RL Assignment: Text Flappy Bird

Léo Buecher[1]

[1]CentraleSupélec, France

Github repository : https://github.com/leobcairo/RL_Assignment_code

In this report, we study Reinforcement Learning methods solving the Text Flappy Bird game. Unlike in the normal game where we have access to the position on the screen, in this environment we only have access to the relative position to the next pipe gap which makes likelier the risk of crushing onto the floor.

1 The two designed agents

We have implemented 2 agents for this Reinforcement Learning task : Mont-Carlo Control and Expected Sarsa. Both come from Sutton and Barto's book [1].

They are both model free (which was required, having no given knowledge of the environment). But they remain different, since one relyes on Monte Carlo method, and the other on the Temporal Difference method, so they are interesting to compare.

As we said, the first agent is inspired from the implementation described in [1], p. 97. The difference is however that our agent does not follow a greedy policy, but an epsilon greedy policy, and that the $Q(s, a)$ value is not updated based on the first time the couple (s, a) appears in the episode, but on every time this couple appears.

The second agent is described in [1], p. 133. It is based on temporal difference. The update is done after each action, based on the previous reward and on an estimation of the next return. The update's target is $R_{t+1} + \gamma \mathbb{E}_{A \sim \pi} [Q(S_{t+1}, A)]$.

2 Convergence

Figure 1 shows how the two agents perform episode by episode. For this, we used the optimal hyperparameters (whose selection is described in 4). The first thing we notice is that the variance of the episode rewards remains very high, even in the long run. The agents do not succeed in finding a safe policy. Our guess is that the state is not enough to know the position on the screen and thus to establish a perfect policy.

We can see on this figure that the MC agent performs slightly better than the ES agent. Over 3000 episodes the former achieves a reward (averaged on the last 100 episodes) of 233 while the latter only achieves a reward of 130. Apart from that, the convergence of the two agents is very similar : their rewards stabilise around 1500 episodes.

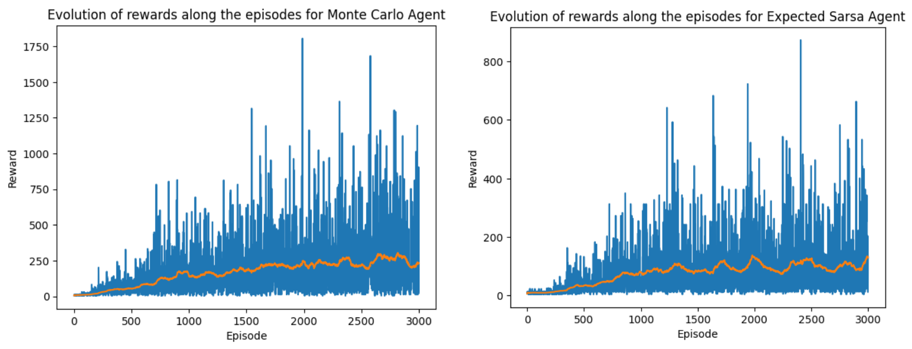

Fig. 1. Evolution of rewards along the episodes for the two agents

3 State value function

Figure 2 shows the state value functions for the two implemented agents. We spatially represented it so that it corresponds to how we see the state on the screen. Both state value functions are similar : the states with high values represent a quite narrow channel which corresponds to the exit from the previous pipe gap, which opens before progressively closing with a conic shape to the next pipe gap. The dark left corners are due to the fact these states are never visited, as it is impossible to arrive there from the previous pipe gap. The dark right corners and the dark space at the bottom are due to the high probability of crashing into the pipe or the floor.

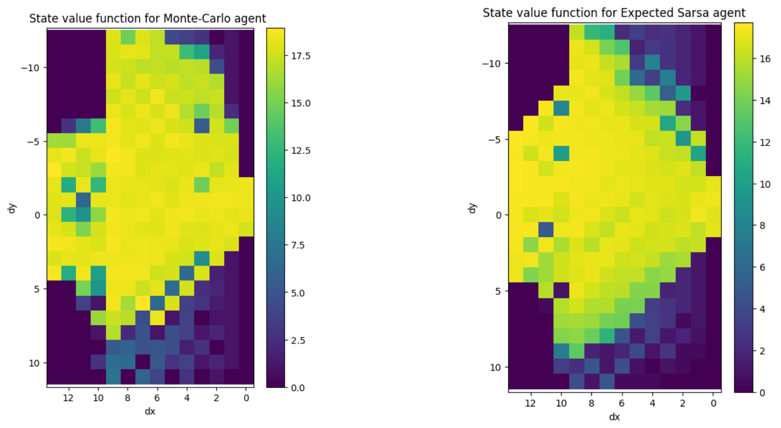

Fig. 2. State-value functions for the two agents

As for the differences, we can see on this figure that the Monte Carlo agent performs slightly better, and we can notice the higher variance. Besides, we can see that it penalises less the states in the upper right centre, which makes us think that the MC agent better deals with the going down movement toward the gap.

4 Hyperparameter selection

The policy of the two agents is fixed as ϵ-greedy. So that they are GLIE, we made ϵ vary as follows $\epsilon = \epsilon_{init}/k_{ep}^{p}$, where k_{ep} is the number of the episode. We studied the influence of ϵ_{init} and p.

More precisely, we looked at the rewards after 1000 episodes, averaged on the 200 last ones. For each parameter choice, we made 10 runs to have a statistical distribution, whose mean is represented by the hard blue line and whose standard deviation is represented by the light blue shape around it. We see on Figure 3 that ϵ_{init} has little impact. However, we concluded the best values were around 0.04 for the MC agent and 0.1 for the ES agent.On the contrary, Figure 4 shows that the power law p of the ϵ update has a clear impact on the final reward, and that the best value is 0 for both agents. This surprised us, since it corresponds to a case that does not satisfy the GLIE assumption.

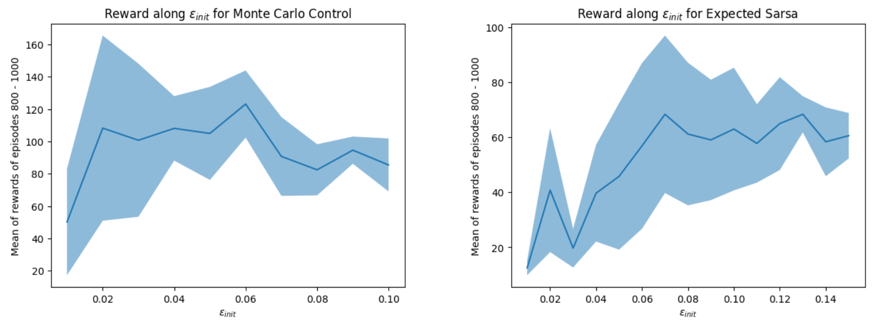

Fig. 3. Influence of ϵ_{init}

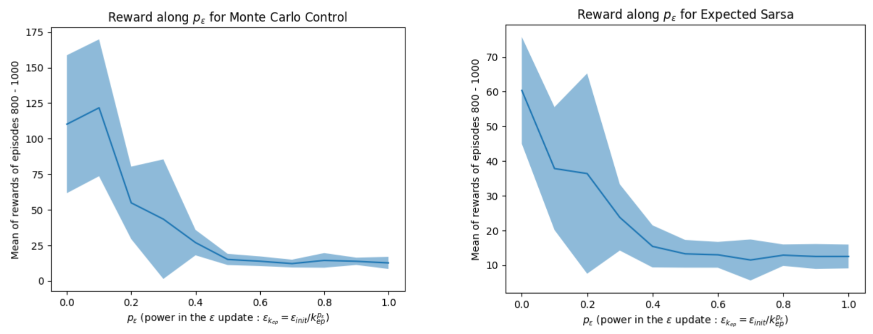

Fig. 4. Influence of p

References

1. Sutton, R.S., Barto, A.G.: Reinforcement learning: An introduction. IEEE Transactions on Neural Networks **16**, 285–286 (2005)