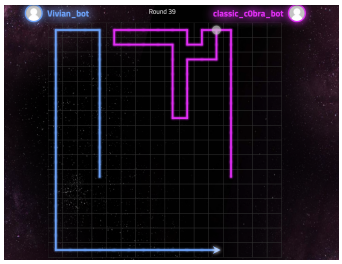
AI Agent for Light Rider

Di Bai (dibai), Yipeng He (yipenghe), Wantong Jiang (wantongj)



Left: Game logo Right: Screenshot of a ongoing game

The project we plan to work on is to develop a smart gaming AI agent for the light riders game on riddles.io platform [1]. The platform allows developers to upload their bot engine and play against bots developed by other players. Light Rider Game is a modified version of the tradition 'Snake" game. In this game two players' bots can move anywhere on the board, but need to avoid crashing into the boundary or any wall any player left behind. Every move of the bot will make the block it steps on occupied and become part of the wall. The bot that first crashes into any wall or boundary loses.

Some challenges for developing the AI agent for this game includes the following:

1. The bot will need to consider the movement of the other bot to make a decision, and how the other bot acts is uncertain.
2. How to model the movement of other bot for the purpose of training.
3. How to optimize the algorithm to generate effective strategies in limited time slot.

We find this game interesting because it has a simple rule and a simple winning criteria, but yet has many aspects to explore. How do different strategies behave during a game? Is there a certain pattern for the walls for a certain strategy, even when the opponent has different behaviors? Can self-play training like the way AlphaGo evolved be used to train such bot and get good performance [2]?

The platform provides starter code and codes of the simulator, which can be found here. It also provides a simple starting bot, which can be our baseline, which is to just move randomly to a legal adjacent position. The starting bot loses to all top 20 players on the leaderboard, so one of our goal of this project would also to develop a bot that can beat all top player bots on the leaderboard.

Few work has been done specifically for this game, but as the game is has some similarity to the classic game "Snake", we found some related works on "Snake". In 2017, Benoit, Michael and Mo completed a project An intelligent Snake in Python as final project for CS221 [3]. Sebastien Duois, Sebastien Levy and Felix also did a project AI For A Multiplayer Snake Game [4]. Their state representation and approach may be good inspirations for our project.

To solve the problem, firstly we could use adversarial game playing strategies like minimax and monte carlo tree search. Furthermore, we could use reinforcement learning methods like Q-learning and SARSA to learn a policy and run the game. Self play training could be a way to develop this bot [5]. We would like to compare the results of reinforcement learning algorithms and other game theory strategies.

Because we run the AI agent on riddles.io platform, our algorithm need to communicate with the system using fixed APIs. The game engine provides the current state: the states of each cell on the game grid, number of game round, time limit, etc. The bot, our algorithm should return an “action” to the engine, whether to move up or down, left or right, or just keep the original direction.

Below is an example of inputs and outputs of our algorithm, provided by riddles.io:
Inputs:
These are the settings of the game, which are only sent at the beginning of the game. The settings contain the player id, bot name, time limit, and size of the game field.

```
update game round 0

update game field .,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.,.

action move 10000
```

In each round, the engine provides current game round number, the state of each cell partitioned by comma, and whether the bot should move this round. This is the input of our bot at each round. A simple example is shown above.

Output: “up”

```
Output from your bot: "up"
```

The engine will take this result and update the states. “up” means our algorithm decides to move the bot up.

The game we choose is an adversarial game, where two players run their bot and compete, so the basic evaluation of each algorithm is whether the algorithm can beat its counterpart. We will run each competition multiple times and compute the possibility that each type of algorithm can beat the other, thus clearly compare which algorithm is better. We could also compete with bots written by other groups submitted to the platform to get a more general result.

Specifically, we plan to play 50 or more random games against bots of the top players on riddles.io platform and then calculate the winning rate of our bot. Another evaluation metric we will use is that we first write a simple bot (for example, take random moves every round), and check how fast our smart bots can win. However, for such problem, we believe that there doesn’t exist an easy-to-code oracle since even human may feel hard to come up with a best strategy given a state. The best bot might be a bot that knows every move of the opponents or their strategies.

References:

[1] Riddles.io. "Light Riders" .URL: https://playground.riddles.io/competitions/light-riders.

[2] Silver, D., et al. "Mastering the game of Go without human knowledge." *Nature* 550.7676 (2017): 354.

[3] Benoit Z., et al. "CS221 project :An Intelligent Snake in Python" December 2017. URL: https://web.stanford.edu/class/cs221/2018/restricted/posters/rschoenh/poster.pdf.

[4] Sebastian D., et al. “CS221 project: AI for a Multiplayer Snake game:” January 2017. URL: https://sds-dubois.github.io/2017/01/03/Multiplayer-Snake-AI.html.

[5] Silver, D., et al. "Mastering chess and Shogi by self-play with a general reinforcement learning algorithm. arXiv 2017." *arXiv preprint arXiv:1712.01815*.