Reinforcement Learning : Connect 4 - Thomas VICAIRE

Tanguy Blerbvaqc, Lucie Clémot, Nicolas Péruchot, Thomas Vicaire

1 Introduction

We aim to develop an agent capable of playing Power 4 using reinforcement learning methods. Reinforcement learning is effective for this kind of game, with many possible configurations. Intuition and theory allow us to anticipate that agents trained on many games will be able to perform well.

In addition, we will develop the game environment around this agent in order to evaluate its performance, and to play with it.

To do so, we will use the module pettingzoo.classic.connect_four_v3 to create the CONNECT4 environment. We will compare the performances of two reinforcement learning model based : Q learning which learns by updating Q-tables, and Deep Q learning which uses deep learning to learn the decision process.

2 Win checks

We implemented some wins checks in order to analyze the behaviour of our agents. Those check if there are some winning configuration in the next move. It is done in a greedy way in comparison to reinforcement learning. Each win_check checks a specific configuration, and in this report, I will analyze a few of them. The checks functions return a Boolean whether there is a winning situation or not.

— In this configuration, we will use the check : *win_ diagonal*. It checks for each column if a play would lead to a victory regarding its 'diagonal neighbors'. In this situation, it returns false as there is an almost

completed diagonal, but there would be an empty cell under the last chip so it is not a possible move.

```
[[0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0]
 [0 0 1 0 0 0 0]
 [0 1 1 2 0 0 0]
 [1 2 2 2 0 0 0]]
```

Figure 1 – Diagonal win check

— In this second configuration, we will use the check *is_valid_play*. It checks if a move is valid or not. Here it returns false as there is a flying chip.

```
[[0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0]
 [0 0 0 1 0 0 0]
 [0 0 1 0 0 0 0]
 [0 1 1 2 0 0 0]
 [1 2 2 2 0 0 0]]
```

Figure 2 – Invalid configuration

3 Training

For the last training, I implemented the training and the saving of the models and its statistics with Tanguy. We led training in parallels in order to ensure results. Training were made on 100 000 epochs each, and I trained locally the Q-learning model : the training was faster locally than Deep Q learning on mydocker's GPU.

4 Statistics on the last training

I wrote the notebook that shows and analyzes the statistics on the last training. All the statistics and analysis are on the notebook

5 Conclusion

The project was concrete application of the reinforcement learning class with a famous child game. It was a way to focus on all the facets of a reinforcement project without having too many time-consuming tasks (I assume that all the checks/unit tests for chess are numerous and complex). Even if I had not implemented the models themselves, I succeeded in better understand reinforcement learning and the way it is used. I think that I would be able to use those type of algorithm in the future.