Assignment - Reinforcement Learning

Antoine Dargier

CentraleSupélec University, Gif-sur-Yvette, France
antoine.dargier@student-cs.fr
Notebook link (contact me if errors): https://t.ly/w8Ue

Abstract. In this assignment, I created two agents able to play the Text Flappy Bird game: a Q-Learning and an Expected Sarsa agent. I have seen that Q-Learning performs better, able to reach an average score of 47 after experiences, whereas Expected Sarsa reaches 32. Then, I fine-tuned the parameters to have the best results: a step size of 0.7 and a discount of 0.8.

Keywords: Reinforcement Learning · Q-Learning · Expected Sarsa

1 Introduction

The primary goal of this assignment is to explore and implement reinforcement learning strategies in a game known as Text Flappy Bird (TFB). TFB is a simpler adaptation of the classic Flappy Bird game, where the player's character is represented by a basic unit element. I aim to teach the game's character to navigate through obstacles and achieve high scores.
There are two versions of the environment: one that returns the position of the bird with regard to the next pipe gap; the other one gives the distance between the bird and the pipe gap. The state space is all possible positions of the bird on the screen. The action space is binary: either the bird flies upwards or hovers and descends. When the bird succeeds at a step, he got a reward of 1, and he got -1 when he dies.

2 Agents

I chose to work on Q-learning and Expected Sarsa agents because they are popular RL algorithms that can be used to solve a variety of problems.
Q-learning is a model-free, off-policy algorithm that learns the optimal action-value function by repeatedly estimating the maximum expected reward for each action at each state.
On the other hand, Expected Sarsa is also a model-free, on-policy algorithm that learns the Q-function by estimating the expected reward of taking a particular action in a given state, following the current policy. In other words, Expected Sarsa learns a Q-function that maps state-action pairs to expected rewards, based on the current policy.

Finally, the difference is that Expected Sarsa takes into account the policy being followed by the agent when estimating the expected reward, whereas Q-learning updates its Q-values based on the maximum expected reward of the next state-action pair.

3 Results

Computing 20,000 episodes, I obtain the following results for the two agents:

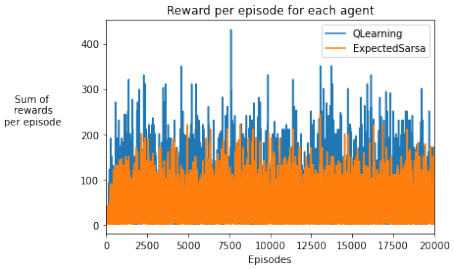

Fig. 1. Reward per episode for each agent

We can see that in this problem, Q-Learning seems to perform better than Expected Sarsa, with higher rewards per episode. It seems that both algorithms have converged very quickly. The average reward is 47 for Q-Learning agent, whereas it is 32 for Expected Sarsa agent.

3.1 State Value functions

Now I will analyse the state value functions for the two agents:

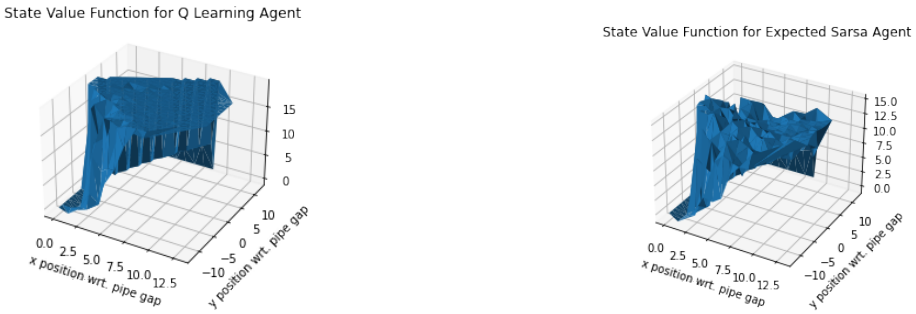

Fig. 2. State Value functions plot

We can see that the state value functions have some differences. High values are given for states that are in front of the pipe gaps: very logical! For the Q-learning, we have a very logical plot: states in the middle and close to the gap have big values, and it decreases when we go far away from the gap. For Expected Sarsa, it is quite surprising that the value changes a lot between two close states.

3.2 Parameter-Sweep Graphs

I then studied the impact of two parameters: the step size and the discount:

Fig. 3. Parameter-Sweep Graphs

We can see that the performances of Expected Sarsa decrease when the step size is increasing. However, for Q-Learning, there is some plateau of performances where this parameter doesn't have a big impact.
For the choice of the discount, we can see that this parameter doesn't have a big impact on performances for both algorithms.

4 Discussions

With the environment TextFlappyBird-screenv0, the state is only the distance to the next pipe gap. There is a big limitation of this state space because, for the same distance, it is not the same result at all when the bird is in front of the gap or not. So, it could be very difficult for the model to find a precise value function.
In the real game, the same agents can be used if we arrive to have good observations of the environment. For that, it will be needed to develop an algorithm to detect the place of the bird and of the next pipe gap.

5 Conclusion

In this project, we have seen that we are able two develop reinforcement learning agents that can play TFP. In our case, the Q-Learning agent seems to perform a little better than Expected Sarsa. In fact, its state value function seems more logical and it performs better results for different values of the step size and of the discount parameter. With a little work to develop a detector on the real game, our pipeline seems good to be used on the real game.