
3） 智能体不能移动出网络，如果下一步的动作命令会让智能体移动出边界，那么这一步将不会执行，即智能体原地不动，得到-1 奖励。
4） 智能体移动到黑色区域，得到-100 奖励。
5） 智能体移动到黄色区域，该回合结束。

由图可知，最优的路线需要 13 步，因此最后智能体获得的奖励指在-13 左右为最佳结果。

（2）Q-Learning 算法

Q-Learning 是一种记录行为值（Q value）的方法，每种行为在一定的状态都会有一个值 Q(s,a)，就是说行为 a 在 s 状态的值是 Q（s,a）。对于迷宫游戏，s 就是当前 agent 所在的地点了。每一步，智能体可以选择四种动作，所以动作 a 有四种

可能性。
在本实验里，已经提供了强化学习基本的训练接口，只需要实现 Q 表格的强化学习方法即可。算法框架如下：

Initialize $Q(s,a)$ arbitrarily
Repeat (for each episode):
  Initialize s
  Repeat (for each step of episode):
    Choose a from s using policy derived from Q (e.g., ε-greedy)
    Take action a, observe r, s'
    $Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right]$
    $s \leftarrow s'$;
  until s is terminal

图 2 强化学习算法框架
可以看到，算法的核心部分是更新 Q 表格。训练目标是在评价阶段，智能体的平均奖励达到-13。做出可视化结果如下图。

图 3 Q-Learning 训练奖励曲线

五、实验目的：

实验使用 Q 表格方法解决迷宫寻路问题，理解强化学习算法原理。

六、实验内容：

阅读代码，补全代码中缺失的部分，完成迷宫实验。

七、实验器材（设备、元器件）：

PC 微机一台

八、实验步骤：

1. choose_action 函数：

```python
    def choose_action(self, state):
        max_epsilon = 0.7
        min_epsilon = 0.05

        self.sample_count += 1
        self.epsilon = min_epsilon + (max_epsilon - min_epsilon) * np.exp(
            -0.0005 * self.sample_count
        )
        if np.random.uniform() > self.epsilon:
            action = self.predict(state)
        else:
            action = np.random.choice(self.action_dim)
        return action
```

使用贪心策略，首先计算 ε 值，在前期鼓励探索，在后期倾向于记忆。之后生成一个随机数，若该随机数大于 ε 则选择 Q 值较大的行为，否则随机。

2. predict 函数：

```python
    def predict(self, state):
        Q_list = self.Q_table[state, :]
        Q_max = np.max(self.Q_table[state, :])
        action_list = np.where(Q_list == Q_max)[0]
        action = np.random.choice(action_list)
        return action
```

首先获取了给定状态下所有动作的 Q 值并找到最大值，然后找出了所有 Q 值等于最大 Q 值的动作并从中随机选择一个动作。

3. update 函数：

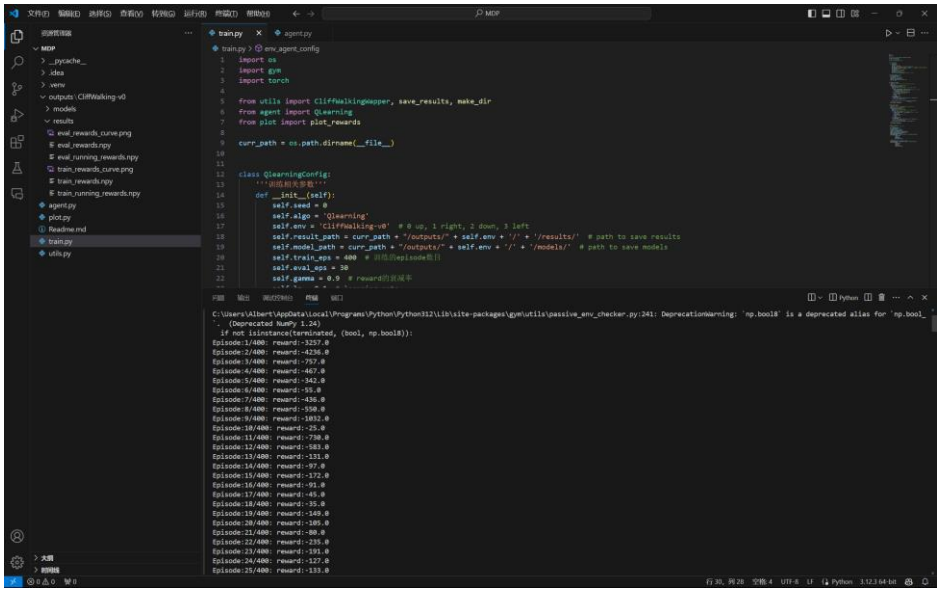
```python
    def update(self, state, action, reward, next_state, done):
        Q_predict = self.Q_table[state, action]
        if done:
            Q_target = reward
        else:
            Q_target = reward + self.gamma * np.max(self.Q_table[next_state, :])
        self.Q_table[state, action] += self.lr * (Q_target - Q_predict)
```

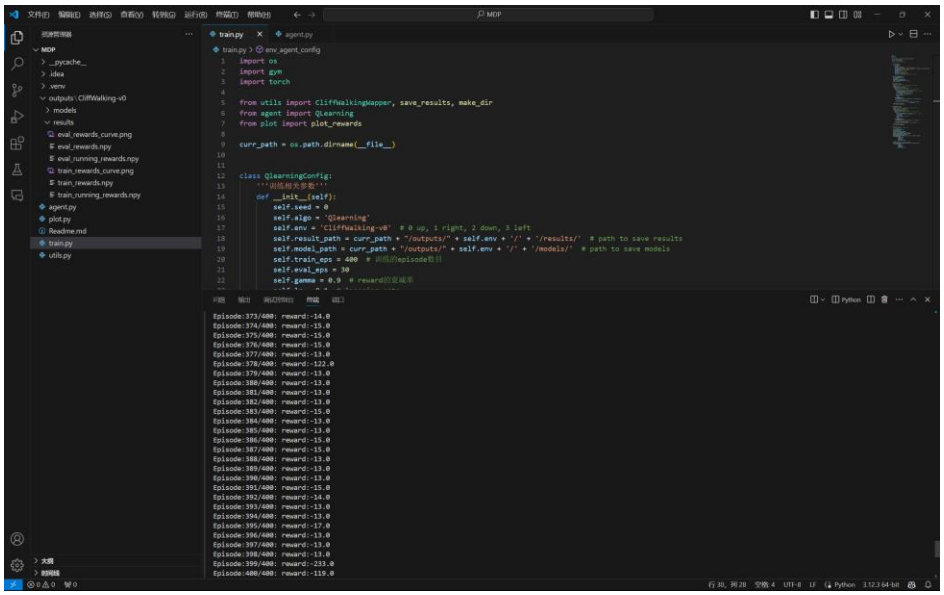
根据是否完成计算 Q 值。如果完成，Q 值就是当前的奖励，否则以当前的奖励与下一个状态的最大 Q 值之和乘以衰减系数。最后根据学习率和目标 Q 值与预测 Q 值的差距，更新 Q 表。

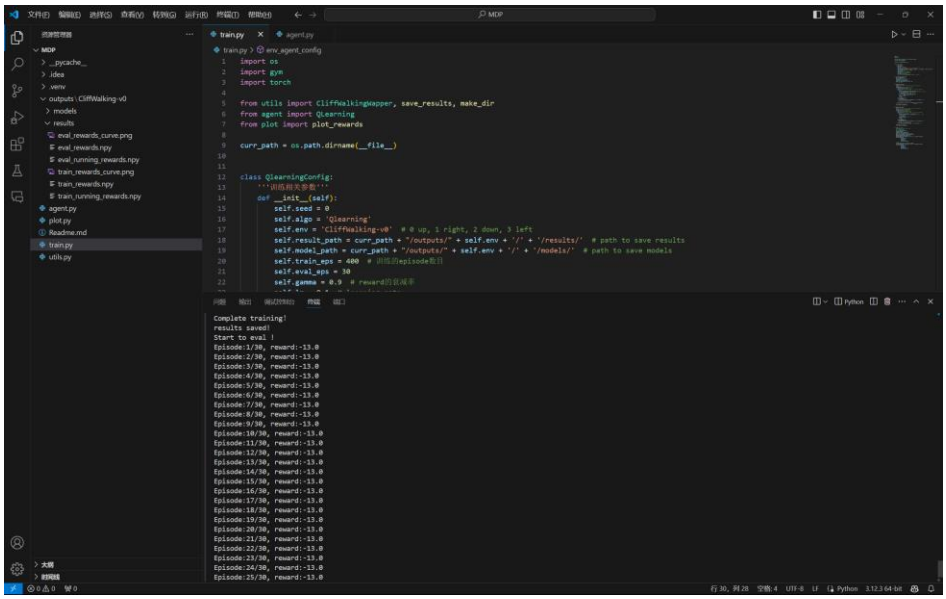
4. 增多了 self.train_eps 的值，训练数目翻倍，并且由于笔者电脑显卡具有 cuda 核心，取消了对
self.device = torch.device(
"cuda" if torch.cuda.is_available() else "cpu") # check gpu
两行代码的屏蔽。

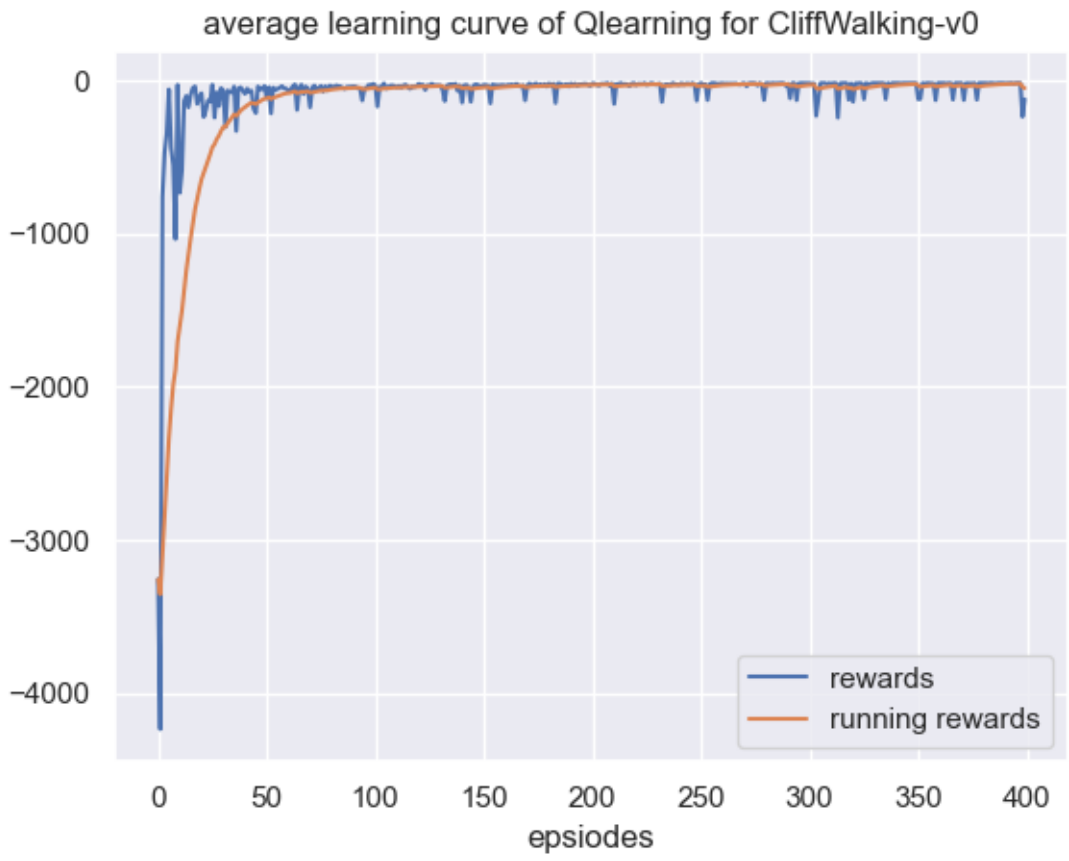
```python
class QlearningConfig:
    '''训练相关参数'''
    def __init__(self):
        self.seed = 0
        self.algo = 'Qlearning'
        self.env = 'CliffWalking-v0'  # 0 up, 1 right, 2 down, 3 left
        self.result_path = curr_path + "/outputs/" + self.env + '/' + '/results/'  # path to save results
        self.model_path = curr_path + "/outputs/" + self.env + '/' + '/models/'  # path to save models
        self.train_eps = 400  # 训练的episode数目
        self.eval_eps = 30
        self.gamma = 0.9  # reward的衰减率
        self.lr = 0.1  # learning rate
        self.render_frqc = 30 # 仿真渲染频率
        self.device = torch.device(
        "cuda" if torch.cuda.is_available() else "cpu")  # check gpu
```

九、实验数据及结果分析：


train.py
agent.py
env_agent_config
import os
import gym
import torch
from utils import CliffWalkingWapper, save_results, make_dir
from agent import QLearning
from plot import plot_rewards
curr_path = os.path.dirname(__file__)
class QlearningConfig:
'''训练相关参数'''
def __init__(self):
self.seed = 0
self.algo = 'Qlearning'
self.env = 'CliffWalking-v0' # 0 up, 1 right, 2 down, 3 left
self.result_path = curr_path + "/outputs/" + self.env + '/' + '/results/' # path to save results
self.model_path = curr_path + "/outputs/" + self.env + '/' + '/models/' # path to save models
self.train_eps = 400 # 训练的episode数目
self.eval_eps = 30
self.gamma = 0.9 # reward的衰减率
C:\Users\Albert\AppData\Local\Programs\Python\Python312\Lib\site-packages\gym\utils\passive_env_checker.py:241: DeprecationWarning: `np.bool8` is a deprecated alias for `np.bool_` (Deprecated NumPy 1.24)
if not isinstance(terminated, (bool, np.bool8)):
Episode:1/400: reward:-3257.0
Episode:2/400: reward:-4236.0
Episode:3/400: reward:-757.0
Episode:4/400: reward:-467.0
Episode:5/400: reward:-342.0
Episode:6/400: reward:-55.0
Episode:7/400: reward:-436.0
Episode:8/400: reward:-550.0
Episode:9/400: reward:-1032.0
Episode:10/400: reward:-25.0
Episode:11/400: reward:-730.0
Episode:12/400: reward:-583.0
Episode:13/400: reward:-131.0
Episode:14/400: reward:-97.0
Episode:15/400: reward:-172.0
Episode:16/400: reward:-91.0
Episode:17/400: reward:-45.0
Episode:18/400: reward:-35.0
Episode:19/400: reward:-149.0
Episode:20/400: reward:-105.0
Episode:21/400: reward:-80.0
Episode:22/400: reward:-235.0
Episode:23/400: reward:-191.0
Episode:24/400: reward:-127.0
Episode:25/400: reward:-133.0


train.py
agent.py
env_agent_config
Episode:373/400: reward:-14.0
Episode:374/400: reward:-15.0
Episode:375/400: reward:-15.0
Episode:376/400: reward:-15.0
Episode:377/400: reward:-13.0
Episode:378/400: reward:-122.0
Episode:379/400: reward:-13.0
Episode:380/400: reward:-13.0
Episode:381/400: reward:-13.0
Episode:382/400: reward:-13.0
Episode:383/400: reward:-15.0
Episode:384/400: reward:-13.0
Episode:385/400: reward:-13.0
Episode:386/400: reward:-15.0
Episode:387/400: reward:-15.0
Episode:388/400: reward:-13.0
Episode:389/400: reward:-13.0
Episode:390/400: reward:-13.0
Episode:391/400: reward:-15.0
Episode:392/400: reward:-14.0
Episode:393/400: reward:-13.0
Episode:394/400: reward:-13.0
Episode:395/400: reward:-17.0
Episode:396/400: reward:-13.0
Episode:397/400: reward:-13.0
Episode:398/400: reward:-13.0
Episode:399/400: reward:-233.0
Episode:400/400: reward:-119.0


train.py
agent.py
env_agent_config
Complete training!
results saved!
Start to eval !
Episode:1/30, reward:-13.0
Episode:2/30, reward:-13.0
Episode:3/30, reward:-13.0
Episode:4/30, reward:-13.0
Episode:5/30, reward:-13.0
Episode:6/30, reward:-13.0
Episode:7/30, reward:-13.0
Episode:8/30, reward:-13.0
Episode:9/30, reward:-13.0
Episode:10/30, reward:-13.0
Episode:11/30, reward:-13.0
Episode:12/30, reward:-13.0
Episode:13/30, reward:-13.0
Episode:14/30, reward:-13.0
Episode:15/30, reward:-13.0
Episode:16/30, reward:-13.0
Episode:17/30, reward:-13.0
Episode:18/30, reward:-13.0
Episode:19/30, reward:-13.0
Episode:20/30, reward:-13.0
Episode:21/30, reward:-13.0
Episode:22/30, reward:-13.0
Episode:23/30, reward:-13.0
Episode:24/30, reward:-13.0
Episode:25/30, reward:-13.0



十、实验结论：

Q-Learning 算法经过学习之后可以接近最优路径，使 rewards 为-13.

十一、总结及心得体会：

Q-Learning 算法相对简单、学习快速，不需要理解环境效率较高。但是比较莽撞，缺乏长期记忆力。

十二、对本实验过程及方法、手段的改进建议：

尝试其他的强化学习算法：例如 Sarsa、DQN 等；并对 Q-learning 和 Sarsa 方法进行对比分析。

报告评分：

指导教师签字：