博 弈 论 实 验 报 告

---及第网站上五子棋的实现

院 （系）：计软智学院 专 业：人工智能

组员信息：58119213 徐丹颖（理解网站逻辑+接口部分）50% ；

58119321 李雨辰（书写五子棋规则类） 50%。

Jidi 网站队名：Chris 实验时间：2021 年 12 月

徐丹颖，女，本科，.E-mail: xdy31415926@126.com. 李雨辰，男，本科，.E-mail: 1290498160@qq.com .

基于静态规则的五子棋博弈游戏

徐丹颖 58119213[1)] 李雨辰 58119321[1)]

[1)](东南大学 计算机科学与工程学院、软件学院、人工智能学院, 江苏省南京市 211189)

摘 要 五子棋是一种两人对弈的纯策略型棋类游戏，棋盘与围棋通用。对战双方分别使用黑白两色的棋子，下在棋盘直线与横线的交叉点上，先形成五子连线者获胜。考虑到我们的组员均喜爱五子棋并有所了解，而且考虑到及第网站所给出的评测要求，因此我们选择使用基于五子棋规则的方法对五子棋进行编写。该规则重点考虑了几种可能在棋局中遇见且能够在一定程度上决定获胜与否的情况，并为每种情况赋予了不同权重。我们根据及第网站的相关开源代码对其对弈游戏的整体架构进行了分析与学习，并按照及第网站要求将我们的五子棋规则嵌入。我们的模型在上传后得到了**状元**的排名。

关键词 五子棋；规则；及第网站；Python

Gobang game based on static rules

Danying Xu 58119213[1)] Yuchen Li 58119321[1)]

[1)](Department of Artificial Intelligence, Southeast University, Jiangsu Nanjing 211189)

Abstract Gobang is a pure strategy board game in which two players play against each other. The board and Go are common. Both sides of the game use black and white chess pieces and place them on the intersection of the straight and horizontal lines of the chessboard. The first to form a line of five pieces wins. Considering that our team members all love and understand gobang, and considering the evaluation requirements given by the website, we chose to use the method based on gobang rules to write gobang. The rule focuses on several situations that may be encountered in a chess game and can decide whether to win or not to a certain extent, and assigns different weights to each situation. We analyzed and studied the overall structure of the game based on the relevant open source code of the Jidi website, and embedded our Gobang rules in accordance with the requirements of the Jidi website. Our model got the top ranking after uploading.

Key words Gobang; Rules; and Section Website; Python

1 模型

1.1 JIDI网站对弈游戏架构分析

及第网站设计有两个部分，一种“run_log.py”文件允许玩家本地设计游戏并允许在本地 debug，二是网站使用“/env/example/main.py”完成在线模型操作与玩家对弈。

图 1 是及第网站在 GitHub 上给出的关于网站上所有博弈游戏的本地运行框架（在线运行的并未列举出来）。

```
|-- platform_lib
        |-- README.md
        |-- run_log.py          // 本地调试运行环境
        |-- examples    // 提交运行文件示例       需包含 my_controller 函数输出policy
            |-- random.py  // 随机策略  需根据环境动作是否连续 调整 is_act_continuous 的值
        |-- replay              // render工具，用于非gym环境，打开replay.html上传run_log 存储的.json文件
        |-- env         // 游戏环境
        |       |-- simulators          // 模拟器
        |       |       |-- game.py
        |       |       |-- gridgame.py // 网格类模拟器接口
        |       |-- obs_interfaces              // observation 观测类接口
        |       |       |-- observation.py              // 目前支持Grid Vector
        |       |-- config.ini          // 相关配置文件
        |       |-- chooseenv.py
        |       |-- snakes.py
        |       |-- gobang.py
        |       |-- reversi.py
        |       |-- sokoban.py
        |       |-- ccgame.py
```

图 1

通过分析该项目的代码可知，在本地运行文件需要通过操作“run_log.py”文件。该文件通过“if __name__ == "__main__"”对“render_game(g, fps=1)”进行调用，进一步调用“get_joint_action_eval(game, multi_part_agent_ids, policy_list, actions_spaces, all_observes)” 函数。

该函数可以对个人有权限修改的文件“submission.py”中的“my_controller()”进行调用，并实现用户的所有设计。其中在每一步中，主函数会获得两个玩家在当前情况下的最佳决策，然后根据当前玩家顺序取得对应玩家的最终决策。

另外，通过阅读网站上图 1 未列出的其他代码文件，我们发现该网站设计逻辑非常精美。网站整体通过class的封装和继承，实现了各种游戏的统一。

网站游戏整体上可以分为两个分类思路：一是判断游戏是否为网格类游戏（比如五子棋是网格类游戏），二是判断游戏是否为连贯性操作（比如贪吃蛇是连贯性操作，五子棋是离散型操作）。

网站操作部分通过主函数获得特定游戏的网站统一设定（比如游戏玩家所需数量等）调用相关游戏类文件（比如五子棋是“gobang.py”）和用户文件进行实现。在线对弈定义了agents类型（base, multi, single）并允许通过及第算法库进行智能体训练并上传模型。

1.2 题目分析

及第网站所给出的规则为：

对战双方双方分别使用黑白两色的棋子，下在棋盘直线与横线的交叉点上，先形成五子连线者获胜。

（1）棋盘初始化为空。

（2）一方获胜或者到达指定步数，游戏结束。

（3）观测为一个字典，其中的键为"state_map"、"chess_player_idx"、"board_width"和"board_height"。"state_map"对应的值为 15*15*1 的矩阵，表示当前棋盘的状态，其中的值为 0、1、2，0 表示没有棋子，1 表示黑棋，2 表示白棋；"chess_player_idx"的值为当前棋手的序号；"board_width"的值为棋盘的宽；"board_height" 的值为棋盘的高。

（4）动作空间为长度为n_action_dim的列表，其中n_action_dim=2，每个元素为Gym当中的Discrete类（ Discrete Link），[Discrete(15), Discrete(15)]

（5）获胜的一方reward为 100，输家为 0，平局双方各得 50。

图 2 展示了网站初始化的“submission.py”相关代码的解释。

```
def my_controller(observation, action_space, is_act_continuous=False):
    """
    @param observation: 字典格式,其中的键为"state_map","chess_player_idx","board_width","board_height".
                        "state_map":对应的值为15*15*1的矩阵,表示当前棋盘的状态,其中的值为0,1,2,0表示没有棋子,1表示黑棋,2表示白棋.
                        "chess_player_idx"的值为当前棋手的序号,"board_width"的值为棋盘的宽:"board_height" 的值为棋盘的高.
                        {state_map:
                        chess_player_idx:
                        board_width:
                        board_height:}
    @param action_space: 动作空间为长度为n_action_dim的列表,其中n_action_dim=2.
                         每个元素为Gym当中的Discrete类(Discrete Link)->[Discrete(15), Discrete(15)]
                         动作空间相当于棋子的坐标,第一个Discrete类代表的是横坐标,第二个Discrete类代表的是纵坐标
                         Discrete类见utils/discrete.py
    @param is_act_continuous: 是否需要连续动作,在五子棋中应该就是false
    @return agent_action: 该决策此次执行的动作,在随机策略中,应该是返回两个长度为150的列表,
                          第一个数组中1代表横标上哪一个位置上放置,第二个数组中的1代表纵坐标上的哪一个位置上放置
    """
    agent_action = []  # 返回的决策,也就是棋子放置的位置
    for i in range(len(action_space)):  # 因为有两个维度(横坐标和纵坐标),所以先采样横坐标,后采样纵坐标
        action_ = sample_single_dim(action_space[i], is_act_continuous)
        agent_action.append(action_)
    return agent_action


def sample_single_dim(action_space_list_each, is_act_continuous):
    each = []
    if is_act_continuous:
        each = action_space_list_each.sample()
    else:
        # 五子棋的两个维度都是离散的,所以会进入这里
        if action_space_list_each.__class__.__name__ == "Discrete":
            # 在五子棋中,先生成初始的十五个点
            each = [0] * action_space_list_each.n
            # 随机采样一个点,在那个位置放置棋子
            idx = action_space_list_each.sample()
            each[idx] = 1
        elif action_space_list_each.__class__.__name__ == "MultiDiscreteParticle":
```

图 2

在网站初始化的“submission.py”文件内，“my_controller (observation, action_space, is_act_continuous=False)” 通过调用下游的函数“sample_single_dim (action_space_list_each, is_act_continuous)“实现了五子棋游戏的坐标初始化。

我们的任务是如何将五子棋游戏的判断嵌入“my_controller()”函数中。该函数传入observation和action_space两个参数。Observation是一个字典，key分别为："state_map"、"chess_player_idx"、"board_width"和"board_height"。

1.3 我们的模型

我们的模型可以主要分为基于规则的五子棋实现和网络接口嵌入两部分。

1.3.1　五子棋部分

五子棋的规则通过“class Robot(object)”进行封装。该类实现了查找当前棋盘上可下棋坐标的函数，包含各种不同局势下赋予价值的函数。我们首先简单说一下关于五子棋不同局势的描述。首先连五很好理解，即下在此位置后会形成赢局。另外，对于已有四子的情况会产生活四和眠四。活四指的是有两个点可以成五的四；眠四指的是只有一个点可以成五的四。同理，我们可以定义出活三、眠三、活二和眠二。以眠三举例，眠三是能够形成冲四的三。如图 3 所示，就是眠三的基本形态，其中白点都代表冲四点。和活三相比，这种玩法的危险系数也会相对较低，主要在于眠三形式及时不放手，也会出现冲四。

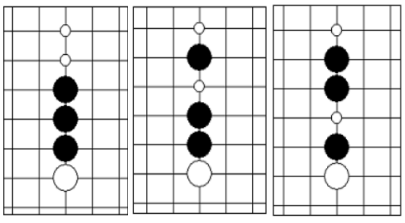

图 3

我们扫描整个棋盘，并在某格为空时扫描该点周围 8*8 的空间下棋情况。我们在每点设置四个 list。List1 表示该点横行前后共 8 个位置所有下棋情况，list2 表示该点纵行前后共 8 个位置所有下棋情况，list3 表示该点左斜行前后共 8 个位置所有下棋情况，list4 表示该点右斜行前后共 8 个位置所有下棋情况。然后我们根据价值赋值获得对我们最有利的点并返回坐标。

如果已有四子且可形成连五，则该点利益最大，我们赋值为 10000。对于剩下的情况，我们认为活局比眠局更好，（n-1）个点的活局比 n 个点的眠局好，能够直接相连比需要隔一子好，因此我们的赋值如下。活四赋值 5000，冲四赋值 1700。活三赋值 1900 或 1600（取决于不同周围情况，比如下在此处后空一格形成隐式活三就不如直接三子相连后形成活三的收益大，因此我们将其赋值略降低以保证能最先形成相连的点），眠三赋值 350 或 300。活二赋值 99，眠二赋值 5，其他赋值为 0。我们将每个可下棋的点的价值通过四个方向相加，得到该点的最终价值，并取最高价值点返回。

1.3.2 网络接口部分

我们通过定义“class check()”类将五子棋价值判断的类封装起来。我们设计了“ai_play（）”的类成员函数，在当前玩家的棋子颜色下调用价值类并计算总价值。

另外，在类外我们又编写了“change_form(x,y)”函数，将 15*15 棋盘的决定坐标转换为主函数规定的格式。

在及第网站规定的“my_controller(observation, action_space, is_act_continuous=False)”函数中，我们通过以上两个函数将决定坐标返回到主函数。

2 算法

2.1 目标

我们的目标是通过五子棋不同局势下通过静态方法进行决策。

2.2 可能方法

我们在最初开展实验时参考了一些已有工作。我们发现目前主要方法除了基于规则的方法外，还有许多采用深度学习，基于蒙特卡洛树，或者是强化学习的方法。考虑到我们的模型需要保证在 1s 内做出判断，因此我们选取了基于五子棋规则的模型。另一种可能情况是通过强化学习得到可以直接进行判断的模型，但由于未学习过强化学习的相关内容，在理解和实现上的难度较大。

2.3 实现

五子棋的基本形式被分成：连五，活四，冲四，活三，眠三，活二，眠二。具体的算法过程与代码实现部分完全一样，具体可见 1.3.1 部分。

2.4 设计优势

我们的模型能够通过不同状态下的局势，返回代表不同重要程度的不同返回值，从而确定当前整体情况下最优下棋策略。

2.5 潜在改进方向

Louis Victor Allis 在 1994 年的Searching for Solutions in Games and Artificial Intelligence[2]一文中提出，五子棋游戏在无禁手前提下，先手有绝对获胜策略。由于考虑到网站题目规定时间因素，即策略需在 1s内做出。因此我们在实现中暂时未实现。未来工作可以通过分别考虑玩家是否为先手进行进一步的改进。

3 评估

3.1 JIDI网站排名

图 4 图 5 是我们的模型提交到及第网站后的 15 小时及一天后的排名情况，积分分别是 0.97 和 1.00，均得到了**状元**排名。第二天虽然并列第一但说明我们的排名还是很靠前的。

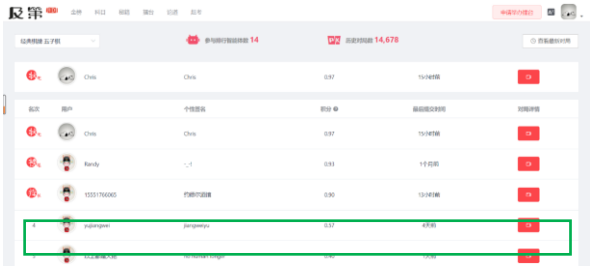

图 4

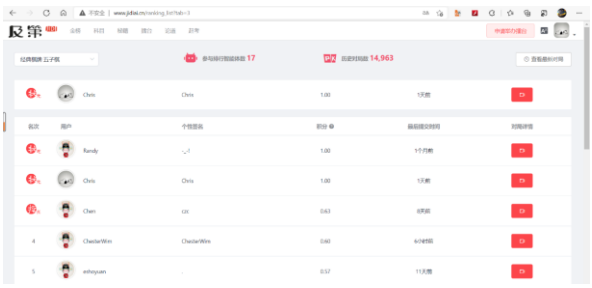

图 5

3.2 模型性能分析

下面通过精度，时间特性的要求和灵活性三个方面对我们的模型进行了性能分析。

3.2.1 精度

通过 3.1 的结果，可以看到我们的模型确实能够达到很高的精度。分别是 0.97 和 1.00。在完成

此报告前（截止至 2021 年 12 月 22 日）均能够达到第一名的水平。

3.2.2 时间特性的要求

及第网站的评测说明要求如下：

（1）该环境在金榜的积分按照最新 30 局均分进行计算并排名。

（2）平台验证和评测时，在单核CPU上运行用户代码（暂不支持GPU），限制用户每一步返回动作的时间不超过 1s，内存不超过 500M。

考虑到我们的模型需要保证在 1s内做出判断，因此我们选取了基于五子棋规则的模型。虽然能达到 1s内做出判断，但我认为在某些情况下判断速度上还是有欠缺的。

3.3 灵活性

我们将五子棋当前棋盘状态下的价值计算部分封装到了“Robot(object)”类，将决策函数包括robot类共同封装到“check()”类。这样我们的代码可以在以后维护和迁移时都有很好的融合性。

4 项目总结

4.1 项目中遇到的困难与解决方法

1. 在开始做项目的第一周，我们卡在了及第网站的游戏架构代码上。理清楚整体架构的逻辑以及我们的工作非常耗费时间。

2. 在配相关环境上，我们也花了许多的时间。后来通过新建 conda 的新虚拟环境得到了最终的解决。

3. 在“run_log.py”中 agent_id==0 是黑棋，agent_id==1 是白棋；而在“submission.py”中传入的 observation “chess_player_idx”是 1 和 2。写代码时会混淆。

4. 在接口 debug 中，我们需要考虑到组员代码的不一致性，需要一遍一遍通过输出发现代码中格式不一致的问题。

4.2 项目总结

在本次项目中，我们收获到了很多知识。

从博弈论方面来讲，我们了解到了相关基于蒙特卡洛树，基于强化学习，基于规则，基于深度学习等方面实现五子棋AI算法的可能性，这加深了我们对于博弈论这门课程的知识掌握与实际拓展。

从及第网站的五子棋代码实现方面来讲，我们通过阅读及第网站博弈游戏的架构代码，学习到了中科院水平的python项目构建知识，主要是如何构建一个具有逻辑性、可操作性和安全性的支持更新的python项目，以及关于python类的相关代码知识。

5 细节说明

5.1 设备

本地设备使用 Windows10-64bit-CPU 系统。

调试软件采用 PyCharm 2018.1.4 x64 的专业版完成。

参 考 文 献

[1] Allis L V. Searching for solutions in games and artificial intelligence[M]. Wageningen: Ponsen & Looijen, 1994.

[2]Lin Hua. Gobang intelligent game robot based on Self-Play[D]. Zhejiang University.

(林华. 基于 Self--Play 的五子棋智能博弈机器人[D]. 浙江大学.)

[3] https://en.wikipedia.org/wiki/Gomoku

附录I

```
from env.gobang import GoBang
from env.obs_interfaces.observation import *
from env.simulators.gridgame import GridGame
class Robot(object):
    '''基于五子棋规则写的一个机器人'''

    def __init__(self, _board):
        self.board = _board

    def haveValuePoints(self, player, enemy, board):
        """算出棋盘中所有有价值的点"""
        points = []

        for x in range(15):
            for y in range(15):
                list1 = []
                list2 = []
                list3 = []
                list4 = []
                if self.board[x][y] == 0:
                    for tmp in range(9):
                        i = x + tmp - 4
                        j = y + tmp - 4
```

```
                    # 搜索的范围是 x,y 的周围 8×8 的空间
                    if i < 0 or i > 14:
                        list1.append(-1)
                    else:
                        list1.append(board[i][y])
                    if j < 0 or j > 14:
                        list2.append(-1)
                    else:
                        list2.append(board[x][j])
                    if i < 0 or j < 0 or i > 14 or j > 14:
                        list3.append(-1)
                    else:
                        list3.append(board[i][j])
                    k = y - tmp + 4
                    if i < 0 or k < 0 or i > 14 or k > 14:
                        list4.append(-1)
                    else:
                        list4.append(board[i][k])

                player_score = self.value_point(player, enemy, list1, list2, list3, list4)
                enemy_score = self.value_point(enemy, player, list1, list2, list3, list4)
                if enemy_score >= 10000:
                    enemy_score -= 500
                elif enemy_score >= 5000:
                    enemy_score -= 300
                elif enemy_score >= 2000:
                    enemy_score -= 250
                elif enemy_score >= 1500:
                    enemy_score -= 200
                elif enemy_score >= 99:
                    enemy_score -= 10
                elif enemy_score >= 5:
                    enemy_score -= 1
                value = player_score + enemy_score
                if value > 0:
                    points.append([x, y, value])
        #print("list1:",list1)
        #print("list2:", list2)
        #print("list3:", list3)
        #print("list4:", list4)
        return points

    def MaxValue_po(self, player, enemy):
        """算出最大价值的点，先求取有价值点，再求其中的最大价值点。"""
        '''
        if player==2 and enemy==1:
            player=1
            enemy=0
        elif player==1 and enemy==2:
            player = 0
            enemy = 1
        '''
        points = self.haveValuePoints(player, enemy, self.board)
        print(self.board)
        #print("points:", points)
        flag = 0
        _point = []
        for p in points:
            if p[2] > flag:
                _point = p
                flag = p[2]
        #如果棋盘上还没有棋子
        if len(_point)==0:
            _point.append(7)
            _point.append(7)
            _point.append(99)
        #print("_point:",_point)
        return _point[0], _point[1], _point[2]  # 返回 x, y, value

    def value_point(self, player, enemy, list1, list2, list3, list4):
        """算出点的价值"""
        flag = 0
        flag += self.five(player, list1)
        flag += self.five(player, list2)
        flag += self.five(player, list3)
        flag += self.five(player, list4)
        flag += self.alive4(player, list1)
        flag += self.alive4(player, list2)
```

```
        flag += self.alive4(player, list3)
        flag += self.alive4(player, list4)
        flag += self.sleep4(player, enemy, list1)
        flag += self.sleep4(player, enemy, list2)
        flag += self.sleep4(player, enemy, list3)
        flag += self.sleep4(player, enemy, list4)
        flag += self.alive3(player, list1)
        flag += self.alive3(player, list2)
        flag += self.alive3(player, list3)
        flag += self.alive3(player, list4)
        flag += self.sleep3(player, enemy, list1)
        flag += self.sleep3(player, enemy, list2)
        flag += self.sleep3(player, enemy, list3)
        flag += self.sleep3(player, enemy, list4)
        flag += self.alive2(player, enemy, list1)
        flag += self.alive2(player, enemy, list2)
        flag += self.alive2(player, enemy, list3)
        flag += self.alive2(player, enemy, list4)
        flag += self.sleep2(player, enemy, list1)
        flag += self.sleep2(player, enemy, list2)
        flag += self.sleep2(player, enemy, list3)
        flag += self.sleep2(player, enemy, list4)
        return flag

    @staticmethod
    def five(player, compare):
        """下在这个点将会得到连五"""
        if compare[0] == player and compare[1] == player
and \
                compare[2] == player and compare[3] ==
player:
            return 10000
        elif compare[5] == player and compare[6] ==
player and \
                compare[7] == player and compare[8] ==
player:
            return 10000
        elif compare[2] == player and compare[3] ==
player and \
                compare[5] == player and compare[6] ==
player:
            return 10000
        elif compare[1] == player and compare[2] ==
player and \
                compare[3] == player and compare[5] ==
player:
            return 10000
        elif compare[3] == player and compare[5] ==
player and \
                compare[6] == player and compare[7] ==
player:
            return 10000
        else:
            return 0

    @staticmethod
    def alive4(player, compare):
        """下在这个点将会形成活四"""
        if compare[0] == 0 and compare[1] == player and
\
                compare[2] == player and compare[3] ==
player \
                and compare[5] == 0:
            return 5000
        elif compare[3] == 0 and compare[5] == player
and \
                compare[6] == player and compare[7] ==
player \
                and compare[8] == 0:
            return 5000
        elif compare[1] == 0 and compare[2] == player
and \
                compare[3] == player and compare[5] ==
player \
                and compare[6] == 0:
            return 5000
        elif compare[2] == 0 and compare[3] == player
and \
                compare[5] == player and compare[6] ==
player \
                and compare[7] == 0:
            return 5000
        else:
            return 0

    @staticmethod
    def sleep4(player, enemy, compare):
        """下在这个点会形成眠四"""
```

```
        if compare[0] == enemy and compare[1] == player
and \
                compare[2] == player and compare[3] ==
player \
                and compare[5] == 0:
            return 1700
        elif compare[1] == enemy and compare[2] ==
player and \
                compare[3] == player and compare[5] ==
player \
                and compare[6] == 0:
            return 1700
        elif compare[2] == enemy and compare[3] ==
player and \
                compare[5] == player and compare[6] ==
player \
                and compare[7] == 0:
            return 1700
        elif compare[3] == enemy and compare[5] ==
player and \
                compare[6] == player and compare[7] ==
player \
                and compare[8] == 0:
            return 1700
        elif compare[0] == 0 and compare[1] == player
and \
                compare[2] == player and compare[3] ==
player \
                and compare[5] == enemy:
            return 1700
        elif compare[1] == 0 and compare[2] == player
and \
                compare[3] == player and compare[5] ==
player \
                and compare[6] == enemy:
            return 1700
        elif compare[2] == 0 and compare[3] == player
and \
                compare[5] == player and compare[6] ==
player \
                and compare[7] == enemy:
            return 1700
        elif compare[3] == 0 and compare[5] == player
and \
                compare[6] == player and compare[7] ==
player \
                and compare[8] == enemy:
            return 1700
        else:
            return 0

    @staticmethod
    def alive3(player, compare):
        """下在这个点会形成活三"""
        if compare[0] == 0 and compare[1] == 0 and \
                compare[2] == player and compare[3] ==
player \
                and compare[5] == 0:
            return 1900
        elif compare[1] == 0 and compare[2] == 0 and \
                compare[3] == player and compare[5] ==
player \
                and compare[6] == 0:
            return 1900
        elif compare[2] == 0 and compare[3] == 0 and \
                compare[5] == player and compare[6] ==
player \
                and compare[7] == 0:
            return 1900
        elif compare[1] == 0 and compare[2] == player
and \
                compare[3] == player and compare[5] ==
0 \
                and compare[6] == 0:
            return 1900
        elif compare[2] == 0 and compare[3] == player
and \
                compare[5] == player and compare[6] ==
0 \
                and compare[7] == 0:
            return 1900
        elif compare[3] == 0 and compare[5] == player
and \
                compare[6] == player and compare[7] ==
0 \
                and compare[8] == 0:
            return 1900
```

```
        elif compare[0] == 0 and compare[1] == player
and \
                compare[2] == player and compare[3] ==
0 \
                and compare[5] == 0:
            return 1600
        elif compare[2] == 0 and compare[3] == player
and \
                compare[6] == player and compare[5] ==
0 \
                and compare[7] == 0:
            return 1600
        elif compare[3] == 0 and compare[5] == player
and \
                compare[7] == player and compare[6] ==
0 \
                and compare[8] == 0:
            return 1600
        elif compare[3] == 0 and compare[5] == 0 and \
                compare[7] == player and compare[6] ==
player \
                and compare[8] == 0:
            return 1600
        elif compare[0] == 0 and compare[1] == player
and \
                compare[2] == player and compare[3] ==
0 \
                and compare[6] == 0:
            return 1600
        elif compare[0] == 0 and compare[1] == player
and \
                compare[2] == player and compare[3] ==
0 \
                and compare[6] == 0:
            return 1600
        else:
            return 0

    @staticmethod
    def sleep3(player, enemy, compare):
        """下在这个点会形成眠三"""
        if compare[1] == enemy and compare[2] == player
and \
                compare[3] == player and compare[5] ==
0 \
                and compare[6] == 0:
            return 350
        elif compare[2] == enemy and compare[3] ==
player and \
                compare[5] == player and compare[6] ==
0 \
                and compare[7] == 0:
            return 350
        elif compare[3] == enemy and compare[5] ==
player and \
                compare[6] == player and compare[7] ==
0 \
                and compare[8] == 0:
            return 350
        elif compare[0] == 0 and compare[1] == 0 and \
                compare[2] == player and compare[3] ==
player \
                and compare[5] == enemy:
            return 350
        elif compare[1] == 0 and compare[2] == 0 and \
                compare[3] == player and compare[5] ==
player \
                and compare[6] == enemy:
            return 350
        elif compare[2] == 0 and compare[3] == 0 and \
                compare[5] == player and compare[6] ==
player \
                and compare[7] == enemy:
            return 350
        elif compare[0] == enemy and compare[1] == 0 and
\
                compare[2] == player and compare[3] ==
player \
                and compare[5] == 0 and compare[6] ==
enemy:
            return 300
        elif compare[1] == enemy and compare[2] == 0 and
\
                compare[3] == player and compare[5] ==
player \
                and compare[6] == 0 and compare[7] ==
enemy:
```

```
            return 300
        elif compare[2] == enemy and compare[3] == 0 and
\
                compare[5] == player and compare[6] ==
player \
                and compare[7] == 0 and compare[8] ==
enemy:
            return 300
        elif compare[0] == enemy and compare[1] ==
player and \
                compare[2] == 0 and compare[3] == player
\
                and compare[5] == 0 and compare[6] ==
enemy:
            return 300
        elif compare[1] == enemy and compare[2] ==
player and \
                compare[3] == 0 and compare[5] == player
\
                and compare[6] == 0 and compare[7] ==
enemy:
            return 300
        elif compare[2] == enemy and compare[3] ==
player and \
                compare[5] == 0 and compare[6] == player
\
                and compare[7] == 0 and compare[8] ==
enemy:
            return 300
        elif compare[0] == enemy and compare[1] ==
player and \
                compare[2] == 0 and compare[3] == player
\
                and compare[5] == 0 and compare[6] ==
enemy:
            return 300
        elif compare[1] == enemy and compare[2] ==
player and \
                compare[3] == 0 and compare[5] == player
\
                and compare[6] == 0 and compare[7] ==
enemy:
            return 300
        elif compare[3] == enemy and compare[5] == 0 and
\
                compare[6] == player and compare[7] ==
player \
                and compare[8] == 0:
            return 300
        elif compare[0] == enemy and compare[1] ==
player and \
                compare[2] == player and compare[3] ==
0 \
                and compare[5] == 0:
            return 300
        elif compare[2] == enemy and compare[3] ==
player and \
                compare[5] == 0 and compare[6] == player
\
                and compare[7] == 0:
            return 300
        elif compare[3] == enemy and compare[5] ==
player and \
                compare[6] == 0 and compare[7] == player
\
                and compare[8] == 0:
            return 300
        elif compare[0] == player and compare[1] ==
player and \
                compare[2] == 0 and compare[3] == 0 \
                and compare[5] == enemy:
            return 300
        elif compare[2] == enemy and compare[3] ==
player and \
                compare[5] == 0 and compare[6] == 0 \
                and compare[7] == player:
            return 300
        elif compare[3] == enemy and compare[5] ==
player and \
                compare[6] == 0 and compare[7] == 0 \
                and compare[8] == player:
            return 300
        elif compare[0] == player and compare[1] == 0
and \
                compare[2] == 0 and compare[3] == player
\
                and compare[5] == enemy:
```

```
            return 300
        elif compare[1] == player and compare[2] == 0
and \
                compare[3] == 0 and compare[5] == player
\
                and compare[6] == enemy:
            return 300
        elif compare[3] == enemy and compare[5] == 0 and
\
                compare[6] == 0 and compare[7] == player
\
                and compare[8] == player:
            return 300
        elif compare[0] == 0 and compare[1] == player
and \
                compare[2] == player and compare[3] ==
0 \
                and compare[5] == enemy:
            return 30
        elif compare[2] == 0 and compare[3] == player
and \
                compare[5] == 0 and compare[6] == player
\
                and compare[7] == enemy:
            return 300
        elif compare[3] == 0 and compare[5] == player
and \
                compare[6] == 0 and compare[7] == player
\
                and compare[8] == enemy:
            return 300
        elif compare[0] == 0 and compare[1] == player
and \
                compare[2] == 0 and compare[3] == player
\
                and compare[5] == enemy:
            return 300
        elif compare[1] == 0 and compare[2] == player
and \
                compare[3] == 0 and compare[5] == player
\
                and compare[6] == enemy:
            return 300
        elif compare[3] == 0 and compare[5] == 0 and \
                compare[6] == player and compare[7] ==
player \
                and compare[8] == enemy:
            return 300
        elif compare[0] == player and compare[1] == 0
and \
                compare[2] == player and compare[3] ==
0 \
                and compare[5] == enemy:
            return 300
        elif compare[1] == enemy and compare[2] ==
player and \
                compare[3] == 0 and compare[5] == 0 \
                and compare[6] == player:
            return 300
        elif compare[2] == player and compare[3] == 0
and \
                compare[5]== 0 and compare[6] == player
\
                and compare[7] == enemy:
            return 300
        elif compare[3] == enemy and compare[5] == 0 and
\
                compare[6] == player and compare[7] ==
0 \
                and compare[8] == player:
            return 300
        else:
            return 0

    @staticmethod
    def alive2(player, enemy, compare):
        """下在这个点会形成活二"""
        if compare[1] == 0 and compare[2] == 0 and \
                compare[3] == player and compare[5] ==
0 \
                and compare[6] == 0:
            return 99
        elif compare[2] == 0 and compare[3] == 0 and \
                compare[5] == player and compare[6] ==
0 \
                and compare[7] == 0:
            return 99
        elif compare[0] == 0 and compare[1] == 0 and \
```

```
            compare[2] == 0 and compare[3] == player
\
            and compare[5] == 0 and compare[6] ==
enemy:
            return 99
        elif compare[1] == 0 and compare[2] == 0 and \
            compare[3] == 0 and compare[5] == player
\
            and compare[6] == 0 and compare[7] ==
enemy:
            return 99
        elif compare[1] == enemy and compare[2] == 0 and
\
            compare[3] == player and compare[5] ==
0 \
            and compare[6] == 0 and compare[7] == 0:
            return 99
        elif compare[2] == enemy and compare[3] == 0 and
\
            compare[5] == player and compare[6] ==
0 \
            and compare[7] == 0 and compare[8] == 0:
            return 99
        else:
            return 0

    @staticmethod
    def sleep2(player, enemy, compare):
        """下在这个点会形成眠二"""
        if compare[2] == enemy and compare[3] == player
and \
            compare[5] == 0 and compare[6] == 0 \
            and compare[7] == 0:
            return 5
        elif compare[3] == enemy and compare[5] ==
player and \
            compare[6] == 0 and compare[7] == 0 \
            and compare[8] == 0:
            return 5
        elif compare[0] == 0 and compare[1] == 0 and \
            compare[2] == 0 and compare[3] == player
\
            and compare[5] == enemy:
            return 5
        elif compare[1] == 0 and compare[2] == 0 and \
            compare[3] == 0 and compare[5] == player
\
            and compare[6] == enemy:
            return 5
        elif compare[1] == enemy and compare[2] == 0 and
\
            compare[3] == player and compare[5] ==
0 \
            and compare[6] == 0 and compare[7] ==
enemy:
            return 5
        elif compare[2] == enemy and compare[3] == 0 and
\
            compare[5] == player and compare[6] ==
0 \
            and compare[7] == 0 and compare[8] ==
enemy:
            return 5
        elif compare[0] == enemy and compare[1] == 0 and
\
            compare[2] == player and compare[3] ==
0 \
            and compare[5] == 0 and compare[6] ==
enemy:
            return 5
        elif compare[2] == enemy and compare[3] == 0 and
\
            compare[5] == 0 and compare[6] == player
\
            and compare[7] == 0 and compare[8] ==
enemy:
            return 5
        elif compare[0] == enemy and compare[1] == 0 and
\
            compare[2] == 0 and compare[3] == player
\
            and compare[5] == 0 and compare[6] ==
enemy:
            return 5
        elif compare[1] == enemy and compare[2] == 0 and
\
            compare[3] == 0 and compare[5] == player
\
```

```
            and compare[6] == 0 and compare[7] ==
enemy:
            return 5
        elif compare[0] == 0 and compare[1] == player
and \
            compare[2] == 0 and compare[3] == 0 \
            and compare[5] == enemy:
            return 5
        elif compare[3] == 0 and compare[5] == 0 and \
            compare[6] == 0 and compare[7] == player
\
            and compare[8] == enemy:
            return 5
        elif compare[0] == 0 and compare[1] == 0 and \
            compare[2] == player and compare[3] ==
0 \
            and compare[5] == enemy:
            return 5
        elif compare[2] == 0 and compare[3] == 0 and \
            compare[5] == 0 and compare[6] == player
\
            and compare[7] == enemy:
            return 5
        elif compare[1] == enemy and compare[2] ==
player and \
            compare[3] == 0 and compare[5] == 0 \
            and compare[6] == 0:
            return 5
        elif compare[3] == enemy and compare[5] == 0 and
\
            compare[6] == player and compare[7] ==
0 \
            and compare[8] == 0:
            return 5
        elif compare[0] == enemy and compare[1] ==
player and \
            compare[2] == 0 and compare[3] == 0 \
            and compare[5] == 0:
            return 5
        elif compare[3] == enemy and compare[5] == 0 and
\
            compare[6] == 0 and compare[7] == player
\
            and compare[8] == 0:
            return 5
        else:
            return 0



#获得题目定义的参数
'''
conf = GridObservation()
gb = GoBang(conf)
chess_player_idx = gb.chess_player #当前电脑玩家 id. 1
是黑棋，2 是白棋，默认黑棋先下
board_width = gb.board_width #==15
board_height = gb.board_height #==15
current_state = gb.current_state #当前棋盘的整体情况
==[x][y][0]
all_grids = gb.all_grids #所有未被占用的格点 ==[i][j]
all_observes = gb.all_observes #目前棋盘上所有黑白子
==[i][j]
player_done = gb.joint_action_space #每个玩家的 action
space list, 可以根据 player_id 获取对应的
single_action_space==[i][j]
'''

'''
gobang 可用函数：
check_at(x, y)：当前位置是否可以落子
'''


class check():
    def __init__(self,obs):
        #super().__init__(conf)

        '''
        self.width = board_width
        self.height = board_height
        #self.state = current_state
        self.grids = all_grids
        self.observes = all_observes
        self.done = player_done #此玩家的已有格点
        '''
        self.player_idx = obs["chess_player_idx"]
        self.current = obs["state_map"] #[[[]]]
```

```
        self.board = self.reshape32(self.current)
#[[[]]]变成[[]]
        self.robot = Robot(self.board)

    def reshape32(self, dlist):
        #print("current:",self.current)
        #r = sum(dlist, [])
        import numpy as np
        temp = np.array(dlist)
        temp1 = temp.reshape(15,15)
        r = temp1.tolist()
        print("r:",r)
        return r

    def ai_play(self):
        """AI 下棋"""
        if self.player_idx == 2:
            # 人执黑==机器是白棋 2
            _x, _y, _z = self.robot.MaxValue_po(2, 1)
            #position_in_matrix = _x, _y
            return _x, _y
        else:
            _x, _y, _z = self.robot.MaxValue_po(1, 2)
            #position_in_matrix = _x, _y
            return _x, _y




#最终被主函数所调用，返回下棋的坐标（同时返回两个维度）
def my_controller(observation, action_space,
is_act_continuous=False):
    agent_action = []

    for i in range(len(action_space)):
        #print("len:",len(action_space)) #2
        #print("action_space:",action_space)
#[Discrete(15), Discrete(15)]

        #observation 会传入 dict，key 分别为："state_map"、
"chess_player_idx"、"board_width"和"board_height"
        #agent_id==0 是黑棋，agent_id==1 是白棋
        #传入的 observation 玩家 idx 是 1 和 2
        #同一种棋盘 state 下分别返回两个玩家的两种策略，通过主
函数中的顺序再决定选择哪个。因此，应该返回当前玩家的最佳下棋坐
标即可。
        ai = check(observation)
        x, y = ai.ai_play()
        print("x,y:",x,y)
        #action_ = sample_single_dim(action_space[i],
is_act_continuous, x, y)\
        action_ = change_form(x, y)
        #agent_action.append(action_)
    #print("agent ACTION:",agent_action) #[[0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0], [0, 0, 1, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]]
    return action_



def change_form(x,y):
    list1 = [[0] * 15 for i in range(2)]
    list1[0][x]=1
    list1[1][y]=1
    return list1



#返回
def sample_single_dim(action_space_list_each,
is_act_continuous, x, y):
    each = []
    if is_act_continuous:
        each = action_space_list_each.sample()
    else:
        #五子棋进这里
================================================
================================================
        if action_space_list_each.__class__.__name__
== "Discrete":
            each = [0] * action_space_list_each.n
            #print("each:",each)
            idx = action_space_list_each.sample() #随
机时所需要
            #idx
            each[idx] = 1

#================================================
```

```
==================================================
==========
        elif
action_space_list_each.__class__.__name__ ==
"MultiDiscreteParticle":
            each = []
            nvec = action_space_list_each.high -
action_space_list_each.low + 1
            sample_indexes =
action_space_list_each.sample()

            for i in range(len(nvec)):
                dim = nvec[i]
                new_action = [0] * dim
                index = sample_indexes[i]
                new_action[index] = 1
                each.extend(new_action)
    return each


def sample(action_space_list_each,
is_act_continuous):
    player = []
    if is_act_continuous:
        for j in range(len(action_space_list_each)):
            each = action_space_list_each[j].sample()
            player.append(each)
    else:
        player = []
        for j in range(len(action_space_list_each)):
            # each = [0] * action_space_list_each[j]
            # idx =
np.random.randint(action_space_list_each[j])
            #五子棋进这里
==================================================
==========================================
            if
action_space_list_each[j].__class__.__name__ ==
"Discrete":
                each = [0] * action_space_list_each[j].n
                idx =
action_space_list_each[j].sample()
                each[idx] = 1
                player.append(each)
            #
==================================================
==================================================
====
            elif
action_space_list_each[j].__class__.__name__ ==
"MultiDiscreteParticle":
                each = []
                nvec = action_space_list_each[j].high
                sample_indexes =
action_space_list_each[j].sample()

                for i in range(len(nvec)):
                    dim = nvec[i] + 1
                    new_action = [0] * dim
                    index = sample_indexes[i]
                    new_action[index] = 1
                    each.extend(new_action)
                player.append(each)
    return player
```

（共计 644 行）