DS4001 · 人工智能原理与技术

Homework 2 强化学习

中国科学技术大学 · 2024 春季学期

April 7, 2024

作业要求

本实验由书面部分（回答问题）和编程部分（代码填空）组成。其中：

- **编程部分**：完整代码可在 `code` 文件夹中找到。你只需要在 submission.py 的

```
# BEGIN_YOUR_CODE

# END_YOUR_CODE
```

  部分完成代码编写。不要对除 `submission.py` 以外的文件进行更改。在完成所有代码编写后，你可以运行 `python learner.py` 学习到 Q 值表，之后运行 `python evaluator.py` 进行评估。

- **书面部分**：文件夹中有一个 tex 压缩包（`report.zip`），里面包含了一个叫 main.tex 的文件，建议你使用 Overleaf 创建相应项目过后，根据 main.tex 中标红的 TODO 部分完成书面部分填写。（注：为方便助教批改，编程填空部分的代码也需要复制粘贴到对应位置）

在完成所有题目后，你需要在 bb 系统上传一个命名为学号 _ 姓名 _HW2.zip 的压缩包，里面只用包含 submission.py 和 report.pdf 两个文件。助教会据此对你的作业进行评分。**注意事项：**

- **本次作业需独立完成，不允许任何形式的抄袭。如被发现，互相抄了一份作业的几名同学分配此作业的分数。**
- **本次作业的截止时间为** {2024/4/30}. **在此之后延迟** 1/2/3/4 **天会有** 10/30/60/100 **的额外扣分。**

如果在做作业时任何问题，可以通过 腾讯文档 向助教提问，也欢迎同学帮助解答问题。

1. 热身 [10%]

引言 考虑一个简单的 MDP：

- 状态空间 $\mathcal{S} = \{-2, -1, 0, 1, 2\}$；
- 动作空间 $\mathcal{A} = \{a_1, a_2\}$；
- 初始状态 $s_0 = 0$，某一时刻 t 的状态 $s_t \in \{-2, 2\}$ 时，游戏结束。
- 定义 $p_{sas'} := \mathcal{P}(S_{n+1} = s' \,|\, S_n = s, A_n = a)$，则环境噪声下的转移概率可以被表示为

$$p_{i,a_1,i+1} = 0.6 \,|\, p_{i,a_1,i-1} = 0.4 \,|\, p_{i,a_2,i+1} = 0.5 \,|\, p_{i,a_2,i-1} = 0.5$$

- 奖励函数 $\mathcal{R}(s, a, s') = \begin{cases} 25, s' = -2 \\ 40, \ s' = 2 \\ -10, \ otherwise \end{cases}$
- 衰减因子 $\gamma = 1$。

(a) [5 分] 在 Value Iteration 中，对于迭代轮次 $i \in \{0, 1, 2\}$，求对应轮次 i 的 $V^{(i)}(s)$ 值，假定初始估计 $V^{(0)}(s) \equiv 0$。（你需要列出一个表格，一共含有 $3 \times 5 = 15$ 个值）

(b) [5 分] 根据 $V^{(2)}(s)$ 的情况，求迭代轮次 $i = 2$ 时对应的确定性策略 $\mu(s)$。（你需要列出 3 个 $(s, \mu(s))$ 数值对，因为 $-2, 2$ 两个终止状态不需要考虑在内）。

2. Q-Learning [15%]

引言 现在让我们对 Q-Learning 算法做进一步讨论。

提示 以下是可供你参考的定义：

- $G_t = \sum_{k=t}^{+\infty} \gamma^{k-t} \mathcal{R}_k$
- $v(s) = \mathbb{E}[G_t \,|\, S_t = s]$
- $q(s, a) = \mathbb{E}[G_t \,|\, S_t = s, A_t = a]$

以及，直接从环境中进行随机采样，其实本质上就是一种 Model-Free 的办法，这样我们就不必再去管状态转移概率在具体的数学形式上是什么了（因为 Monte-Carlo 采样本身就能给出随机性的描述）。

(a) [2 分] 根据 MDP 的定义和性质，为什么 $v(s), q(s, a)$ 与 t 无关？

(b) [8 分] 将估计值的初始值设置为 0。对于定义在状态空间 $\mathcal{S} = \{0, 1\}$ 和动作空间 $\mathcal{A} = \{a_1, a_2\}$ 上的某一轨迹（trajectory）$\tau = \{(s_i, a_i, r_i)\}_4 = \{(0, a_1, 1), (1, a_1, 2), (0, a_2, -1), (1, a_2, 0)\}$，请手动写出估计值 $\hat{q}(s, a)$ 在时间 $1 \le t \le 4$ 的更新过程（Monte Carlo Q-Learning）。

(c) [5 分] 简单解释 Q-Learning 算法为什么能够收敛（请看参考材料 https://zhuanlan.zhihu.com/p/365814943。理解并复述文中证明的大致思路和直觉即可，不需要对其中的数学细节进行复现）。

3. Gobang Programming [55%]

引言 你现在应当初步理解了强化学习的原理。现在，我们可以考虑更加现实的一个游戏——五子棋的简化版，“三子棋”。为方便起见，我们在这里将“三子棋”的规则统一为：

- 1. 棋盘大小：$n = 3$；
- 2. 先手规则：黑棋先下；
- 3. 终局判定：一旦某一方在**行、列、正反对角线任一方向**上达成连续的三个棋子，则这一方赢。否则，若直到棋盘被填满仍未分出胜负，则双方平局。

在本题中，假定我们要训练的智能体是黑棋棋手，而白棋是一个只会随机落子的小白（因此可以把白棋落子视为环境噪声）；那么我们也可以将其表述为一个规范的 MDP：

- 状态空间 $\mathcal{S}$ =｛黑棋落子前所有棋盘可能的状态 s｝（某一位置 $s_{ij} = 0$ 表示无子，1 表示落黑子，2 表示落白子）；
- 动作空间 $\mathcal{A} = \{(x, y) \mid s_{xy} = 0\}$；
- 初始状态 $s_0 = 0_{n\times n}$，终局判定达成时，游戏结束。
- 奖励函数 $\mathcal{R}(s, a, s') = [L_b^2(s') - L_w^2(s')] - [L_b^2(s) - L_w^2(s)]$，其中 $L_b(s)$ 是状态 s 下黑棋的最长连续长度，$L_w(s)$ 是状态 s 下白棋的最长连线长度。
- 衰减因子 $\gamma = 0.5$。

(a) [2 分] 分析直接通过 Q Learning 获得每个状态下的 Q^* 值是否是一个可行的方案（对于 $n = 3$）?

(b) [33 分] 按照 submission.py 中的要求，完成代码填空。按照代码正确性和能否完整运行给分。在本实验中，你需要实现 Q-Learning 算法，训练智能体进行三子棋的游戏。为了减少你的工作量，助教已经实现了三子棋的框架、Q-Learning 的基本代码以及一些辅助代码。你只需要额外补全以下几个函数即可。同时这里再做一些简单的提示：

- `get_next_state`[5 分]，根据当前状态 `self.board` 和动作 `action`，返回下一个状态 `next_board`。你无须在这一部分对 self.board 进行修改。
- `sample_noise`[3 分]，根据当前的状态和动作空间，随机生成一个白棋的落子位置。
- `get_connection_and_reward`[5 分]，根据当前状态 `self.board` 和动作 `action`，返回下一个状态的奖励 `reward`，你可以使用 `get_next_state` 以及 `self.count_max_connections` 辅助实现 `get_connection_and_reward`；
- `sample_action_and_noise`[8 分]，根据当前状态 `self.board`，依据 epsilon-greedy 策略输出一个动作 `action`，同时返回一个白棋的落子位置。助教已经完成了调用 `sample_noise` 函数生成白棋的落子位置的返回逻辑，你需要补充实现根据 epsilon-greedy 策略返回 `action` 的部分。epsilon-greedy 策略的具体实现可参考 PPT 第 5 章《强化学习》第 39 页。此外，若状态-动作对 (s, a) 在 `self.Q` 中未被记录，则应返回一个随机动作。动作确定后，需从动作空间中移除该动作。

- `q_learning_update`[12 分]，你需要按照 Q-Learning 的更新公式来更新 `self.Q`。同时约定以下几个规则：1. 在数学上默认所有 `self.Q[s][a]` 的初始值为 `0`。另外，状态 s 对应的动作空间为空（即棋盘已满或者游戏已经结束的情况）导致无法在空集上取 max 时，直接以 0 代替即可；2. `self.Q` 被初始化为一个空字典 `{}`，在后续中记录的 (s, a) Pair 也是有限的（如某一时刻，`self.Q` 的值可能是 `{`s_1`:{`a_1`:0.5, `a_2`:1.0}, `s_2`:{`a_1`:0.3}, `s_3`:{}}` 的形式，而 s_4 尚未被记录）。但是这样已经足以让我们实现正确的 Q-Learning 更新，请你想一想解决办法。

(c) [10 分] 请你运行 learner.py 进行 Q Learning，之后运行 evaluator.py 进行测试。你需要截取训练以及评估最后的截图，按照复现结果是否合理给分。（建议将评估轮次设置的稍大一些，这样会更准确）

(d) [10 分] 将 learner.py 和 evaluator.py 中的参数更改为 $n = 4$（其他任何参数和代码均不做改变），然后在 4*4 的棋盘上尝试执行 Q Learning 算法，并给出 evaluator.py 的运行结果。回答该结果是否符合你的预期，并给出解释。

4. Deeper Understanding [10%]

引言 强化学习的评估和优化本质上与 Bellman 算子密切相关。

- 1. 评估。随机策略 $\pi(s, a) \in [0, 1]$ 的价值估计可以用算子 $\mathcal{T}_\pi$ 表示：

$$(\mathcal{T}_\pi v)(s) = \mathbb{E}_{a \in \mathcal{A}}\{r_{sa} + \gamma \cdot \sum_{s' \in \mathcal{S}} p_{sas'} \cdot v(s')\} = \sum_{a \in \mathcal{A}} \pi(s, a)[r_{sa} + \gamma \cdot \sum_{s' \in \mathcal{S}} p_{sas'} \cdot v(s')]$$

- 2. 优化。例如 Value Iteration，实质上就是通过迭代求解最优策略，这个过程同样可以用算子 $\mathcal{T}$ 表示

$$(\mathcal{T} v)(s) = max_{a \in \mathcal{A}}\{r_{sa} + \gamma \cdot \sum_{s' \in \mathcal{S}} p_{sas'} \cdot v(s')\}$$

- 3. 有如上定义之后，$v_\pi, v^* \in \mathbb{R}^{|\mathcal{S}|}$ 实际上是对应 Bellman 方程的解

$$\mathcal{T} v^* = v^*$$

$$\mathcal{T}_\pi v_\pi = v_\pi$$

其中 $v_\pi(s) = \mathbb{E}[G_t | S_t = s, A \sim \pi], v^*(s) = max_\pi\{v_\pi(s)\}$。

(a) [5 分] 确定性策略 μ 同样可以被表示为某个随机策略 π。基于 $\mathcal{T}_\pi$，请你给出 $\mathcal{T}_\mu$ 的定义。

(b) [5 分] 证明 $\mathcal{T}$ 是压缩映射，即对于 $v_1, v_2 \in \mathbb{R}^{|\mathcal{S}|}$，我们恒有 $||\mathcal{T} v_1 - \mathcal{T} v_2||_\infty \leq \gamma \cdot ||v_1 - v_2||_\infty$。（有这个条件之后，我们就不难证明 Value Iteration 能够在无穷范数意义下收敛到最优策略下的 v 值了）

5. Feedback [10%]

引言 你可以写下任何反馈，包括但不限于以下几个方面：课堂、作业、助教工作等等。

必填 你在本次作业花费的时间大概是？觉得难度如何？

选填 你可以随心吐槽课程不足的方面，或者给出合理的建议。若没有想法，直接忽略本小题即可。