Flappy Bird - Solo Assignment Report

Ugo Muhieddine

CentraleSupélec

Abstract. This paper compares the behaviour of two agents to solve the Flappy Bird game as well as their parameter tuning: Q-Learning, SARSAMAX. Here is the link for the github

Keywords: Reinforcement Learning · Flappy Bird · SARSA · Q-Learning

1 Environment description

The flappy bird game idea is to make a little bird pass between pipes as many time as possible. To do so, two actions are possible: jump (1), stay still (0) which with gravity make the bird fall. The environment allows 22*14 states that are based on the vertical and horizontal distance of the agent to the center of the next hole. The reward is equal to 1 for each survived step, 0 in case of death (touching the pipes or the bottom line). Flappy Bird is a popular mobile game that requires the player to control a bird's flight through a series of pipes without hitting them. This environement is the TextFlappyBird-screen-v0, using text-based input. As asked for the assignment, I focused on the one giving the x and y distance to the pipe hole. The environment I will be using has two possible actions: jump (1) or do nothing (0), and the state is represented by the bird's x and y distance from the pipes' hole for a total of 525 states:

- x : from 0 to 14 (int)
- y : from -12 to 22 (int)

There is however a second version of the environment using image-based input, where the agent receives a visual representation of the game as pixels. The main limitation of using the same agent for both versions is that the agent needs to be able to handle different types of input. Also, the agent that works well with the text-based input may not work well with the image-based input and vice versa.

2 Agent descrpition

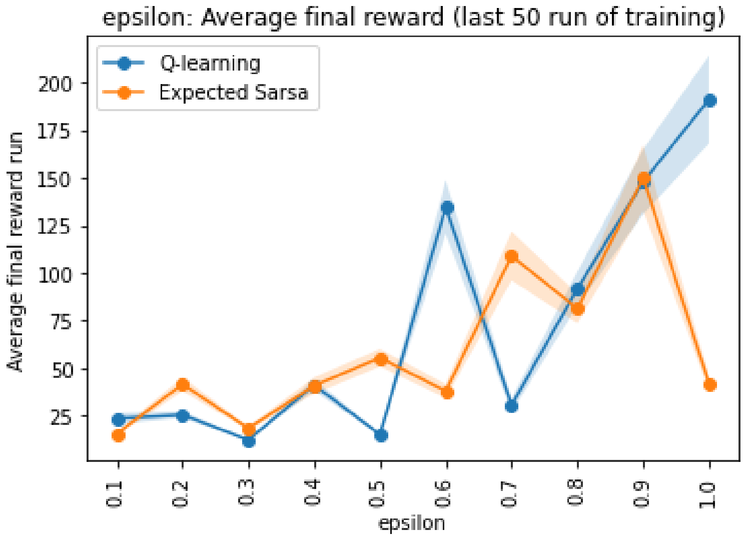
We chose Q-Learning and Expected SARSA agents as they are popular algorithms in Reinforcement Learning and have shown good performance in similar tasks. The two agents differ in their sensitivity to parameters, convergence time, and how they handle rewards and scores.

Q-Learning is known to converge to an optimal policy given enough iterations and has a high sensitivity to its learning rate and discount factor. In contrast, Expected SARSA converges faster and is less sensitive to its parameters. It also handles rewards and scores differently from Q-Learning as it calculates the expected value of all actions instead of selecting the action with the highest Q-value.

2.1 Parameters tuning

To optimize the two agents, one have the hand on few parameters:

- **Exploration**: The epsilon parameter controls the exploration vs. exploitation trade-off. If the epsilon parameter is too low, the agent may not explore enough and get stuck in sub-optimal policies. If the epsilon parameter is too high, the agent may not exploit the knowledge it has already acquired and waste time exploring. The **epsilon decay** parameters is also important to consider in order to refine the exploration process.



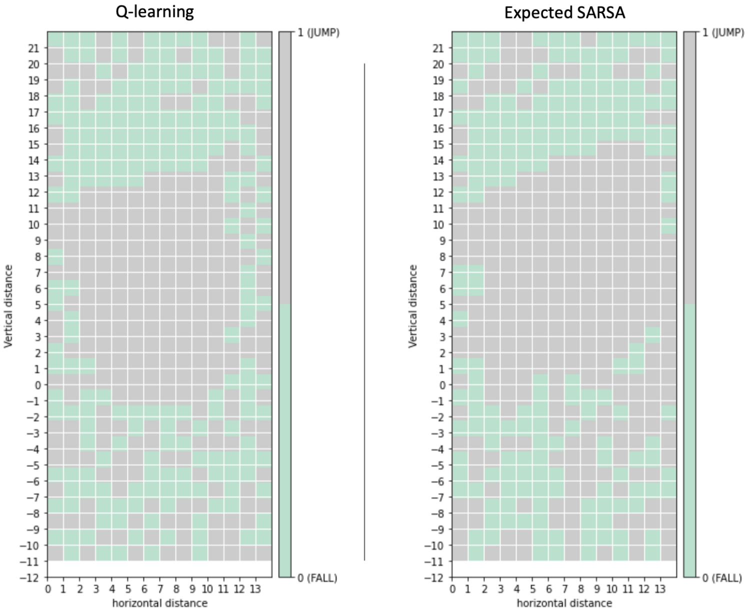
- **Learning rate**: The step_size (or alpha) parameter controls how much the agent updates its Q-values after each step. If the step_size parameter is too low, the agent may take too long to converge to the optimal policy. If the step_size parameter is too high, the agent may overshoot and fail to converge. The **step size decay** may help find the good balance for the trade-off
- **Discount factor**: The discount parameter controls the relative weight of immediate rewards vs. future rewards. If the discount factor is too low, the agent may only focus on immediate rewards and overlook the long-term consequences of its actions. If the discount factor is too high, the agent may overestimate the value of future rewards and

get stuck in sub-optimal policies.

3 Result comparison

Even with the parameter optimisation, the reward over the agents' training is not increasing as expected, could it come from an implementation error? However, one can clearly see that the agents are learning how to play: the following figure highlights the policies of the trained agent. For both agent, the birds is jumping when he almost aligns with the hole. The purpose here is to maintain the vertical position. One can also underline the fact thatfor the Q-Learning, the central grey area is larger on its bottom. It emphasizes that the bird will jumping earlier to meet its favourite position. But these policies show also the fact that the agents didn't fully train: the situation where the vertical distance is very low should push the bird to jump. However for both agent, we see that this area is noisy.

In sum, the policy of the Q-Learning looks better but the training appears unfinished.



State-Value comparison of the agents