Reinforcement Learning in Text Flappy Bird

Emilien Biré

1 Introduction

This report presents the implementation and comparison of two reinforcement learning agents, a Monte Carlo (MC) agent and an Expected Sarsa agent, for solving the Text Flappy Bird (TFB) environment. The TFB environment provides two observation modes: one returning a complete screen render and another providing the distance of the player from the center of the nearest pipe gap along the x and y axes. The objective of this assignment is to analyze the performance, convergence properties, and generalization abilities of both agents, on the second version of the environment.

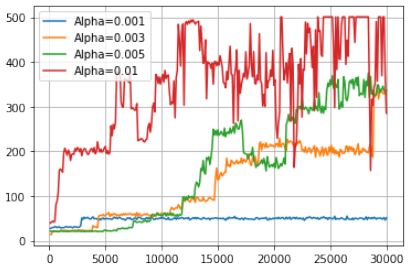
2 Experimental Setup

The two agents were implemented using the OpenAI Gym interface provided for TFB. The MC agent updates value estimates only at the end of an episode, relying on complete episode trajectories to improve the policy. In contrast, the Sarsa agent uses temporal difference (TD) learning with eligibility traces, allowing it to update estimates within an episode based on intermediate rewards. After implementing the agents, we focused on finding the best step size (α parameter) and exploration parameter (ϵ) for both agents.

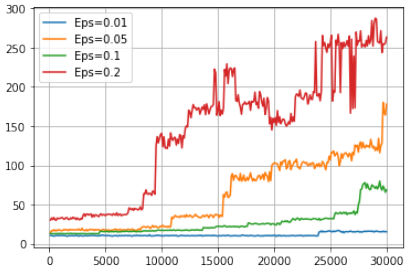
2.1 Parameters search and results

Figure **??** showcase the search of α and ϵ parameters to find the best combinations for the MCC and Sarsa agents. We can clearly see that the MCC agent has a really high variance in the end of the training, for bigger step sizes. When epsilon increases, the maximum reward is higer for a fixed step size. As for the Sarsa agent, the training is more stable (we start to see high variance for bigger step sizes), and the same behaviors appear when we increase epsilon. The MC agent exhibits high variance in learning as it updates policies only after full episodes are completed. The Sarsa agent, benefiting from on-policy updates and eligibility traces, adapts more quickly to new situations and converges faster.

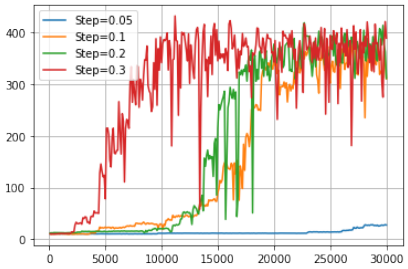
We can extract the best set of parameters and compare the two agents on Figure 2. We can see that MCC can achieve a higher end reward after 30000 steps, but Sarsa indeed converges faster and seems more stable.



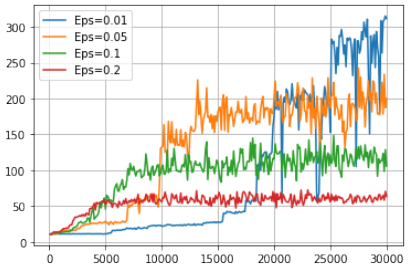
(a) α sweep for MCC with $\epsilon = 0.02$



(b) ϵ sweep for MCC with $\alpha = 0.02$



(c) α sweep for Sarsa with $\epsilon = 0.02$



(d) ϵ sweep for Sarsa with $\alpha = 0.02$

Figure 1: α and ϵ parameter sweeps

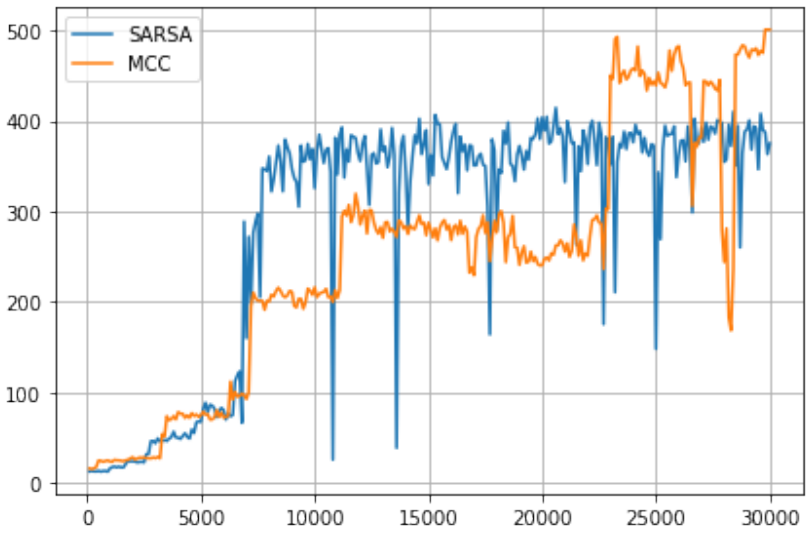
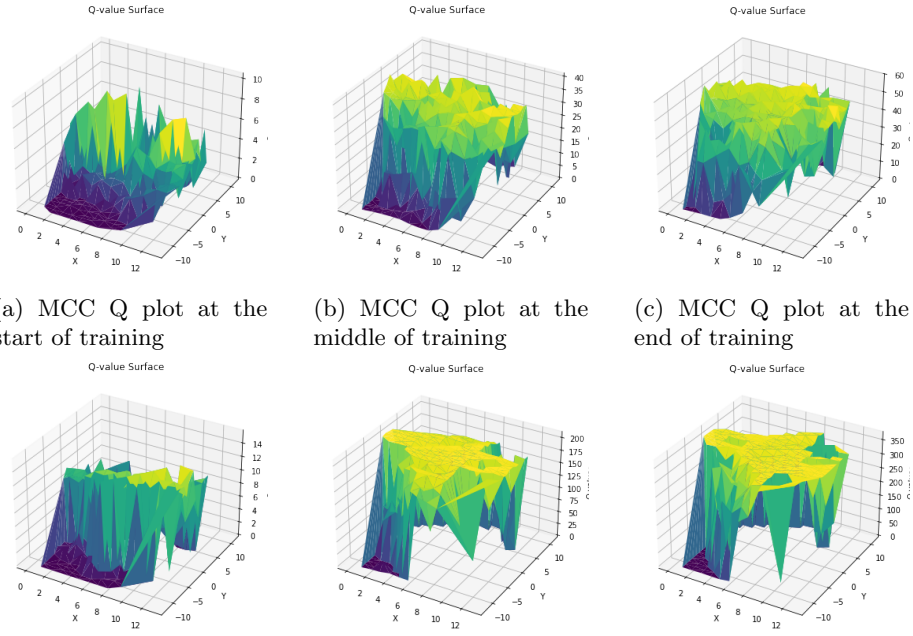


Figure 2: Comparison of best Sarsa and MCC agents



(a) MCC Q plot at the start of training

(b) MCC Q plot at the middle of training

(c) MCC Q plot at the end of training

(d) Sarsa Q plot at the start of training

(e) Sarsa Q plot at the middle of training

(f) Sarsa Q plot at the end of training

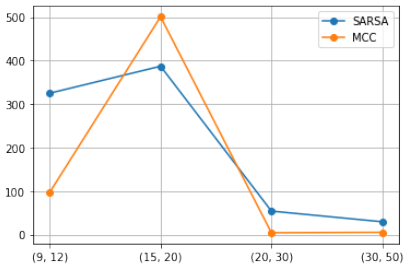
Figure 3: Q-values surface plots throughout training of the two agents

2.2 State-value function visualization

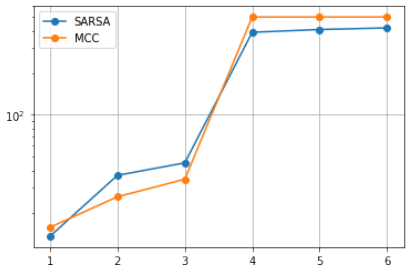
After having trained the best MCC and Sarsa agents, we can see how the value s look like, at different steps of training. Fig. 3 showcases that the shape of the Q function for MCC is noisier and less clearer, whereas for Sarsa, we can see a clear plateau on which the expected return is higher, even if a weird gap can be seen around the state (10,-5).

3 Generalization Across Configurations

A trained agent's performance was evaluated across different environment configurations, including variations in pipe gap size and screen dimensions. The default configuration was $(w, h) = (15, 20)$ in screen size, and $p = 4$ for the pipe gap. Both agents faced difficulties to generalize: when the pipe gap decreases, we face a huge drop in performance, as pictured in Fig. 4. Same as when the environment begins to be bigger than what the agent have seen during training. Regardless of the performance on the original environment, the Sarsa agent seems to be a little bit better on new environment configurations, showcasing its stability.



(a) Generalization of agents on different size of environments.



(b) Generalization of agents on different pipe gaps.

Figure 4: Generalization of agents on different environment configurations.

4 Application to the Original Flappy Bird Game

The original Flappy Bird environment provides observations being the current frame of the game, thus an image. The sate space is in that case really big, making MC and Q-learning practically impossible, due to memory issues.

5 Conclusion

The comparison between MC and Sarsa highlights fundamental trade-offs in reinforcement learning. The MC agent, though capable of achieving superior long-term performance, requires extensive training. The Sarsa(λ) agent, benefiting from incremental updates, demonstrates faster convergence and better adaptability.