强化学习：作业二

linze.H

2024 年 11 月 18 日

1 作业内容

在 gridworld 环境中实现 Q-learning 算法，并进行调试.

2 实现过程

2.1 代码构建

首先，构建基本代码结构：

- `train.py`：`q_learning()` 为算法 `Q-learning` 主要框架；`val()` 为测试函数；`train()` 为主要代码流程框架，和源框架代码一致，为源代码框架的模块化.
- `algo.py`：实现了对 Agent 的抽象，实现了对 `Q` 函数的抽象方法，包括对 `Q` 的安全调用和策略迭代；对策略 π 的抽象方法，包括策略评估；此外还提供了方便调试 `QAgent` 的打印信息，直接 `print(QAgent())` 可以打印 `Q` 函数和当前策略 π，如图1.
- `utils.py`：实现了保存模型和环境图像的函数 `save_result()`；利用库 `matplotlib` 实现的绘制方法 `Drawer`，可以实时绘制 `info` 对象；将 `plot()` 方法移动到此.
- `arguments.py`：实现命令行参数解析和 Python 脚本超参结构体 `conf` 解析，以命令行参数为优先.
- `main.py`：读取命令行参数设置，进行超参设定，进行实验；便于使用 Python 脚本进行超参调试.

然后，在命令行设置超参，或在 `main.py` 中的 `conf` 结构体中设置超参，传入 `main(conf: table)` 或 `get_score(conf: table) -> (converge_steps, mean_reward, max_reward, min_reward)` 中，进行训练或调参.

2.2 Q-learning 实现说明

算法核心如下所示

```python
def q_learning(
    epoch: int, env: Make_Env, agent: QAgent, T: int, epsilon: float
) -> None:
    obs = tuple(env.reset().astype(int))  # initial observation
    for _ in tqdm(range(T), leave=False):
        action = agent.select_action(obs, epsilon=epsilon)  # select action using ε-greedy.
        obs_next, reward, done, info = env.step(action)  # interact with environment
        obs_next = tuple(obs_next.astype(int))
        agent.eval_pol(obs, obs_next, action, reward)  # evaluate the Q function
        agent.update_pol(state=obs, act=action)  # update the policy
        # if the episode has terminated, we need to reset the environment.
        obs = tuple(env.reset().astype(int)) if done else obs_next
```

```python
class Agent:
......
    def select_action(
        self, state: tuple, epsilon: Union[float, None] = None
    ) -> int:
        """ε-greedy policy."""
        if epsilon is None or np.random.rand() >= epsilon:
            return self._select_action(state)
        else:
            return self.env.action_sample()
......

class QAgent(Agent):
......
    def _select_action(self, state: tuple) -> int:
        """greedy policy, return default action if not set."""
        return self._policy.get(state, self.default_action)

    def eval_pol(
        self, state: tuple, next_state: tuple, act: int, reward: Union[float, int],
    ):
        """update the Q function."""
        self.Q[state, act] += self.lr * (
            reward
            + self.discount_factor * self.Q[next_state, self._select_action(state)]
            - self.Q[state, act]
        )

    def update_pol(
        self, state: Union[tuple, None] = None, act: Union[int, None] = None
    ):
        """
        update the policy given Q function. only update the relevant part of the policy
        if given `state` and `action`, else update the whole policy.
        """
        if state and act is None:
            for s in self.Q.states():
                self._policy[s] = max(list(range(4)), key=lambda i: self.Q[s, i])
        elif self.Q[state, self._select_action(state)] < self.Q[state, act]:
            self._policy[state] = act
```

```
41 ......
```

3 复现方式

1. 安装版本为 3.7 的 Python ，并按照文件 requirements.txt 安装依赖环境.
2. 然后使用命令行调用：

```
python main.py [arguments]
```

3. 如果有调参需要，可以使用 Python 脚本在文件 main.py 中设置不同超参 conf，传入 get_score(conf: table) -> (converge_steps, mean_reward, max_reward, min_reward) 进行调参.

4 实验效果

4.1 主要实验效果

实验运行结果如图2. 此运行设置的超参如表1，其中可以使用命令 python main.py -h 获得各个参数的说明，此模型在文件 Qmodel.pkl 中，此模型的最终运行效果如图3.

hyper-paremeter	setting
num_stacks	4
T	100
val_T	500
epochs	1000
env_mode	2
log_interval	40
save_img	True
save_interval	10
lr	0.8
discount_factor	0.8
default_action	0
epsilon	0.2
render_epoch	-1
log_file	./log
val_save_dir	./imgs

表 1: 实验超参设置

4.2 其他环境设置效果

当设置超参 `env_mode` 为 1 时，可以加载另一个地图模型，但此模型似乎不方便进行实验. 因为此环境不是连通图. 而且此环境返回的 `info` 和内部的 `env.no_wall_position()` 的信息并不一致：`info` 返回如图3，但 `env.no_wall_position()` 返回的信息如图 4.

因此再次不详细说明此环境模式下的效果说明,具体 performance.png 可以查看文件 ./code/imgs/11-16-24_16:57:11/performance.png.

4.3 学习率 `lr` 对代码效率影响

在 `main.py` 撰写了 `test_lr()` 函数，用于研究学习率对 Q 函数收敛的影响. 最终结果如图5.

5 小结

经过这次实验，我的代码编写能力有所提升，对 Q-learning 算法的理解提升了.

```
---
Q:
→-4.9  →-3.4  →-3.0  #####  #####  →-0.4  →0.72  →2.16
←-3.8  ←-4.9  ←-4.6  #####  #####  ←-1.3  ←-1.3  ←-0.4
↓-3.4  ↓-4.5  ↓-2.6  #####  #####  ↓-0.4  ↓0.72  ↓3.95
↑-3.9  ↑-4.1  ↑-3.0  #####  #####  ↑-1.3  ↑-0.4  ↑2.16

→-2.6  →-2.0  →-1.3  →-0.3  →-1.3  →0.73  →2.16  →3.95
←-3.0  ←-3.0  ←-2.6  ←-2.6  ←-3.0  ←-1.3  ←-0.4  ←-0.4
↓-3.1  ↓-4.4  ↓-2.0  ↓4.00  ↓0.76  ↓-0.4  ↓0.72  ↓6.18
↑-4.9  ↑-3.5  ↑-3.4  ↑2.15  ↑-1.3  ↑-1.3  ↑-0.4  ↑2.16

#####  →-4.1  #####  →-0.2  →0.77  #####  #####  →6.18
#####  ←-4.2  #####  ←4.00  ←2.18  #####  #####  ←6.16
#####  ↓-4.4  #####  ↓6.26  ↓0.91  #####  #####  ↓8.98
#####  ↑-3.8  #####  ↑2.20  ↑-2.3  #####  #####  ↑3.96

→-4.5  →-4.4  #####  →0.78  →2.16  #####  →8.98  →12.4
←-4.8  ←-4.9  #####  ←6.24  ←3.98  #####  ←26.3  ←20.0
↓-4.8  ↓-4.4  #####  ↓9.07  ↓2.19  #####  ↓35.8  ↓12.4
↑-4.6  ↑-4.0  #####  ↑3.94  ↑0.77  #####  ↑22.2  ↑3.93

#####  #####  #####  →8.97  #####  #####  →29.0  #####
#####  #####  #####  ←8.96  #####  #####  ←35.8  #####
#####  #####  #####  ↓12.5  #####  #####  ↓46.1  #####
#####  #####  #####  ↑6.18  #####  #####  ↑27.6  #####

→8.96  →12.5  →17.0  →22.4  →29.3  →37.9  →48.6  →61.5
←6.18  ←6.18  ←8.96  ←12.5  ←16.8  ←22.8  ←23.4  ←45.7
↓0.71  ↓8.97  ↓4.47  ↓16.9  ↓22.3  ↓29.4  ↓58.8  ↓78.2
↑6.18  ↑9.04  ↑12.4  ↑6.18  ↑22.6  ↑29.2  ↑35.3  ↑61.5

→2.14  #####  →6.39  #####  #####  #####  →61.5  →78.2
←2.15  #####  ←6.28  #####  #####  #####  ←48.2  ←58.8
↓0.72  #####  ↓2.47  #####  #####  #####  ↓74.8  ↓99.0
↑3.94  #####  ↑8.96  #####  #####  #####  ↑37.8  ↑61.5

→0.72  #####  →37.5  →48.2  →61.5  →78.2  →99.0  ddddd
←0.72  #####  ←29.0  ←29.0  ←37.5  ←48.2  ←61.5  ddddd
↓0.72  #####  ↓29.0  ↓37.5  ↓48.2  ↓61.5  ↓78.2  ddddd
↑2.15  #####  ↑4.16  ↑37.5  ↑48.2  ↑61.5  ↑48.2  ddddd
policy:
↓ → ↓ #  #  → → ↓

→ → → ↓ ← → → ↓

#  ↑ #  ↓ ← #  #  ↓

→ ↑ #  ↓ ← #  ↓ ←

#  #  #  ↓ #  #  ↓ #

→ → → → → → ↓ ↓

↑ #  ↑ #  #  #  → ↓

↑ #  → → → → → d
```

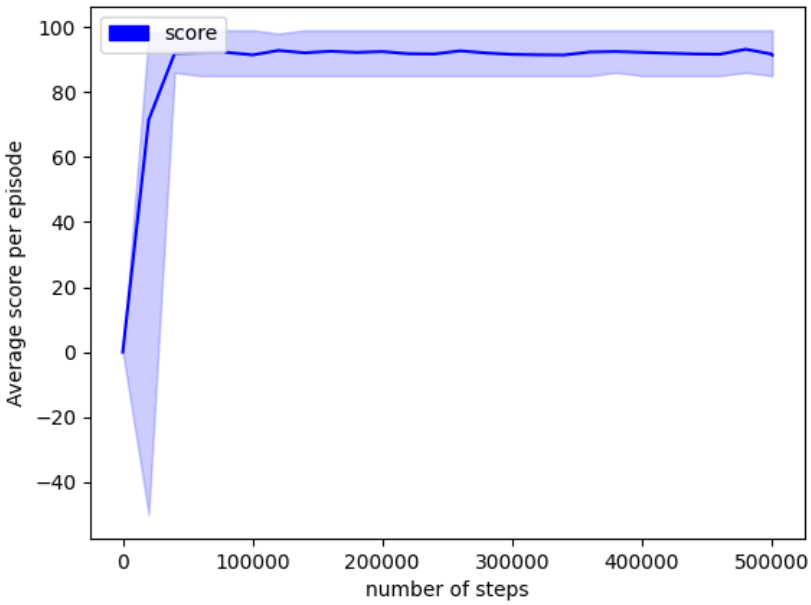
score
100
80
60
40
20
0
−20
−40
Average score per episode
0
100000
200000
300000
400000
500000
number of steps

图 2: 实验运行效果

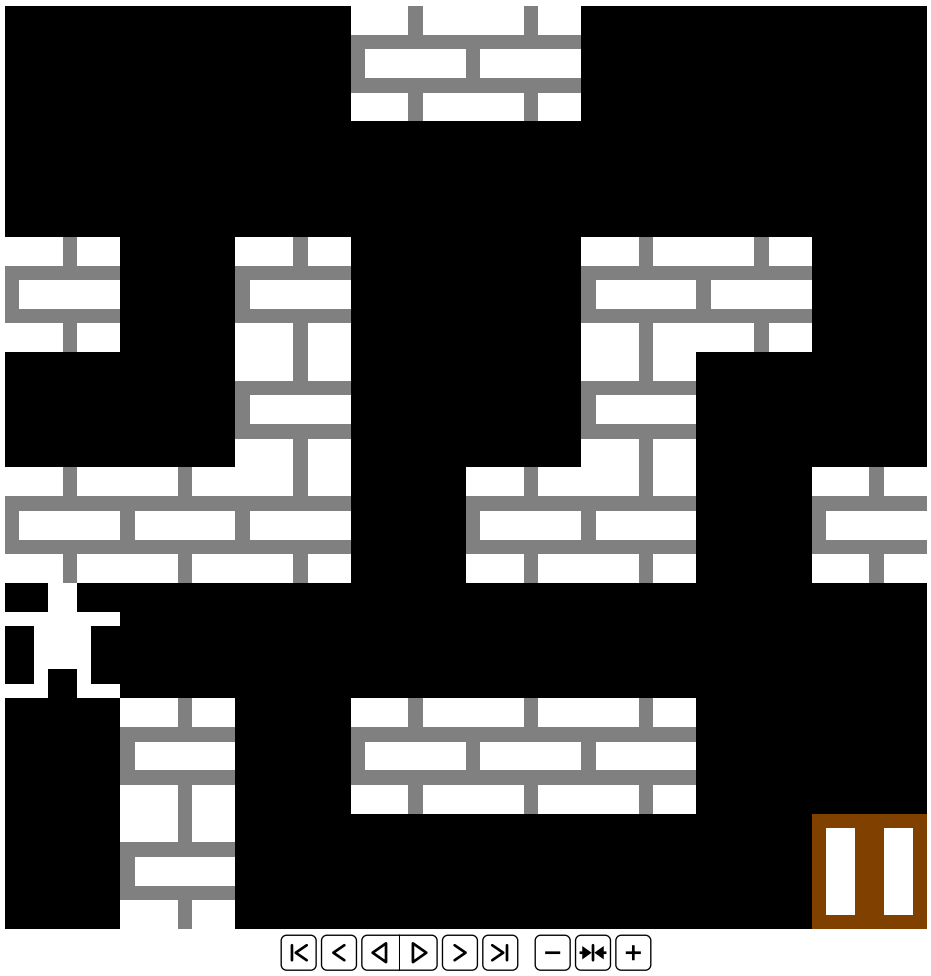

图 3: 实验运行效果

```
Q:
→-4.2  →-4.0  →-3.8  →-4.9  #####  →0     →0     →0---
←-4.5  ←-4.4  ←-4.9  ←-4.9  #####  ←0     ←0     ←0---
↓-4.9  ↓-4.9  ↓29.0  ↓-4.6  #####  ↓0     ↓0     ↓0---
↑-4.7  ↑-4.9  ↑22.2  ↑-4.7  #####  ↑0     ↑0     ↑0---

→-4.9  →-4.9  →-4.6  →-4.7  #####  →0     →0     →0---
←-4.8  ←-4.8  ←-4.9  ←-4.9  #####  ←0     ←0     ←0---
↓-4.9  ↓-4.9  ↓37.5  ↓-4.5  #####  ↓0     ↓0     ↓0---
↑-4.8  ↑-4.6  ↑22.2  ↑-4.9  #####  ↑0     ↑0     ↑0---

#####  #####  →37.5  #####  #####  →0     →0     →0---
#####  #####  ←37.5  #####  #####  ←0     ←0     ←0---
#####  #####  ↓48.2  #####  #####  ↓0     ↓0     ↓0---
#####  #####  ↑29.0  #####  #####  ↑0     ↑0     ↑0---

ddddd  →61.5  →48.2  →48.2  #####  →0     →0     →0---
ddddd  ←99.0  ←78.2  ←61.5  #####  ←0     ←0     ←0---
ddddd  ↓78.2  ↓61.5  ↓48.2  #####  ↓0     ↓0     ↓0---
ddddd  ↑78.2  ↑29.0  ↑48.2  #####  ↑0     ↑0     ↑0---

#####  #####  #####  #####  #####  #####  #####  #####
#####  #####  #####  #####  #####  #####  #####  #####
#####  #####  #####  #####  #####  #####  #####  #####
#####  #####  #####  #####  #####  #####  #####  #####

→0     →0     →0     →0     #####  →0     →0     →0---
←0     ←0     ←0     ←0     #####  ←0     ←0     ←0---
↓0     ↓0     ↓0     ↓0     #####  ↓0     ↓0     ↓0---
↑0     ↑0     ↑0     ↑0     #####  ↑0     ↑0     ↑0---

→0     →0     →0     →0     #####  →0     →0     →0---
←0     ←0     ←0     ←0     #####  ←0     ←0     ←0---
↓0     ↓0     ↓0     ↓0     #####  ↓0     ↓0     ↓0---
↑0     ↑0     ↑0     ↑0     #####  ↑0     ↑0     ↑0---

→0     →0     →0     →0     #####  →0     →0     →0---
←0     ←0     ←0     ←0     #####  ←0     ←0     ←0---
↓0     ↓0     ↓0     ↓0     #####  ↓0     ↓0     ↓0---
↑0     ↑0     ↑0     ↑0     #####  ↑0     ↑0     ↑0---
policy:
→ → ↓ ↓ # → → →

↑ ↑ ↓ ← # → → →

# # ↓ # # → → →

d ← ← ← # → → →

# # # # # # # #
```

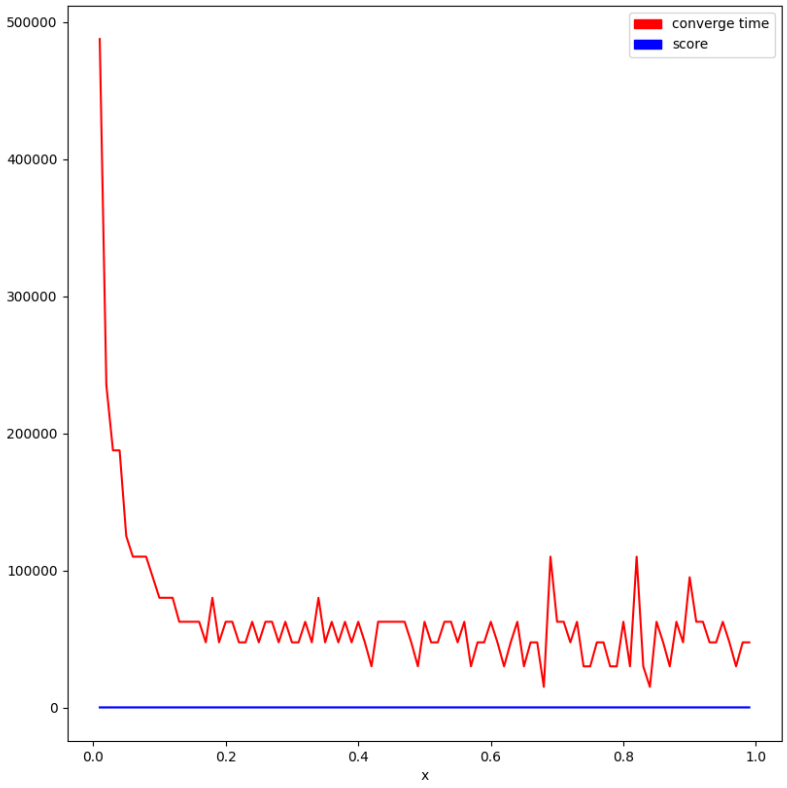

图 5: 学习率与收敛速度关系图