USTC AI3002 Final-Project 实验报告

王梓浩　　刘谨睿　　崇时瑞

2026 年 1 月 31 日

1 环境配置

崇时瑞：

- Pytorch: 2.6.0+cu126
- CUDA 可用
- 显卡： NVIDIA GeForce RTX 4070 Laptop GPU

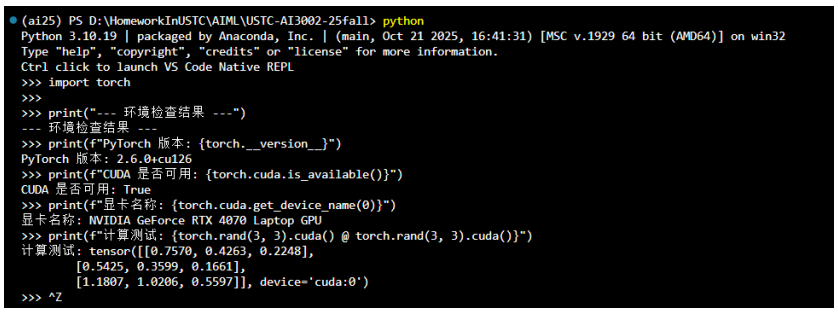


图 1: 相关配置截图

2 Warm-up

2.1 Monte Carlo Learning

2.1.1 First-visit MC核心思想

原理简述：

First-Visit MC 通过计算状态 s 在多个回合中首次出现后的平均回报来估算其价值。其核心逻辑基于大数定律，即经验平均值在样本足够多时会收敛于期望值。

算法核心步骤：

1. 采样：按照策略 π 生成一个完整的回合（Episode）。
2. 识别：在该回合中，仅标记每个状态 s 第一次出现的时刻 t。
3. 更新：计算从 t 开始的回报 $G_t = \sum_{k=0}^{T-t-1} \gamma^k R_{t+k+1}$，并更新价值函数：

$$V(S_t) \leftarrow V(S_t) + \alpha[G_t - V(S_t)] \tag{1}$$

特点总结：

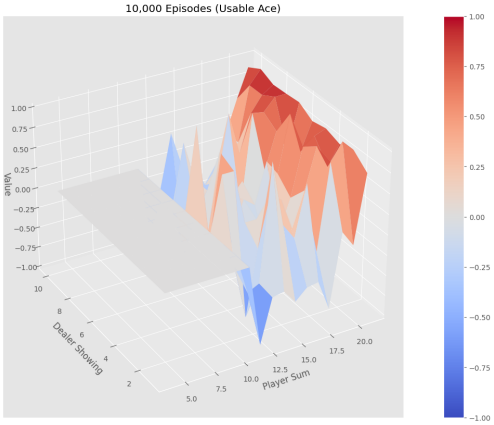
- 无偏性：直接利用实际回报，不依赖其他状态的估计值。
- 收敛慢：必须完成整个回合才能更新，方差通常较大。

2.1.2 Black Jack 实验结果及分析

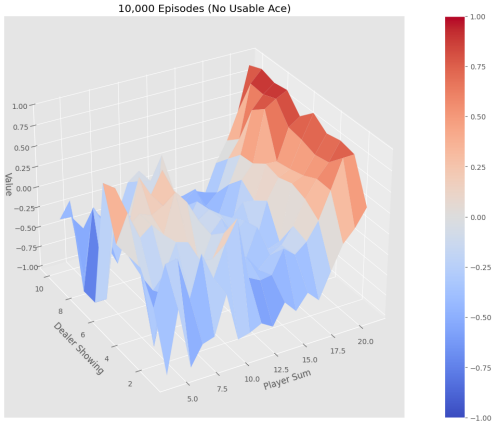
本实验对比了不同训练轮数下，First-Visit MC 算法对 Blackjack 环境价值函数的估计效果。图 2 展示了在是否有 Usable Ace 的情况下，10,000 次与 500,000 次迭代的结果。

结果讨论

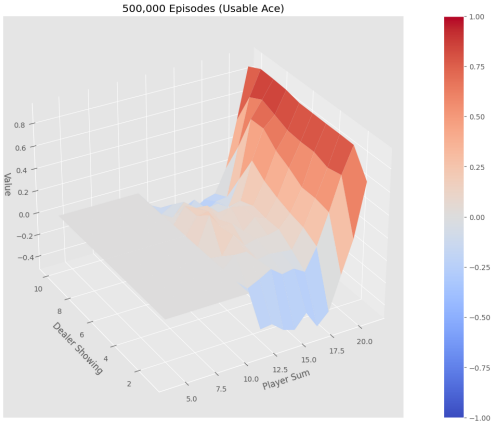
- 样本量对平滑度的影响：从图中可以观察到，10k 次迭代时曲面存在明显的锯齿状波动，这是由于 MC 方法方差较大导致的。当迭代次数增加至 500k 时，价值函数曲面变得平滑且稳定，证明了算法的收敛性。



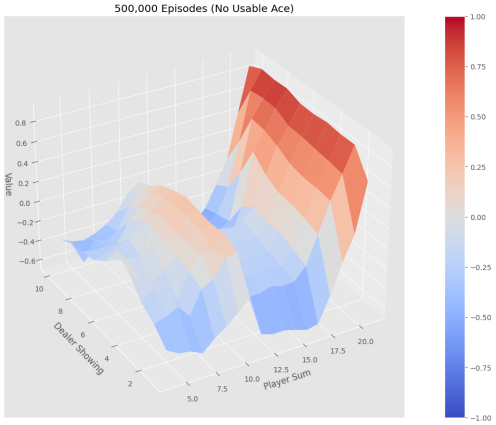
(a) 10,000 Episodes (Usable Ace)



(b) 10,000 Episodes (No Usable Ace)



(c) 500,000 Episodes (Usable Ace)



(d) 500,000 Episodes (No Usable Ace)

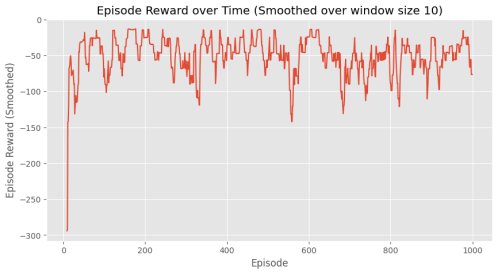
图 2: 不同迭代次数下 MC 算法估算的价值函数曲面对比

- 状态价值分布：在所有情况下，玩家总和达到 20 或 21 时，状态价值均接近 1.0（红色区域），而当庄家明牌为 A 且玩家总和较低时，价值明显降低。
- Usable Ace 的作用：拥有 Usable Ace 的曲面（a, c）在中间区域比无 Ace（b, d）更加平缓，反映了 A 牌作为 11 点使用时提供的额外生存机会。

2.2 Temporal-Difference Learning

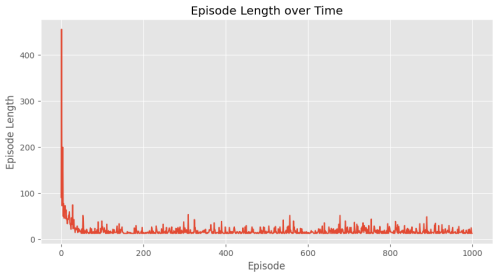
2.2.1 两种算法在 CliffWalking 的训练曲线对比

在 CliffWalking 环境中，我们对比了 Q-learning（Off-policy）与 SARSA（On-policy）的表现。图 3 展示了两者的每回合回报（Episode Reward）与回合步数（Episode Length）的收敛过程。

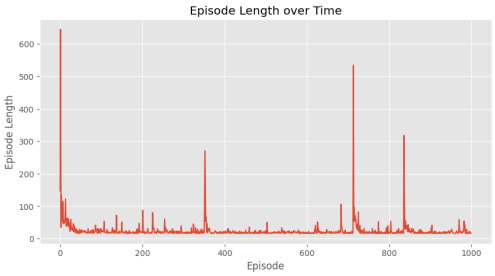


(a) Q-learning 每回合回报

(b) SARSA 每回合回报



(c) Q-learning 回合步数



(d) SARSA 回合步数

图 3: Q-learning 与 SARSA 在 CliffWalking 环境下的训练统计对比

实验观察：

- 收敛速度：从步数曲线可以看出，两种算法在前 100 个回合内均能迅速将步数从数百步降低至 50 步以内，体现了 TD 学习的高效性。
- 回报波动：对比图 (a) 与 (b)，Q-learning 在收敛后的回报曲线上存在密集的负向“深坑”（Spikes），即即便在后期也经常触发 -100 的掉落惩罚。相比之下，SARSA 的收敛曲线更加平稳，其平滑后的平均回报明显高于 Q-learning。

2.2.2 SARSA 与 Q-learning 的差异分析

结合“更安全路径 vs 更激进路径”的现象，两者的核心差异在于对探索风险的处理方式：

1. Q-learning (Off-policy) 的激进策略： Q-learning 学习的是目标策略（Optimal Policy），即沿着悬崖边走的最近路径。然而，由于训练中存在 ϵ-greedy 探索，智能体有概率随机动作。在悬崖边进行随机动作极易导致掉落。因此，虽然它找到了理论最优路径，但在训练过程中的在线表现较差。
2. SARSA (On-policy) 的安全策略： SARSA 在更新价值函数时考虑了当前的探索行为。它“意识到”在悬崖边探索会导致惨重损失，因此它学习到了一条虽然路径稍长、但远离悬崖边缘的安全路径。这使得 SARSA 在存在探索随机性的情况下，能够获得更高的累积奖励。

结论：Q-learning 追求的是“理论最优”，适用于环境模型固定且执行时无随机扰动的场景；而 SARSA 追求的是“行为最优”，在探索阶段或环境具有随机性时表现得更加稳健。

2.3 推荐选做部分

3 Main Task

3.1 利用MCTS进行的模型优化

3.1.1 核心逻辑实现

在 AlphaZero 框架中，我们利用MCTS承担了策略改进器的角色。我们通过在每个状态下进行多次模拟搜索，得到一个比纯神经网络输出更稳健的动作概率分布。为了解决 12×12 大规模棋盘带来的数值不稳定问题，我们在代码中实现了带有温度参数的 Log-Sum-Exp 稳定逻辑。

```
def get_action_probs(self, state, temp=1.0):
    # 执行 n_simulations 次模拟搜索 (n=100)
    for _ in range(self.n_simulations):
        self._simulate(state)

    counts = np.array([child.visit_count for child in root.children.values()])
    actions = list(root.children.keys())

    # 针对温度极低的情况进行分支处理，防止除以零
    if temp < 1e-3:
        probs = np.zeros(self.board_size**2)
        probs[actions[np.argmax(counts)]] = 1.0
        return probs

    # 核心优化：使用 Log 空间计算防止数值溢出
    log_counts = np.log(counts + 1e-8) * (1.0 / temp)
    exp_counts = np.exp(log_counts - np.max(log_counts))
    probs_on_actions = exp_counts / np.sum(exp_counts)

    full_probs = np.zeros(self.board_size**2)
    full_probs[actions] = probs_on_actions
    return full_probs
```

3.1.2 实验结果与效率分析

1. 训练吞吐量统计

在本实验环境下，针对 12×12 规模的五子棋任务，系统在 12 小时的连续训练中完成了约 3800 个自博弈回合（Episodes）。虽然单回合生成的耗时较长（平均约 11.4 秒），但这保证了每一个样本都经过了深度的 MCTS 搜索（每次动作模拟 100 次），从而提供了高质量的梯度方向。

2. 模型性能评估

尽管训练步数有限，但模型的收敛质量超出了预期。经过约 3,000 轮的神经网络参数优化后，我们将该模型与执行完全随机策略的智能体进行了 100 场对局测试。结果显示：模型达到了 100% 的胜率。

这表明模型已经从最初的随机噪声中脱颖而出，成功掌握了五子棋的基础落子逻辑与防守技巧，验证了 MCTS 引导下的策略提升逻辑在 12×12 棋盘上的有效性。由与时间有限，以及MCTS方法的开销较大，后续的训练并未继续。