上海交通大學

SHANGHAI JIAO TONG UNIVERSITY

AI1603-3-五子棋项目报告

第 13 组

小组成员学号 521030910356, 521030910395, 521030910397

姓名 王凯灵，卜家梓，李奇睿

1、 简介

主要实现了两个智能体。分别对应两条算法设计路线：1）纯策略赋分五子棋 AI，通过扫描棋盘上双方落子情况，对每个可落子点分配不同的分数，再根据分数的高低评估在各个点落子行为的优先级，选取最优行为进行落子。2）基于 DQN 强化学习思想[1]，实现了 GitHub 上首个基于 TensorFlow 2.x 的多模式五子棋强化学习网络模型，并部署了相应智能体。

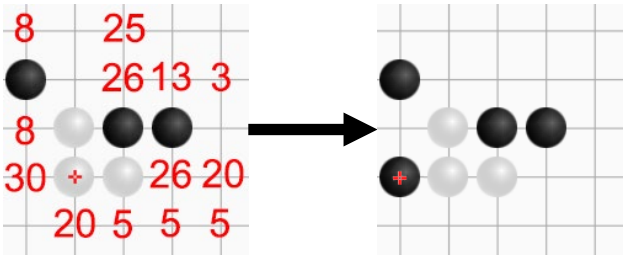
2、 策略实现方案及其可行性分析

2.1 纯策略赋分智能体

本方法最初实现于 6 月 30 日。基本思想是仿照专家系统列举棋盘情况，枚举最优解。使用的棋盘情况列举完全来自于组员个人的五子棋实战，为避免重复，设计思路无互联网参考，皆为原创。

2.1.1 基本逻辑（图一）

列举的棋盘情况是线性序列，长度最大为 7（亦可为 6、5）。为避免棋盘边缘损失，从 observation 中读取棋盘状态后，先添加四周长度为 2 的边界，将其变为一个 19×19 的方阵。之后从横竖和对角线四个方向用一个长度为 7 的序列扫描棋盘方阵，根据棋局落子情况，将每个可落子点的得分填入和棋盘方阵尺寸相同的分数矩阵 score 中，再随机选取最高得分落子点之一作为最优解进行落子。



图一 基本逻辑

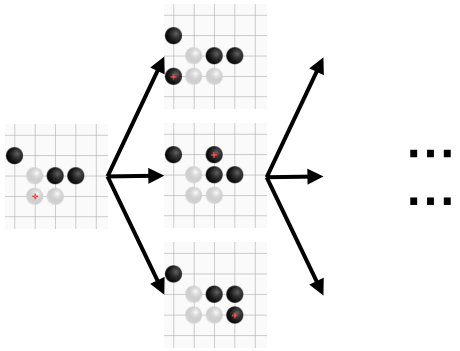
2.1.2 得分表设计

我们在设计落子点的分数时，遵循对称原则和自下而上的分级原则。对称原则是将同一种情况分为自己的和对手的落子。如果某一点对于对方进攻有利，则就是对我方防守有利。改变双方落子对总分的配比可以实现进攻和防守的策略调整。分级原则保证每一级的分数都至少是下一级分数的 4 倍以上(例如落子在 A 点，我们可将已经存在的己方活三进一步连成活四；落子在 B 点，我们可以堵截对手的活三。那么 A 点在本次扫描中的得分将至少高于 B 点得分的 4 倍，即 A 的优先级将会超过 B，除非在扫描其他方向时，B 处出现比构成活四更佳的落子方案以证明 B 更优)，以防止由于扫描网格不同方向扫描得分叠加造成的次优解取代最优解现象。

2.1.3 改进思路

这一智能体在和人对下时已经有不错的水平，但是对局到了后期时，防守能力偏弱，且当对手利用不同方向威胁较小的落子同时构成多个威胁较大的情况，我方没有防守能力。于是在这一智能体的基础上，增加了两个相应对策。

首先，优化赋分策略，并且使用两套不同赋分策略，一套为均衡，一套偏防守。当棋盘落子情况满足一定条件，则智能体赋分策略有均衡变为防守，通过不求胜利只求平局的方式避免失败。其次，进行搜索（图二）。在模拟多个较优落子情况并代替对手模拟落子的条件下，生成 **n** 个回合以后的棋盘。这样，智能体就拥有了一定的预防能力。我们讲初始实现记为方法 **151**，两次增加对策先后记为 **151v2** 和 **151v3**，以方便下文进行比较。



图二 搜索方法

此外，组员还给出另一种思路：**1.** 额外设计两个 **15×15** 的矩阵，分别存储每个落子点的“**win_tag**”和“**lose_tag**”**(**每一点这两个参数都初始化为 **0)**，每当该点出现一个己方次优解时，**win_tag+=1**，最后扫描一遍 **win_tag** 矩阵，当某一点的 **win_tag>=2** 时，它的得分上升为此时除决胜步外最优解的分数；同理，每当该点出现一个对方次优解时，**lose_tag+=1**，最后扫描一遍 **lose_tag** 矩阵，当某一点的 **lose_tag>=2** 时，它的得分上升为此时除决胜步之外最优解的分数。以此来规避部分“配合”问题。**2.** 将扫描网格由条状网格更改为 **5×5** 的方格，使问题维度上升，这样可以考虑列举出 **5×5** 网格内所有的连子情况。

进过测试，最后的智能体已经稳定和本组组员打成平手。确定该算法具有较高智能，可行度高。

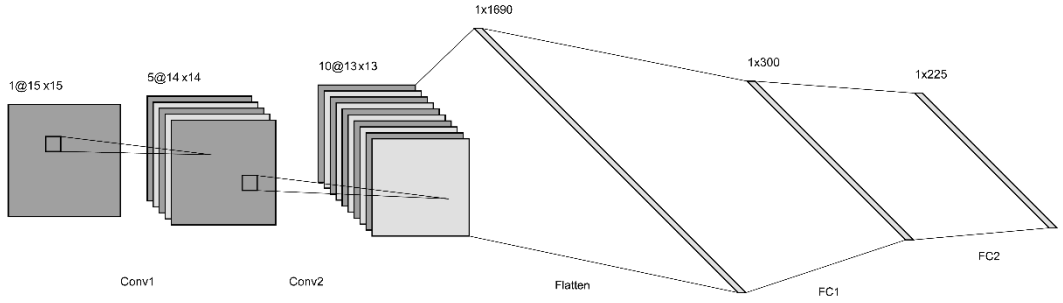
2.2 强化学习智能体

考虑时间因素，选择使用了原理较为简单的 **DQN** 进行强化学习。最初于 **7** 月 **4** 日使用 **pyTorch10.2-cuda11.6** 实现，此后一周进行模型修改、调参和训练。

2.2.1 网络结构

模型网络结构（图三）为：两个 **filter** 数量分别为 **5** 和 **10** 的卷积层和两个全连接层。该网络结构设计思路为通过卷积得到棋盘上一个棋块的特征信

息，再使用两个全连接映射到动作。由于算力和时间关系，没有使用网格搜索调整超参数。



图三 网络结构

2.2.2 价值函数

如果仿照 **AlphaGo** 的设计，应当使用蒙特卡洛数搜索等方法得到动作价值。奈何算力限制，只能实现使用单独设计的扫描棋盘赋分的 **demo** 进行训练。如果用前一方法的赋分直接一致，容易导致因为对手不断出招，使得己方动作价值一直为负，己方赌掉对方落子的价值不如放任对方胜利，学习效率底下，智能体摆烂。所以使用了另外的价值评价方法。此外，如果落子周围有己方落子就给奖励，这样的价值函数也能取得有限效果。

2.2.3 重新实现

因对 **pyTorch** 熟练度不高，出于性能方面的考虑，几日后使用了最新的 **tensorflow2.9** 重新实现了所有功能并使用 **GPU** 进行训练，实测训练速度大约达到 **pyTorch** 实现的 **3-5** 倍。

2.2.4 训练方法

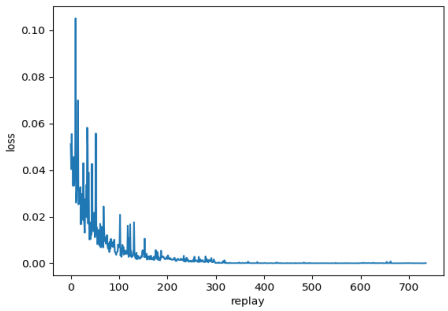
训练时可选择采用多模式，采用前一算法（实际为 **151v2** 微调得来的 **151v4**）作为 **coach**，可以用 **agent** 与 **coach** 进行训练，也可以用 **coach** 自对弈生成对局经验以训练 **agent**，或者 **agent** 自对弈。由于前面的 **coach** 落子具有一定随机成分，所以不会产生过于重复的训练样本。

实际测试时进行了两次正式的训练，1）用 **6000** 场的 **coach** 对局训练模型，再使模型自对弈互相训练 **30000** 场。自对弈前 **3000** 场 **mes loss** 如图四。有收敛迹象。2）初始化后自对弈 **30000** 场，前 **3000** 场 **loss** 如图五。由于给出的 **reward** 函数不能很好的应对乱下的情况，第二钟训练的 **loss** 一致较低但浮动大。

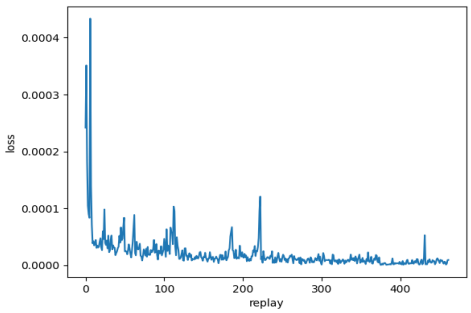
3、结果分析

纯策略手段的智能程度依赖于实现文件包含的落子情况数量以及分数配置的科学性。在模型设计初期，由于列举情况数量较少以及分数分配的误差，导致模型的实际对局可能出现严重失误。基于前文所述的两个应对策略，我们更新的 **151v2**、**151v3** 模型能够适应更加复杂的对局；同时我们重新调

整了分数的分配原则，精准赋分，这也使模型选出最优解的能力增强不少。



图四 loss1



图五 loss2

关于搜索方法，虽然代码中包含了较强的剪枝，但剪枝过程不一定总能发生，这也导致了实际测试时可能出现超时性能问题。最后未采用这一方法。

3.1 模型胜率对比

各对局 1000 场计算胜（败）率如下表（先后手各 500，不含平局）：

概率 / 对局	胜	败	平
151vs151	0%	0%	100%*
151vs151v2	39%	45%	16%
151vs151v3	50%	50%	0%
151v2vs151v2	25%	29%	46%
151v2vs151v3	41%	44%	45%
151v3vs151v3	50%	50%	0%

*151 无随机选取

关于强化学习获得的结果。由于参数修改重新训练的周期长，这一方法还有很大改进空间。目前实现的的模型经过测试，仅在前期具有可观的智能，可以 1000：0 必胜 random 智能体，且可以和 coach 进行 20 个回合的对局。随着价值函数的改进和继续训练，该模型可以达到更好的效果。

4、结论

静态纯策略赋分方法在整体设计上相对简便，只需要记录人类经验，省去了繁杂的模型训练调参过程。但缺点是，对于本课程而言，代码难度较低，实现门槛低，智能体质量完全取决于人对五子棋的理解，且一旦代码完成，

棋力就确定了。搜索方法虽然具备更高的智能，可以演化未来的棋局，但时间上存在门槛，剪枝方法不能处理最坏时间。

强化学习方法虽然训练调参耗时，但部署以后十分轻量，且理论上没有棋力上限。虽然本组最后只提交了最稳定的静态赋分方法，但我们相信我们的强化学习模型有潜力可以超过静态策略。

5、项目分工

王凯灵（队长）：主要算法设计和建模、代码实现、报告撰写，海报文案、绘图

卜家梓：部分棋谱设计和对局研究、评分表设计、**tag** 方法思路、报告撰写、海报文案

李奇睿：部分棋谱设计和对局研究、评分表设计、搜索算法思路

参考文献

[1]. Datawhale, *Easy RL: 强化学习教程*, 2022.

[2]. 代码实现 repo