**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**

**电 子 科 技 大 学**

**实 验 报 告**

**学生姓名：李天 学 号：2020080904021**

**指导教师：段立新 张彦如 顾实**

**实验地点： 主楼A2-413-1 实验时间：2022.12.3**

**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**

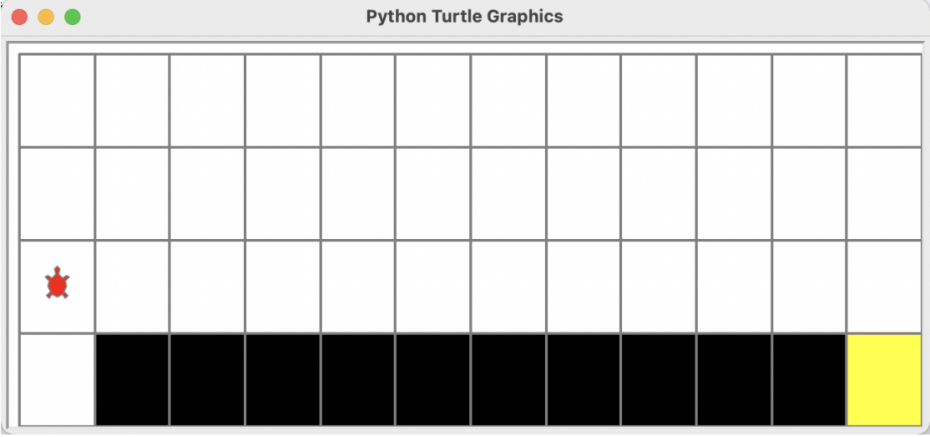


图 1 迷宫游戏示例

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：

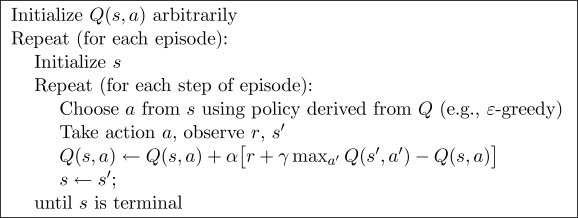


图 2 强化学习算法框架

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。

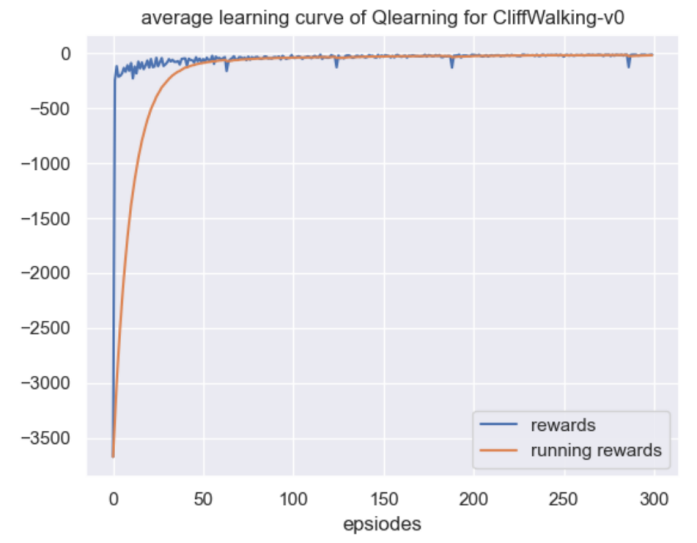


图 3 Q-Learning训练奖励曲线

**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.补全./agent.py中choose\_action()方法**

choose\_action()方法用于训练过程中针对一个state选取action，因此按照Q-Learning算法来看，应该选取该state对应的四个Q(s,a)中最大的Q所采取的action，即如下公式所示：

同时为了提升该算法的性能，我引入了**贪心策略（ε-greedy）**用于进一步优化choose\_action()方法。其表示在智能体做决策时，有一很小的正数ε(<1)的概率**随机选择未知的一个动作**，剩下1-ε的概率选择已有动过中动作价值最大的动作。这样做的好处是，如果我们每次都选择最好的动作，那会有很多动作没有被选择，ε-greedy则可以解决这个问题。进一步思考会发现，所有的动作被选择的概率都满足**π(a|s)≥ε/|A|**，这就保证了每个（状态-动作）二元组都会有一定概率被访问到。

同时，为了使后期训练时尽可能选择最优路径，在这里我设置**ε随着迭代次数而逐渐衰减（最开始为0.95，最终收敛至0.01）**，使得最后Q-Learning可以达到收敛。由此，可以补全choose\_action()方法，如图4所示。

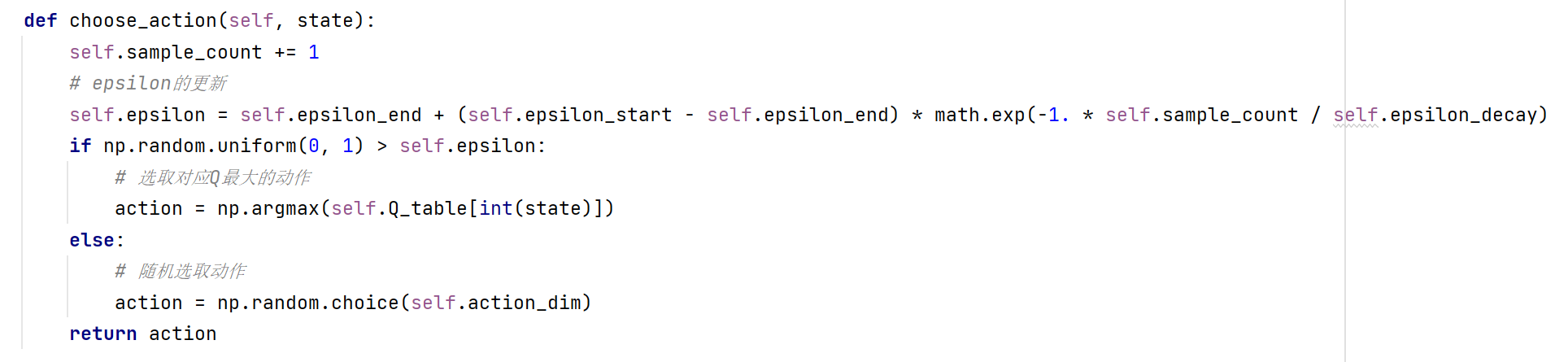


图 4 choose\_action方法

**2. 补全./agent.py中predict()方法**

predict()方法用于评估模型时使用，即在模型训练好后，根据Q表格选取每个state下最优的action。因此，实现该方法只需要取Q表格中对应当前state的最大值所采取的action，predict()方法实现如图5所示。

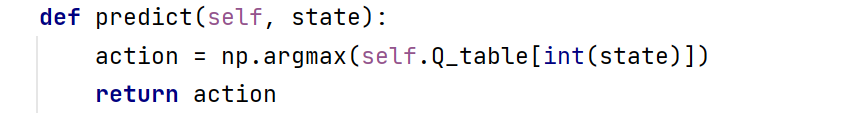


图 5 predict方法

**3.补全./agent.py中update()方法**

update()方法用于训练模型时更新Q表格。因此，实现该方法需要实现如下公式：

但需要注意，如果当前状态已经达到终点，那么其Q值更新时对应的未来Q值应该就等于当前的reward。由此，update()方法实现如图6所示。

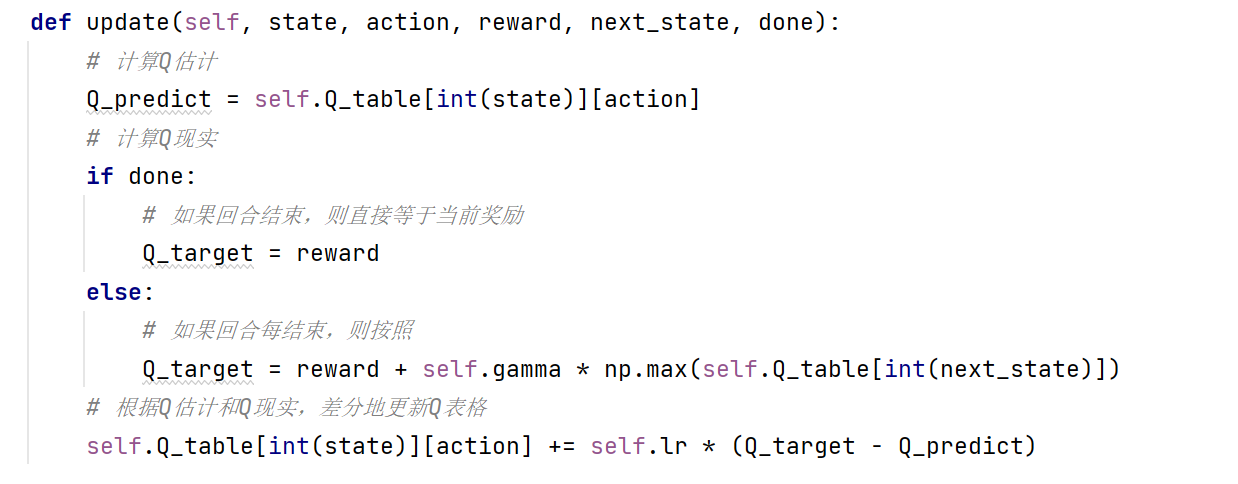
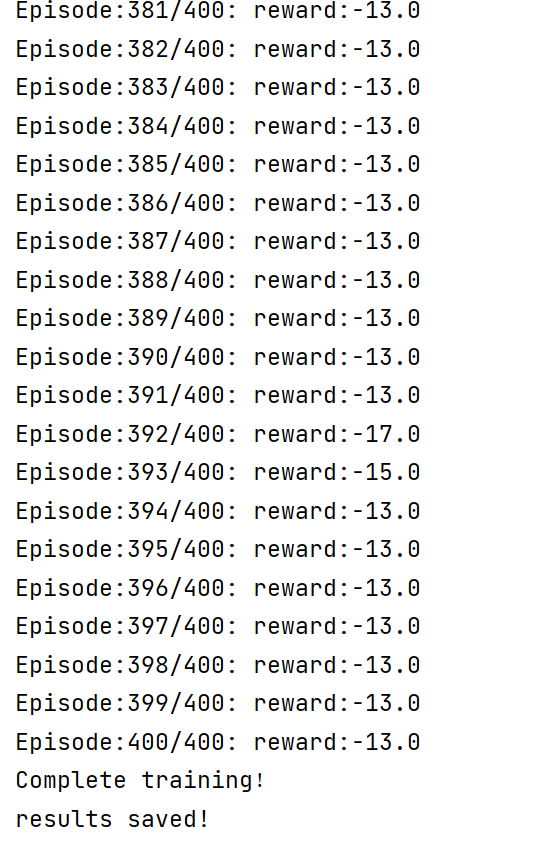


图 6 update方法

**九、实验数据及结果分析：**

**（1）训练过程输出展示**

训练过程中的输出为：



本次实验我将QlearningConfig中的train\_eps参数增加到了400，可以看到（如图7），到了训练后期，实验结果基本**趋于收敛**（有少数非-13的结果是因为**ε-greedy**机制的存在）。

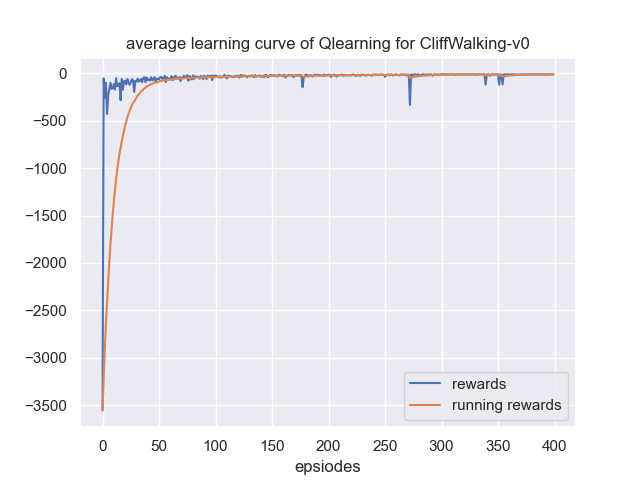
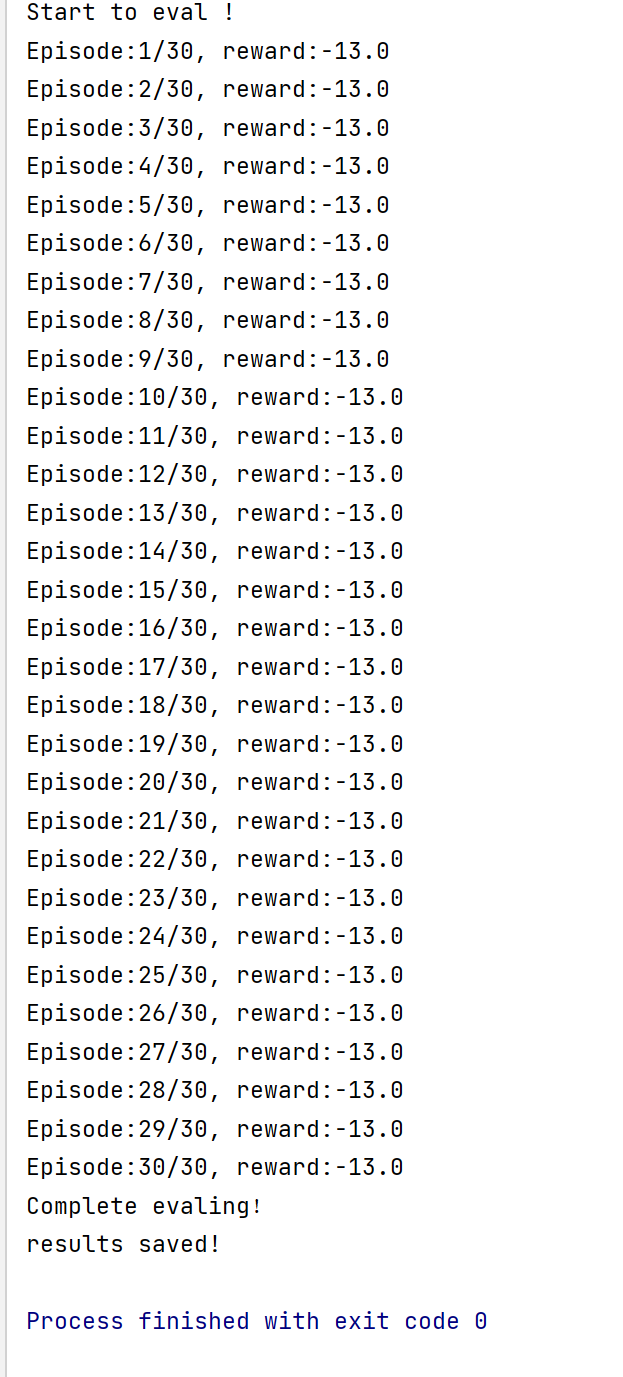


图 7 training\_reward曲线

**（2）评估模型**

评估模型过程输出为：



可以发现，评估结果较为理想，总共30次episode都选择了最短的路径，**智能体的奖励每次都是-13**，成功达到了训练目标（如图8）。

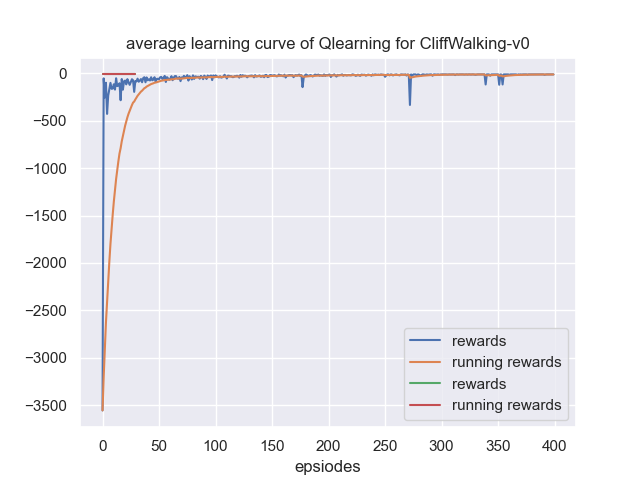


图 8 eval\_reward曲线

**十、实验结论：**

本次实验圆满完成，成功实现了所有实验要求，成功达到了训练目标。

**十一、总结及心得体会：**

本次实验我熟悉和掌握了Q-table算法，理解了算法原理，并利用该算法成功实现训练悬崖寻路问题，最终在评估过程中能达到预期的训练目标，使我对Q-Learning算法有了更深入的理解和思考，对该算法的应用有了一定的了解。

**十二、对本实验过程及方法、手段的改进建议：**

在实际实验过程中，对gym库的使用不太熟悉，花了很多的时间去了解这个库，以及训练的环境，我认为之后可以先简要的讲解一下整个项目。

**十三、代码附件**

|  |
| --- |
| **实验三、补全agent.py文件** |
| 1. import numpy as np 2. import math 3. class QLearning(object): 4. def \_\_init\_\_(self, state\_dim, action\_dim, cfg): 5. self.action\_dim = action\_dim  *# dimension of acgtion* 6. self.lr = cfg.lr  *# learning rate* 7. self.gamma = cfg.gamma *# 衰减系数* 8. self.epsilon = 0.1 9. self.epsilon\_start = 0.95 10. self.epsilon\_end = 0.01 11. self.epsilon\_decay = 300 12. self.sample\_count = 0 13. self.Q\_table = np.zeros((state\_dim, action\_dim))  *# Q表格* 14. def choose\_action(self, state): 15. self.sample\_count += 1 16. *# epsilon的更新* 17. self.epsilon = self.epsilon\_end + (self.epsilon\_start - self.epsilon\_end) \* math.exp(-1. \* self.sample\_count / self.epsilon\_decay) 18. if np.random.uniform(0, 1) > self.epsilon: 19. *# 选取对应Q最大的动作* 20. action = np.argmax(self.Q\_table[int(state)]) 21. else: 22. *# 随机选取动作* 23. action = np.random.choice(self.action\_dim) 24. return action 25. def predict(self, state): 26. action = np.argmax(self.Q\_table[int(state)]) 27. return action 28. def update(self, state, action, reward, next\_state, done): 29. *# 计算Q估计* 30. Q\_predict = self.Q\_table[int(state)][action] 31. *# 计算Q现实* 32. if done: 33. *# 如果回合结束，则直接等于当前奖励* 34. Q\_target = reward 35. else: 36. *# 如果回合每结束，则按照* 37. Q\_target = reward + self.gamma \* np.max(self.Q\_table[int(next\_state)]) 38. *# 根据Q估计和Q现实，差分地更新Q表格* 39. self.Q\_table[int(state)][action] += self.lr \* (Q\_target - Q\_predict) 40. def save(self, path): 41. np.save(path + "Q\_table.npy", self.Q\_table) 42. def load(self, path): 43. self.Q\_table = np.load(path + "Q\_table.npy") |

**报告评分：**

**指导教师签字：**