Lab4 实验报告

PB22111695 蔡孟辛

1 实验流程

使用强化学习训练agent游玩 gymnasium 库中的MountainCar。

PART 1： Value Iteration

(1) 完成computeQ 函数：根据当前的$V(s)$计算$Q(s,a)$.

- succAndRewardProb：一个字典，将 `(state, action)` 映射到 `(next_state, prob, reward)` 的列表。
- 对于每个可能的下一个状态，计算其对 Q 值的贡献。
- 贡献值是转移概率 prob 乘以即时奖励 reward 和折扣后的下一个状态的值 discount * V[next_state] 之和。

(2) 完成computePolicy 函数：根据当前的 $V(s)$ 返回当前的策略 π.

- 遍历所有状态和对应的动作列表，不断更新最佳 Q 值 和 最佳动作 ，并将当前状态的最佳动作存储在策略字典policy 中。

(3) 完成 valueIteration 函数中的迭代循环,更新 $V(s)$.

- 使用 computeQ 函数更新 V 值，寻找并记录 V 值变化的最大值，重复此过程，直到所有状态的 V 值变化不超过 epsilon，则认为算法已收敛，退出循环。

PART 2：Model-based Monte Carlo

(1) 实现 getAction 方法,使用 $\epsilon-$greedy策略确定下一步的动作

- 首先检查是否处于探索模式（explore为 `True`），并且使用 `random.random()` 生成一个 0 到 1 之间的随机数是否小于探索概率 explorationProb。
- 如果满足探索条件，则从所有可能的动作 self.actions 中随机选择一个动作并返回。
- 如果当前状态 state在策略 self.pi 中，返回当前状态 state对应的最优动作。
- 如果当前状态不在策略 self.pi 中，则从所有可能的动作 self.actions 中随机选择一个动作并返回。

(2) 实现 incorporateFeedback 方法,根据当前环境的反馈更新我们对环境的建模以及策略.

- 遍历所有状态和动作的转移计数，计算从 (state, action) 转移到所有 next_state 的总次数。
  - 遍历所有可能的下一个状态及其对应的转移计数，算从 (state, action) 转移到 next_state 的概率。
  - Hint 1: `prob(s, a, s') = (counts of transition (s,a) -> s') / (total transtions from (s,a))`
  - Hint 2: `Reward(s, a, s') = (total reward of (s,a) -> s') / (counts of transition (s,a) -> s')`
  - 将计算得到的 next_state、prob 和 reward 添加到 succAndRewardProb 字典中。
- 运行值迭代算法，并使用计算得到的 succAndRewardProb 和折扣因子 self.discount 来更新策略 self.pi。

PART 3：Q-learning

(1) 实现 getAction 方法,使用 ϵ—greedy策略确定下一步的动作

- 首先检查是否处于探索模式，生成一个 0 到 1 之间的随机数是否小于探索概率，如果满足探索条件，则从所有可能的动作 self.actions 中随机选择一个动作并返回。
- 如果不进行探索，则选择当前状态下 Q 值最大的动作。

(2) 实现 incorporateFeedback 方法,根据当前环境的反馈更新Q表.

- 首先检查当前状态是否为终止状态，如果当前状态是终止状态，则目标值为当前 reward（终止状态的V值为 0）；否则，计算在下一个状态下所有可能动作的最大 Q 值。
- 更新 Q 值：获取当前状态和动作的 Q 值，使用 Q-learning 更新公式更新 Q 值。
  - Q-learning 更新公式: `self.Q[state, action] = q_value + self.getStepSize() * (target - q_value)`
  - self.getStepSize()：获取步长（学习率）。
  - target - q_value：TD 误差（目标值与当前 Q 值之差）

2 超参数

```
parser.add_argument("--mcvi_exprob", type=float, default=0.5, help="ExplorationProb for mcvi training.")
parser.add_argument("--mcvi_episodes", type=int, default=1000, help="The number of episodes for mcvi training.")
parser.add_argument(
    "--tabular_exprob", type=float, default=0.15, help="ExplorationProb for TabularQLearning training."
)
parser.add_argument(
    "--tabular_episodes", type=int, default=2000, help="The number of episodes for TabularQLearning Training."
)
```

3 最好的游戏结果 (输出的图片)

PART 2 运行:

```
python train.py --agent value-iteration
```

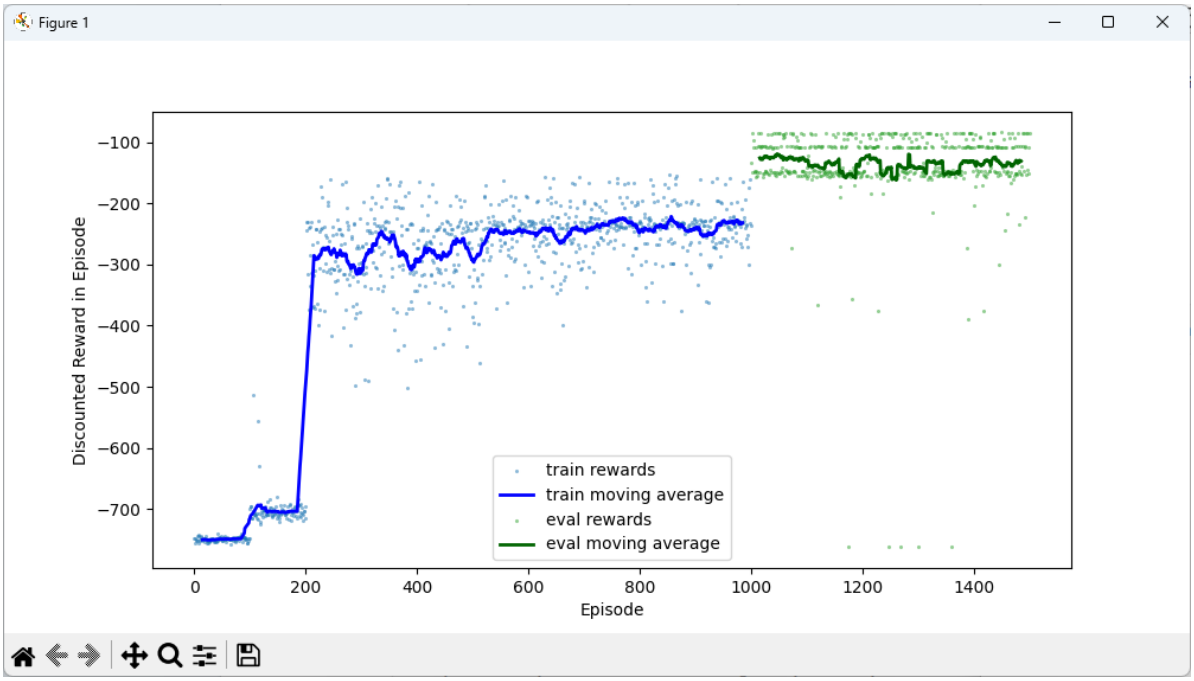


Figure 1
Discounted Reward in Episode
−100
−200
−300
−400
−500
−600
−700
0
200
400
600
800
1000
1200
1400
Episode
train rewards
train moving average
eval rewards
eval moving average

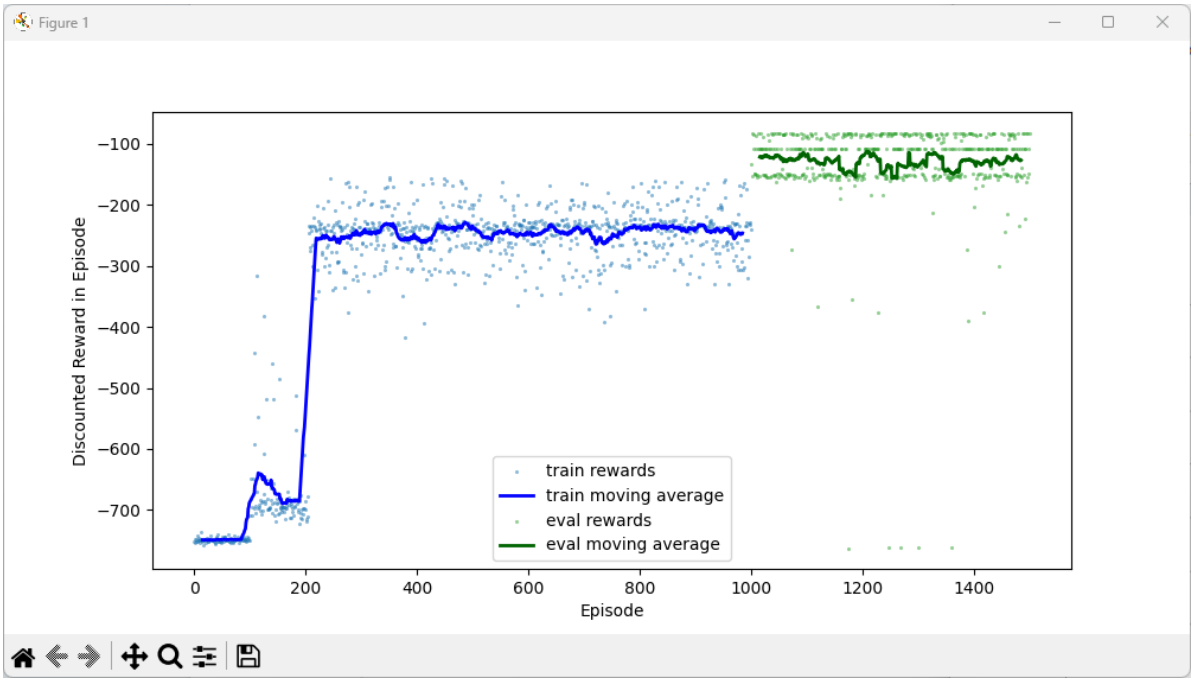


Figure 1
Discounted Reward in Episode
−100
−200
−300
−400
−500
−600
−700
0
200
400
600
800
1000
1200
1400
Episode
train rewards
train moving average
eval rewards
eval moving average

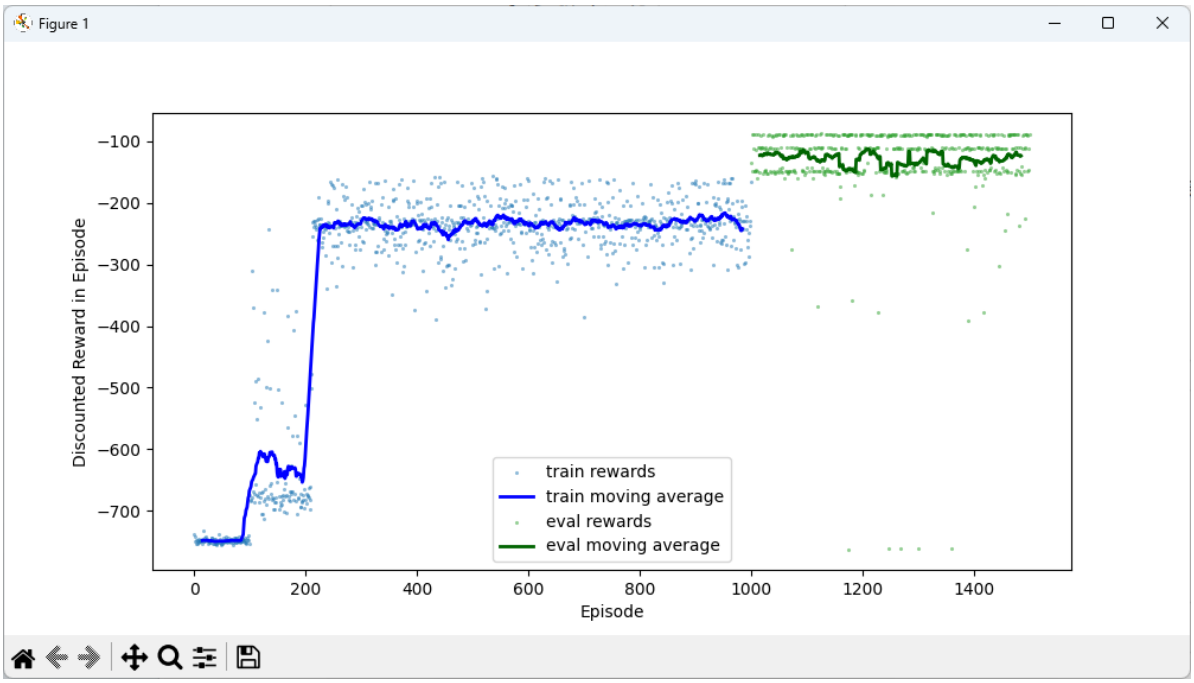
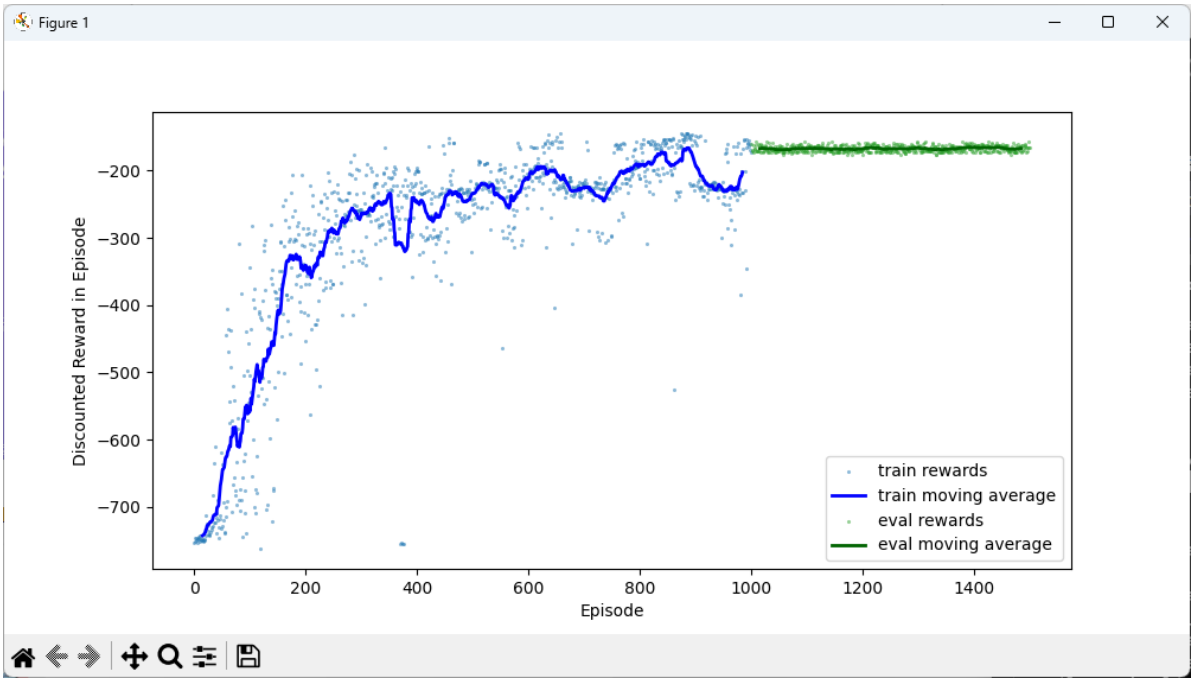


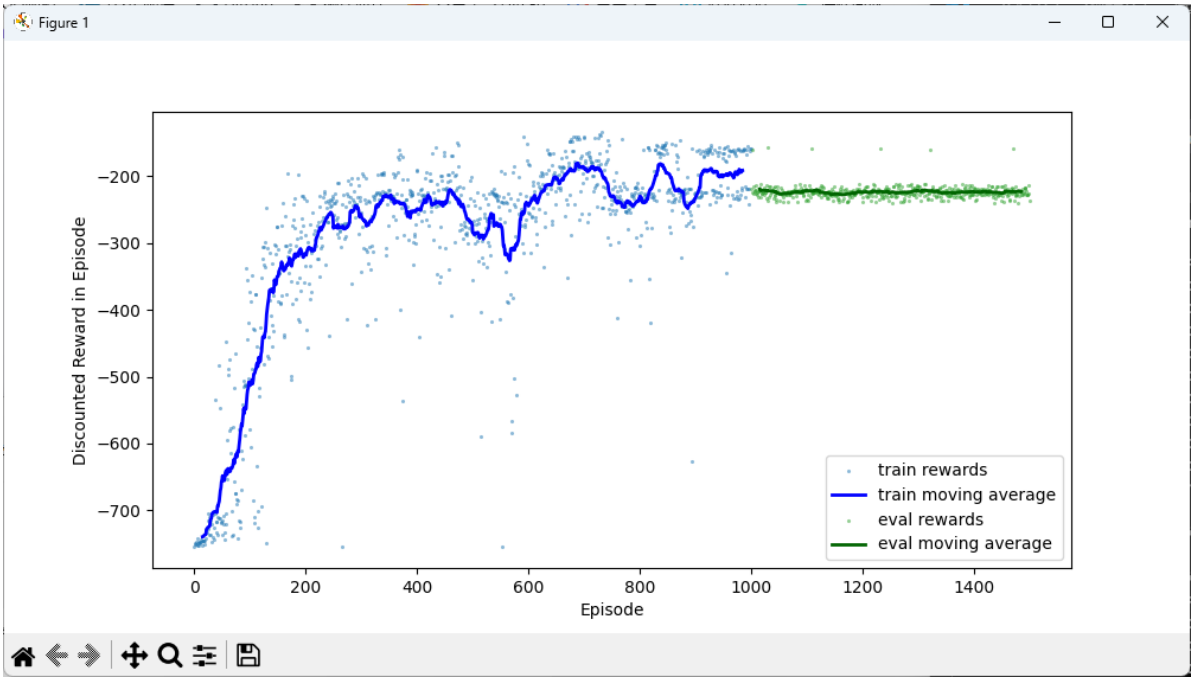
Figure 1
Discounted Reward in Episode
−100
−200
−300
−400
−500
−600
−700
0
200
400
600
800
1000
1200
1400
Episode
train rewards
train moving average
eval rewards
eval moving average

PART 3 运行：

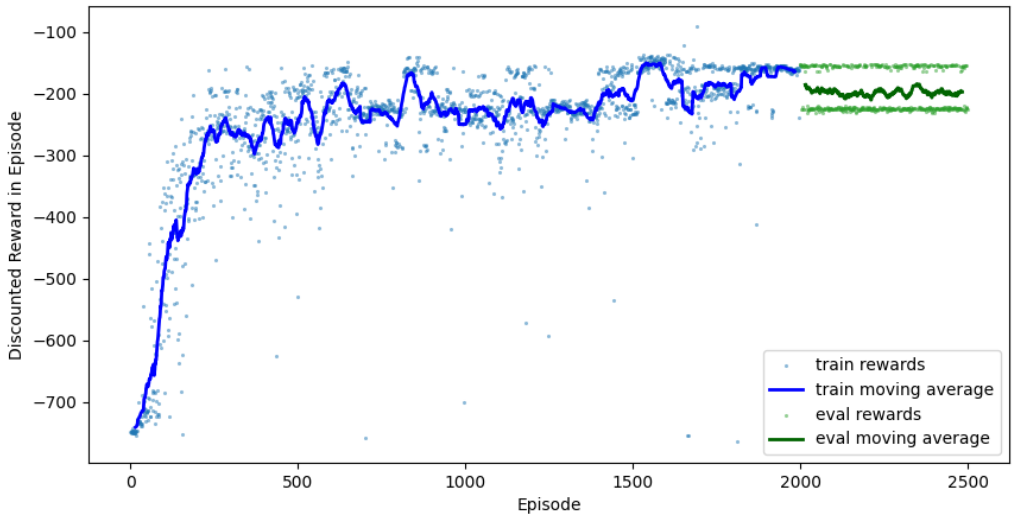
```
python train.py --agent tabular
```

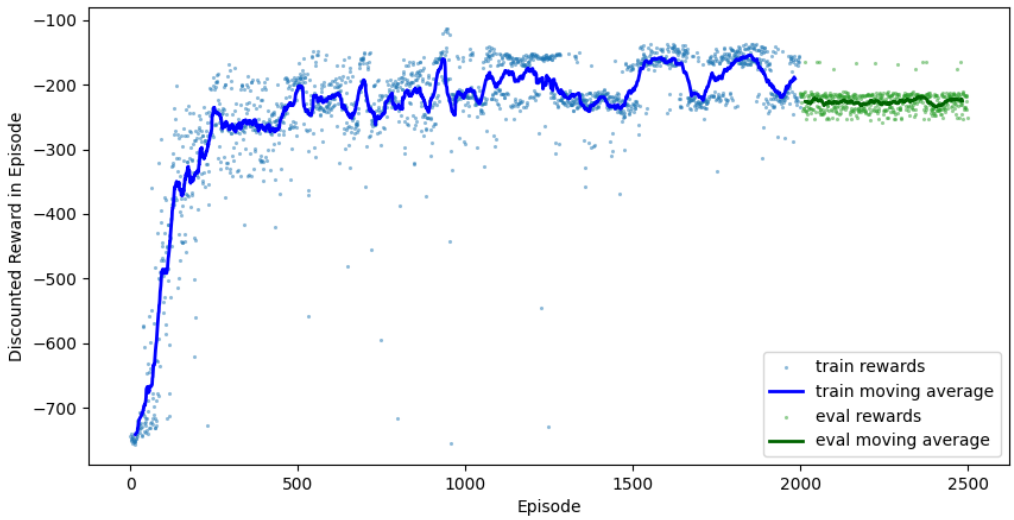
未调参时：

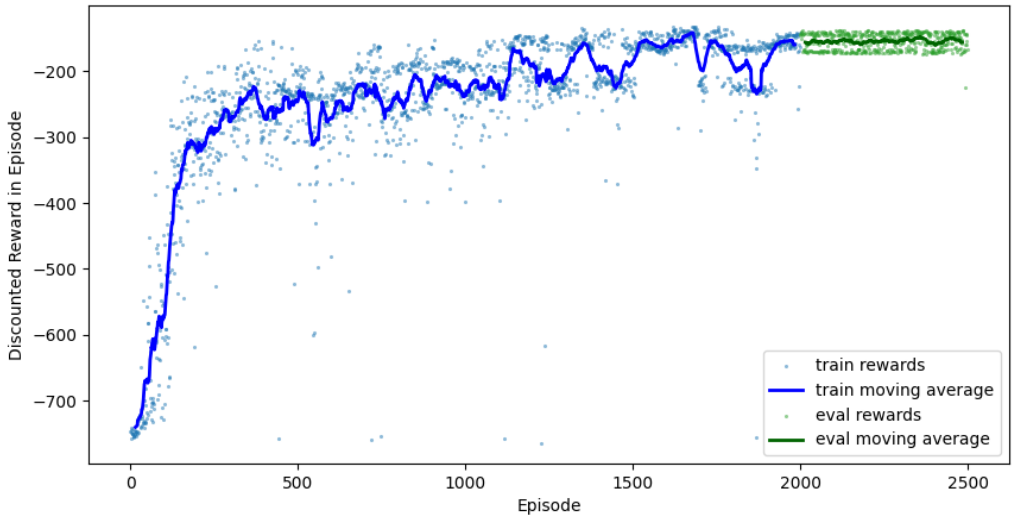




调参 default = 2000








```
管理员: Anaconda Prompt
Trial 1900 (totalReward = -223.6052, Length = 237)
Training complete! Running evaluation, writing weights to ..\results\tabular/tabular.safetensors and generating reward p
lot...
**********************************************************
Trial 3 out of 3
**********************************************************
Trial 0 (totalReward = -747.1591, Length = 1000)
Trial 100 (totalReward = -546.5442, Length = 644)
Trial 200 (totalReward = -273.6391, Length = 277)
Trial 300 (totalReward = -314.08536, Length = 325)
Trial 400 (totalReward = -201.49876, Length = 207)
Trial 500 (totalReward = -265.0538, Length = 272)
Trial 600 (totalReward = -246.46901, Length = 259)
Trial 700 (totalReward = -286.4447, Length = 301)
Trial 800 (totalReward = -215.77498, Length = 215)
Trial 900 (totalReward = -163.31456, Length = 156)
Trial 1000 (totalReward = -265.34555, Length = 264)
Trial 1100 (totalReward = -165.54597, Length = 158)
Trial 1200 (totalReward = -270.54932, Length = 279)
Trial 1300 (totalReward = -220.18393, Length = 229)
Trial 1400 (totalReward = -204.40894, Length = 219)
Trial 1500 (totalReward = -228.20793, Length = 249)
Trial 1600 (totalReward = -228.94954, Length = 251)
Trial 1700 (totalReward = -143.50378, Length = 143)
Trial 1800 (totalReward = -147.08237, Length = 142)
Trial 1900 (totalReward = -230.12401, Length = 245)
Training complete! Running evaluation, writing weights to ..\results\tabular/tabular.safetensors and generating reward p
lot...

(ml24) E:\cylia\USTC-ML24-Fall-main\lab4\src>python grade.py
100%|| 1000/1000 [00:01<00:00, 638.59it/s]
100%|| 1000/1000 [00:01<00:00, 540.39it/s]
ModelBasedMonteCarlo: -130.5150909423828 +/- 37.110103607177734, Win Rate: 0.961
TabularQLearning: -158.80430603027344 +/- 14.416419982910156, Win Rate: 1.0
You got a score of 29.415 out of 30!

(ml24) E:\cylia\USTC-ML24-Fall-main\lab4\src>python grade.py
100%|| 1000/1000 [00:01<00:00, 638.40it/s]
100%|| 1000/1000 [00:01<00:00, 539.59it/s]
ModelBasedMonteCarlo: -130.51548767089844 +/- 37.11042022705078, Win Rate: 0.961
TabularQLearning: -158.80430603027344 +/- 14.416419982910156, Win Rate: 1.0
You got a score of 29.415 out of 30!

(ml24) E:\cylia\USTC-ML24-Fall-main\lab4\src>
```

4 回答问题

- 根据输出的图片讨论与分析算法的性能,在更一般的情况下,有哪些因素会导致Model-based Value Iteration算法失效?
    - **模型不准确**：如果用来估计状态转移概率和奖励的模型不够准确，那么基于此模型的学习过程可能会导致错误的策略。特别是在高维或连续空间中，构建精确的环境模型是非常困难的。
    - **局部最优解**：在某些情况下，算法可能会陷入局部最优解而无法找到全局最优解。特别是当环境具有复杂的结构时，这个问题更加突出。

- **计算资源限制**：对于非常大的状态空间，Value Iteration可能需要大量的时间和内存来进行完整的遍历和更新。这种情况下，即使理论上能找到最优解，实际操作上也可能不可行。
- **稀疏奖励问题**：当奖励信号非常稀疏时（即大部分时间都没有奖励反馈），算法很难有效地引导探索过程，从而难以发现通往目标的有效路径。
- **非马尔可夫性质**：如果环境不是完全马尔可夫决策过程（MDP），也就是说未来状态不仅依赖于当前状态和动作，还依赖于历史信息，那么基于MDP假设的Value Iteration将不再适用。
- **参数选择**：不适当的折扣因子、初始值设定、停止条件（如epsilon阈值）等参数设置都可能影响算法的效果。
- **初始值**：如果Q值或其他相关变量的初始化方式引入了偏差，那么这可能会误导后续的学习过程，使得算法收敛到次优解。

- 填空: 上面实现的 TabularQlearning 是一个 **Model-free**, **Temporal Difference**, **off-Policy** , **Value-based**, **on-line**RL Algorithm.

5 反馈

2days.