Reinforcement Learning for the Halite challenge

Nicola Dainese, Stefano Mancone, Francesco Vidaich

30 March 2019

1 Objective

In this project we aim to solve a particular challenge, that is the Halite challenge, using a new kind of AI paradigm called Reinforcement Learning (RL).

2 The Halite challenge

Halite is a programming game played by two or four players, which compete for the resources of the map. To win the game one must have more resources than his opponents at the end of the game. The game ends after a certain number of turns, that can vary from match to match, but usually is about 400 or 500 turns.

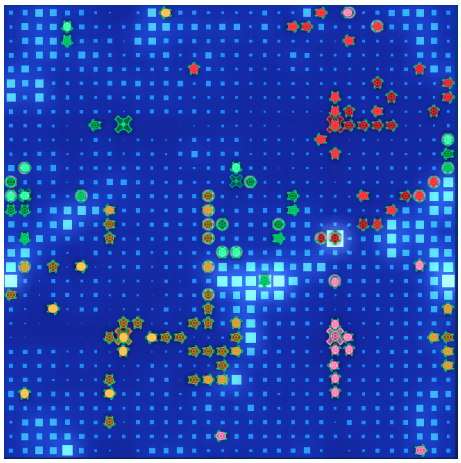

Figure 1: A screenshot from the game.

The map is a squared lattice, with toroidal border conditions. The resource that one has to maximize during the game is called halite and can be extracted from each cell of the map. The basic entities of the game are ships (moving units) and shipyards (a kind of warehouses). Ships at each turn can move in

the cardinal directions with a unitary step or stay still. Both ships and cells can contain up to 1000 halite. For a ship moving has a cost of 10% of the halite contained in the cell where it stands at the begin of the turn, whereas staying still has the positive consequence of extracting 25% of the halite contained in the cell. To effectively store the halite, a ship has to return to the shipyard and in this way it deposits its halite. The shipyard can spawn new ships at the cost of 1000 halite. Last but not least, a ship can be converted in a dropoff at a cost of some halite (4000), in order to have a strategic storage near zones rich of halite. A more complete overview of the game rules, please read: https://2018.halite.io .

The principal actions that an AI player has to choose are:

1. where and if to move a ship;
2. when to spawn new ships;
3. when and where to build new dropoffs.

Each of this choices presents a specific trade-off and the task of an AI player is to find the optimal policy of action that maximizes the final amount of resources stored in the shipyard at the end of the game (notice that they have not to be spent to be counted in the final score).

3 Program of the project

1. Setup Halite local tools in order to automate the training procedure;
2. Understand how the challenge can be translated into the RL formalism;
3. Explore some study-cases and models in the RL literature that were used with similar challenges and choose the model that fits the best our specific problem;
4. Define the agents, the action space, the observation space and the rewards;
5. Choose what to learn (e.g. the policy, action-value functions, value functions ...);
6. Develop and debug the learning algorithm;
7. Train the algorithm and make a benchmark for ulterior developments.

4 Technical things to do

1. Setup flourine (github: https://github.com/fohristiwhirl/fluorine) and eventually other tools to automate local training procedure;
2. Install and learn to use the main libraries for RL (probably TensorFlow, PyTorch and some library developed by openAI);

3. Program from scratch a couple of RL algorithms to solve simpler tasks;

4. Develop our Halite bot in python;

5. Train and evaluate performances of the bot for different hyperparameters (in order to tune them).

5 References and bibliography

1. Richard S. Sutton and Andrew G. Barto, Reinforcement Learning: An Introduction (pdf complete draft: incompleteideas.net/book/bookdraft2017nov5.pdf)

2. OpenAI Spinning Up in Deep RL (https://spinningup.openai.com/en/latest/index.html)

3. Key papers in Deep RL (https://spinningup.openai.com/en/latest/spinningup/keypapers.html)