Deep Reinforcement Learning on Quoridor

Anonymous ??? submission

Paper ID 1

Abstract

For a start with deep reinforcement learning concept, we will be trying to use the algorithm Actor Critic Agent for achieving beating an opponent which tries only to follow the shortest path in the board game called "Quoridor".

1. Introduction

The idea of the project started because of my curiosity on the deep reinforcement part of the Artificial Intelligence, and how I should apply it to a board game. "Quoridor" is a board game, that contains 3 pieces: a board of 81 square spaces (9x9), pawns, and walls, and the objective is to get your pawn to the opposite side of the board. This game could be played in 2 or 4 players, but in our solution to the problem, we will only have 2 players. The idea with this board game was given by my license teacher, Adrian Viorel, and the Advantage Actor Critic was proposed by my Deep Learning and Computer Vision teacher, Diana-Laura Borza. There are a lot of contributors to my solution, like Deep Mind, one of the most important contributors in deep reinforcement learning, or like Kyutae Lee, which implemented Monte Carlo search tree and the one who inspired me the most to continue with this idea (a link with his work: https://github.com/gorisanson/quoridor-ai). Another contribution comes from Stable Baselines, which implemented Advantage Actor Critic, and also helped me along the way with some examples and also with a paper about their work. The objective of this project is to apply an Actor Critic Agent algorithm and to have a better understanding of reinforcement learning.

2. Literature Review

For my first solution, Actor Critic method was not one of the options, but after finding about stable baselines and their implementation of Advantage Actor Critic (A2C), I wanted to give it a try. First solution I've seen for this project was Kyutae Lee's repository from git hub [1]. Its methods were all implemented in Java Script, with a lot of code, but good explanations and metrics written. His 0.2 version had some drawbacks because of the hyperparamters, but in the next one, he would've obtained better . After A2C's proposal, my orientation moved to Stable Baselines's documentation and to " sentdex" who explained how to use and create your own environment for using A2C (https://www.youtube.com/@sentdex). In his tutorial, he has created an environment for Snake, and showed more about the metrics for reward and loss. He also explained cases depending on the environment: with a discrete distribution, where the moves are more clearly (left or right moves, put a wall in the first box etc.), or continuous distribution (a move between a range, rotate 0.2 degrees clockwise, move 2.3 feet to the right etc.). In the case of Quoridor, it has to be used the discrete distribution, because the actions are specific, it is not needed a range to make a move. For having a better understanding of the A2C algorithm, I also studied a paper, by Volodymyr Mnih, Adrià Puigdomènech Badia *et al*., called "Asynchronous Methods for Deep Reinforcement Learning", which contains the formulas and the logic behind the algorithm. The paper also talks about the problem and the drawback with other deep reinforcement learning algorithms: 'they use experience replay, which consumes a lot of memory, but they provided a very different paradigm: instead of experience replay, they asynchronously execute multiple agents in parallel, on multiple instances of the environment. [...] This simple idea enables a much larger spectrum of fundamental on-policy RL algorithms, such as Sarsa, n-step methods, and actor-critic methods, as well as off-policy RL algorithms such as Q-learning, to be applied robustly and effectively using deep neural networks.' [3]

3. Proposed solution

In this proposed solution, the Advantage Actor Critic will learn how to beat an opponent that does not put walls, and only moves through the shortest path possible. Here we will look through 2 main parts: the opponent pawn and the pawn that learns how to play.

3.1. The opponent

The opponent uses Lee algorithm to set the shortest path to the opposite side and to move through the board. If the ”AI” (the pawn that learns) put a wall on the way he walks, then he will reconfigure his route in the same way. The Lee algorithm is one possible solution for maze routing problems. It always gives an optimal solution, if one exists, but is slow and requires large memory for dense layout. The algorithm is a breadth-first based algorithm that uses queues to store the steps.

3.2. Advantage Actor Critic

Before we talk about the ”AI” pawn and how it learns to play the game, we need to understand what is Advantage Actor Critic algorithm, and how it works. ’The Advantage Actor Critic (A2C) algorithm combines two types of Reinforcement Learning algorithms (Policy Based and Value Based) together. Policy Based agents directly learn a policy (a probability distribution of actions) mapping input states to output actions. Value Based algorithms learn to select actions based on the predicted value of the input state or action.’ [4] For an example to help understand the Actor-Critic, ’imagine you play a video game. You can play with a friend that will provide you some feedback. You’re the Actor, and your friend is the Critic. You don’t know how to play at the beginning, so you try some actions randomly. The Critic observes your action and provides feedback. Learning from this feedback, you’ll update your policy and be better at playing that game. On the other hand, your friend (Critic) will also update their way to provide feedback so it can be better next time.’ [1]

So in simpler words, the actor has a neural network that works as a policy gradient and gives actions, or better said, a discrete distribution with each probability of each action, and it updates after the advantage, which is created by the ”Q Value” and the Value of the critic, who it’s also a neural network, with the same input as the actor, but also have the actor’s action, but in the end, there will be only a value. In the same time, the actor learns what actions should use in a specific state, and the critic learns how to evaluate an action. Another similar arhitecure would be ’This type of architecture is in Generative Adversarial Network(GAN) where both discriminator and generator participate in a game. The generator generates the fake images and discriminator evaluate how good is the fake image generated with its representation of the real image. Over time Generator can create fake images which cannot be distinguishable for the discriminator. Similarly, Actor and Critic are participating in the game, but both of them are improving over time, unlike GAN.’ [2]

3.3. "AI" (The pawn that learns)

So how does the pawn learns? After understanding the concept of the actor and the critic, now we only have to define the input layer, which this will be our ”observation space” in the code, and the reward. The observation space will be created by the coordinates of AI pawn and the opponent’s pawn, how many walls the AI have left and the status of every coordinate in which there can be a wall (0 will be for no wall, 1 for a vertical wall and 2 for an horizontal wall). As for the reward part, we need to think at least as a player of ”Quoridor”. From a personal opinion and experience, it is more important to move than to put walls, at least at the start of the game, so we will favor that the actor should choose move actions. The output distribution starts equally for each move, there 64 actions for walls, and 4 for pawn moves, so of course it will favor to put a wall, instead moving through the board. The reward grows if he make a valid move, but it grows even more if it prefers to move, instead of putting a wall. Also it punishes a lot if the actor puts a wall and it doesn’t have any walls left. A huge reward for an actor move is by putting a wall that will make the opponent’s shortest path longer than the last time and by winning the game.

4. Experimental results

As this project uses only deep reinforcement learning, it doesn’t require a dataset. The section will be separated in the metrics result and a comparison with other projects which approached applying reinforcement learning on the ”Quoridor” game.

4.1. Metrics

The quality of the learning of the ”AI” pawn can be considered by the reward of an episode. For the first system of rewarding described at 3.3, the graphs look like this:

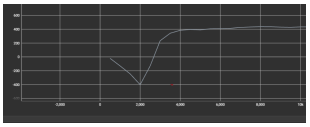

Figure 1. Image with reward graph, and vanilla rewards

As a first step, the algorithm works good and the reward grows as expected, but unfortunately, the actor learned that putting many walls are a bad action, and if it moves it is a good action. The actor doesn’t put any wall at all, and just moves. So for a second trial, we should change the rewards: no more increasing the reward if it did a correct move, it will only increase if he wins or if a wall lengthens the opponent’s shortest path.

The graph for the new reward system would look like:

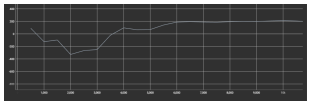

Figure 2. Image with reward graph, with a more punishable system reward

Even if the reward is smaller with 100, the results really paid off: the actor learns better that it is a good move to put walls that changes the path of the opponent, but again, there still a problem with the mechanism of moving to the goal. So let's reward the actor if the move will make him come closer to the goal. It will be implemented in the same way it was implemented for the opponent, but it will only be used for the move the "AI" pawn did.

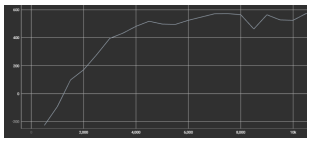

Figure 3. Image with reward graph, with a more complex system reward

The reward grows even higher than before, and this time, the "AI" pawn can actually win. The actor learned that if it only follows the shortest path, because the "AI" pawn is the first one who moves and it will arrive faster to the destination. Sometimes the actor tries to put a wall, but without success (maybe there should be a higher reward for putting a wall that lengthen the opponent's path).

4.2. Other solutions

The other solutions are better than this one, because their opponent took more cases, and not only the shortest path. I think that is one of the biggest disadvantages in my solution. Another thing different is that the majority of the projects used Monte Carlo Search Tree, which is a reinforcement learning algorithm. The game does not have a big complexity, so we do not need necessarily an algorithm like A2C, which is part of deep reinforcement learning, plus that it combines two reinforcement algorithms. The other game versions did not allow overlap between the pawns, and used the original rules, which were to jump over the opponent or to move diagonally.

5. Conclusion

In a personal matter, the project was a very good first step in extending my area for reinforcement learning. The Advantage Actor Critic algorithm though was way too complex for a beginner in this field and very hard to understand.

As a conclusion for the project part, the actor did learn how to play against hat specific opponent, but it is still not good enough to play vs a human, or other algorithms.

5.1. How it can improve

At first, a lot of changes to the hyperparameters of the A2C from stable baselines, and changing a lot in the opponent's moves: starting to put walls at random times, and still following the shortest path, also maybe following other types of algorithms, not only the shortest path (to diverse the actor's actions). Also, the project has a lot of spaghetti code, and it should be changed, maybe even from scratch.

References

[1] Thomas Simonini "huggingface". Advantage actor critic (a2c) unit 7, of the deep reinforcement learning class with hugging face, 2022. 2

[2] Dhanoop Karunakaran. The actor-critic reinforcement learning algorithm, 2022. 2

[3] Volodymyr Mnih, Adrià Puigdomènech, Mehdi Mirza1, Alex Graves, Tim Harley, Timothy P., David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning, 2016. https://arxiv.org/pdf/1602.01783v2.pdf. 1

[4] Mike Wang. Advantage actor critic tutorial: mina2c, 2021. https://towardsdatascience.com/advantage-actor-critic-tutorial-mina2c-7a3249962fc8. 2