作业 2：强化学习基础

清华大学软件学院
软件工程探索与实践（人工智能板块），2023 年春季学期

1 介绍

本次作业需要提交说明文档（PDF 形式）和 Python 的源代码。注意事项如下：

- 本次作业满分为 100 分。
- 作业按点给分，因此请在说明文档中按点回答，方便助教批改。
- 请不要使用他人的作业，也不要向他人公开自己的作业，否则将受到严厉处罚，作业分数扣至-100（即倒扣本次作业的全部分值）。
- 统一文件的命名：{学号}_{姓名}_hw2.zip

2 REINFORCE

在本题中，你将基于 CartPole-v0[1]环境，利用强化学习算法控制平衡木。本题需要完成以下内容，提交代码和实验报告，代码见 `./code/policy_gradient`。

1. 补充 `REINFORCE` 类中的 `learn` 函数，实现 REINFORCE 算法。
2. 请绘制 REINOFRCE 的训练曲线，包括训练过程的损失函数的变化和最终奖励值。由于强化学习的不稳定性，你的结果需要基于至少 3 个种子。

提示：

1. 动手之前，请仔细阅读代码中的注释，确保你已了解问题定义和代码框架。
2. 你可以解除位于 122 行的注释以获取可视化结果，env.render() 会渲染环境，让你看到平衡木的控制结果。
3. 本题已经提供 Policy Gradient 算法的代码框架，希望你完成损失函数部分，可以参考论文[2]与课件，REINOFRCE 算法位于第 3 讲课件第 51 页。
4. PyTorch 框架在本题中的用法等价于 numpy，比如可以通过 `torch.mean` 计算均值，通过 `torch.std` 计算方差。在 debug 过程中，如果你无法确定一个 Tensor 的形状，你可以使用

[1] `https://www.gymlibrary.dev/environments/classic_control/cart_pole/`

[2] `https://homes.cs.washington.edu/~todorov/courses/amath579/reading/PolicyGradient.pdf`

Tensor.shape 获取之。如果你之前没有安装过 PyTorch，推荐通过 conda 安装 cpu 版本，具体命令请参考[3]。

5. PyTorch 在计算时会保存计算图，以供 autograd 模块自动求导。因此请注意在计算时不要使用 torch.tensor() 创建中间变量，因为这样创建出的变量是值拷贝，不在计算图上，不会计算梯度。可以使用 torch.cat() / torch.stack() / torch.gather() 创建中间变量。

3 围棋的强化学习环境

我们的期末大作业是使用 AlphaZero 框架实现一款围棋智能体，本次作业是期末大作业的环境模块。在本题中，你将根据给出的规则，在现有代码框架的基础上实现一个围棋强化学习环境。本题需要完成以下内容，提交代码和实验报告，代码见 `./code/alphazero`。

1. 补全 Board 和 GoGame 类，实现围棋的强化学习环境，你可以参考之前给出的井字棋棋盘与环境。
2. 利用助教给出的测例（`GoTest.py`）和自己生成的测例，进行单元测试。
3. 确保 GoGame 类和你之前实现的 MCTS 模块是适配的。（无需报告，完成即可）

规则：

注意：为了方便编程实现和训练模型，本次作业中的围棋规则和真实的围棋规则略有不同。

1. **基本规则：** 棋盘为 $n \times n$ 网格，每个格子只能放一枚棋子。n 为奇数。对局双方各执黑色、白色棋子。棋子数量无限。
2. **棋子的气：** 一枚棋子上、下、左、右四方向相邻的空格点称为气。被对方棋子占据的格点不能称为气。通过上、下、左、右四方向相邻的同色棋子，视为不可分割的整体，其中任意一枚棋子的气都与其它棋子共享。一枚棋子必须有气，否则会被消除。
3. **落子：** 黑棋先手，黑白双方轮流落子，每次可落一枚棋子，且只能落子在空格点上。双方都可以在轮到自己落子时，选择不进行落子，这称为 “虚着”。选择虚着后，继续对方的落子环节。若无处可落子，则只能选择虚着。
4. **提子：** 己方落子后，如果有对方棋子无气，将所有对方无气的棋子从棋盘上移除；己方落子后，如果双方都有无气棋子，也只将对方棋子从棋盘上移除。
5. **禁着：** 如果在某格点落子后，提取所有对方无气棋子后，仍存在己方棋子无气，则该格点禁止落子。
6. **打劫：** 落子后，如果正好提掉对方的一个子，下回合对手落子时，禁止在此处落子以提掉本回合的落子。
7. **终局：** 连续的两次落子，黑白双方都虚着，则游戏结束。若双方落子超过 $n \times n \times 4$，则游戏结束。游戏结束后，棋盘上每个棋子为对应颜色的一方记一分；棋盘上的每个空格点，如果仅与某颜色上下左右四方向相邻，且通过上下左右四方向连通的所有空格点也仅与该颜色相邻，则为该颜色记一分。得分高者胜利，若得分相同，则为平局。(样例请参见 `GoTest.py`)

提示：

1. 打劫规则是为了防止以下情况 (图1)：（1）时白棋在 A 处落子，提取 B 处的黑子，然后（2）中黑子再在 B 处落子，会导致局面（3）和（1）完全相同，且双方可以无限重复上述操作。

[3] `https://pytorch.org/get-started/locally/`

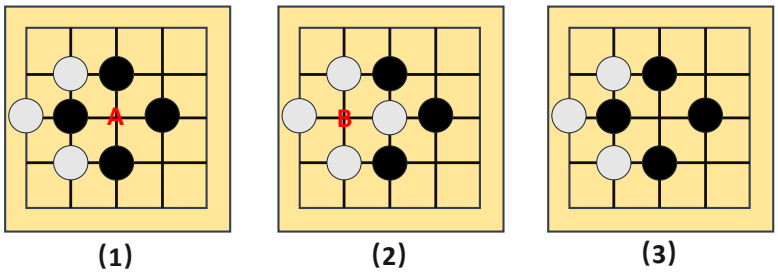

图 1

在实际实现中，你只需要保证在本次落子和上次对手落子都提取且恰好提起一个棋子的情况下，本次落子位置不是上次提取位置即可。

2. MCTS 中使用由棋盘生成的字符串作为字典的键值，请在实现过程中注意需要哪些信息才能唯一确定一个棋盘状态（仅仅记录棋子的位置可以吗?）。
3. Board 的实现仅为参考，我们推荐将落子、提子、计算得分、计算合法落子位置等操作在 Board 中实现，但你也可以根据自己的思路编写。助教提供的测试脚本基于本次作业提供的 Board 框架，仅作参考，你可以使用里面的测例，也可以根据自己的实现思路进行修改。但请注意，我们提供的测例相对于本次作业的规则要求和 MCTS 的需求是不充分的，你需要自行补充一些测例以确保程序的正确性。错误的围棋环境实现可能会为下一次作业带来很大的调试困难。