计算概论 B 大作业：小北的冰未名湖挑战

Topic: 算法实现、文件读写、模拟、强化学习

Name: 孙 海 岳
Student ID: 2100013127

Peking University

2024-11-23

Abstract

在本次大作业中，你将利用 python 构建自己的策略以帮助小北完成冬天即将到来的冰未名湖挑战（不是）。本次作业大致分为两部分，基础部分（90% 分数）需要同学们完成基于蒙特卡洛搜索的算法，附加部分（10% 分数）则需要同学们对算法的部分参数进行微调，并试图分析参数改变对结果影响的原因。还在等什么？小北需要你的帮助，你难道希望小北在冬季未名湖挑战中掉入冰窟窿吗?

1 项目信息

本节包含完成该项目所需的环境配置、基础知识、代码结构、评测方法，请同学们看完这一部分再开启“拯救小北之旅”。

注：相关知识只是相关，具体和本项目有关的，你只需要了解蒙特卡洛方法的思想，以及如果你选择数学证明作为附加题，可能需要了解的信息。

1.1 项目说明

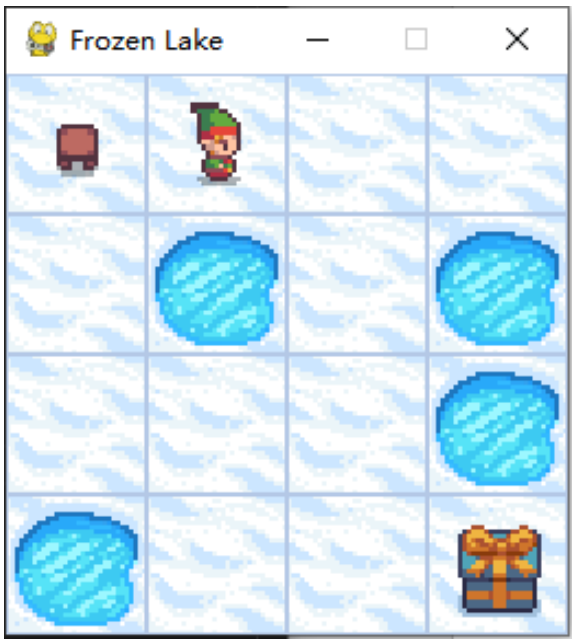

Figure 1: 未名湖挑战示意图

寒冷的冬天又来了，新来的朋友们，你们是否听闻过北大未名湖的种种传奇？所谓腹有诗书气自华，气是否自华我们暂时按下不表，至少未名湖冰面的某些地方难以承载这些诗书的重量，倘若踏错，便会落入深渊，然后重生在校园警卫的身旁，成为北大学生的茶后饭谈。小北深知这种“冰湖挑战”的危害，因此今年提前预谋以身犯险，希望借助算法的力量向新来的同学证明冰湖的危险。受到小北的委托，我们提出了这一项目，希望借助同学们的力量，使小北免于在最终展示中落入未名湖底。（ps. 冬季私自踏入未名湖是非常危险的行为，冰面虽好玩，切忌随意

踏入，“未名溜冰场”会在相关老师确认安全后对外开放，大家请务必注意自身安全)。

本项目将会以 list 的形式提供给大家未名湖冰面的状况图，大家的根本目的是在给定位置的情况下，根据状况图给出当前位置下最好的行动，使得小北能到达终点，并避免在途中掉入冰窟窿。更多详细的描述将会随题目要求一起介绍。

1.2 环境配置

numpy: python 史上最强大的数学运算库之一，其对张量运算的广播处理能极大加速相关运算，除此之外，他也为我们申明多维数组提供了很多便利。本实验只会用到其申明多维数组的功能。

gym：Gym 库 (https://gym.openai.com) 是 OpenAI 推出的强化学习实验环境库。不过大家不要害怕，使用该库仅仅是为了省去大家自己实现生成实验环境这一复杂且乏味的过程，从而能将精力集中在更有意思的实现上。

安装命令：

pip install numpy

pip install gym

1.3 相关知识

蒙特卡洛方法 (Monte Carlo method)，是一种 “统计模拟方法”。20 世纪 40 年代，为建造核武器，冯. 诺伊曼等人发明了该算法。因赌城蒙特卡洛而得名。这种方法是思想并不复杂，举一个很简单的例子，对于一个圆，我们想计算其 pi 的面积，于是我们可以利用蒙特卡洛方法，在圆的外部画一个外切正方形，然后用大量细小的米粒以等概率的方式投入到这个正方形区域内，通过统计落在圆内的米粒的比例 θ 并计算正方形的面积 S，我们可以获得圆面积的粗略估计 $S' = \theta S$。这正是概率的魅力。

具体到该项目中，在给定位置 s 和合法动作集合 $A = \{a_1, a_2, a_3, ...\}$ 的情况下，我们将会使用蒙特卡洛搜索算法给出合理的下一步动作，使小北能顺利到达中点。

最简单的蒙特卡洛搜索包含以下三个步骤：

1. 模拟采样

根据当前位置随机采样一个动作，在环境中执行该动作，然后重复此操作，直到掉入冰窟窿或者到达终点，记录这条路径和最终的 Reward，一次采样结束。

2. 反向传播更新

利用上一步采样的路径和 Reward，我们可以更新我们的策略。具体而言，我们可以更新这条路径上每一个状态及其对应动作的成功概率，从而有利于我们最终选取一个成功概率最高的动作。

3. 信任度上限

为了更好的平衡探索与利用，我们采用信任度上限的方式，为我们的采样过程制定动作选择策略。

一个动作的信任度为：

$$\frac{w_i}{n_i} + c * \sqrt{\frac{lnN_i}{n_i}} \tag{1}$$

其中 w_i 是动作 i 的胜利次数，即走到终点的次数，n_i 是该动作的模拟次数，N_i 是所有的模拟次数，即统计总共走过的步数，c 是探索常数，理论值应该取 $\sqrt{2}$。

1.4 代码结构

该项目的代码可以大致分为两部分 game.py 和 policy.py。

其中关于游戏实现部分将完全在 game.py 中完成，关于策略的部分将完全在 policy.py 中完成。需要注意的是，我们需要优先完成 game.py 中的部分代码，才能正常运行 policy.py 的算法。因此不建议直接去做算法，虽然算法可能看上去更有趣。

大部分代码的具体说明都以注释的形式体现，请认真看注释，因为有些部分可能不希望你去修改，如果你非要修改，可能会需要连带着修改其它的地方。

1.5 评测方法

这里的评测方法只有 policy 的评测，简而言之，你只需要运行 python policy.py 即可，看到一个成功率，理论而言你的成功率应该稳定在 15% 左右，甚至更低，但是放宽心，这并不是你的问题，这是由于蒙特卡洛方法在如此稀疏的 Reward 条件下，本身就十分难以更新价值（毕竟曾经我输出过采样过程，曾经达成了 1000 次采样没有一次到达终点）。

实际上只要成功率合理，并且你对你的代码的解释符合算法要求，即可。

具体评测函数见文末，其原理为根据你输入的策略 pi，在给定地图 map 上，进行多轮测试，最终给出成功率。

2 基础要求

本节包含多项基本要求，以及实现该要求可能用到的部分代码、数学相关知识，请同学们务必看完这部分说明（这其实是废话，不看完你怎么知道要做什么）。

2.1 游戏开发：未名湖挑战 v1.0

这部分的代码将完全在 game.py 中完成，下面将会为大家介绍你一定需要补全的函数，以及其作用了。此后是你可以自由发挥的部分（当然也是要求）。

2.2 必须完成的函数

action_mask3函数用于查看对于地图 map，当前状态 state，返回所有合理的动作。具体的动作见 game.py 的开头，有定义。

step4函数用于查看对于地图 map，当前状态 state 的下一次行动后，游戏是否结束，同时返回游戏的下一个状态，环境反馈的 Reward 和游戏是否接受的 Flag。其中关于地图的 Reward, 每一步的 Reward 都是 0, 到达终点获得 Reward 为 1.

2.3 自由发挥的部分

这部分并没有严格的条条框框，你可以充分发挥你的想象力（只要能完成小北的要求即可）。

要求如下：

1. 实现可以实时操作的冰湖挑战小游戏。你可以使用控制台，每次根据输入的动作，刷新地图或返回动作不合法等。

2. 实现可以存档的游戏。小北想要随时复盘，因此，你的游戏需要在游戏过程中，通过一些举动（比如输入 save 等），实现存档（文件读写，把当前局面存在硬盘上）。并能在游戏开始菜单界面加载已有的存档，并开始继续游戏。

2.4 算法开发：基于蒙特卡洛搜索的小北外置大脑

Algorithm 3 First-Visit Monte-Carlo 策略评估

Input: 待评估策略 π

Initialization: 值函数 $V(s) \in \mathbb{R}$； $V() = 0$； 每个状态的计数器 $N(s) = 0$
for 每个回合 **do**
    使用 π 生成一个回合： $S_0, A_0, R_0, \cdots, S_{T-1}, A_{T-1}, R_{T-1}, S_T$
    $G \leftarrow 0$
    for $t \leftarrow T-1$ to 0 **do**
        $G \leftarrow \gamma G + R_{t+1}$
        if S_t **not in** $S_0, \cdots, S_{t-1}$ **then**
            $N(S_t) \leftarrow N(S_t) + 1$
            $V(S_t) \leftarrow V(S_t) + \frac{1}{N(S_t)}\left[G - V(S_t)\right]$
        end if
    end for
end for

Output: 策略 π 对应的值函数 V

Figure 2: First-Visit 蒙特卡洛算法伪代码

这部分的代码将完全在 policy.py 中实现, 具体的算法思路见上文伪代码2. 这部分你需要补全两个函数, 三部分代码.

sample_actions5函数以地图 map, 当前状态 state 作为输入, 根据等概率采样, 返回一个合法的动作.

compute_qpi_MC6的 greedy 部分, 需要你实现 $\epsilon - greedy$ 采样算法. 该算法的想法是, 我以 $1-\epsilon$ 的概率选择当前的最优动作 (即你的策略), 同时以 ϵ 的概率在所有合法动作中随机选择. 该策略作为一种简单但是广泛应用的方法, 十分优秀的平衡了探索与利用的过程.

compute_qpi_MC7部分需要你按照前文提到的伪代码实现具体的 Reward 的更新, 动作价值函数 (Q 函数) 的更新.

3 附加要求

这部分为同学们准备了多项附加要求，小北并不期望同学们全部完成，选择其中一项完成即可收获小北的 bonus 得分，为了反对“卷王”，完成了多项的同学将会受到小北的惩罚——只按得分最高的一项计分，但是会得到项目组的奖励（视完成人数情况为喜茶、蜜雪冰城）。

3.1 附加选项 1. 强化学习：蒙特卡洛采样的超参数分析

你成功完成了小北的需求, 但是小清对你的作业并不满意. 他认为, 小北的要求太简单了, 我们小清都是算法大佬. 因此他要求你修改前文实现的蒙特卡洛算法的参数贪心参数 ϵ 或衰减函数 γ, 并做实验分析其变化对算法的影响. 为了向小清证明我们的能力 (否则我们可能需要健身), 小北拜托你完成这个任务.

要求:

调整参数, 并给出实验结果. 实验结果可以为图表的形式, 如 γ-训练 epoch-winRate, 来证明参数改变对算法收敛速度的影响等其它你认为符合逻辑的形式.(不确定可以把自己的想法告诉助教). 我们不会在具体形式上难为你, 主要目的是为了鼓励你调试参数, 同时感受强化学习中, 参数对结果的影响.

3.2 附加选项 2. 可视化设计：游戏值得一个更加生动的形式

你成功完成了小北的需求, 但是由于你的游戏过于好玩, 在北大掀起了一股冰湖挑战热潮. 然而许多北大学子并没有像你一样对计算机如此了解, 因此他们渴望一个可视化界面, 而不是对着枯燥的终端, 你能帮帮他们吗?

要求:

可选: 实现可视化 (图形化) 界面, 要求不能直接使用 gym 的图形界面.

可选: 用鼠标控制动作选择.

3.3 附加选项 3. 震惊！未名湖挑战大变革！出现了新的目标！

小北成功完成了挑战, 但是小北的反常现象让北大上下为之哗然, 为了维护北大作为一流大学的颜面, 北大学生会对挑战难度进行增加, 增加了额外的奖励点, 这时小北是否还需要到达终点呢?

要求:

更新环境, 要求每次重置环境时, 在冰面上随机生成一个宝箱位置 G(gold), 到达该位置小北将会获得 Reward 为 100, 此外, 小北在冰面上每走一步将获得 Reward -1. 修改环境, 并尝试你的实验, 看看策略是否有变化.

3.4 附加选项 4. 数学证明：置信度上界 UCT 的推导过程，以多臂老虎机为例

震惊! 由于有大量的数院大佬涌入, 他们表示, 计算机什么的, 哪有算法证明有意思. 小北迫于无奈出了这个题.

要求:

以多臂老虎机为例, 证明前文提到的 UCT 的推导. 具体内容可以参考网上的证明, 但是需要自己清楚每一步的原因, 并写在报告中.

4 实验报告

实验报告应包含你完成的内容。需要注意，你至少应该说明你的环境配置方案（让助教至少能运行）、完成这些任务的思路、简单的代码实现方法、思考题、实验数据。此外，你还可以记录你对该任务的思考，这些思考有可能是算法的改进，或者是你发现了新的库文件能够更好的完成任务等。实验报告无字数、模板要求，把完成内容说明白即可。

5 评分标准

游戏实现设计，能够实现人机交互（即能够正常开始游戏、游玩游戏、能够正常结束）共计 40% 分数;

实现游戏信息存储和加载（能够在游戏过程中保存结果，并能够根据存档重开游戏）共计 20% 分数;

补全蒙特卡洛模拟算法的空（总共两个空，分别为 $\epsilon - greedy$ 采样、计算累计 Reward 并更新动作价值函数）共计 20% 分数

完成实验报告（内容可以包含你的代码结构、配置的环境、你的奇思妙想、你觉得非常出彩的地方等等）共计 10% 分数

附加选项:

1. 蒙特卡洛超参数分析（你可以做 ϵ 变化分析，或 γ 衰减的分析，或自行设计衰减函数等）实验报告中写明你的实验结果，以及你对这种结果的可能的解释，即可 10% 分数。

2. 可视化设计（只要需要实现图形界面，比如做成像前文提到的那个图形样子? 或者实现鼠标操控等等，只要你认为能改进可视化游戏即可，不过可能会得分比较主观，你也可以向助教说明你的工作量来证明价值）共计 10% 分数。

3. 大变革，将环境 Reward 按照说明修改，并重新巡训练得到结果。同时将不同 Reward 的策略进行比较，试分析策略不同的原因。共计 10% 分数。

4. 数学证明。可以从网上搜索别人的证明方法，但是你需要在实验报告中明确写出每一步这么做的原因以体现你的理解，而不是直接抄的（否则会让其它同学觉得不公平），共计 10% 分数

6 附录

1. https://zhuanlan.zhihu.com/p/589652397
2. https://zhuanlan.zhihu.com/p/52339556

Listing 1 游戏框架部分 game.py

```python
def state_int2tuple(state:int)->tuple[int, int]:
    '''
    将 state 从 int 类型转换为 tuple 类型

    对应顺序为:
    [[0, 1, 2, 3],
     [4, 5, 6, 7],
     [8, 9, 10, 11],
     [12, 13, 14, 15]]
    '''
    return (state // 4, state % 4)
def state_tuple2int(state:tuple[int, int])->int:
    '''
    将 state 从 tuple 类型转换为 int 类型
    '''
    return state[0] * 4 + state[1]

def create_env()->list:
    '''
    创建游戏环境，返回当前游戏的状态
    :return: 返回当前游戏的整张地图的 list 形式
    例如:
    [['F', 'F', 'F', 'F'],
     ['F', 'S', 'H', 'F'],
     ['H', 'F', 'F', 'F'],
     ['F', 'F', 'F', 'G']]
    其形状一定是 4x4 矩阵。
    '''
    env = gym.make("FrozenLake-v1")  # 创建环境
    env.reset()
    state = env.render('ansi')
    state = list(filter(lambda x : x == 'F' or x == 'S'\
    or x == 'H' or x == 'G', state))
    state = [state[i * 4: (i + 1) * 4] for i in range(4)]
    return state
```

Listing 2 policy.py 评测函数

```
def test_pi(map:list[list], pi, num_episodes=1000):
    """
    测试策略。
    参数:
        env -- OpenAI Gym 环境对象。
        pi -- 需要测试的策略。
        num_episodes -- 进行测试的回合数。

    返回值:
        成功到达终点的频率。
    """

    count = 0
    for e in range(num_episodes):
        ob = reset(map)
        while True:
            a = pi[ob]
            ob, rew, done = step(map, ob, a)
            if done:
                count += 1 if rew == 1 else 0
                break
    return count / num_episodes

    return state
```

Listing 3 处理请求

```
def action_mask(map:list[list], state:int)->list[int]:
    '''
    根据所在地图位置，返回合法的动作集合
    '''


    '''
    你的代码
    '''
    pass


    return [0, 1, 2, 3]
```

Listing 4 处理请求

```
def step(map:list[list], state:int, action:int)->tuple[int, int, bool]:
    '''
    根据当前地图位置和动作，返回下一步的地图位置和 Reward 值和是否结束游戏的 Flag
    '''

    pass

    return (state, 0, False)
```

Listing 5 处理请求

```
def sample_actions(map:list[list], state:int)->int:
    '''
    根据所在位置等概率采样所有可能动作
    '''
    pass
    return action_mask(map, state)[0]
```

Listing 6 处理请求

```
# 根据策略选择动作，采用 epsilon-greedy 方式采样（需要补全）
            # ====================================================
            action = 0
            pass
            # if np.random.rand() < epsilon:
            #     action = ?
            # else:
            #     action = ?
            # ====================================================
```

Listing 7 处理请求

```
        # 计算回报值的累积，并更新动作价值函数
        pass
        for _, (state, action, reward) in enumerate(reversed(episode)):
            # 需要补全 G 的更新方式
            # ==================================
            G = 0
            # ==================================
            if action == None:
                continue
            sa = (state, action)
            # 如果该状态-动作对没有被访问过，更新 N 和 Q
            if sa not in visited:
                visited.add(sa)
                state = int(state)
                action = int(action)
                # =========== 补全 Q 的更新方式，并且添加访问 =================
                N[state][action] += 0
                Q[state][action] += 0
                # ================================================
```