ConnectX

David Barbas Rebollo

Artificial Neural Networks
February, 2021

1 Introduction

In this project, the topic of reinforcement learning is explored by participating in the competition of ConnectX available by Kaggle. The competition consists of giving a solution to the famous game connect four.

This game has two players who alternate turns dropping colored discs into a into a seven-column, six-row vertically suspended grid. The pieces fall straight down, occupying the lowest available space within the column. Each player uses a different color, and the objective is to be the first player to get four discs in a row.

Figure 1: Connect four game example.

Kaggle, a subsidiary of Google, is an online community of data scientists and machine learning practitioners. Kaggle allows users to find and publish data sets, explore and build models in a web-based data-science environment, work with other data scientists and machine learning engineers, and enter competitions to solve data science challenges.

For the competition ConnectX, OpenIA's gym is used as an environment to create agents which can play the game. Which then can be uploaded to Kaggle. The objective of this work is to explore reinforcement learning techniques to give the best solution possible in this competition.

2 Agents defined

2.1 One-Step Lookahead

The first agent to be defined is done using simplifying the value function algorithm and using a selected heuristic. This is done because connect four has over 4 trillion game boards so it would not be feasible to compute until the final state of each board. Therefore, only the next move is taken in account.

In this way, different moves a rewarded differently. With this estimation the moves with higher rewards are more likely to result in a win for the agent. The different possible moves considered and its rewards are:

- Placing 4 discs in a row (win): 1,000,000
- Placing 3 discs in a row: 10
- Placing 2 discs in a row: 1
- Opponent places 2 discs in a row: -1
- Opponent places 3 discs in a row: -1,000

2.2 N-Step Lookahead

Deepening into the idea from before, is this scenario instead of only taking into consideration the next move, the 3 next moves are explored. As there are some cases in which only taking into consideration the next move is not sufficient to win or at least draw.

For this agent the minimax algorithm was implemented to select the best move given the information of the game tree. Also, alpha-beta pruning was added to the algorithm to make decisions more efficiently.

Additionally, the heuristic was extended to take into account the situations were the agent loses. The new reward for this situation is:

- Opponent places 4 discs in a row (loss): -10,000

2.3 Deep Reinforcement Learning

In this section, a different approach is taken. Instead of having established policies with different rewards and searching for the best rewarded set of actions, the policy of the agent is searched.

Therefore, deep reinforcement learning is explored using a neural network and different policy gradients. In essence, policy gradient methods update the probability distribution of actions so that actions with higher expected reward have a higher probability value given an observed state.

Given the small board size of the game, a small CNN is used to model the network.

2.3.1 Advantage Actor-Critic

This approach uses two models. Firstly, there is an actor model which performs the task of learning which actions to make under the current state of the environment. Once, the action is done it is evaluated by the critic model. The critic, rates the action and gives passes the loss calculated to the actor so it updates its policy.

Due to having two models that must be trained, there are two set of weights to be optimized separately.

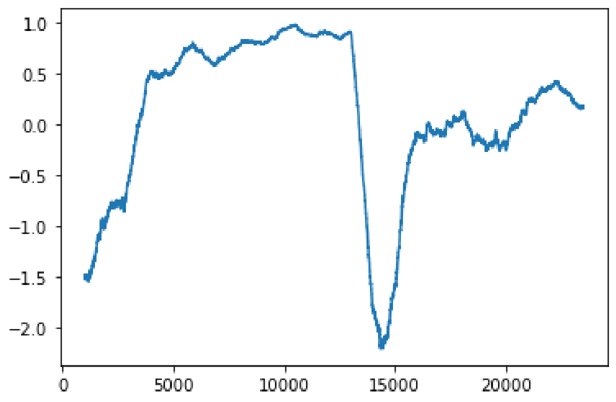

Figure 2: Cumulative reward during training with A2C algorithm.

2.3.2 Proximal Policy Optimization

The PPO algorithm was introduced by OpenAI team in 2017 and it is based on advantage actor critic. PPO collects a small batch of experiences interacting with the environment and then uses that batch to update its decision-making policy. Once the policy is updated the old experiences are discarded and new ones are collected to update the policy. Meaning, this is an pseudo online policy learning approach.

The idea behind this approach is updating progressively the policy so there is less variance in training. Although, it ensures some bias, it also provides a smoother training.

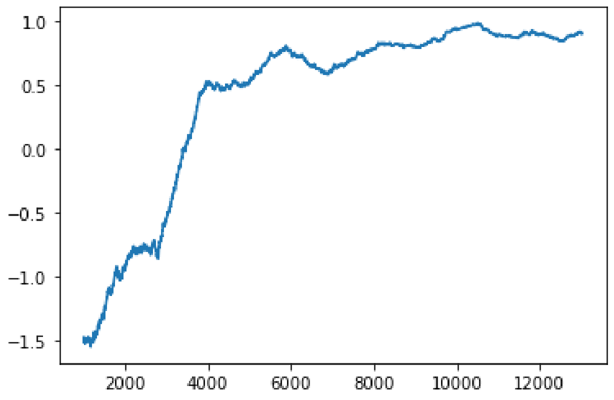

Figure 3: Cumulative reward during training with PPO algorithm.

3 Competition results

Once an agent is submitted to Kaggle's competition, it plays several games against other agents. This way it is ranked in the leaderboard according to its results against other submitted agents.

The obtained results are shown below:

Table 1: Results of experimentation for ConnectX task.

Agent	Score	Placement
One-step	265.8	365/452
3-step	661.5	252/452
A2C	954.1	86/452
PPO	1063.8	45/452

The results obtained are the expected ones. As each iteration of the solution proposed yields better results than the previous one.

The most noticeable difference in scores a from one-step to 3-step and from 3-step to deep reinforcement learning.

In the end, PPO obtains the best result. Which also enabled the agent to be in the top 50 or top 12% of the competitors.

It is important to note that submitted agents are continuously playing games against each other. This results are recorded previous to the delivery of the assignment and may have fluctuated when revised.

4 Conclusion

After doing this project it was been really interesting delving into deep reinforcement learning algorithms and expanding the knowledge seen in class.

Given the environment of OpenAI the implementation of the agents was fairly easy to use, giving the opportunity to focus on the parameters of the model.

As shown by the results obtained, deep reinforcement learning is becoming very dominant at attaining good performing behaviors in dynamic environments.