Agent Comparison on Text Based Flappy Bird

Corentin Davion[1]

CentraleSupélec, Gif-sur-Yvette, 91190,France

Abstract. This report summarize the findings of two agents, one Monte Carlo, one based on the λ-SARSA algorithm described by Sutton et al [1] on the Text based Flappy Bird custom Gym environment. We compare both agents, discuss how it behaviors when the parameters of the environment changes and cover a succinct parameter sweeps through exhaustive search.

Keywords: Monte Carlo · λ SARSA · Reinforcement learning

1 Introduction

1.1 Environment description

We use the *Text Flappy bird v0* gym environment, which represents the well know Flappy Bird game in the terminal with ASCII symbols. We can change the environment width, heights and pipe gap, all of which will have influence on the performance agents. The observations states is the possible position of the bird which depends on the size of the environment with respect to the center of the next pipe gap. The action space is simply flap (go up by one unit) or do nothing. The reward is the amount of timestep the bird survives in game, with 'death' being falling to the ground or colliding with a pipe.

1.2 Agent description

In this section we briefly describe the two different agents used to compare and train within the environment.

The Monte Carlo agent Monte Carlo (MC) reinforcement learning is a model-free method that learns the optimal policies by averaging rewards over entire episodes rather than making updates at each time step. Unlike Temporal Difference (TD) methods, which update based on bootstrapped estimates, this agent waits until an episode ends to compute the total discounted return G and update state-action values with

$$Q(s,a) \leftarrow Q(s,a) + \alpha(G - Q(s,a))$$

when playing the episode backward. There are two main variants: First-Visit MC, which updates only the first occurrence of a state-action pair in an episode, and Every-Visit MC, which updates all occurrences. Since this method converges asymptotically, this makes MC suitable for environments with delayed rewards but requires full episode trajectories

λ SARSA The λ-SARSA (Lambda SARSA) is a reinforcement learning algorithm that combines SARSA (on-policy TD learning) with eligibility traces to balance between an Monte Carlo (MC) update and a Temporal Difference (TD) learning. The main idea keep track of each encountered state-action pair and decayed them every step with rate λ to allow update information to propagate through different timesteps. When $\lambda = 0$, the algorithm behaves like a standard SARSA algorithm, updating only the most recent state-action pair with

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left(r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)\right)$$

When $\lambda = 1$, it approximates Monte Carlo methods, considering the full episode before updating.

2 Training the agent

We trained both agent on a height=15, width=20 and pipe gape=4 environment over 20000 episodes with ϵ greedy policies. The Q value function is randomely initialize between $[-1, 1]$ Below are the plots of the rewards over the episodes, where, for the sake of visualization, we have plotted every 100 points and smoothed it out with a exponential moving average of weight 0.05.
The λ Sarsa agent was trained with $\alpha = 0.2, \lambda = 0.1, \epsilon = 0.1$ while the MC agent was trained with $\alpha = 0.1$ and the same ϵ.

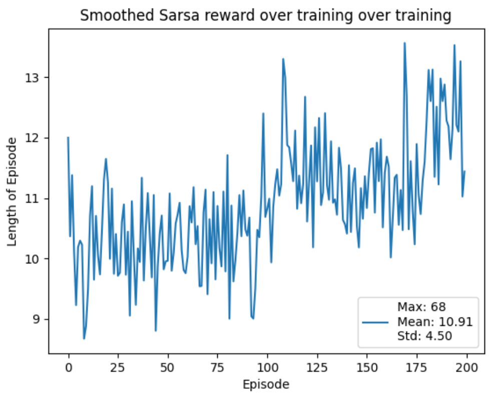

Fig. 1: λ SARSA agent

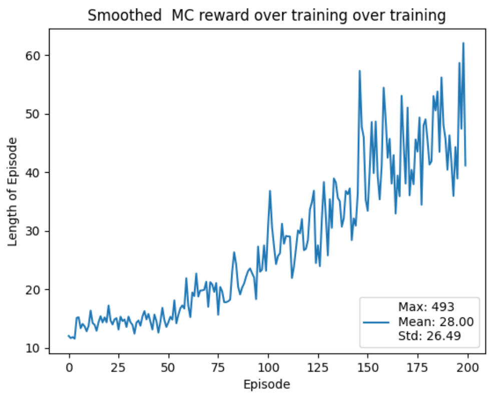

Fig. 2: MC agent

Fig. 3: Training of both agents

We see that the training is very unstable, but has an overall upward trends, which means that both agent is slowly learning how to play the Flappy bird game. We do note that the MC agent is performing significantly better than the λ SARSA agent, but is more prone to variations. To see the decision making of the agent, we show below a heatmap of the Q value functions for both actions with respect to the state tuple.

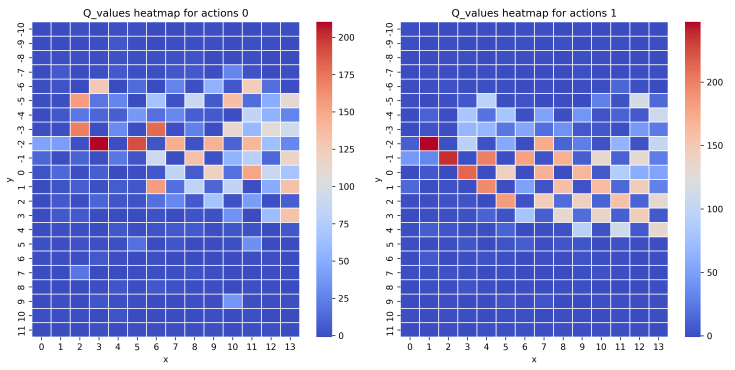

Fig. 4: λ SARSA Q-value

We see that λ Sarsa is less prone to attribute high value to a lot of states (but it can mean that the since the agent didn't survive very long, it didn't got to see a lot of states)

However the Monte Carlo exhibited proper behavior. We do see that for both agents, different actions for the same states are complementary, which means that agent understood that each possible state have a

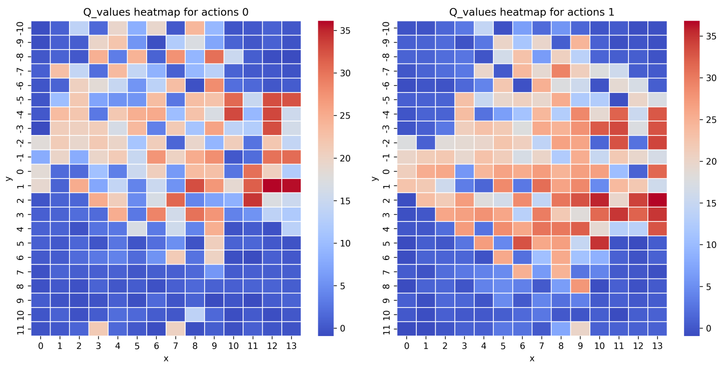

Fig. 5: MC Q-value

well defined action and that when the bird is lower than the gap, both showcase a tendency to flap, which is the optimal way to play.

3 Performance of the agent on different environment

Here we put the agent in different environment and ask them to play the game according to a greedy policy.Since changing the environment also changed the observation space, we have handpicked some parameters so that the agent can indeed play the game. The rewards are averaged over 10 runs with fixed heigh=15 fixed a maximum playtime of 2000

λ SARSA fixed	pg=2	pg=4	pg=6	pg=8
w=15	8.0	8.0	25.6	30.1
w=20	10.6	12.2	52.5	60.3

Table 1: λ SARSA fixed results

MC agent	pg=2	pg=4	pg=6	pg=8
w=15	22	76.4	126.6	221.2
w=20	19	74.5	678.2	2000

Table 2: MC agent results

Table 3: Comparison of λ SARSA fixed and MC agent results

We see in Table3 a higher pipe gap makes the game easier to play, which seems logical. The overall shape of the environment also makes the game easier play, both account for the fact that the bird has more leeway.

4 Parameter sweeps

Here conduct some parameters sweep for both agents. With varying parameters, we trained both agents and task them to play the game greedily on the default environment parameters.
For the λ Sarsa, we have varied α versus λ and ϵ versus λ.We see that with a very small exploration incentive during training, the λ parametes matters a lot less and that a sweet balance between the learning rates and decay needs to found for a good performance.

As for the MC agent, we see that both a small learning rate and high learning (α) with rather average ϵ yield very good results, whereas extreme such as small learning rate and small ϵ yield very poor results. On the flip side, one high and one low yield average results, showcasing that the stability of a MC method results in the trade off of choosing a lot of good action in the episode.

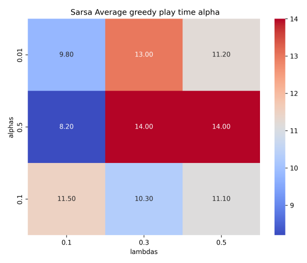

Fig. 6: λ SARSA α VS λ

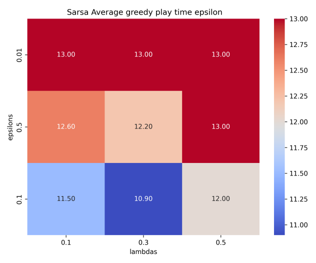

Fig. 7: λ SARSA ϵ VS λ

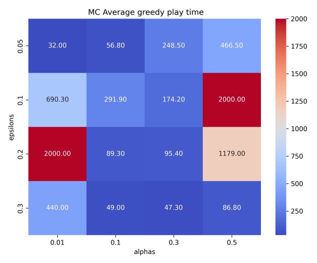

Fig. 8: MC α VS ϵ

Fig. 9: Playtime with in parameter sweep

5 Discussion

Overall, we have seen that the Monte Carlo agent performs much better than the λ SARSA agent, meaning that the Flappy bird environnement play into the strengh of the Monte Carlo techniques (However, bad implementation of the λ Sarsa cannot be ruled out). We also seen that the Monte carlo method works best when there is a good tradeoff between learning the past episode and exploratoty tendencies. The experiments and plots can be replicated at https://github.com/C-Davion/CSRL.

References

1. Sutton, Richard S. and Barto, Andrew G.. Reinforcement Learning: An Introduction. Second : The MIT Press, 2018 .