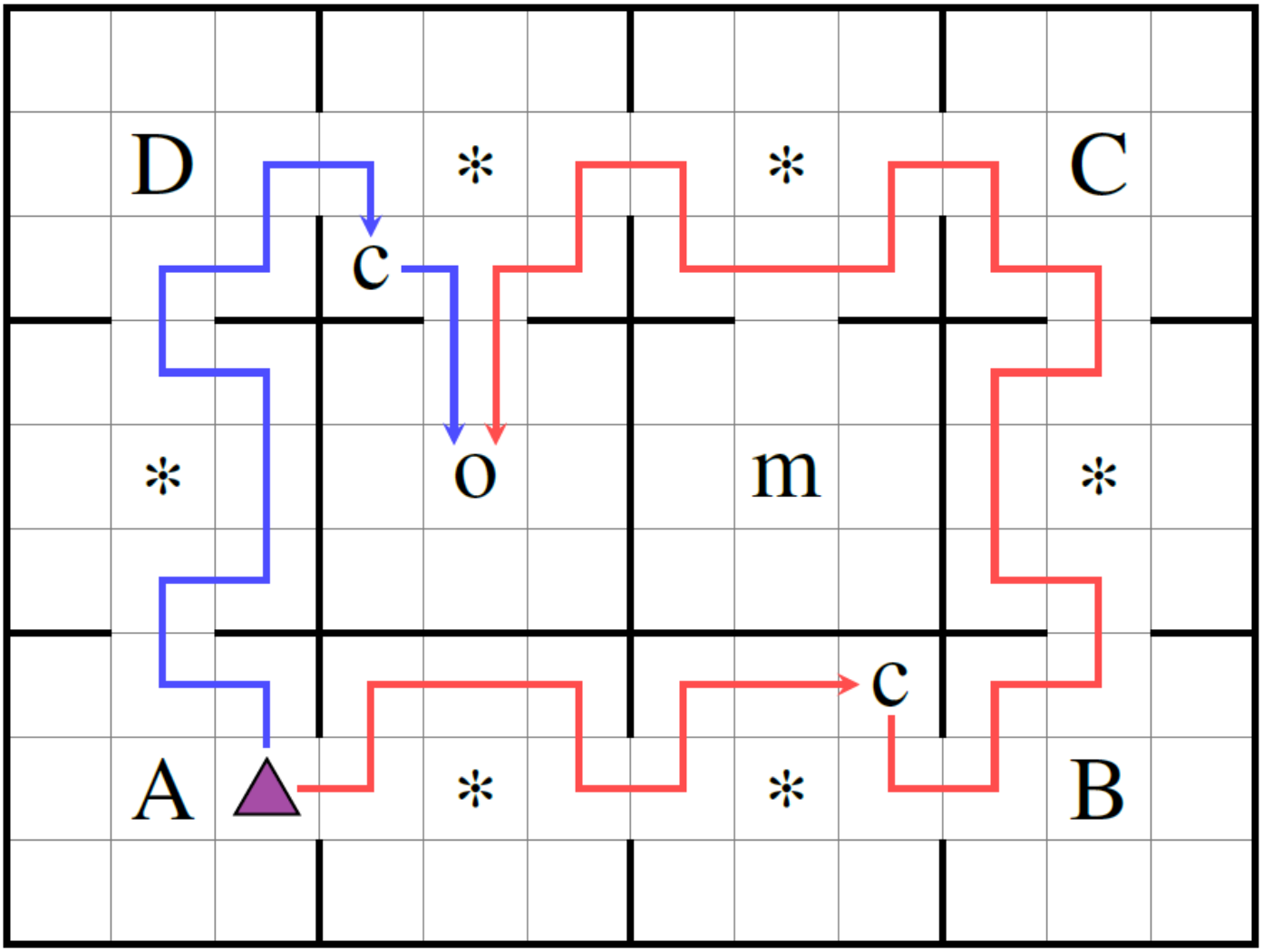
第9次作业

1.强化学习: 表格型Q-Learning算法

如下图所示, 本次作业的OfficeWorld环境为9*12网格, 其中三角形表示智能体, 黑色粗线表示无法跨越的墙壁, A,B,C,D为四个地标, c表示coffee, m表示mail, o表示office, *表示盆栽. 该环境的可选任务有 `'to_a', 'to_b', 'to_c', 'to_d','get_mail', 'get_coffee', 'to_office'`, 分别表示在不触碰盆栽的前提下到达地标或者物品所在的网格. 下图的蓝色路径表示了任务 `'to_office'` 的最优解, 而红色路径是一个次优解.



`office_world.py`为该环境的代码, 其中的 `play()`函数供人工与环境交互(通过键盘wasd操作智能体)并可视化. `rl_test.py`中给出了执行一次测试的函数 `run_test()`.

作业要求: 在该环境上实现表格型Q-Learning算法. 要点如下:

- 将学习算法以及各种辅助方法集成到类 `QLAgent`中, 自定义Q函数的数据格式(建议使用 `numpy`数组)
- 在 `learn()`方法中实现Q函数更新, 传入的参数自定; 在 `get_action()`方法中实现基于Q-Learning的策略, 传入的参数必须为 `state, eval_mode`且不可修改.

作业示例代码 `hw9_21000000_zhangsan.py`中给出了:

- 学习参数 `LearningParams`与测试参数 `TestingParams`, 可自行修改参数的值或添加新的参数
- `QLAgent`为智能体类, 是本次作业的核心. 其中 `learn()`方法为每次训练时智能体的更新过程, `get_action()`方法输入为状态 `state` 与参数 `eval_mode`, 输出为该状态下应该采取的动作. 其中参数 `eval_mode`在训练时为 `False`, 测试时为 `True` (可以理解为训练和测试时智能体的策略未必相同).
- `run_experiment()`函数, 为强化学习的大致训练与测试框架.

提示

1. 一般情况下只需要编写 `QLAgent`并调整参数便可完成本次作业. 若有新想法可自行修改示例代码中的其他部分.
2. 与环境交互 `env.step()`返回四个参数 `s, r, done, events`
   - `s`是采取动作后的新状态, 是one-hot形式, 可使用argmax转化为整数
   - `r`是环境返回的奖励, 定义如下: 若完成了任务, 则$r=1$; 若触碰到盆栽则$r=-1$; 其他情况$r=-0.01$.
   - `done`表示当前episode是否结束. 若任务完成或者触碰盆栽则 `done=True`
   - `events`表示当前事件, 例如 `'None', 'to_a'`等. 若 `events==task`则任务完成.
3. 每次测试 `run_test()`返回的总奖励是不带折扣的累积奖励
4. 最后作业只需交一个代码文件即可.

思考题

1. Q-Learning中的epsilon参数有什么作用?

2. 为什么智能体 `get_action()`时需传入 `eval_mode`参数以区分训练策略和测试策略? 这样做有什么好处?
3. 如果将任务修改为"先去地标B, 然后去地标D", 那么Q-Learning算法理论上可以学出来吗?
4. 如果任务是"去地标A或者C", 并且到达地标A的奖励是1, 到达地标C的奖励是10, 这时候Q-Learning的表现如何? 若无法学出到达地标C的策略, 可能是什么原因?

实验环境参考

python=3.8

```
torch               2.0.0
numpy               1.24.2
```