# 四子棋 AI

DATA130008 2023 秋季学期 期末项目 第一阶段

梁敬聪　22110980009@m.fudan.edu.cn
张辛农　22110850001@m.fudan.edu.cn

2023 年 10 月 11 日

## 目录



## 提交说明

- 请在 **Kaggle 平台**[1] 上提交 AI 代码文件，参与评测；
  - 每位同学都需要提前注册 Kaggle 账号，用于组队参加比赛；
  - 首次访问请通过 **邀请链接**[2] 报名参赛；
  - 线上组队后，队伍名须修改为**任意一名组队同学的学号**；

[1] https://www.kaggle.com/c/sds-ai-connect-four
[2] https://www.kaggle.com/t/1f78e3ba557b47bc960dc15b4f8cf9bc

- 必须使用 **Python 3** 实现 AI，并按规定的格式编写代码后提交；
  - **使用其它语言无法参与测评，也不得分！**
  - 每次只能提交一个 `.py` 文件，大小不超过 100M；
  - 每队每天**至多提交两次**，请不要多次重复提交；
- 有关四子棋、MCTS 和 Kaggle 评测机制的更多信息，请参考 `Final Project 1 Tutorial.pptx`。如有任何其它问题，可以联系两位助教。
- **特别声明：作业抄袭者一律零分！！！**

## 提交节点（第一阶段）

- 比赛平台开放时间：**2023 年 10 月 11 日 12：00**；
- 报告提交截止时间：**2022 年 10 月 28 日 24：00**。

## 代码格式

按照下列格式编写 AI 代码，保存为 `.py` 文件（如 `submission.py`）：

```
# Import or define anything IN OR BEFORE your agent function
from random import choice

def act(observation, configuration):
    # Your MCTS AI code goes here
    board = observation.board
    columns = configuration.columns
    free = [c for c in range(columns) if board[c] == 0]
    # Remember to return an integer as the move
    return choice(free)
```

### 参数说明（部分）

- `observation`：当前棋局
  - `observation.board`
    一维数组，表示棋盘状态（格子按从左到右，从上到下的顺序记录），0 表示空，1 表示先手，2 表示后手。
  - `observation.mark`
    本方顺序，1 表示本方为先手，2 表示本方为后手。
- `configuration`：游戏配置
  - `configuration.columns`
    棋盘列数。
  - `configuration.rows`
    棋盘行数。
  - `configuration.inarow`
    获胜条件，即需要多少本方棋子相连才能获胜。
  - `configuration.actTimeout`
    每步最长用时，默认为 2 秒；AI 可以在有超时余量（见下）时动用超时余量额外思考。

  - `configuration.remainingOverageTime`
    超时余量，初始默认为 60 秒；如果某一步思考超出每步用时并用完所有超时余量，则 AI 将被立即判负。

**其它说明**

- 代码只能使用 Kaggle 官方环境提供的包；
  - 官方环境基本涵盖了机器学习常用的包，如 `numpy`、`pytorch` 等；
- 代码中出现的**最后一个函数**将作为主函数调用，因此任何其它语句（如包引用 `import` 或子函数定义）必须出现在**该函数之前或函数体内**。

# 评测方式

请首先按照下列步骤报名参赛并组队，然后前往 **比赛页面**[3]，点击 Submit Agent 提交 `.py` 文件；每队每天至多提交两次。

## 报名步骤

- 注册 Kaggle 账号；
- 通过 **邀请链接**[4] 进入报名前比赛页面；
- 点击 Join Competition 报名参赛，此时是单人组队状态；
- 线下组队完成后，由队长通过 Team 页面的 Send Invitation 功能向队员发送组队邀请，队员接受后即完成线上组队。

**注意：报名前比赛页面无法通过网址直接访问！**

## 评测机制

- Kaggle 首先进行一场验证游戏，检查提交的 `.py` 文件是否有效；
  - 如果验证失败，提交会标记为 Error，不计入每天提交次数；
- 检查通过后，Kaggle 将提交的 AI 加入竞技场，赋予初始能力分 $\mu_0 = 600$ 和起始方差 $\sigma_0^2$；
- 每天 Kaggle 会多次根据当前 AI 能力分挑选分数相近的其它 AI 捉对厮杀；
  - 每轮的胜者加分，败者扣分，打平则低分者加分，高分者扣分；
  - 分数变动幅度取决于当前 AI 的能力分方差 $\sigma^2$；
- 每次比赛后，能力分方差 $\sigma^2$ 会减小，使得 AI 的能力分 $\mu$ 逐渐趋于稳定。

## 本地调试

- 可以在本地 Python 环境安装 `kaggle_environments` 程序包，用于本地调试 AI；

[3] https://www.kaggle.com/c/sds-ai-connect-four
[4] https://www.kaggle.com/t/1f78e3ba557b47bc960dc15b4f8cf9bc

  - kaggle_environments 的使用方法可以参考 connectx-getting-started.ipynb，其中第一段代码可用于安装程序包；
- 四子棋环境（connectx）提供两个预设 AI：random 和 negamax，其中后者实现了朴素的对抗搜索。

## 常见问题

- 安装 kaggle_environments 期间遇到与 vec-noise 安装相关的报错：
  - 如有类似 AttributeError: 'dict' object has no attribute '__NUMPY_SETUP__' 的提示，请尝试在 Conda 环境（Anaconda）中安装。
  - （仅限 Windows 系统）如有类似 error: Microsoft Visual C++ 14.0 or greater is required 的提示，请按照下列步骤（或参考 **该链接**[5]）安装 Microsoft C++ 生成工具：
    * 通过 **该链接**[6] 下载安装包；
    * 打开安装包，进入 Visual Studio Installer，点击下载安装 Visual Studio 生成工具（如果以前装过，点修改）；
    * 在打开的对话框里勾选红圈所示的组件，确认安装或修改。

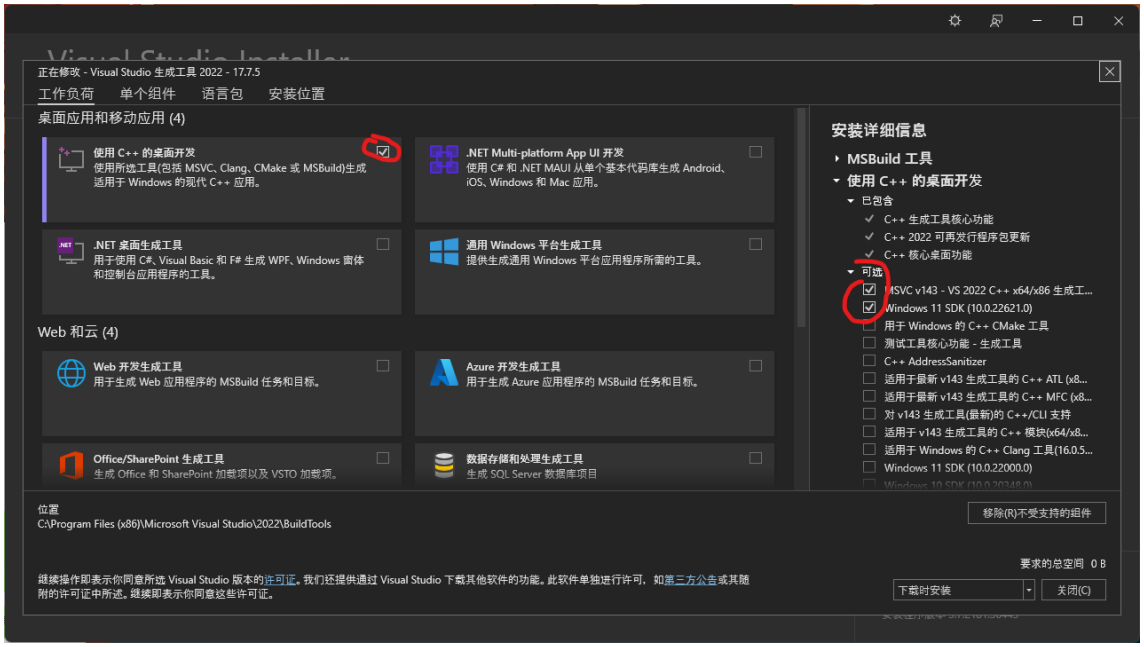


图 1: 需要安装的组件

  - 另外，也可以安装 1.12.0 或以下版本：pip install "kaggle-environments<=1.12.0"，从而跳过安装 vec-noise；此时创建四子棋环境时会提示 vec-noise 未安装，可以忽略。
- 导入 kaggle_environments 时提示部分程序包未安装：

[5] https://stackoverflow.com/questions/64261546
[6] https://visualstudio.microsoft.com/visual-cpp-build-tools/

  - 通常这些包是由其它游戏环境或 Jupyter Notebook 请求导入的，四子棋环境的各项功能不受影响，可以正常使用；
  - 可能提示的程序包有 `vec-noise`（见上）、`termcolor`、`pygame`、`ipywidgets` 等，如果不希望每次都看到未安装提示可以另外用 `pip` 安装它们。
- Jupyter Notebook/Lab 或 VSCode 中无法使用人机对抗功能（`env.play`）：
  - `env.play` 只能在 Jupyter Notebook v6 或以下版本正常工作，可以通过 `pip install "notebook<7"` 安装；
  - 除人机对抗外，也可以调用预设的 `negamax` 调试 AI。

# 评分规则

## 组队要求

- 每个小组由**两到三人**组成（单挑也可以，但不推荐）；
- 请在**报告**中列出参赛队伍名与**所有**成员的姓名、学号和 Kaggle 用户名。

## 成绩设置

- 报告（70%）：
  - 不超过 6 页，每个队伍提交一份即可；
- 基线评测（30%）：
  - 每个队伍提交的所有 AI 中分数最高的一个将在本阶段结束后与基线 AI 单独较量，胜率高于 50% 即得满分。

# 项目参考

## 相关网站

- Connect X 公开比赛官网：https://www.kaggle.com/c/connectx
- `kaggle_environments` 仓库主页：https://github.com/Kaggle/kaggle-environments

## 参考文献（五子棋 AI）


- Go-moku and threat-space search(1993), Louis Victor Allis and Hj Van Den Herik.
- Searching for Solutions in Games and Artificial Intelligence(1994), Louis Victor Allis.
- Go-Moku Solved By New Search Techniques(1996), Louis Victor Allis, H. Jaap van den Herik, and M. P. H. Huntjens.
- Self-teaching adaptive dynamic programming for Gomoku(2012), Dongbin Zhao, Zhen Zhang, and Yujie Dai.

- Evolving Gomoku Solver by Genetic Algorithm(2014), Junru Wang and Lan Huang.
- Effective Monte-Carlo tree search strategies for Gomoku AI(2016), J. H. Kang and H. J. Kim.
- ADP with MCTS algorithm for Gomoku(2016), Zhentao Tang, Dongbin Zhao, Kun Shao, and Le Lv.